

Enrico Macii  
Vassilis Paliouras  
Odysseas Koufopavlou (Eds.)

LNCS 3254

# Integrated Circuit and System Design

Power and Timing Modeling,  
Optimization and Simulation

14th International Workshop, PATMOS 2004  
Santorini, Greece, September 2004  
Proceedings

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*New York University, NY, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Moshe Y. Vardi

*Rice University, Houston, TX, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

This page intentionally left blank

Enrico Macii Vassilis Paliouras  
Odysseas Koufopavlou (Eds.)

# Integrated Circuit and System Design

Power and Timing Modeling,  
Optimization and Simulation

14th International Workshop, PATMOS 2004  
Santorini, Greece, September 15-17, 2004  
Proceedings

eBook ISBN: 3-540-30205-0  
Print ISBN: 3-540-23095-5

©2005 Springer Science + Business Media, Inc.

Print ©2004 Springer-Verlag  
Berlin Heidelberg

All rights reserved

No part of this eBook may be reproduced or transmitted in any form or by any means, electronic, mechanical, recording, or otherwise, without written consent from the Publisher

Created in the United States of America

Visit Springer's eBookstore at:  
and the Springer Global Website Online at:

<http://ebooks.springerlink.com>  
<http://www.springeronline.com>

# Preface

Welcome to the proceedings of PATMOS 2004, the fourteenth in a series of international workshops. PATMOS 2004 was organized by the University of Patras with technical co-sponsorship from the IEEE Circuits and Systems Society.

Over the years, the PATMOS meeting has evolved into an important European event, where industry and academia meet to discuss power and timing aspects in modern integrated circuit and system design. PATMOS provides a forum for researchers to discuss and investigate the emerging challenges in design methodologies and tools required to develop the upcoming generations of integrated circuits and systems. We realized this vision this year by providing a technical program that contained state-of-the-art technical contributions, a keynote speech, three invited talks and two embedded tutorials. The technical program focused on timing, performance and power consumption, as well as architectural aspects, with particular emphasis on modelling, design, characterization, analysis and optimization in the nanometer era.

This year a record 152 contributions were received to be considered for possible presentation at PATMOS. Despite the choice for an intense three-day meeting, only 51 lecture papers and 34 poster papers could be accommodated in the single-track technical program. The Technical Program Committee, with the assistance of additional expert reviewers, selected the 85 papers to be presented at PATMOS and organized them into 13 technical sessions. As was the case with the PATMOS workshops, the review process was anonymous, full papers were required, and several reviews were received per manuscript.

Beyond the presentations of the papers, the PATMOS technical program was enriched by a series of speeches offered by world-class experts, on important emerging research issues of industrial relevance. This year's keynote speech was offered by Prof. Hugo De Man, IEEE Fellow, IMEC and KUL, Belgium. The title of the keynote speech was "Connecting E-Dreams to Deep-Submicron Realities." Furthermore, the technical program was augmented with three invited talks, given by leading industry experts, namely Dr. Nick Kanopoulos, Director of Multimedia Communications, ATMEL, USA, Prof. Kiyo Itoh, Hitachi Fellow, Hitachi, Japan and Dr. Carlo Dallavalle, Telecom Group Process Technology Director, STMicroelectronics, Agrate Brianza, Italy. Finally, two tutorials were embedded within the technical program on two very hot emerging design issues. In particular, Prof. W. Nebel, OFFIS, Germany, gave a tutorial on the "Importance and Sources of Leakage Power in DSM-CMOS Circuits," while Sachin S. Sapatnekar, University of Minnesota, USA, presented "The Certainty of Uncertainty: Randomness in Nanometer Design."

Last, but not least, the rich social program coupled with the unique Santorini settings provided the additional opportunities for establishing new relationships

among the workshop participants. We sincerely hope that the attendees found PATMOS 2004 at Santorini a stimulating and fruitful experience and that they enjoyed their stay on Santorini throughout the duration of the event. We would like to thank the several people who voluntarily worked to make this edition of PATMOS possible, the expert reviewers, the members of the technical program and steering committees, and the invited speakers who offered their skill, time, and deep knowledge to make PATMOS 2004 a memorable event.

September 2004

Enrico Macii  
Vassilis Paliouras  
Odysseas Koufopavlou

# Organization

## Organizing Committee

General Co-chairs	Assist. Prof. Vassilis Paliouras, U. of Patras, Greece Prof. Odysseas Koufopavlou, U. of Patras, Greece
Technical Program Chair	Prof. Enrico Macii, Politecnico di Torino, Italy
Industrial Chair	Dr. Roberto Zafalon, STMicroelectronics, Italy
Conference Co-ordinator	Dr. Konstantina Karagianni, U. of Patras, Greece

## PATMOS Technical Program Committee

B. Al-Hashimi - University of Southampton, UK  
A. Alvandpour - Linköping University, Sweden  
D. Auvergne - LIRMM, France  
D. Bertozzi - University of Bologna, Italy  
A. Bogliolo - University of Urbino, Italy  
J. Bormans - IMEC, Belgium  
J.A. Carballo - IBM, USA  
N. Chang - Seoul National University, Korea  
J. Figueras - University of Catalunya, Spain  
E. Friedman - University of Rochester, USA  
C.E. Goutis - University Patras, Greece  
J.L. Guntzel - Universidade Federal de Pelotas, Brazil  
A. Guyot - TIMA laboratory, France  
R. Hartenstein - University of Kaiserslautern, Germany  
J. Juan Chico - University of Sevilla, Spain  
S. Khatri - University of Colorado, USA  
P. Larsson-Edefors - Chalmers Technical University, Sweden  
V. Moshnyaga - University of Fukuoka, Japan  
W. Nebel - University of Oldenburg, Germany  
J.A. Nossek, Technical University Munich, Germany  
A. Nunez, University of Las Palmas, Spain  
M. Papaefthymiou - University of Michigan, USA  
F. Pessolano - Philips, The Netherlands  
H. Pfeleiderer - University of Ulm, Germany  
C. Piguët, - CSEM, Switzerland  
M. Poncino, University of Verona, Italy  
R. Reis - University of Porto Alegre, Brazil  
M. Robert - University of Montpellier, France  
A. Rubio - University of Catalunya, Spain  
D. Sciuto - Politecnico di Milano, Italy  
D. Soudris, - University of Thrace, Greece  
J. Sparsø - DTU, Denmark



## VIII Organization

A. Stauffer - EPFL, Lausanne, Switzerland  
A. Stempkowsky - Academy of Sciences, Russia  
T. Stouraitis - University of Patras, Greece  
A.M. Trullemans - University of Louvain-la-Neuve, Belgium  
R. Zafalon - STMicroelectronics, Italy

### **PATMOS Steering Committee**

D. Auvergne, University of Montpellier, France  
R. Hartenstein, University of Kaiserslautern, Germany  
W. Nebel, University of Oldenburg, Germany  
C. Piguet, CSEM, Switzerland  
A. Rubio, University of Catalunya, Spain  
J. Figueras, University of Catalunya, Spain  
B. Ricco, University of Bologna, Italy  
D. Soudris, University of Thrace, Greece  
J. Sparsø, DTU, Denmark  
A.M. Trullemans, University of Louvain-la-Neuve, Belgium  
P. Pirsch, University of Hannover, Germany  
B. Hochet, EIVD, Switzerland  
A.J. Acosta, University of Sevilla/IMSE-CSM, Spain  
J. Juan, University of Sevilla/IMSE-CSM, Spain  
E. Macii, Politecnico di Torino, Italy  
R. Zafalon, STMicroelectronics, Italy  
V. Paliouras, University of Patras, Greece  
J. Vounckx, IMEC, Belgium

### **Executive Steering Subcommittee**

President	Joan Figueras, University of Catalunya, Spain
Vice-president	Reiner Hartenstein, University of Kaiserslautern, Germany
Secretary	Wolfgang Nebel, University of Oldenburg, Germany

### **Additional Reviewers**

L. Fesquet	N. Kaczoreck	S. Mamagkakis
C. Layer	M. Hillers	M. Dasygenis
O. Pfänder	D. Helms	E. Lattanzi
J. Rauscher	A. Stammermann	A. Bona
W. Schlecker	A. Nanarelli	V. Zaccaria
E. Sönmez	T. Bjerregaard	G. Bertoni
H. Kabza	S.-M. Yoo	M. Loghi
R. Michalzik	F. Gebara	S. Salerno
H. Schumacher	G. Koytroumpetis	P. Maurine
A. Schulz	K. Tatas	

# Table of Contents

## Keynote Speech

Connecting E-Dreams to Deep-Submicron Realities . . . . .	1
<i>Hugo De Man</i>	

## Invited Talks

Design Methodology for Rapid Development of SoC ICs Based on an Innovative System Architecture with Emphasis to Timing Closure and Power Consumption Optimization . . . . .	2
<i>Nick Kanopoulos</i>	
Low-Voltage Embedded RAMs – Current Status and Future Trends . . . . .	3
<i>Kiyoo Itoh, Kenichi Osada, Takayuki Kawahara</i>	
Adaptive Subthreshold Leakage Reduction Through N/P Wells Reverse Biasing . . . . .	16
<i>Carlo Dallavalle</i>	

## Embedded Tutorials

Leakage in CMOS Circuits – An Introduction . . . . .	17
<i>D. Helms, E. Schmidt, W. Nebel</i>	
The Certainty of Uncertainty: Randomness in Nanometer Design . . . . .	36
<i>Hongliang Chang, Haifeng Qian, Sachin S. Sapatnekar</i>	

## Session 1: Buses and Communication

Crosstalk Cancellation for Realistic PCB Buses . . . . .	48
<i>Jihong Ren, Mark R. Greenstreet</i>	
A Low-Power Encoding Scheme for GigaByte Video Interfaces . . . . .	58
<i>Sabino Salerno, Enrico Macii, Massimo Poncino</i>	
Dynamic Wire Delay and Slew Metrics for Integrated Bus Structures . . . . .	69
<i>Markus Tahedl, Hans-Jörg Pfliederer</i>	
Perfect 3-Limited-Weight Code for Low Power I/O . . . . .	79
<i>Mircea R. Stan, Yan Zhang</i>	
A High-Level DSM Bus Model for Accurate Exploration of Transmission Behaviour and Power Estimation of Global System Buses . . . . .	90
<i>Claudia Kretzschmar, Torsten Bitterlich, Dietmar Müller</i>	

**Session 2: Circuits and Devices (I)**

Performance Metric Based Optimization Protocol ..... 100  
*X. Michel, A. Verle, P. Maurine, N. Azémard, D. Auvergne*

Temperature Dependence in Low Power CMOS UDSM Process ..... 110  
*B. Lasbouygues, R. Wilson, P. Maurine, N. Azémard, D. Auvergne*

Yield Optimization by Means of Process Parameters Estimation:  
 Comparison Between ABB and ASV Techniques ..... 119  
*Mauro Olivieri, Mirko Scarana, Giuseppe Scotti,  
 Alessandro Trifiletti*

High Yield Standard Cell Libraries: Optimization and Modeling ..... 129  
*Nicola Dragone, Michele Quarantelli, Massimo Bertoletti,  
 Carlo Guardiani*

A Study of Crosstalk Through Bonding and Package Parasitics  
 in CMOS Mixed Analog-Digital Circuits ..... 138  
*Gabriella Trucco, Giorgio Boselli, Valentino Liberali*

**Session 3: Low Power (I)**

Sleepy Stack Reduction of Leakage Power ..... 148  
*Jun Cheol Park, Vincent J. Mooney III, Philipp Pfeiffenberger*

A Cycle-Accurate Energy Estimator for CMOS Digital Circuits ..... 159  
*Eunseok Song, Young-Kil Park, Soon Kwon, Soo-Ik Chae*

Leakage Reduction at the Architectural Level and Its Application  
 to 16 Bit Multiplier Architectures ..... 169  
*Christian Schuster, Jean-Luc Nagel, Christian Piguët,  
 Pierre-André Farine*

Reducing Cross-Talk Induced Power Consumption and Delay ..... 179  
*André K. Nieuwland, Atul Katoch, Maurice Meijer*

Investigation of Low-Power Low-Voltage Circuit Techniques  
 for a Hybrid Full-Adder Cell ..... 189  
*Ilham Hassoune, Amaury Neve, Jean-Didier Legat, Denis Flandre*

Leakage Power Analysis and Comparison  
 of Deep Submicron Logic Gates ..... 198  
*Geoff Merrett, Bashir M. Al-Hashimi*

**Session 4: Architectures**

Threshold Mean Larger Ratio Motion Estimation in MPEG Encoding  
 Using LNS ..... 208  
*Jie Ruan, Mark G. Arnold*

Energy- and Area-Efficient Deinterleaving Architecture for High-Throughput Wireless Applications . . . . .	218
<i>Armin Wellig, Julien Zory, Norbert Wehn</i>	
Register Isolation for Synthesizable Register Files . . . . .	228
<i>Matthias Müller, Andreas Wortmann, Dominik Mader, Sven Simon</i>	
Discrete-Event Modeling and Simulation of Superscalar Microprocessor Architectures . . . . .	238
<i>C. Brandolese, W. Fornaciari, F. Salice</i>	
Design of High-Speed Low-Power Parallel-Prefix VLSI Adders . . . . .	248
<i>G. Dimitrakopoulos, P. Kolovos, P. Kalogerakis, D. Nikolas</i>	
<b>Session 5: Asynchronous Circuits</b>	
GALSification of IEEE 802.11a Baseband Processor . . . . .	258
<i>Miloš Krstić, Eckhard Grass</i>	
TAST Profiler and Low Energy Asynchronous Design Methodology . . . . .	268
<i>Kamel Slimani, Yann Rémond, Gilles Sicard, Marc Renaudin</i>	
Low Latency Synchronization Through Speculation . . . . .	278
<i>D.J. Kinniment, A.V. Yakovlev</i>	
Minimizing the Power Consumption of an Asynchronous Multiplier . . . . .	289
<i>Yijun Liu, Steve Furber</i>	
A Channel Library for Asynchronous Circuit Design Supporting Mixed-Mode Modeling . . . . .	301
<i>T. Bjerregaard, S. Mahadevan, J. Sparsø</i>	
<b>Session 6: System Design</b>	
L0 Cluster Synthesis and Operation Shuffling . . . . .	311
<i>Murali Jayapala, Tom Vander Aa, Francisco Barat, Francy Catthoor, Henk Corporaal, Geert Deconinck</i>	
On Combined DVS and Processor Evaluation . . . . .	322
<i>Anders Brødløs Olsen, Finn Büttner, Peter Koch</i>	
A Multi-level Validation Methodology for Wireless Network Applications . . . . .	332
<i>C. Drosos, L. Bisdounis, D. Metafas, S. Blionas, A. Tatsaki</i>	
SoftExplorer: Estimation, Characterization, and Optimization of the Power and Energy Consumption at the Algorithmic Level . . . . .	342
<i>Eric Senn, Johann Laurent, Nathalie Julien, Eric Martin</i>	

Run-Time Software Monitor of the Power Consumption of Wireless  
Network Interface Cards ..... 352  
*Emanuele Lattanzi, Andrea Acquaviva, Alessandro Bogliolo*

Towards a Software Power Cost Analysis Framework  
Using Colored Petri Net ..... 362  
*Meuse N. Oliveira Júnior, Paulo R. Martins Maciel,  
Raimundo S. Barreto, Fernando F. Carvalho*

**Session 7: Circuits and Devices (II)**

A 260ps Quasi-static ALU in 90nm CMOS ..... 372  
*F. Pessolano, R.I.M.P. Meijer*

Embedded EEPROM Speed Optimization Using System  
Power Supply Resources ..... 381  
*Jean-Michel Daga, Caroline Papaix, Marylene Combe,  
Emmanuel Racape, Vincent Sialelli*

Single Supply Voltage High-Speed Semi-dynamic Level-Converting  
Flip-Flop with Low Power and Area Consumption ..... 392  
*Stephan Henzler, Georg Georgakos, Jörg Berthold,  
Doris Schmitt-Landsiedel*

A Predictive Synchronizer for Periodic Clock Domains ..... 402  
*Uri Frank, Ran Ginosar*

Power Supply Net for Adiabatic Circuits ..... 413  
*Jürgen Fischer, Ettore Amirante, Agnese Bargagli-Stoffi,  
Philip Teichmann, Dominik Gruber, Doris Schmitt-Landsiedel*

**Session 8: Interconnect and Physical Design**

A Novel Layout Approach Using Dual Supply Voltage Technique  
on Body-Tied PD-SOI ..... 423  
*Kazuki Fukuoka, Masaaki Iijima, Kenji Hamada,  
Masahiro Numa, Akira Tada*

Simultaneous Wire Sizing and Decoupling Capacitance Budgeting  
for Robust On-Chip Power Delivery ..... 433  
*Jingjing Fu, Zuying Luo, Xianlong Hong, Yici Cai,  
Sheldon X.-D. Tan, Zhu Pan*

An Efficient Low-Degree RMST Algorithm  
for VLSI/ULSI Physical Design ..... 442  
*Yin Wang, Xianlong Hong, Tong Jing, Yang Yang,  
Xiaodong Hu, Guiying Yan*

Wirelength Reduction Using 3-D Physical Design .....	453
<i>Idris Kaya, Silke Salewski, Markus Olbrich, Erich Barke</i>	
On Skin Effect in On-Chip Interconnects .....	463
<i>Daniel A. Andersson, Lars "J" Svensson, Per Larsson-Edefors</i>	

## Session 9: Security and Safety

A Low and Balanced Power Implementation of the AES Security Mechanism Using Self-Timed Circuits .....	471
<i>D. Shang, F. Burns, A. Bystrov, A. Koelmans, D. Sokolov, A. Yakovlev</i>	
A Power Consumption Randomization Countermeasure for DPA-Resistant Cryptographic Processors .....	481
<i>Marco Bucci, Michele Guglielmo, Raimondo Luzzi, Alessandro Trifiletti</i>	
A Flexible and Accurate Energy Model of an Instruction-Set Simulator for Secure Smart Card Software Design .....	491
<i>Ulrich Neffe, Klaus Rothbart, Christian Steger, Reinhold Weiss, Edgar Rieger, Andreas Muehlberger</i>	
The Impact of Low-Power Techniques on the Design of Portable Safety-Critical Systems .....	501
<i>Athanasios P. Kakarountas, Vassilis Spiliotopoulos, Spiros Nikolaidis, Costas E. Goutis</i>	

## Session 10: Low Power (II)

Modular Construction and Power Modelling of Dynamic Memory Managers for Embedded Systems .....	510
<i>David Aienza, Stylianos Mamagkakis, Francky Catthoor, J.M. Mendias, D. Soudris</i>	
PIRATE: A Framework for Power/Performance Exploration of Network-on-Chip Architectures .....	521
<i>Gianluca Palermo, Cristina Silvano</i>	
Power Consumption of Performance-Scaled SIMD Processors .....	532
<i>Anteneh A. Abbo, Richard P. Kleihorst, Vishal Choudhary, Leo Sevat</i>	
Low Effort, High Accuracy Network-on-Chip Power Macro Modeling ....	541
<i>Andrea Bona, Vittorio Zaccaria, Roberto Zafalon</i>	
Exploiting Dynamic Workload Variation in Offline Low Energy Voltage Scheduling .....	553
<i>Lap-Fai Leung, Chi-Ying Tsui, Xiaobo Sharon Hu</i>	

**Session 11: Low-Power Processing (Poster)**

Design of a Power/Performance Efficient Single-Loop Sigma-Delta Modulator for Wireless Receivers ..... 564  
*Ana Rusu, Alexei Borodenkov, Mohammed Ismail, Hannu Tenhunen*

Power Aware Dividers in FPGA ..... 574  
*Gustavo Sutter, Jean-Pierre Deschamps, Gery Bioul, Eduardo Boemo*

A Dual Low Power and Crosstalk Immune Encoding Scheme for System-on-Chip Buses ..... 585  
*Zahid Khan, Tughrul Arslan, Ahmet T. Erdogan*

The Effect of Data-Reuse Transformations on Multimedia Applications for Different Processing Platforms ..... 593  
*N. Vassiliadis, A. Chormoviti, N. Kavvadias, S. Nikolaidis*

Low Power Co-design Tool and Power Optimization of Schedules and Memory System ..... 603  
*Patricia Guitton-Ouhamou, Hanene Ben Fradj, Cécile Belleudy, Michel Auguin*

Hardware Building Blocks of a Mixed Granularity Reconfigurable System-on-Chip Platform ..... 613  
*K. Masselos, S. Blionas, J-Y. Mignolet, A. Foster, D. Soudris, S. Nikolaidis*

Enhancing GALS Processor Performance Using Data Classification Based on Data Latency ..... 623  
*S. López, O. Garnica, J.M. Colmenar*

Application Analysis with Integrated Identification of Complex Instructions for Configurable Processors ..... 633  
*Nikolaos Kavvadias, Spiridon Nikolaidis*

Power Modeling, Estimation, and Optimization for Automated Co-design of Real-Time Embedded Systems ..... 643  
*Amjad Mohsen, Richard Hofmann*

Mapping Computational Intensive Applications to a New Coarse-Grained Reconfigurable Data-Path ..... 652  
*M.D. Galanis, G. Theodoridis, S. Tragoudas, D. Soudris, C.E. Goutis*

Power Estimation for Ripple-Carry Adders with Correlated Input Data ..... 662  
*Kenny Johansson, Oscar Gustafsson, Lars Wanhammar*

LPVIP: A Low-Power ROM-Less ALU for Low-Precision LNS ..... 675  
*Mark G. Arnold*

## Session 12: Digital Design (Poster)

Low Level Adaptive Frequency in Synthesis of High Speed Digital Circuits .....	685
<i>Leonardo Valencia</i>	
A Novel Mechanism for Delay-Insensitive Data Transfer Based on Current-Mode Multiple Valued Logic .....	691
<i>Myeong-Hoon Oh, Dong-Soo Har</i>	
Pipelines in Dynamic Dual-Rail Circuits .....	701
<i>Jing-ling Yang, Chiu-sing Choy, Cheong-fat Chan, Kong-pong Pun</i>	
Optimum Buffer Size for Dynamic Voltage Processors .....	711
<i>Ali Manzak, Chaitali Chakrabarti</i>	
Design Optimization with Automated Cell Generation .....	722
<i>A. Landrault, N. Azémard, P. Maurine, M. Robert, D. Auvergne</i>	
A New Transistor Folding Algorithm Applied to an Automatic Full-Custom Layout Generation Tool .....	732
<i>Fabricio B. Bastian, Cristiano Lazzari, Jose Luis Guntzel, Ricardo Reis</i>	
A Novel Constant-Time Fault-Secure Binary Counter .....	742
<i>Dimitris Karatasos, Athanasios Kakarountas, George Theodoridis, Costas Goutis</i>	
Buffer Sizing for Crosstalk Induced Delay Uncertainty .....	750
<i>Dimitrios Velenis, Eby G. Friedman</i>	
Optimal Logarithmic Representation in Terms of SNR Behavior .....	760
<i>P. Vouzis, V. Paliouras</i>	
A New Logic Transformation Method for Both Low Power and High Testability .....	770
<i>Y.S. Son, J. W. Na</i>	
Energy-Efficient Hardware Architecture for Variable N-point 1D DCT ...	780
<i>Andrew Kinane, Valentin Muresan, Noel O'Connor, Noel Murphy, Sean Marlow</i>	

## Session 13: Modeling and Simulation (Poster)

Two Level Compact Simulation Methodology for Timing Analysis of Power-Switched Circuits .....	789
<i>Stephan Henzler, Georg Georgakos, Jörg Berthold, Doris Schmitt-Landsiedel</i>	



A Generic Timing Mechanism for Using the APPLES Gate-Level Simulator in a Mixed-Level Simulation Environment . . . . .	799
<i>Alexander Maili, Damian Dalton, Christian Steger</i>	
Modeling Temporal and Spatial Power Supply Voltage Variation for Timing Analysis . . . . .	809
<i>Howard Chen, Daniel Ostapko</i>	
On Timing and Power Consumption in Inductively Coupled On-Chip Interconnects . . . . .	819
<i>T. Murgan, A. García Ortiz, C. Schlachta, H. Zimmer, M. Petrov, M. Glesner</i>	
Signal Sampling Based Transition Modeling for Digital Gates Characterization . . . . .	829
<i>Alejandro Millán, Jorge Juan, Manuel J. Bellido, Paulino Ruiz-de-Clavijo, David Guerrero, Enrique Ostúa</i>	
Physical Extension of the Logical Effort Model . . . . .	838
<i>B. Lasbouygues, R. Wilson, P. Maurine, N. Azémard, D. Auvergne</i>	
An Extended Transition Energy Cost Model for Buses in Deep Submicron Technologies . . . . .	849
<i>Peter Caputa, Henrik Fredriksson, Martin Hansson, Stefan Andersson, Atila Alvandpour, Christer Svensson</i>	
Moment-Based Estimation of Switching Activity for Correlated Distributions . . . . .	859
<i>Alberto García-Ortiz, Tudor Murgan, Manfred Glesner</i>	
Table-Based Total Power Consumption Estimation of Memory Arrays for Architects . . . . .	869
<i>Minh Q. Do, Per Larsson-Edefors, Lars Bengtsson</i>	
A Physically Oriented Model to Quantify the Noise-on-Delay Effect . . . . .	879
<i>Tobias Gemmeke, Tobias G. Noll</i>	
Noise Margin in Low Power SRAM Cells . . . . .	889
<i>S. Cserveny, J.-M. Masgonty, C. Piguet</i>	
Delay Evaluation of High Speed Data-Path Circuits Based on Threshold Logic . . . . .	899
<i>Peter Celinski, Derek Abbott, Sorin D. Cotofana</i>	
<b>Author Index . . . . .</b>	<b>907</b>

# Connecting E-Dreams to Deep-Submicron Realities

Hugo De Man

Professor K.U.Leuven  
Senior Research Fellow IMEC, Belgium

Today's electronics industry is making the transition from a PC centric world towards Weiser's world of pervasive computing and communication. Mass produced intelligent micro-systems will be present in virtually every person and every object he/she is in touch with. System scenarioists create a plethora of e-Dreams whereby the world starts to adapt to the wishes of the consumer wherever he/she may be.

The adaptiveness of such systems means that these global distributed systems must autonomously organise themselves and adapt to new services without much user interaction. Therefore such systems are software intensive by nature and run on extremely complex and nomadic silicon platforms. In addition, they require human centric interfaces to augment our senses, control our health, wellness, comfort, security and mobility. Most are wearable or permanently hidden in the infrastructure and hence must be low- to ultra-low energy devices.

On the other hand, the technology push for these e-Dreams comes from the relentless CMOS scaling that will reach the 30nm scale in the next decade. Then device sizes start to match the size of bio-and other molecules. Mems are evolving into self-assembled Nems, CMOS will become compatible with all sorts of new materials and multiple dies will be packaged to extremely intelligent e-grains possible scavenging their energy from the environment.

Hence, we are witnessing two worlds that, on the one end, enable themselves but, on the other hand, we see the conceptual gap between these two worlds increase at a Moore type rate.

This talk will focus on a number of challenges resulting from the growing distance between "atoms" and "ities". Atoms refer to the nano-scale physics of the underlying devices while "ities" refers to the expectations of the e-Dreams such as: reliability, dependability, security, adaptivity, interoperability, manufacturability etc.

More in particular the following aspects will be discussed:

1. How to cope with the energy-flexibility-cost issue given the computational power required for nomadic and consumer devices as well as smart sensors for personal wellness and infrastructure monitoring and control. Particular attention must be paid to novel compute and communication architectures, low power memory architectures but also to the creation of the software layers to map e-Dreams on these architectures to operate at lowest possible energy and with guaranteed quality of service in spite of the physical uncertainties caused by nano-scale physics. This provides an enormous challenge to master design complexity far above what today's EDA industry is delivering or that software engineering can produce in a reliable way.
2. Nano-scale physics will profoundly jeopardize the digital abstraction paradigm that allowed us to design very complex digital systems. When scaling below 90nm we have to design systems of 1 billion transistors while, at the same time, coping with increasing gate- and subthreshold leakage power, uncertainty due to the stochastic and discrete nature of doping and lithography and delay dominance by interconnect. Hence designing for worst-case conditions is no longer an option if we want to exploit further scaling. Doing better will require novel design flows based on optimal mapping of applications based on run-time monitoring of timing-power trade-off of the architectural components on-chip. We will present some possible solutions based on multiple Pareto-optimal mappings at design time combined with a run-time optimal control algorithm based on run-time monitoring of timing-power and temperature of chip components.

# Design Methodology for Rapid Development of SoC ICs Based on an Innovative System Architecture with Emphasis to Timing Closure and Power Consumption Optimization

Nick Kanopoulos

Atmel Corporatio  
Multimedia and Communications  
3800 Gateway Centre Blvd.  
Morrisville, NC 27560  
U.S.A

**Abstract.** Complex System on Chip (SoC) ICs require a high development effort and extensive testing in order to meet performance and power consumption specifications. Those requirements are usually achieved through laborious and iterative procedures which engage synthesis, timing, power analysis and back-end tools. Traditional design methodologies deal with the above issues from a synthesis or technology point of view, trying to optimize the propagation delay of primitive cells, simplifying the logic and disabling the clocks when possible. This presentation describes how a design methodology can be combined with an innovative architecture, called COMBA, for system on chip ICs in order to help the designer to achieve timing closure and meet performance and power consumption requirements in short time. This methodology is supported by tools that produce the gate level description of the system using pre-optimized building blocks for the COMBA architecture, leading to minimum development and testing time while preserving the performance and power requirements.

# Low-Voltage Embedded RAMs – Current Status and Future Trends

Kiyoo Itoh, Kenichi Osada, and Takayuki Kawahara

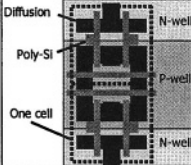
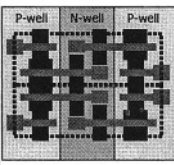
Central Research Laboratory, Hitachi, Ltd.,  
Kokubunji, Tokyo, 185-8601, Japan.  
k-itoh@cr1.hitachi.co.jp

**Abstract.** Low-voltage high-density embedded (e-) RAMs, focusing on RAM cells and peripheral circuits, are described. First, challenges and trends in low-voltage e-RAMs are described based on the S/N issue of RAM cells, and leakage and speed-variation issues of peripheral circuits. Next, state-of-the-art low-voltage e-DRAMs and e-SRAMs are investigated, focusing on leakage-reduction circuits. Finally, future prospects for e-RAM cells and peripheral circuits are discussed in terms of low-voltage designs.

## 1 Introduction

Low-voltage high-density embedded RAMs (e-RAMs) are becoming increasingly important because they play critical roles in reducing power dissipation and chip size of MPU/MCU/SoC for power-aware systems. Thus, research and development [1-6] aiming at sub-1-V RAMs has been active, as exemplified by the presentation of a 0.6-V 16-Mb e-DRAM [5]. To create such e-RAMs, however, many challenges remain with RAM cells and peripheral circuits [1,2]: For RAM cells, in addition to being the smallest and simplest cell possible, signal-to-noise ratio (S/N) must be maintained high to ensure stable and reliable operations because the ratio is always worsened at low voltages as the signal charge of non-selected cells and the signal voltage developed on the data (bit) line decrease. It is also worsened when leakage in a cell increases. The traditional six-transistor (6-T) SRAM cell is well known to suffer from its large size, large soft-error rate (SER), and narrow static noise margin even at about 1.2V. This poses the difficult question of what makes for the best alternative. In the meantime, non-volatile RAMs are emerging as candidates. For peripheral circuits, both leakage and speed variations must be reduced [1, 2] because they are prominent at a lower voltage even for a fixed design parameter variation. Unfortunately, the parameter variations are more enhanced as the devices are scaled. Thus, in addition to new devices such as a fully-depleted (FD) SOI [7] and high-k gate insulators, a variety of solutions such as raising the threshold voltage ( $V_T$ ) of MOST and controlling internal power-supply voltages [1, 2] have been proposed to cope with the issues.

In this paper, a view of circuit designs on RAM cells and peripheral circuits in low-voltage high-density e-RAMs is presented. First, challenges and trends in low-voltage e-RAMs are described in terms of the S/N issue in RAM cells that includes the signal charge, signal voltage and leakage, and leakage and speed-variation issues of peripheral circuits. Next, state-of-the-art low-voltage e-DRAMs and e-SRAMs are investi-

	Conventional cells	LS cells
		
Cell aspect ratio (height/width)	>1	<1
Well layer	Parallel to WL	Parallel to DL
Diffusion layer	Bended	Straight and parallel to DL
Poly-silicon layer	Two directions	Straight and parallel to WL
Application of advanced lithography (phase shift or OPC*)	Difficult	Easy
Pattern fluctuation	Large	Small
Immunity to mask misalignment	Poor	Good
Electrical balance	Poor	Good
Scalability	Poor	Good
Low voltage operation	Difficult	Possible

\*OPC: Optical Proximity Correction

Fig. 1. Comparisons between conventional SRAM cell and LS cell [18].

gated, focusing on leakage-reduction circuits. Finally, future prospects for RAM cells and peripheral circuits are discussed in terms of low-voltage designs.

## 2 Challenges and Trends in Low-Voltage E-RAMs

### 2.1 RAM Cells

**Signal Charge:** Maintaining the signal charge ( $Q_s$ ) of non-selected cells [1,2] is crucial because  $Q_s$  is usually decreased as the power supply voltage ( $V_{DD}$ ) is lowered, and the storage-node capacitance of cell ( $C_s$ ) is decreased with device scaling. Otherwise, data-retention characteristics are degraded. For example, the SER of a 6-T cell [2] rapidly increases with device scaling due to a small and parasitic  $C_s$ , while that of a 1-T DRAM cell is gradually decreased due to an intentionally-added larger  $C_s$ . In addition to using an on-chip ECC as a system solution, adding a capacitor to the cell node is another solution [2].

**Signal Voltage:** Reductions in  $V_T$  variation ( $\alpha(V_T)$ ) and  $V_T$  mismatch ( $\delta V_T$ ) between paired MOSTs [1,2] lower the minimum value of operable  $V_{DD}$ . In particular, a reduction in or compensation for  $\delta V_T$  in flip-flop circuits (i.e., an SRAM cell itself and an SRAM/DRAM sense amplifier) effectively increases the previously described  $Q_s$  or signal voltage on the data line. The reductions are crucial especially for the 6-T SRAM because ever-larger variations in the sub-100-nm era are major obstacles to low-voltage operations. A lithographically symmetric cell layout (LS cell) [18] shown in Fig. 1 reduces the variations, and thus it is becoming the defacto standard in the industry. Unfortunately, however, both  $\alpha(V_T)$  and  $\delta V_T$  of e-SRAMs are statistically and physically larger than those of e-DRAMs despite the use of the LS cell. This is because more flip flops are used, and the FET size accepted in each flip flop is smaller, causing  $\alpha(V_T)$  and  $\delta V_T$  to be more susceptible to process variations: The number in an e-SRAM almost equals  $M$ , while that in an e-DRAM is  $M/n$  ( $M$ : memory

capacity,  $n$ : the number of cells connecting to one sense amplifier, e.g., 64 for a high-speed design and 1,024 for a low-cost design). The MOST size in the 6-T SRAM cell cannot be enlarged without enlarging the memory cell size, while that in the DRAM sense amplifier can effectively utilize the space. Gain cells such as 3-T cells [2] can overcome the issue with enough signal voltage (i.e.,  $V_{DD}$  in some cases) even in an extremely low-voltage region.

**Leakage:** Reducing leakage (i.e., gate tunneling current or subthreshold current) is essential not only to maintain the refresh time of DRAM cells but also to suppress an increase in the retention current of SRAM cells [1, 2]. The gate tunneling current [8, 10] is sensitive to the gate oxide thickness ( $t_{ox}$ ) of a MOSFET, while it is not sensitive to the junction temperature ( $T_j$ ) and gate voltage ( $V_G$ ). For example, a leakage reduction of one order of magnitude requires a  $t_{ox}$  increment of only 2-3 Å, while it requires a  $V_G$  reduction of as much as 0.5 V in a low- $V_{DD}$  region. Thus, although a few reduction circuits such as power switching [9], power-supply lowering [10], and gate-voltage reducing [8] ones have been proposed, developing leakage-free high- $k$  gate insulators is more effective. However, the subthreshold current is not sensitive to MOST structures. Even a conventional FD-SOI device with a reduced subthreshold swing (i.e.,  $S$ -factor) cannot reduce the leakage satisfactorily. However, it is quite sensitive to  $T_j$  and  $V_G$  or  $V_T$  [1, 2, 10]. For example, a leakage reduction of one order of magnitude calls for a  $V_G$  reduction or  $V_T$  increase of only 0.1 V. Thus, circuit development to control  $V_G$  and  $V_T$  is more effective in reducing the leakage. Fortunately, the control is relatively easy thanks to a small voltage swing being required.

The  $V_T$  of the transfer MOST in the 1-T DRAM cell must be high and is not scalable [1, 2] to ensure the maximum refresh time ( $t_{REFmax}$ ) and a full- $V_{DD}$  write to the cell. The minimum  $V_T$  ( $=V_{TLD}$ ) must meet  $t_{REFmax}$ -specifications by suppressing the subthreshold current of cell node to data line during “L” data-line disturbances [1]. Furthermore, the word-line voltage ( $V_w$ ) must be higher than the sum of  $V_{DD}$  and  $V_{TFW}$  to perform the full- $V_{DD}$  write. Here,  $V_{TFW}$  is the sum of the  $V_{TLD}$  and a  $V_T$ -increase due to the body effect developed by raising the source (i.e., storage node) by  $V_{DD}$ , and it must thus be quite high. Consequently, the  $V_w$  must also become high, exemplified by 3.3V at  $V_{DD} = 2V$  for a low-cost 64-Mb design and 1.7V at  $V_{DD} = 1V$  for a high-speed 64-Mb design [19]. This  $V_w$  is provided by the so-called word bootstrapping. Even with a low-actual  $V_T$ , a high  $V_T$  (i.e.,  $V_{TFW}$ ) is effectively achieved as the sum of a negative voltage applied to non-selected word lines and a low actual  $V_T$ . This is the so-called negative word-line (NWL) scheme that produces gate-source offset driving. It is discussed later. The NWL scheme is better than the word bootstrapping in terms of the MOST gate stress voltage during activation [1].

The  $V_T$  of the 6-T SRAM cell must also be quite high and is not scalable to meet retention-current specifications because subthreshold currents in the cell rapidly increase as  $V_T$  decreases, as shown in Fig. 2, and eventually dominate even the total active current of an e-SRAM [1,2]. For a low-power design of a 1-Mb e-RAM that allows consuming a leakage of 0.1  $\mu A$  at  $T_{jmax} = 75^\circ C$ , for example, the  $V_T$  at  $25^\circ C$  might be as high as or higher than 0.71 V. For a high-speed design accepting a larger leakage, the  $V_T$  can be as low as 0.49 V with a 1-Mb e-RAM that allows consuming 10  $\mu A$  at  $T_{jmax} = 50^\circ C$ .

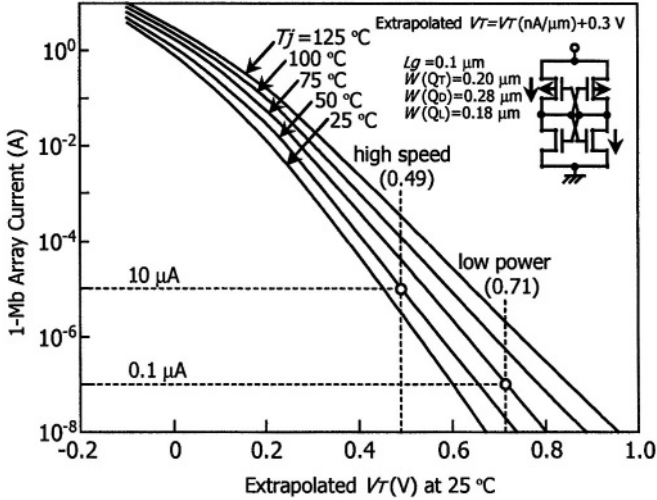


Fig. 2.  $V_T$  of cross-couple MOSTs versus subthreshold current of 1-Mb SRAM.

## 2.2 Peripheral Circuits

**Leakage:** A high-speed reduction scheme applicable even to the active mode is essential because leakage eventually dominates the total active current of an e-RAM as  $V_T$  is lowered [1, 2]. The scheme not only reduces leakage in the standby mode but also performs mode transitions at a high speed while reducing leakage. This performance is exemplified by a fast recovery time from standby to active mode that is a key to mobile applications. Thus, it is better if the scheme reduces more leakage with a smaller voltage swing (i.e., better reduction efficiency) or confines the load capacitance to the small. Leakage reduction is maximized if the high-speed scheme is combined with another scheme that enables confining active (i.e., selected) blocks in an e-RAM to the small every instance within a cycle time [2]. This is because if the high-speed scheme is applied to the remaining inactive (i.e., non-selected) blocks that dominate the total leakage of the e-RAM, the leakage of the e-RAM in the active mode is minimized. Fortunately, peripheral logic circuits in e-RAMs accept such schemes and reduce leakage easily and effectively, whereas random logic circuits in MPU/MCU/SOC [1, 2] do not. This stems from the following features of RAMs: Peripheral circuits consist of many iterative circuit blocks such as decoder and driver blocks, the leakage of which dominates the total leakage of an e-RAM. All circuits in each block in the active mode, however, are non-selected, except one. This configuration minimizes the active area, enabling simple leakage controls without an area penalty [2]. Moreover, some of the circuits are composed of robust circuits (e.g., NAND decoders) and input-predictable logic so that designers can prepare reduction schemes in advance. Furthermore, the cycle time is slow enough to permit additional time for leakage control. For DRAMs, multi-static  $V_T$  is easily available by utilizing internal voltages necessary for the operations.

Basic circuit concepts for reducing subthreshold current are to use a **high- $V_T$**  MOST that is achieved with a high-actual  $V_T$  or a **low-actual- $V_T$**  MOST. Various reduction

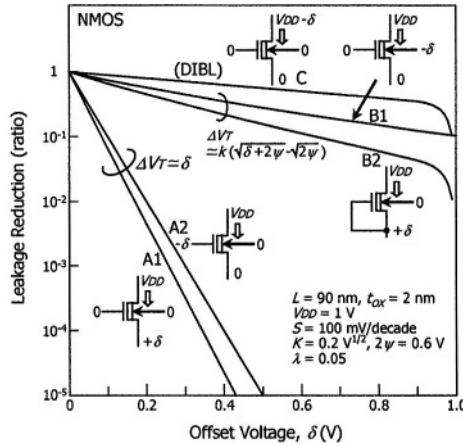


Fig. 3. Leakage reduction efficiency of various schemes [2].

schemes [2] to raise the  $V_T$  of a low-actual  $V_T$  MOST statically or dynamically have been proposed dating back to 1992 [14-17], as shown in Fig. 3. Of the schemes, gate-source backbiasing categorized as gate-source self-backbiasing (A1) and gate-source offset driving (A2) is best in terms of reduction efficiency. In particular, the gate-source self-backbiasing enables an automatic reduction.

**Speed Variations:** Suppressing speed variations in peripheral circuits is essential because the degree of speed variation for any given variation in design parameters is increased by lowering  $V_{DD}$ , exemplified by  $\alpha(V_T)/(V_{DD} - V_T)$  [2]. Unfortunately, design parameter variations such as  $\alpha(V_T)$  increase with technology scaling (Fig. 8). The challenge, for example, is to control  $V_T$  stringently, compensate for  $V_T$  variation through controlling substrate biases or internal  $V_{DD}$  generated by a voltage-down converter, or use new devices with less  $V_T$  variation.

### 3 State-of-the-Art E-DRAMs

A 0.09- $\mu\text{m}$  16-Mb e-DRAM [5] operating at a record-setting low voltage of 0.6 V was recently presented. It used a 0.195- $\mu\text{m}^2$  trench capacitor ( $C_c = 40$  fF) 1-T cell. The total operating power at 0.6 V and at a 20-ns row cycle was only 39 mW, and the standby and sleep-mode currents at 0.9 V and 105°C were as low as 328 and 34  $\mu\text{A}$ , respectively. Figure 4 shows the data lines, each connecting 128 cells, and sense amps (SAs). Many internal voltages are generated by on-chip converters mainly consisting of charge pumps. An excessively boosted word voltage of 3 V is probably needed for a high-speed charging of a large  $C_c$ , rather than for a full- $V_{DD}$  write. Substrate biases of the SA and other periphery circuits are statically controlled in accordance with process and temperature variations to suppress the speed variations and subthreshold currents. Consequently, the speed of an inverter was improved by 63% for slow process conditions (i.e., high  $V_T$ ), and the subthreshold current was reduced by 75% for fast process conditions (i.e., low  $V_T$ ). Instead, these significant improvements suggest that



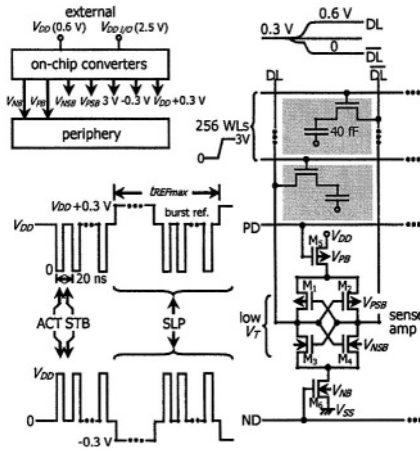


Fig. 4. Cell relevant circuits of 0.6-V 16-Mb e-DRAM [5].

operations would be very sensitive to substrate noises. However, a triple-well structure for isolating the floating array substrate from floating p/n peripheral substrates, coupled with half- $V_{DD}$  sensing, keeps these substrates quiet. A low  $V_T$  of 0.2 V is used for SA MOSTs (M1-M4) to enable high-speed half- $V_{DD}$  sensing, while a normal  $V_T$  of 0.3 V is used for SA-driver MOSTs (M5, M6) to reduce the dc current in standby mode. The substrate biases of SA and driver MOSTs are independently controlled because of different  $V_T$  implants. In the sleep mode, the gate-source offset driving of the NMOSTs and PMOST (above  $V_{DD}$  and below  $V_{SS}$  by 0.3 V) completely cuts the subthreshold currents. The RAM data is retained by periodically existing sleep mode and performing burst refresh cycles at 20 ns, as shown in the figure. The refresh scheme minimizes pump currents of the converters for -0.3 V and  $V_{DD} + 0.3$  V, enabling a simple design for the converters. Figure 5 shows sense-amp driver, and row-decoder block and word-driver block. The gate-source offset driving and various switched-source impedances (SSIs) [1, 2, 15-17] were used throughout the driver and blocks to reduce subthreshold currents. Mode transitions between active(ACT), standby(STB), and sleep(SLP) modes are fast because loads that on-chip voltage converters must drive are kept small. However, possible problems [1] are power-on surge currents and CMOS latch-up despite their occurring less often at a low  $V_{DD}$ . Another possibility is that substrate-voltage degradations might occur when pumps in the converters cannot sink increased substrate current at an extremely high-speed operation or a high  $V_{DD}$  applied during a burn-in stress test [1].

A 1.3- $\mu\text{m}$  16-Mb e-DRAM operating at 312 MHz uses a stacked capacitor [6]. Lowering the  $V_{DD}$  from 1.2 V to 1 V in the standby mode despite reduced  $Q_s$ , coupled with turning off the  $V_{DD}$  of several blocks that are not operated, enabled a retention power as small as 77  $\mu\text{W}$  at a refresh cycle of 4  $\mu\text{s}$ . Half- $V_{DD}$  sensing with a raised ground level was adapted successfully. A long recovery time of 0.4  $\mu\text{s}$  to the normal mode was due to the power control, although it is much shorter than the aforementioned refresh cycle. In addition to the 1-T cell, a 3-T DRAM cell [2] is a promising candidate for low-voltage RAM cells, as suggested by a CAM cell with an additional stacked capacitor ( $C_s = 30$  fF) [11].

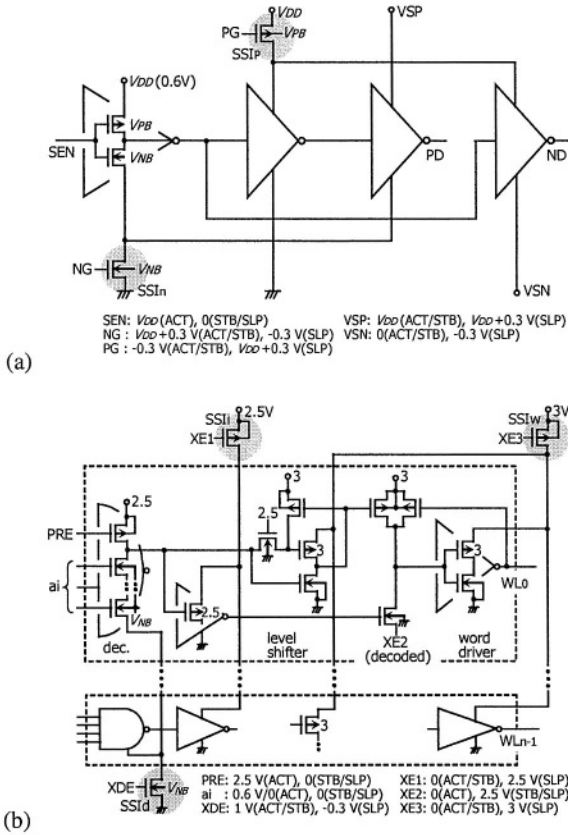


Fig. 5. Sense-amp driver (a), and row-decoder block and word-driver block(b)[5].

## 4 State-of-the-Art E-SRAMs

Two circuit schemes [4, 10] have been used to reduce the leakage: raising the cell's NMOST source in the standby modes despite reduced  $Q$ , and maintaining the power supply of the cell high so that the  $V_T$  and  $t_{ox}$  can be preserved high and thick [2], as discussed later.

Figure 6 illustrates a 0.13- $\mu\text{m}$  300-MHz 1-Mb e-SRAM module operating at 1.2 V [3, 4]. The leakage is reduced with three schemes: confining the active area to the small [1, 2, 17] (#1 in the figure) by dividing a module into 4 banks (BK0-BK3) to reduce the leakage to one-fourth by selecting only one bank with turning on M3 and M5; reducing the leakage in non-selected banks with switched-source impedance (SSI) [1, 2](SSIa and SSIw); and cutting the leakage of the peripheral circuits in the sleep mode with a power switch[2](#5). In non-selected banks in the active or standby modes of the module, an impedance (M1) in SSIa automatically creates a small raised voltage ( $\delta$ ) at the common source of cells as a result of the subthreshold-current flows from many cells. Thus, the resultant substrate-source backbiasing (#3, B2 in Fig. 3) to

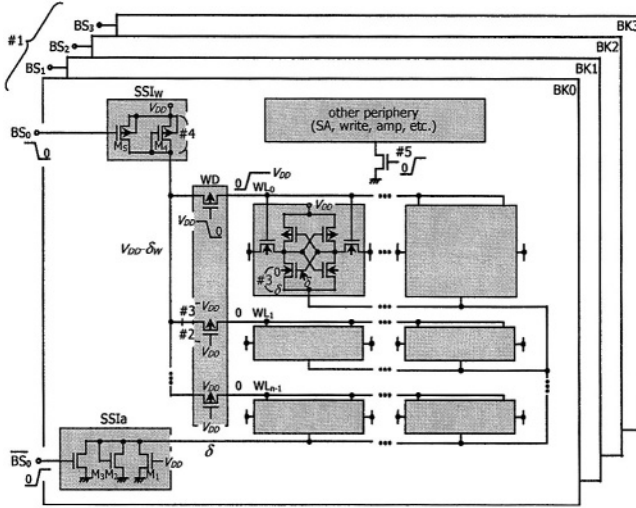
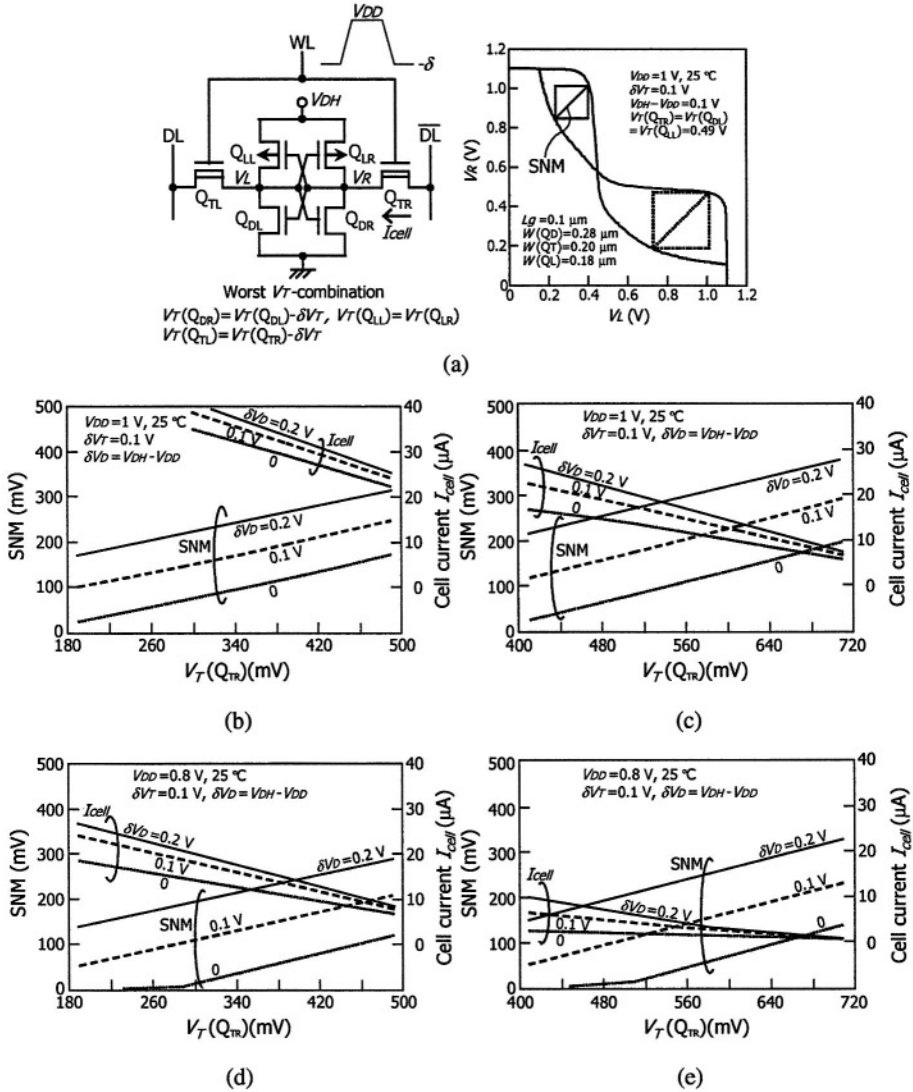


Fig. 6. 0.13- $\mu\text{m}$  1.2-V 1-Mb e-SRAM module [4].

the cross-coupled off-NMOST in each cell raises the  $V_T$  enabling the leakage to be reduced. The necessary  $\delta$ , however, is large (e.g., 0.3 V) due to the bad reduction efficiency of the scheme, and the common source is heavily capacitive due to many cells being connected. Hence, high-speed operation is not possible. Here, the diode (M2) clamps the source voltage, so  $Q_s$  is not reduced by an excessive  $\delta$ . SSIw also reduces the leakage of each off-PMOS in the word driver block three ways [2]: gate-source self-backbiasing(#2), substrate-source backbiasing (#3), and a drain-induced barrier lowering (DIBL) effect (#4). The gate-source backbiasing, however, has the best reduction efficiency [1, 2], necessitating a small  $\delta_w$  for a large leakage reduction. In addition, the power-line capacitance of the word drive block can be light. Thus, leakage in the block is greatly and quickly reduced. Results indicated that leakage currents were reduced by 25%, 67%, and 95% for the high-speed active mode with only SSIw turned on, for the slow-speed active mode with both SSIa and SSIw turned on, and for the sleep mode, respectively. Even so, however, a 1.2-V  $V_{DD}$  is still high, and a reduced  $Q_s$  in the standby mode would pose a problem.

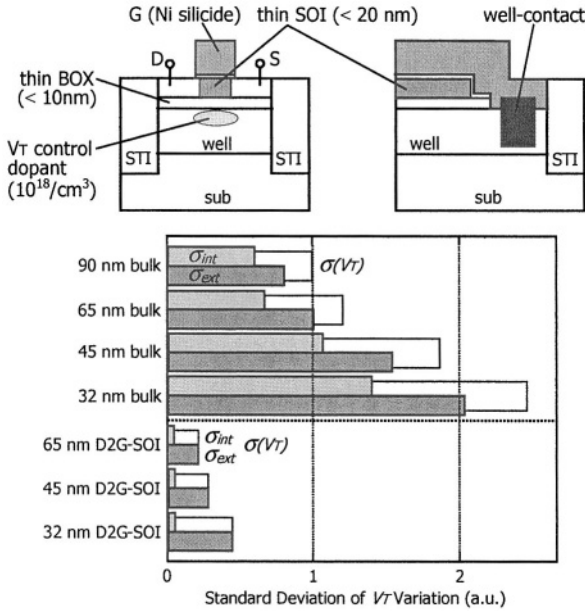
Figure 7 shows the raised power-supply ( $V_{DH}$ ) scheme [2]. High- $V_T$  cross-coupled MOSTs are used to reduce subthreshold currents. Low- $V_T$  transfer-MOSTs, coupled with the NWL scheme (i.e., gate-source offset driving) to cut leakage during non-selected periods, increase the cell read current. The resultant degraded static noise margin (SNM) is compensated for by increased conductance of cross-coupled MOSTs by the raised supply ( $V_{DH}$ ) that is generated by a charge pump or a voltage-down converter of the I/O power supply. Furthermore, the  $V_{DH}$  maintains the  $Q_s$  and drivability of cross-coupled MOSTs despite a high  $V_T$  and large  $\alpha(V_T)$  and  $\delta V_T$ . Obviously, the cell current and SNM for the worst  $V_T$  combination are greatly increased for both a high-speed design of  $V_T = 0.49$  V and a low-power design of  $V_T = 0.71$  V even with a small boost of 0.1 V. Thus, the  $V_{DH}$  scheme is likely to make a 1-V operation possible even in the usual design conditions of a 100-mV SNM and 20- $\mu\text{A}$  cell current, while the conventional  $V_{DD}$  scheme (i.e., no raised power supply and a fixed high- $V_T$  transfer MOSTs) makes a 1-V operation at  $V_T = 0.71$  V impossible. Note that a 0.8-V opera-



**Fig. 7.** Raised power supply ( $V_{DH}$ ) scheme applied to 6-T SRAM cell [2]: (a) for cell circuit, the worst  $V_T$  combination for SNM and  $I_{cell}$ , and a butterfly curve in the worst condition, (b) and (c) for SNM and  $I_{cell}$  versus  $V_T(Q_{TR})$  for a high-speed design with  $V_T(Q_{DL}) = V_T(Q_{LL}) = 0.49$  V and for a low-power design with  $V_T(Q_{DL}) = V_T(Q_{LL}) = 0.71$  V. Data are for  $V_{DD} = 1$  V. (d) and (e) for the same, but for  $V_{DD} = 0.8$  V. A  $0.13\text{-}\mu\text{m}$  device model was used.

tion is extremely difficult even for  $V_T = 0.49$  V. Despite keeping MOST sizes to the same while others are scaled down, the cell size could be reduced for each technology generation.

Figure 8 shows a new FD-SOI device, called a dynamic-double-gate SOI (D2G-SOI) [7]. It reduces the  $V_T$  variation thanks to reductions in short and narrow channel



**Fig. 8.** D2G-SOI device [7]: (a) for structure, (b) for predicted standard deviations of  $V_T$  variation (arb. unit,  $\sigma(V_T)$ ) for the total,  $\sigma_{int}$  for the intrinsic,  $\sigma_{ext}$  for the extrinsic).

effects and dopant atoms. Note that the variations in  $V_T$  mismatches ( $\delta V_T$ ) are also decreased due to  $\sigma(\delta V_T) = \sqrt{2}\sigma(V_T)$ . In addition, a double gate MOST structure increases the on-current while decreasing the off-current, if the gate and well are connected. Thus, if applied to the 6-T cell, the SNM should be greatly improved.

## 5 Future Prospects

Figure 9 compares existing RAM cells in terms of cell size, signal charge  $Q_s$ , and signal voltage. For the 6-T cell, the  $V_{DD}$  scaling down to sub-1-V is extremely difficult, as explained previously, due to the requirement of a high  $V_T$  for a small retention current and ever-larger variations of  $V_T$  and  $\delta V_T$  with device scaling. Moreover, decreased  $Q_s$  as  $V_{DD}$  is lowered unavoidably requires additional capacitance ( $C_s$ ) at the cell node to prevent soft errors from occurring. Thus, alternatives such as the raised power supply scheme and FD-SOI cells are required. Even so, the large cell size of the 6-T cell will eventually limit the use of the RAM cell to relatively small-memory-capacity applications. Although the 1-T cell is smallest, it requires a large  $C_s$  for enough signal voltage, while the 3-T and 6-T cells need only a small  $C_s$  against soft errors. This calls for quite complicated processes for stacked or trench capacitors. Thus, much simpler capacitors are preferable. Structures that were intensively studied when the transition of planar capacitors to three-dimensional capacitors took place in the early 1980s might be candidates, if a small data-line capacitance array is com-

Circuit	1-T DRAM	3-T DRAM	SRAM	
Cell Size*	15 F <sup>2</sup>	45 F <sup>2</sup>	125-160 F <sup>2</sup>	
Q <sub>s</sub>	V <sub>W</sub> = V <sub>DD</sub>	C <sub>s</sub> (V <sub>DD</sub> - V <sub>T</sub> )/2	C <sub>s</sub> [V <sub>DD</sub> - V <sub>T</sub> (M <sub>1</sub> ) - V <sub>T</sub> (M <sub>2</sub> )]	
	V <sub>W</sub> > V <sub>DD</sub> + V <sub>T</sub> (M <sub>1</sub> )	C <sub>s</sub> V <sub>DD</sub> /2	C <sub>s</sub> [V <sub>DD</sub> - V <sub>T</sub> (M <sub>2</sub> )]	
	V <sub>T</sub> (M <sub>1</sub> )	1.3 V (low cost) 0.7 V (high speed)	1.3 V (low cost) 0.7 V (high speed)	0.7 V (low power) 0.5 V (high speed)
	α(ΔV <sub>T</sub> ) bulk/FD-SOI	—	—	large/small
C <sub>s</sub> (required)	large	small	small	
V <sub>sig</sub> **	C <sub>s</sub> V <sub>DD</sub> /2(C <sub>s</sub> + C <sub>g</sub> ) - ΔV <sub>T5</sub>	(50-100) mV - ΔV <sub>T5</sub>	(50-100) mV - ΔV <sub>T5</sub>	

\* : logic compatible process  
 \*\* ΔV<sub>T5</sub> : sense-amp offset

Fig. 9. Comparisons between 1-T and 3-T DRAM cells and 6-T SRAM cell.

bined to relax the requirement for the necessary C<sub>s</sub>. Gain cells [2] such as the 3-T cell widely used in the early 1970s might be used again because of the advantages they provide: low-voltage operation capability, relatively small size, and simple structures compatible with a logic process. This could happen if high-speed sensing is developed. Note that the word bootstrapping and NWL, which have been commonly used for modern DRAMs, will also improve the design flexibility of the 3-T and 6-T cells, as discussed earlier.

In the long run, high-speed, high-density non-volatile RAMs [2] show strong potential for use as low-voltage RAMs. In particular, they have leakage-free and soft-error-free structures, and the non-destructive read-out and non-charge-based operations that they could provide are attractive in terms of achieving fast cycle times, low power with zero standby power, and stable operation, even at a lower V<sub>DD</sub>. Simple planar structures, if possible, would cut costs. In this sense, magnetic RAMs (MRAMs) and Phase-Transition RAM (PRAM) are appealing propositions, and a 4-Mb MRAM [12] and a 64-Mb PRAM [13] have been presented. In MRAMs, however, major drawbacks remain. The magnetic field needed to switch the magnetization of the storage element as well as the variations must be reduced. Additionally, wearing out of the retention characteristics needs to be resolved in PRAMs. Furthermore, larger cell sizes and scalabilities and the stability required to ensure non-volatility are still in the early stages of development.

A stable and reliable sensing will be extremely important at a lower V<sub>DD</sub> for peripheral circuits. Precise controls for internal voltages and low-power design for on-chip voltage converters and noise suppression-especially at substrates-will also be keys to solving subthreshold-current and speed-variation issues and stable operations. If FD-SOI devices such as D2G-SOI resolve V<sub>T</sub>-variation issue even with increased cost, these devices will simplify peripheral circuit designs that make coping with the issue increasingly complicated. In addition, new gate insulators free from gate-tunneling currents must be developed soon.

## 6 Conclusion

This paper presented challenges and trends in low-voltage RAMs in terms of signal charge, read signal voltage, and leakage in RAM cells and subthreshold current and speed variations in peripheral circuits. State-of-the-art e-DRAMs and e-SRAMs were then discussed. Based on these considerations, future prospects were considered with emphasis on the need for gain DRAM cells, new SRAM cells such as a boosted power-supply cell and fully-depleted SOI cell, on-chip low-power voltage converters, and precise controls for internal voltages with converters for stable and reliable operations in both RAM cells and peripheral circuits.

## References

- [1] Kiyoo Itoh, *VLSI Memory Chip Design*, Springer-Verlag, New York, 2001.
- [2] Y. Nakagome et al., "Review and prospects of low-voltage RAM circuits," *IBM J. R & D*, vol. 47, no. 5/6, pp. 525-552, Sep./Nov. 2003.
- [3] T. Kamei et al., "A Resume-Standby Application Processor for 3G Cellular Phones," 2004 ISSCC Dig. Tech. Papers, pp. 336-337, Feb. 2004.
- [4] M. Yamaoka et al., "A 300 MHz **25 $\mu$ A/Mb** Leakage On-Chip SRAM Module Featuring Process-Variation Immunity and Low-Leakage-Active Mode for Mobile-Phone Application Processor," 2004 ISSCC Dig. Tech. Papers, pp. 494-495, Feb. 2004.
- [5] K. Hardee et al., "A 0.6V 205MHz 19.5ns tRC 16Mb Embedded DRAM," 2004 ISSCC Dig. Tech. Papers, pp. 494-495, Feb. 2004.
- [6] F. Morishita et al., "A 312MHz 16Mb Random-Cycle Embedded DRAM Macro with **73  $\mu$ W** Power-Down Mode for Mobile Applications," 2004 ISSCC Dig. Tech. Papers, pp. 202-203, Feb. 2004.
- [7] M. Yamaoka et al., "Low Power SRAM Menu for SOC Application Using Yin-Yang-Feedback Memory Cell Technology," *Symp. VLSI Circuits Dig.*, June 2004.
- [8] K. Nii et al., "A 90 nm Low Power 32K-Byte Embedded SRAM with Gate Leakage Suppression Circuit for Mobile Applications," *Symp. VLSI Circuits Dig.*, pp. 247-150, June 2003.
- [9] T. Inukai and T. Hiramoto, "Suppression of Stand-by Tunnel Current in Ultra-Thin Gate Oxide MOSFETs by Dual Oxide Thickness MTCMOS(DOT-MTCMOS)," *Ext. Abst. 1999 Int'l Conf. SSDM*, Tokyo, pp. 264-265, 1999.
- [10] K. Osada et al., "16.7fA/cell Tunnel-Leakage-Suppressed 16Mb SRAM for Handling Cosmic-Ray-Induced Multi-Errors," 2003 ISSCC Dig. Tech. Papers, pp. 302-303, Feb. 2003.
- [11] H. Noda et al., "A 143MHz 1.1W 4.5Mb Dynamic TCAM with Hierarchical Searching and Shift Redundancy Architecture," 2004 ISSCC Dig. Tech. Papers, pp. 208-209, Feb. 2004.
- [12] J. Nahas et al., "A 4Mb **0.18 $\mu$ m** 1T1MTJ Toggle MRAM Memory," 2004 ISSCC Dig. Tech. Papers, pp. 44-45, Feb. 2004.
- [13] W. Y. Cho et al., "A **0.18 $\mu$ m** 3.0V 64Mb Non-Volatile Phase-Transition Random-Access Memory(PRAM)," 2004 ISSCC Dig. Tech. Papers, pp. 40-41, Feb. 2004.
- [14] Y. Nakagome et al., "Symp. VLSI Circuits Dig., pp. 82-83, June 1992.
- [15] T. Kawahara et al., "Subthreshold current reduction for decoded-driver by self-reverse biasing," *IEEE J. Solid-State Circuits*, vol. 28, pp. 1136-1144, Nov. 1993.
- [16] M. Horiguchi et al., "Switched-source-impedance CMOS circuit for low standby subthreshold current giga-scale LSI's," *IEEE J. Solid-State Circuits*, vol. 28, pp. 1131-1135, Nov. 1993.

- [17] T. Sakata et al., “Subthreshold-current reduction circuits for multi-gigabit DRAM’s,” IEEE J. Solid-State Circuits, vol. 29, pp. 761-769, July 1994.
- [18] K. Osada et al., “Universal-Vdd 0.65-2.0V 32 kB Cache using Voltage-Adapted Timing-Generation Scheme and a Lithographical-Symmetric Cell,” 2001 ISSCC, Dig. Tech. Papers, pp. 168-169, Feb. 2001.
- [19] K. Itoh et al., “Review and Future Prospects of Low-voltage Embedded RAMs,” CICC2004 Dig. October 2004.



# Adaptive Subthreshold Leakage Reduction Through N/P Wells Reverse Biasing

Carlo Dallavalle

Telecom Group Process Technology Director  
ST Microelectronics,  
Agrate Brianza, Italy

**Abstract.** In the deep submicron era, leakage power has become a first rank component, not only in the battery powered devices but in high performance broadband and networking processors. SoC complexity implies hundred millions transistors then large die size which combined with 30cm diameter wafers lead to large process variations, therefore to highly spread leakage power. A method aimed to reduce this leakage power variation through the combined use of dedicated cell libraries, monitor structures and programmable charge pump reverse bias substrate generators is proposed.

# Leakage in CMOS Circuits – An Introduction

D. Helms<sup>1</sup>, E. Schmidt<sup>2</sup>, and W. Nebel<sup>1</sup>

<sup>1</sup> OFFIS Research Institute

domenik.helms@offis.de,

<http://www.lowpower.de>

<sup>2</sup> ChipVision Design Systems AG

D-26121 Oldenburg, Germany

**Abstract.** In this tutorial, we give an introduction to the increasingly important effect of leakage in recent and upcoming technologies. The sources of leakage such as subthreshold leakage, gate leakage, pn-junction leakage and further GIDL, hot-carrier effect and punchthrough are identified and analyzed separately and also under PTV variations. Since leakage will dominate power consumption in future technologies, we also review leakage optimization techniques and leakage estimation approaches supporting optimizations especially at higher abstraction levels.

## 1 Introduction

The three sources of power dissipation in digital CMOS circuits can be summarized in the following equation:

$$P_{avg} = P_{short} + P_{switch} + P_{static} = I_{sc}V_{dd} + \alpha C_L V_{dd}^2 f + I_{leak}V_{dd} \quad (1)$$

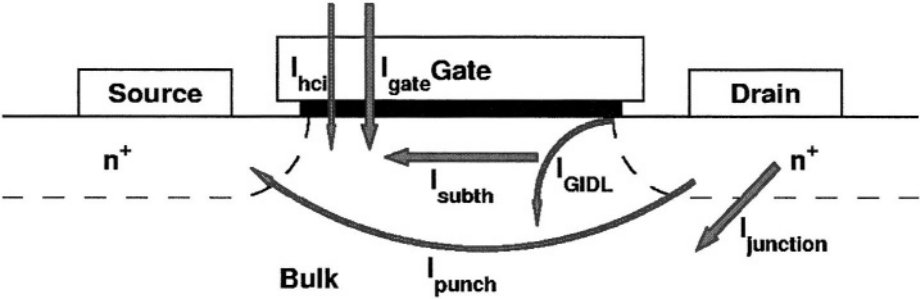
Between 1970 and 1980, the power consumption of a high-end processor rose from 0.2W to 2W. Power was mainly consumed in the NMOS technology of those days, since half of all transistors short-circuited their output to pull it down to logic 0. In the early 1980's, the first power-aware technology, the CMOS technology was introduced. It was able to reduce the power requirements to levels that much less complex systems some 10 years ago consumed. Static power  $P_{short}$  as the major contributor to the systems power consumption seemed to have been removed for good.

A small remaining  $P_{short}$  in CMOS logic is due to the direct path short circuit current  $I_{sc}$ , which occurs when both the NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground.

The second term of Equation 1,  $P_{switch}$ , refers to the dynamic component of power, where  $C_L$  is the load capacitance,  $f$  the clock frequency and  $\alpha$  the node transition activity factor.  $P_{switch}$ , which is dominating power consumption in today's circuits has been reduced by various techniques like voltage and threshold voltage scaling since then.

It is exactly these techniques which lead to another change in power consumption dominance. Prognoses [1] assume that in the middle of this decade static effects will again become the most dominant source of power consumption.

As analyzed in [2,3,4], leakage currents can be divided into 6 major contributors which will be reviewed in Section 2. In Section 3, we discuss the impact of PTV<sup>1</sup> variations on leakage currents. Section 4 presents existing work on high-level leakage modeling under PTV variations and Section 5 introduces leakage optimization techniques. Finally, Section 6 concludes this work.



**Fig. 1.** The 6 most prominent leakage mechanisms inside a MOS transistor

## 2 Leakage Sources in MOS Transistors

A MOS transistor can be represented as an electrical device with 4 different potentials: source  $\varphi_S$ , drain  $\varphi_D$ , gate  $\varphi_G$  and bulk  $\varphi_B$  of which  $\varphi_B$  can be assumed to be 0 and the others can be described as the potential difference to bulk (e.g.  $V_D = \varphi_D - \varphi_B = \varphi_D$ ).

In a classical view of a semi-conductor, we assume the connection between each pair of these potentials to have a nearly infinite resistance except for the drain-source connection for which the resistance of course can be switched by the gate voltage between high and low impedance.

**Table 1.** Occurrence of the leakage mechanisms for different n-channel transistor states. Due to low drain-source resistance for  $V_G = 1$ ,  $V_D$  and  $V_S$  can just differ transiently

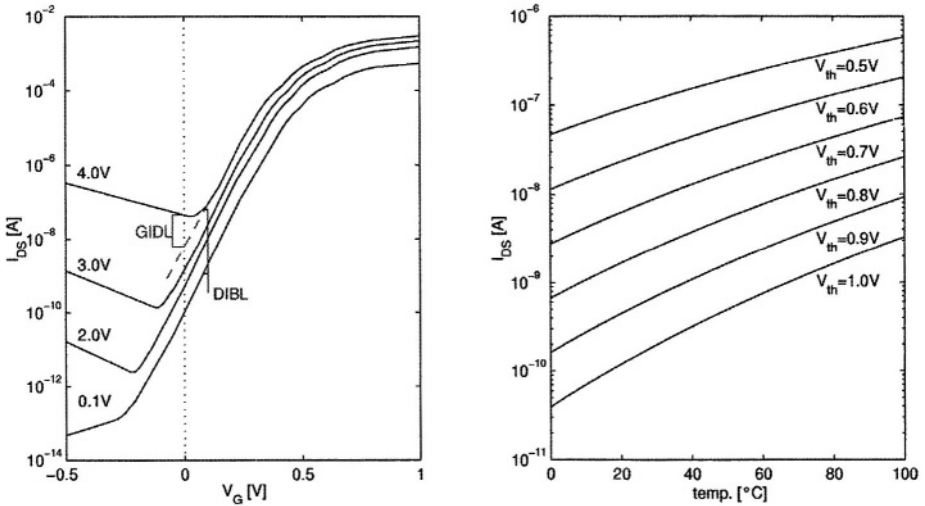
$V_G$	$V_D$	$V_S$	Sub-th	Gate	pn-jun.	GIDL	punch	HCI
0	0	0	-	-	-	-	-	-
0	0	1	✓	-	✓	✓	✓	✓
0	1	0	✓	-	✓	-	✓	✓
0	1	1	-	-	✓	✓	-	-
1	0	0	-	✓	-	-	-	-
1	1	1	-	✓	✓	-	-	-

<sup>1</sup> Process, thermal and voltage variations

As the feature size and all related technology parameters continue to be scaled down aggressively, the resistances become lifted to dimensions in which they markedly influence as well the transistor's functional behavior as the power behavior. Leakage currents in MOS transistors can be divided into 6 different mechanisms, occurring in different states as shown in Table 1. In the following, we will review each of these mechanisms.

## 2.1 Subthreshold Current

In Figure 2 (left) the drain to source current, which is shown mainly depends on the gate voltage. The curves can be separated at the threshold voltage into a linear rising and a saturating, almost constant region. Even if the gate voltage is low - far below this threshold voltage - the current through the channel is not exactly 0. This is because a transistor is not a simple switch but in contrast to a high-level view it is a complex analog component. A potential difference between source and drain will thus result in a current through the channel which is called subthreshold current. This is the best-known kind of leakage and it also has the highest impact on the overall static power consumption at least for recent technologies [5].



**Fig. 2.** Left: Drain to source current versus gate voltage for different drain voltages taken from [4]. The drain voltage's effect on drain-source current can be explained by second order effects like DIBL, reducing the threshold voltage and GIDL, decreasing the junction's barrier. Right: Plot of Equation 3: subthreshold current versus temperature for different threshold voltages.

In the weak inversion region, the subthreshold current's dependence on the most important technology parameters is given is [6,7] as

$$I_{SUB} = KV_T^2 (W/L) e^{(V_{gs}-V_{th})/(nV_T)} \left(1 - e^{-V_{ds}/V_T}\right), \quad (2)$$

where

$$n = 1 + \frac{C_{dm}}{C_{ox}} \quad \text{and} \quad K = \mu_0 \sqrt{\frac{q\epsilon_{si}N_{ch}}{2\Phi_s}} \quad (3)$$

are technology parameters,  $W$  and  $L$  are the channel's width and length, and  $V_{ds}$  is the drain-source,  $V_{gs}$  is the gate source voltage.  $V_T = k_B T/q$  is the thermal voltage, which is a direct proportional measure to the absolute temperature, as  $T$  is the temperature,  $k_B$  is the Stefan-Boltzmann-constant and  $q$  is the elementary charge.  $\mu_0$  is the zero bias mobility,  $N_{ch}$  is the effective channel doping,  $\Phi_s$  is the potential of the surface and  $C_{ox}$  and  $C_{dm}$  are the capacitances of the oxide layer and the depletion layer. From Equation 3 we can immediately derive the subthreshold current's dependencies to different technological and dynamical parameters. As can be seen in Figure 2 (right) where Equation 3 is evaluated for different temperatures and threshold voltages, the subthreshold current varies by several orders of magnitude as threshold voltage and temperature just differ by a few ten percents. For upcoming technologies Figure 2 determines the following: as the threshold voltage continues to be scaled down, the temperature's influence is decreased. In order to analyze this behavior in detail we evaluate the deviations  $\partial/\partial V_{th} \ln(I_{sub})$  and  $\partial/\partial T \ln(I_{sub})$ , assuming independence between  $V_{th}$  and  $V_T$  and a high drain-source voltage  $V_{ds}$ .

$$\left(\frac{\partial \ln(I_{sub})}{\partial V_{th}}\right)^{-1} = -\frac{nk_B T}{q} \quad (4)$$

$$\left(\frac{\partial \ln(I_{sub})}{\partial T}\right)^{-1} = \frac{nk_B T^2}{2} \left(\frac{1}{q(V_{th} - V_{gs}) + nk_B T}\right) \quad (5)$$

Using Equations 4 and 5 and assuming  $C_{dm} = 2C_{ox} \Rightarrow n = 3$ , the importance of temperature and threshold voltage can be evaluated as presented in Table 2.

Since subthreshold leakage is the most important leakage mechanism in recent technologies, it is worthwhile analyzing effects that influence subthreshold

**Table 2.** Dependence of the subthreshold current to threshold and temperature variations in Millivolt and Kelvin per decade change. As can be seen, the temperature dependence is reduced the more the threshold voltage is scaled down: for a  $V_{th} = 1V$  technology, subthreshold rose by a factor of 10 every 20.2K, for a  $V_{th} = 0.5V$  technology, the temperature has to rise by 35.5K to have the same effect.

$T[K]$	$V_{th}[V]$	$\Delta V[mV/dec.]$	$\Delta T[K/dec.]$
300	1.0	-77.6	20.2
300	0.5	-77.6	35.5
400	1.0	-103.5	34.3
400	0.5	-103.5	58.6

leakage. Besides the temperature, dynamically influencing the subthreshold current, there are two more dynamical effects, which are the body effect and the DIBL, increasing leakage by decreasing the threshold voltage which is given as [8,9]

$$\begin{aligned}
 V_{th} = & V_{fb} + \Phi_S + q \frac{\pi N_{sub} X_d^2}{2 C_{ox} W} \\
 & + \gamma \sqrt{\Phi_S - V_s} \left( 1 - \frac{\lambda X_d}{L_{eff}} \right) \\
 & + V_{DS} \left( e^{-L/2l_c} + 2e^{-L/l_c} \right).
 \end{aligned}$$

**Narrow Width Effect and  $V_{th}$  Roll-Off.** In this simplified equation (ref. [10] for more details), the first line describes the static component of threshold voltage, with  $N_{sub}$  being the substrate doping and  $X_d$  being the width of the depletion zone,  $V_{fb}$  is the flat-band voltage, the voltage at which there is no electrical charge in the semiconductor.  $X_d$  is the thickness of the depletion layer,  $\gamma$  is the body factor,  $\Phi_S$  is the surface potential and  $\lambda \approx 1$  is a fitting parameter. In addition,  $V_{th}$  can be reduced by  $V_{th}$  roll-off due to an effective length reduction of the channel. In short channel devices this simple relation is changed by a high influence of the source and drain depletion regions to the channel's potential profile [6].

**Body Effect.** The body effect is described by the second line of the threshold equation. Without going into detail [4], it can be linearized to a source-bulk voltage dependent threshold voltage shift

$$\Delta_{Body}(V_{th}) = \eta' V_s. \quad (6)$$

We can see that via the body effect, the source-bulk voltage can decrease the subthreshold current by increasing the subthreshold voltage. As will be described in Section 3.5, this is the main reason for data dependent leakage in CMOS circuits, called stacking effect.

**Drain Induced Barrier Lowering DIBL.** The last line shows a drain-source voltage dependent shift  $\Delta_{DIBL}(V_{th})$  where

$$l_c = \sqrt{\frac{\epsilon_{Si} T_{ox} X_d}{\epsilon_{ox} \eta}} \quad (7)$$

is the characteristic length of a MOS transistor. As can be seen in Figure 2 (left), the DIBL biases especially the transistor's off-current.

**Table 3.** Prediction of the development of the subthreshold current  $I_{sub}$ , the BTBT-current  $I_{pn-jun}$  and the gate tunneling current  $I_{gate}$  as an approximation derived from [1, 10]

Generation	Year	$I_{sub}$	$I_{pn-jun}$	$I_{gate}$
90nm	2004	840pA	25pA	13pA
50nm	2010	21nA	3.0nA	52nA
25nm	2016	260nA	120nA	510nA

## 2.2 Gate Leakage

Subthreshold leakage is the major source of static current in recent technologies. But today, static currents are just a small problem at all<sup>2</sup>, since they are limited to about 10% of the overall power consumption for high performance applications and to below 20% for mobile applications.

For upcoming technologies the exponential rise of leakage will lead to a leakage domination in terms of power. The break-even point of dynamic vs. static power for logic gates will be reached with the 65nm technology.

Gate leakage is almost negligible today, as subthreshold is approximately 60 times as high (ref. Table 3). But as subthreshold will rise by a factor of 25 from the 90nm to the 50nm technology, gate leakage will rise by the factor of 4000 at the same time - thus overtaking subthreshold in terms of importance. According to [11], gate leakage alone will have a contribution of 15% to the total power consumption in a 2004 technology generation device<sup>3</sup>. Thus it is worthwhile to regard other sources of leakage as well - starting with gate leakage in this section and reviewing pn-junction leakage in the next one.

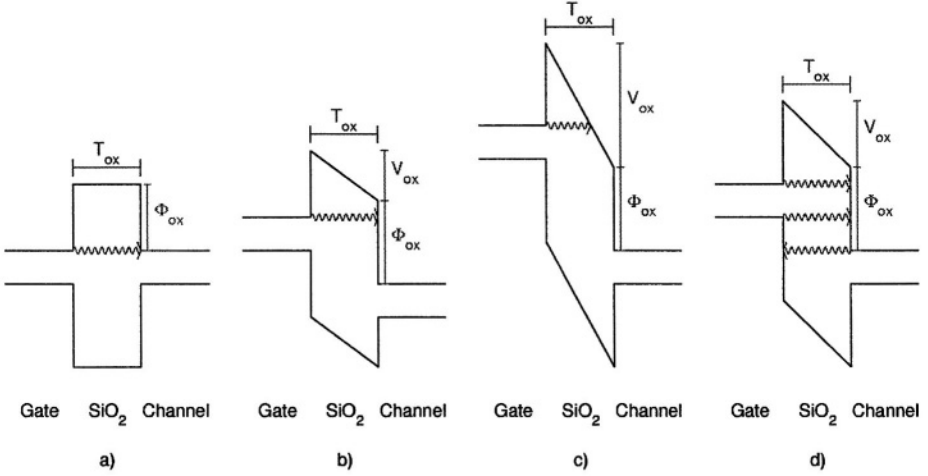
Gate leakage mainly consists of one major effect, the gate tunneling, and one minor effect: hot carrier injection.

**Gate Tunneling.** Gate tunneling is the effect of carriers having a finite probability tunneling in or through the SiO<sub>2</sub> layer, isolating the gate from the channel. When the gate voltage is high, the carriers inside the gate will *see* an energy band potential as presented in part a) of Figure 3 which is easy to handle quantum mechanically. Due to the gate bias, the conduction band of the oxide layer is bent to a trapezoidal shape<sup>4</sup> as presented in b) of Figure 3. Depending on the barrier height  $\Phi_{ox}$  of the oxide layer and the gate bias  $V_{ox}$ , only one of two possible gate tunneling mechanisms will occur:

<sup>2</sup> Except for ultra-low power devices like hearing aids and devices with a very low average switching activity like memories

<sup>3</sup> The dominance of the gate leakage may be avoided due to the introduction of high-k materials (ref. Section5.1).

<sup>4</sup> Of course also the energy bands of the gate and the channel are bent, but this can be ignored for the sake of simplicity



**Fig. 3.** Gate tunneling in a MOS transistor [4]. a) Tunneling at a rectangular obstacle. b) Introducing gate bias results in a trapezoidal reshape of the insulators energy bands. c) Fowler-Nordheim tunneling into the insulators conduction band. d) Direct tunneling through the insulator: conduction band to conduction band electron tunneling, conduction band to valence band electron tunneling and conduction band to conduction band hole tunneling.

**Fowler-Nordheim Tunneling.** In the case of  $V_{ox} > \Phi_{ox}$  the carriers need not have to tunnel through the entire oxide layer as shown in part c) of Figure 3. Instead, the carriers just have to pass a triangular energy potential of the reduced width  $T_{FN} < T_{ox}$ . Fowler-Nordheim tunneling carriers pass directly from the semiconductors conduction band to the insulators conduction band. Knowing  $\Phi_{ox}$  and the field across the insulator  $E_{ox}$ , the total FN-tunneling current  $I_{gate-fn}$  can be calculated as [6]

$$I_{gate-fn} = \alpha e^{\beta} \quad \text{with} \quad \alpha = \frac{L_g W_g q^3 E_{ox}^2}{16\pi^2 \hbar \Phi_{ox}}, \quad \beta = -\frac{4\sqrt{2m^*} \Phi_{ox}^{3/2}}{3\hbar q E_{ox}}, \quad (8)$$

where  $L_g$  and  $W_g$  are the effective gate's length and width subjected to FN-tunneling,  $q$  is the carriers charge,  $\hbar$  is Planck's constant over  $2\pi$  and  $m^*$  is the effective mass of a carrier in the conduction band of silicon. But as for  $\text{SiO}_2$   $\Phi_{ox} = 3.1\text{eV}$ , FN-tunneling is nearly negligible as usually  $V_{ox} < \Phi_{ox}$ .

**Direct Tunneling.** In most cases, gate leakage current is carried by electrons or holes, directly tunneling through the complete oxide layer - thus passing a trapezoidal obstacle. With today's insulator thickness of 20 – 30Å and a projected thickness of 15 – 20Å in the next five years [1], the tunneling probability will lead to dominant static currents<sup>5</sup>.

<sup>5</sup> For 2006, the ITRS [1] determines gate leakage to be too high for today's oxo-nitride insulators



Direct tunneling occurs in 3 possible ways as shown in part d) of Figure 3: Electrons may tunnel from the Conduction or the Valence band. Additionally, holes can tunnel through the channel in inverse direction. As for all these effects the energy barrier has the same shape, only the effective height of the obstacle and the effective mass of the carrier differ, so that all 3 effects can be calculated by

$$I_{gate-dt} = \alpha \exp\left(\beta - \beta [1 - V_{ox}/\Phi_{ox}]^{3/2}\right), \quad (9)$$

using the same  $\alpha$  and  $\beta$  as introduced in Equation 8.

In [12], the gate leakage Equations 8 and 9 are approximated and empirically reduced to the parameters  $W$ ,  $T_{ox}$  and  $V_{DD}$  as [2]

$$I_{gate} = K \cdot W \left(\frac{V_{DD}}{T_{ox}}\right)^2 e^{-\alpha T_{ox}/V_{DD}} \quad (10)$$

where  $K$  and  $\alpha$  can be derived experimentally or numerically.

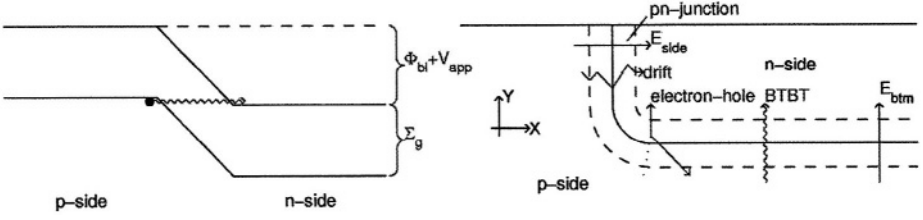
**Hot Carrier Injection HCI.** In the pinch-off region of the channel, large voltage drops result in high electric fields. Mobile carriers in high electric fields may lead to leakage losses by a variety of mechanisms [13] as shown in Table 4. These high-energy carriers are called hot-carriers. Carriers with energies above the ionization threshold of  $1.3eV$  are able to strike out electrons from their orbit, thus producing electron hole pairs with arbitrary directions, which is called impact ionization. The energy of some of the carriers moving towards the gate-insulation is high enough to pass the insulator barrier. Since the barrier height of  $\text{SiO}_2$  for electrons ( $\approx 3.1eV$ ) is lower than that for holes ( $\approx 4.8eV$ ), hot-electron injection is more likely to occur than hot-hole injection. In addition, as the mobility of holes inside the  $\text{SiO}_2$  barrier is lower than that of the electrons, more holes get trapped in the insulator before reaching the gate.

**Table 4.** Leakage mechanisms due to high field electrons

<i>Carrier Energy</i>	<i>Mechanism</i>
$E > 1.12eV$	Light emission
$1.3eV < E < 1.8eV$	Impact ionization
$E > 3.1eV$	Hot-electron injection
$E > 4.8eV$	Hot-hole injection

### 2.3 pn-Junction Leakage

The behavior of a pn-junction in the reverse-biased situation is well known from the analysis of reverse biased diodes. The two classical but marginal mechanisms causing pn-junction current are the diffusion by drift of minority carriers [14] and electron-hole generation in the depletion region. For transistors, the gate also introduces gated diode device action in the drain-bulk junction under the gate.



**Fig. 4.** Left: Band-to-band tunneling in a reverse biased pn-junction. BTBT requires a junction bias  $\Phi_{bi} + V_{app}$  higher than the band gap  $\Sigma_g$ , thus the n-side's conduction band to be lower than the p-side's valence band. Right: Geometry of the pn-junction and the 3 main reverse bias pn-junction leakage mechanisms: a) minority carriers drifting through the junction b) electron-hole generation in high fields c) tunneling through the junction.

**Band-to-Band Tunneling.** With a rising doping concentration, a further effect dominating pn-junction leakage appears. It is caused by a direct tunneling of electrons from the p-side's valence band to the n-side's conduction band.

Due to energy conservation reasons, electrons can never tunnel *upwards* in an energy-band view but just *downwards*, getting rid of the extra energy by emission of a phonon. Thus, band-to-band tunneling can only occur if the voltage drop across the junction, consisting of the reverse bias  $V_{app}$  and the built-in voltage  $\Phi_{bi}$ , is higher than the silicon band gap  $\Sigma_g$ , as shown in Figure 4 (left).

The band-to band tunneling current approximately results [8] as

$$I_{BTBT-side} = W_{eff} Y_{j-eff} \frac{A E_{side} V_{app}}{\Sigma_g^{1/2}} \exp\left(-\frac{B \Sigma_g^{3/2}}{E_{side}}\right) \quad \text{and} \quad (11)$$

$$I_{BTBT-btm} = W_{eff} X_{j-eff} \frac{A E_{btm} V_{app}}{\Sigma_g^{1/2}} \exp\left(-\frac{B \Sigma_g^{3/2}}{E_{btm}}\right) \quad \text{where} \quad (12)$$

$$A = \frac{\sqrt{2m^*} q^3}{4\pi^3 \hbar^2} \quad \text{and} \quad B = \frac{4\sqrt{2m^*}}{3q\hbar} \quad (13)$$

are constants. In order to handle the complex geometry of this problem, average effective electric fields for the junction below the source or drain  $E_{btm}$  and for the junction between source and drain  $E_{side}$  as effective height  $Y_{j-eff}$  and length  $X_{j-eff}$  of the junction was introduced (refer [8] for the details).

## 2.4 Gate Induced Drain Leakage GIDL

Gate induced drain leakage GIDL describes the influence of the gate-field on the depletion layer of the drain. The higher the gate voltage, the thinner the depletion layer becomes in the gate-drain overlap area. According to [15], GIDL currents are carried by an increased band-to-band tunneling probability which is given by

$$I_{GIDL} = A \cdot E_s(V_{gs}) \exp\left(-\frac{B}{E_s(V_{gs})}\right) \quad (14)$$

$$\text{where } A = \frac{2\pi q^2 m_e^{1/2}}{9\hbar^2 \Sigma_G^{3/2}} \text{ and } B = \frac{\pi m_e^{1/2} \Sigma_g}{\sqrt{8q\hbar}}, \quad (15)$$

where  $E_s$  is the field of the surface, which is dependent on the gate-source voltage, approximately given by

$$E_s \approx \frac{V_g - 1.2V}{3T_{ox}}. \quad (16)$$

## 2.5 Punchthrough

A transistor's source and drain depletion region are of course no rectangular regions; instead, the junction lines follow a complex field-dependent profile. With shrinking device size the effective field strength is rising because voltage is scaled much slower than geometric properties.

Thus, for high voltages in short channel devices, the depletion regions may be enlarged in such a way, that they electrically touch deep in the channel, which leads to a punchthrough current flow.

## 3 Impacts on Leakage

In the last section, we reviewed what leakage is from a transistor level point of view. We have seen that all these leakage sources depend on parameters - static ones like gate length  $L$  as well as dynamic ones like temperature. From the system level point of view, we can identify several processes having a high impact on leakage currents. In this section, we will review the most prominent ones which are process-temperature-voltage variations (PTV variations) and data dependency, starting with leakage dependence on process variations. Then we will discuss variations of the two most important dynamical parameters, temperature and power supply. In the end we will discuss aspects of data-dependent leakage.

### 3.1 Inter-die Variations

Inter-die variations, also called *die-to-die variations*, are the collection of effects on leakage due to variation of circuit parameters that vary from die to die. Variations in the coating, doping and etching processes will lead to slight variations in structure sizes and doping profiles. In [16], the leakage power due to these variations was reported to vary by a factor of 20 between the best and the worst die on a wafer.

As inter-die variations describe the variation of the total leakage power of different dies, they cannot be used to forecast a single die's leakage power. But statistical approaches [17], predicting average leakage current and variation, at least allow a prediction of the yield-loss due to too high leakage power.

### 3.2 Intra-die Variations

Intra-die variations, also called *within-die-variations*, are co-generated between design and process, depending on the details of the design [18]. They can affect leakage by causing threshold voltage changes and effective length variations spread over the die. They also must be handled in a statistical way as only average value and variance of the process parameters are known. But in contrast to inter-die variations, intra-die effects affect the power distribution of the transistors of a single system.

The intra-die variation's influence e.g. on the subthreshold leakage (ref. Equation 3) can be modeled by a geometric approach [19] derived from the real physical level. Considering intra-die variations of the channel length results in a bimodal distribution as

$$I_{leak} = \frac{I_p^0 w_p}{k_p} e^{\frac{\sigma_p^2}{2\lambda_p^2}} + \frac{I_n^0 w_n}{k_n} e^{\frac{\sigma_n^2}{2\lambda_n^2}}, \quad (17)$$

where  $w_p$  and  $w_n$  are the total device width in the chip;  $k_p$  and  $k_n$  are factors to determine the percentage of PMOS and NMOS device widths that are in *OFF* state;  $I_p^0$  and  $I_n^0$  are the expected mean subthreshold leakage currents per unit width of PMOS and NMOS devices;  $\sigma_p$  and  $\sigma_n$  are the standard deviations of the channel length variation within a particular chip;  $\lambda_p$  and  $\lambda_n$  are constants that relate channel length of PMOS and NMOS devices to their corresponding subthreshold leakages.

The average leakage current sums up to the respective multiple of the leakage current's mean value. Using Equation 17, the expected overall deviation of the summed up leakage resulting from an intra-die varying channel length of  $\sigma_p$  and  $\sigma_n$  can then be estimated as

$$\sigma = \lambda \sqrt{2 \ln \left( \frac{k I_{leak}}{w I^0} \right)}. \quad (18)$$

The same approach can be used (starting with another equation) for other sources of leakage, always leading to a prediction of the influence of parameter variations to the power consumption distribution.

### 3.3 Thermal Variations

Leakage current, especially subthreshold current is very sensitive to the temperature. This temperature at a specific gate and time depends on the power behavior over time of all other surrounding gates  $P(\vec{x}, t)$ . In addition global parameters like ambient temperature and state of the cooling device should be regarded. Thus, for a precise thermal leakage analysis, a complete system (including package and cooling device) must be regarded as a complex network of heat sources and sinks. Due to the high number of inhomogenities, the temperature distribution inside the system and even the two dimensional one of the die's surface is a complex function, even for a simple steady state solution evaluation.

Due to the exponential leakage-temperature relation accurate thermal models are essential: as can be seen from Table 2, for the chosen parameters  $T = 300K$  and  $V_{th} = 1.0$ , a temperature difference of  $1K$  is sufficient to increase the leakage by 12%. Useful approaches trying to handle thermal variation are presented in Section 4.3.

### 3.4 Power Supply Variations

Varying supply voltage caused by supply voltage drops (IR-drops) can significantly affect the leakage on an IC. For  $V_{DD}$  noise calculation, the package can be regarded as an effective coil of inductance  $L$ . The grid can be modeled by a capacitive and resistive system with  $R_g$  being the grid's resistance and  $C_d$  and  $R_d$  describing the grid's de-cap in parallel to the load [20].

The load, i.e. the complete number of gates, can be described by a varying resistance  $R_L(t)$ . With the mean supply voltage  $V_{DD}$  given, the current deviation  $\mu$  can be calculated as

$$\mu = \frac{V_{DD}}{dR/dt} \quad (19)$$

and the  $V_{DD}$  noise thus results as [18]

$$dV_{DD}(t)/dt = \mu t R_g + \mu L - \mu R_d^2 C_d (1 - e^{t/\tau}) \quad (20)$$

where  $\tau$  is the inverse characteristic frequency of the system.

### 3.5 Data Dependency

In the first place, the leakage of a transistor is just determined by its state - *ON* or *OFF*. Of course, in CMOS designs the number of transistors being *ON* is exactly known, because for every transistor in the *ON* state, there is a transistor in the *OFF* state in the other half of the CMOS logic. But of course there are effects that introduce a data dependency of the leakage current.

**Stacking Effect.** A single inverter's leakage is low in terms of data-leakage dependency with a difference of 2 to 3 between *ON* and *OFF* state. But as soon as more than one transistor is used in the complementary networks, leakage shows huge data dependency as presented in Table 5 from [21].

As can be easily seen, the gate's overall leakage current can vary by factor 50 between different states. This can be explained as follows [22]: If a transistor which is switched *OFF* is placed between the supply and a transistor of which leakage current has to be estimated, the first transistor will cause a slight reverse bias between the gate and the source of the second transistor. In this situation, the body effect will increase the threshold voltage and the reduced drain-source voltage will reduce DIBL and thus effectively further increase threshold voltage (ref. Equation 6). Because the subthreshold current (the most dominant leakage current in actual designs) is exponentially dependent on the threshold voltage, a substantial leakage reduction is obtained.

**Table 5.** Example of the data dependency of the leakage currents of the NAND and the NOR gate.

Inputs		Leakage power [pA]	
A	B	NAND	NOR
0	0	10	308
0	1	173	540
1	0	304	168
1	1	544	112

**Table 6.** Example of the minimal and maximal leakage current of various circuits dependent on the input vector.

Circuit	# cells	$I_{min}$ [ $\mu A$ ]	$I_{max}$ [ $\mu A$ ]	$\sigma$ [%]
c432	187	59.7	73.9	2.4
c499	222	154	215	3.4
c880	383	95.8	132	3.4
c1355	566	128	173	3.0
c1908	996	211	313	6.1
c2670	1255	325	427	3.6
c5311	2485	670	842	2.8
c7752	3692	665	714	4.8

But as reported in [23], the data dependency of bigger systems seems to even out this effect as presented in Table 6.

As can be seen, the relative standard deviation  $\sigma$  is below 6.1% for all analyzed circuits. Thus, for high-level leakage analysis and optimization, data-dependencies are not the most necessary topic. Nevertheless, there are approaches that try to handle the components leakage dependency (ref. Section 4.1).

## 4 High-Level Leakage Estimation Techniques

In Section 2 low level analysis of leakage was presented as a set of equations using in-detail technology information. Section 3 then explained why several of these parameters are not accurately available at low abstraction levels. Furthermore, as the leakage power problem emerges, leakage awareness will become a severe design constraint. Leakage optimization has to be addressed during the complete design flow and thus leakage estimation should be available at all levels of abstraction. As Section 2 also provided lowest level estimation equations, this section briefly reviews proposed higher level estimation techniques.

### 4.1 High-Level Data-Dependent Leakage Analysis

[22] tries to accurately model the exact influence of a transistor stack. This approach results in highly accurate predictions, but of course needs gate level

information which makes it prohibitive in terms of computational effort in the domain of high-level modeling.

The approach of [23] is a little more abstract. Here, statistical measures are propagated through the components. The outcome is a  $n^{\text{th}}$  order Taylor expansion of the leakage current depending on the single component's inputs. Correlation between the inputs comes with the higher correlation terms. This approach is called *leakage sensitivity*. It has an estimation error of below 4% relative standard deviation.

[24] is a highly sophisticated leakage estimation approach where a *feed forward neuronal network* is used to distinguish between states of related leakage behavior. Even though it is very easily evaluated, the estimation capabilities are good for single gates, but poor for complex circuits.

## 4.2 A Static Power Model for Architects

Butts et al. [25] have proposed a relatively simple empirical static power model for architects. The leakage power is calculated as

$$P_{leak} = V_{dd} N k_{design} I_{leak} \quad (21)$$

where  $N$  is the number of transistors of the design,  $k_{design}$  is a design dependent parameter and  $I_{leak}$  is a technology dependent parameter. This equation allows to separate the contributions in a high-level and a low-level dominated factor:  $I_{leak}$  depends on technology parameters like  $V_{th}$ , while  $k_{design}$  depends on design parameters like the fraction of transistors at logic one.

Separating the supply voltage  $V_{dd}$  and the number of transistors in Equation (21) is just a first order correction which is quite straight-forward. This does not mean that in reality the remainder,  $k_{design}$  and  $I_{leak}$ , is independent of supply and transistor count.

While the  $I_{leak}$  parameter is relatively well defined and easily tractable, the problems obviously arise when trying to predict the  $k_{design}$  parameter. The parameter can only be determined empirically and will then represent the characteristics of an average device and is meant to be approximately constant for similar designs.

## 4.3 A Temperature-Aware Static Power Model for Architects (HotLeakage)

The approach of Section 4.2 is simplistic and there are further approaches trying to increase its estimation capabilities, to cope with its flaws and to introduce further parameters like temperature. [26,27] introduces a modified leakage model, which has the same structure as Equation (21) but where the definition of the parameters is refined.

Like in Section 4.2,  $V_{dd}$  and  $N$  are the supply voltage and the number of transistors in the design. But in contrast to Section 4.2, the definition of  $k_{design}$  is separated for N and P transistors into  $k_n$  and  $k_p$ . For calculation of  $I_{leak}$ , the

authors tried to separate every physical effect in order to make the remainder as invariant as possible.  $I_{leak}$  is defined as

$$I_{leak} = \mu_0 \cdot C_{OX} \cdot \frac{W}{L} \cdot e^{b(V_{dd} - V_{dd0})} \cdot V_T^2 \cdot \left(1 - e^{-\frac{V_{dd}}{V_T}}\right) \cdot e^{-\frac{|V_{th}| - V_{off}}{n \cdot V_T}}, \quad (22)$$

which is the subthreshold leakage equation. The low level parameters can be derived using transistor level simulations.  $V_{dd}$  is the current supply voltage,  $V_{dd0}$  is the nominal default supply voltage which is technology dependent. The parameters  $\mu_0$ ,  $C_{OX}$ ,  $W/L$  and  $V_{dd0}$  are absolutely invariant with all parameters in this approach and only depend on the technology used. In contrast, the DIBL factor  $b$ , the subthreshold swing coefficient  $n$  and  $V_{off}$  are derived from a curve fitting on measurements based on transistor level simulations.  $V_{dd}$ ,  $V_{th}$  and  $V_t$  are calculated dynamically during the estimation [26].

## 5 Leakage Optimization Techniques

### 5.1 Transistor Engineering for Performance-Leakage Tradeoff

While scaling down devices, the parameters of a transistor have to obey the rules of constant field scaling in order to control the performance, the reliability and the power consumption of a device: starting at a gate length scaling factor of  $k$ ,  $T_{ox}$  and junction thickness have to be scaled the same way in order to keep short channel effects (SCE) under control. Down-scaling  $V_{DD}$  is then necessary to keep fields constant in order to prevent high field effects like hot-electron effects. Since  $V_{DD}$  reduction would immediately reduce performance,  $V_{th}$  has to be scaled as well too to compensate.

This way, nearly every degree of freedom of choice of parameters is fixed. The remainder of leakage optimization on transistor level is mainly due to modifying the doping profile and the dielectricity of the oxide [4].

**Well Engineering.** There are two doping techniques, mainly influencing leakage power. *Halo doping* means increasing the doping level of the p-substrate close to the inversion layer. It was introduced in order to limit the influence of the channel length on the threshold voltage. Thus, halo doping also limits drain induced barrier lowering by ‘guarding’ the ends of the channel, lowering drain’s and source’s influence.

*Retrograde doping* means changing the doping profile of the bulk in vertical direction: Directly under the oxide layer the doping concentration is lowered and deeper towards the bottom it is increased. Retrograde doping was introduced in order to shift the channel away from the impurities in the oxide interface thus reducing scattering at impurities. This technique increases threshold voltage and thus reduces subthreshold currents, but the high-p doped layer also acts as a punchthrough protection ‘guarding’ the bottom of the channel.



**Oxide Dielectricity.** In most cases, modifying the channel's doping profile reduces drain-source leakage effects such as subthreshold, punchthrough and DIBL. In order to minimize gate-bulk leakage, the dielectricity of the gate insulator is the key parameter. By introduction of other high- $k^6$  materials with  $k > 25$  factors, gate tunneling can be reduced by several orders of magnitude thus possibly pushing away the gate leakage problem for a very long time. The ITRS [1] projects introduction of high- $k$  techniques for the end of this decade.

## 5.2 Low Leakage with Conventional Technology Using Transistor Stacking Effect

As seen in Section 3.5, the subthreshold leakage of a gate differs markedly for different input combinations due to the stacking effect. Without the need for a special low leakage technology, this can be used in two different ways:

At first, idle components can be set to a low leakage state. Finding low leakage states can be done by testing all  $2^n$  states for an  $n$  input component or by applying heuristic approaches [21]. Using state assignment, 5-30% of leakage power can be saved during idle time, where the overhead for generating this state as for switching to this state is not taken into account.

Another way of using stack effect is called power gating and is done by inserting a sleep-transistor in series to the CMOS gate [28]. By doing power gating, 35-90% of the leakage can be saved in idle phases but at the cost of a 5-60% performance loss.

## 5.3 Usage of Special Low-Leakage Technologies

Assuming that all possible techniques presented in Section 5.1 are performed resulting in transistors with the lowest possible leakage under performance constraints, we have to increase the level of abstraction in order to further reduce leakage power.

There are several techniques reported which all work by identifying available slack in the gate net-list and then assigning lower performance components with better power behavior to the gates, having sufficient slack that the timing constraints of the circuit are not violated.

These approaches differ in the way the performance-power tradeoff are biased, the way they performed and the way the gates are replaced.

**Multiple Threshold.** The multiple threshold technology (MTCMOS) contains conventional transistors with low threshold and thus high performance and high leakage power and contains high threshold transistors where leakage currents are reduced for the cost of performance.

---

<sup>6</sup>  $k$  is the dielectricity constant of a material

The threshold can be increased by reducing the doping concentration of the channel, by enlarging the channel length  $L$ , by modifying the body bias<sup>7</sup> or by enlarging the oxide thickness. All methods reduce subthreshold leakage, but the oxide thickness enlargement has further advantages: As tunneling probability exponentially depends of the thickness of the barrier, also gate leakage can be reduced. In addition, the gate capacitance is reduced as the oxide thickness is enlarged. Thus this technique also reduces dynamic power consumption. Beneath all these advantages, the disadvantage of  $T_{ox}$  scaling is that short channel effects are increased and consequently the channel length has to be enlarged to compensate  $V_{th}$  rolloff.

**Multiple  $V_{dd}$ .**  $V_{dd}$  reduction was proposed quite early because it is also a very powerful dynamical power countermeasure. Since the total leakage power results as a product of supply voltage and leakage current  $P_{leak} = I_{leak}V_{dd}$ , leakage depends at least linearly on the supply voltage. But also potential dependent leakage effects like DIBL are reduced. [29] reports a reduction in the order of  $P_{subth} \propto V_{dd}^3$  and  $P_{gate} \propto V_{dd}^4$ .

Multiple voltage techniques have highest impact on the power consumption, but the overhead for supplying several voltages to the gates and the need of introducing level shifters make it more inconvenient to use.

## 5.4 Thermal Control

As we have seen in Section 5.1, by using high-k insulators gate leakage can be sufficiently reduced by a design modification that is very simple to understand, even though it is not that easy to really perform. Gate leakage is one of the two most important leakage sources - at least for the next 5 years. The other one, the subthreshold current, can also be reduced by orders of magnitude by a technique that again is simple to understand but maybe not that easy to perform:

Since small temperature differences lead to huge leakage variations, aggressive cooling techniques up to cryogenic cooling can be applied.

Cooling techniques can strongly control subthreshold current by reducing it by several orders of magnitude.

## 6 Conclusion

Leakage power will become a major issue for upcoming technologies. Leakage analysis will become more complex as different leakage sources like gate leakage and band-to-band tunneling gain influence on the total power. A complete framework for high-level leakage estimation may be available soon as this is one major recent research concern. High-k materials and aggressive cooling may be

<sup>7</sup> This can easily be done for silicon on insulator SOI techniques. For conventional techniques multiple well techniques have to be applied in order to allow varying bulk voltages

the solution to leakage problems for future technologies, but the recent and upcoming technologies have to be controlled by more conventional, less invasive optimizations. As on lowest levels a lot of optimization was proposed, higher level approaches addressing leakage are missing so far.

## References

1. *International Technology Roadmap for Semiconductors ITRS 2003*: public.itrs.net/
2. A Chandrakasan, W Bowhill, F Fox: Design of High-Performance Microprocessor Circuits. *IEEE Press*, 2001.
3. A Keshavarzi, K Roy, C F Hawkins: Intrinsic Leakage in Low Power Deep Submicron CMOS ICs. *Proc. Int'l Test Conf. 1997 (ITC'97)*, p. 146ff, 1997.
4. K Roy, S Mukhopadhyay, H Mahmoodi-Meimand: Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits. *Proc. of the IEEE Vol.91 No.2*, 2003.
5. J Srinivasan: An Overview of Static Power Dissipation. *Internal report*, rsim.cs.uiuc.edu/~srinivsn/Pubs/other/leakage.ps
6. Y Taur, T H Ning: Fundamentals of Modern VLSI Devices, *Cambridge University Press*, 1998.
7. D Fotty: MOSFET Modeling With SPICE, *Prentice Hall PTR*, 1997.
8. S Mukhopadhyay, A Raychowdhury, K Roy: Accurate Estimation of Total Leakage Current in Scaled CMOS Logic Circuits Based on Compact Current Modeling. *Proc. on Design Automatisation Conference DAC*, 2003.
9. Z H Liu et al.: Threshold Voltage Model for Deep-Submicrometer MOSFET's. *IEEE Trans. On Elec. Dev.*, 1993.
10. S Mukhopadhyay, K Roy: Modeling and Estimation of Total Leakage Current in Nano-scaled CMOS Devices Considering the Effect of Parameter Variation. *Int'l Symp. on Low Power Electronics and Design ISLPED*, 2003.
11. R M Rao, J L Burns, A Devgan, R B Brown: Efficient Techniques for Gate Leakage Estimation. *Int'l Symp. on Low Power Electronics and Design ISLPED*, 2003.
12. N S Kim et al., K Flautner, J S Hu et al.: Leakage Current: Moore's Law Meets Static Power. *IEEE Computer Vol.36 No. 12*, 2003.
13. M P Pagey: Characterization and Modeling of Hot-Carrier Degradation in Sub-Micron NMOSFETs. [etd.library.vanderbilt.edu/ETD-db/theses/available/etd-0619102-103401/unrestricted/thesis.pdf](http://etd.library.vanderbilt.edu/ETD-db/theses/available/etd-0619102-103401/unrestricted/thesis.pdf), 2002.
14. R Pierret: Semiconductor Device Fundamentals. *Reading, MA: Addison-Wesley*, 1996.
15. O Semenov, A Pradzynski, M Sachdev: Impact of Gate Induced Drain Leakage on Overall Leakage of Submicrometer CMOS VLSI Circuits. *IEEE Trans. on Semicond. Manufacturing Vol. 15, No.1*, 2002.
16. S Narendra: Leakage Issues in IC Design: Trends and challenges in scaling CMOS devices, circuits and systems. *Proc. of Int'l Conf. on Computer Aided Design ICCAD*, 2003.
17. R Rao, A Srivastava, D Blaauw, D Sylvester: Statistical Estimation of Leakage Current Considering Inter- and Intra-Die Process Variations. *Int'l Symp. on Low Power Electronics and Design ISLPED*, 2003.
18. A Devgan: Leakage Issues in IC Design: Process and Environmental variations. *Proc. of Int'l Conf. on Computer Aided Design ICCAD*, 2003.

19. S Narendra, V De, S Borkar, D Antoniadis, A Chandrakasan: Full-Chip Sub-threshold Leakage Power Prediction Model for  $0.18\mu\text{m}$  CMOS. *Int'l Symp. on Low Power Electronics and Design ISLPED*, 2002.
20. H Su, F Liu, A Devgan, E Acar, S Nassif: Full chip leakage estimation considering power supply and temperature variations. *Int'l Symp. on Low Power Electronics and Design ISLPED*, 2003.
21. F A Aloul, S Hassoun, K A Sakallah, D Blaauw: Robust SAT-Based Search Algorithm for Leakage Power Reduction. *Proc. on Int'l Workshop on Power and Timing Modeling, Optimization and Simulation PATMOS*, 2002.
22. Z Chen, M Johnson, L Wei, K Roy: Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks. *Int'l Symp. on Low Power Electronics and Design ISLPED*, 1998.
23. E Acar, A Devgan, R Rao, Y Liu, H Su, S Nassif, J Burns: Leakage and Leakage Sensitivity Computations for Combinational Circuits. *Int'l Symp. on Low Power Electronics and Design ISLPED*, 2003.
24. L Cao: Circuit Power Estimation Using Pattern Recognition Techniques. *Proc. of Int'l Conf. on Computer Aided Design ICCAD*, 2002.
25. J A Butts, G S Sohi: A static power model for architects. *Proc. Int'l Symp. on Microarchitecture*, 2000.
26. Y Zhang, D Parikh, M Stan, K Sankaranarayanan, K Skadron: HotLeakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects. *Tech Report CS-2003-05, Univ. of Virginia Dept. of Computer Science*, 2003.
27. HotLeakage estimator executable: [lava.cs.virginia.edu/HotLeakage](http://lava.cs.virginia.edu/HotLeakage)
28. M C Johnson, D Somasekhar, L Y Chiou, K Roy: Leakage Control With Efficient Use of Transistor Stacks in Single Threshold CMOS. *IEEE Trans. on VLSI Systems Vol.10 No.1* , 2002.
29. S Tyagi et al.: A 130 nm Generation Logic featuring 70 nm transistors, dual  $V_t$  transistors and 6 layers of Cu interconnects. *Dig. Tech. Papers Int. Electron Devices Meeting*, 2000.

# The Certainty of Uncertainty: Randomness in Nanometer Design

Hongliang Chang<sup>1</sup>, Haifeng Qian<sup>2</sup>, and Sachin S. Sapatnekar<sup>2</sup>

<sup>1</sup> CSE Department

<sup>2</sup> ECE Department

University of Minnesota, Minneapolis, MN 55455, USA

**Abstract.** Randomness and uncertainty are rearing their heads in surprising and contradictory ways in nanometer technologies. On the one hand, uncertainty and variability is becoming a dominant factor in the design of integrated circuits, and on the other hand, algorithms based on randomness are beginning to show great promise in solving large scale problems. This paper overviews both aspects of this issue.

## 1 Introduction

A historical look at integrated circuit technologies shows several inflection points that have characterized the 250nm, 180nm and 130nm nodes. The move to the sub-100nm regime is projected to bring about the most revolutionary of these changes, in terms of how it impacts the way in which design is carried out. Most notably, randomness will become a fact of life that designers will be forced to confront, and perhaps, paradoxically, the only certainty in nanometer designs will be the presence of uncertainty. Several issues related to uncertainty and randomness will be discussed in this paper.

We will begin, in Section 2, by exploring the origins of randomness in nanometer circuits, and will then discuss methods that must be used in next-generation designs to handle such variations in Section 3. This first aspect of randomness, caused by process and environmental variations, is “problematic” and requires new solutions to overcome its effects, since such variations manifest themselves as changes in the delay and power dissipation of a circuit. As a consequence, the analysis of timing will move from a purely deterministic setting to a statistical analysis, as will the analysis of leakage power, which is becoming a major component of the total power dissipation. This has already lead to intense efforts in statistical static timing analysis (SSTA) and statistical power analysis in recent years. Finding efficient solutions to these problems presents numerous new challenges, and while some first steps have been taken, many problems remain unsolved.

Amid all these problems also lies an opportunity: there is a second facet of randomness that is likely to have very positive consequences in the future, as discussed in Section 4. As the electronic design automation world becomes more educated in the use of stochastic techniques, new opportunities will arise on the

algorithmic side, as novel statistical approaches will be developed for solving design problems. This has already been set into motion: problems as diverse as capacitance extraction, power estimation, V<sub>dd</sub> net analysis, crosstalk analysis, placement, and ESD analysis are seeing viable stochastic solution techniques. An attractive feature of the random techniques is that when used in appropriate settings, they can scale extremely well with increasing problem sizes, and for several problems, they have the potential for localized computation. This paper will overview such algorithms and raise the challenge of harnessing the power of such methods for solving the problems of tomorrow.

## 2 Sources of Uncertainty

Current-day integrated circuits are afflicted with a wide variety of variations that affect their performance. Essentially, under true operating conditions, the parameters chosen by the circuit designer are perturbed from their nominal values due to various types of variations. As a consequence, a single SPICE-level transistor or interconnect model (or an abstraction thereof) is seldom an adequate predictor of the exact behavior of a circuit. These sources of variation can broadly be categorized into two classes

**Process variations** result from perturbations in the fabrication process, due to which the nominal values of parameters such as the effective channel length ( $L_{eff}$ ), the oxide thickness ( $t_{ox}$ ), the dopant concentration ( $N_a$ ), the transistor width ( $w$ ), the interlayer dielectric (ILD) thickness ( $t_{ILD}$ ), and the interconnect height and width ( $h_{int}$  and  $w_{int}$ , respectively).

**Environmental variations** arise due to changes in the operating environment of the circuit, such as the temperature or variations in the supply voltage ( $V_{dd}$  and ground) levels. There is a wide body of work on analysis techniques to determine environmental variations, both for thermal issues [8,7,20,10], and for supply net analysis [18].

Both of these types of variations can result in changes in the timing and power characteristics of a circuit.

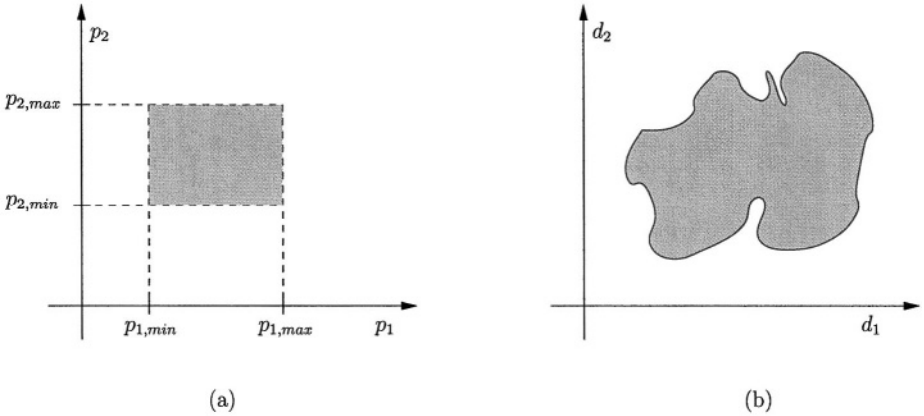
Process variations can also be classified into the following categories:

**Inter-die variations** are the variations from die to die, and affect all the devices on same chip in the same way, e.g., they may cause all of the transistor gate lengths of devices on the same chip to be larger or all of them to be smaller.

**Intra-die variations** correspond to variability within a single chip, and may affect different devices differently on the same chip, e.g., they may result in some devices having smaller oxide thicknesses than the nominal, while others may have larger oxide thicknesses.

Inter-die variations have been a longstanding design issue, and for several decades, designers have striven to make their circuits robust under the unpredictability of such variations. This has typically been achieved by simulating the

design at not just one design point, but at multiple “corners.” These corners are chosen to encapsulate the behavior of the circuit under worst-case variations, and have served designers well in the past. In nanometer technologies, designs are increasingly subjected to numerous sources of variation, and these variations are too complex to capture within a small set of process corners.



**Fig. 1.** The feasible region in (a) the performance parameter space and (b) the design/process parameter space.

To illustrate this, consider the design of a typical circuit. The specifications on the circuit are in the form of limits on performance parameters,  $p_i$ , such as the delay or the static or dynamic power dissipation, which are dependent on a set of design or process parameters,  $d_i$ , such as the transistor width or the oxide thickness. In Figure 1(a), we show the behavior of a representative circuit in the performance space of parameters,  $p_i$ , whose permissible range of variations lies within a range of  $[p_{i,min}, p_{i,max}]$  for each parameter,  $p_i$ , which corresponds to a rectangular region. However, in the original space of design parameters,  $d_i$ , this may translate into a much more complex geometry, as shown in Figure 1(b). This may conservatively be captured in the form of process corners at which the circuit is simulated.

In nanometer technologies, intra-die variations have become significant and can no longer be ignored. As a result, a process corner based methodology, which would simulate the entire chip at a small number of design corners, is no longer sustainable. A true picture of the variations would use one process corner in each region of the chip, but it is clear that the number of simulations would increase exponentially with the number of such regions. If a small number of process corners are to be chosen, they must be very conservative and pessimistic. For true accuracy, a larger number of process corners may be used, but this number may be too large to permit computational efficiency.

The sources of these variations may be used to create another taxonomy:

**Random variations** (as the name implies) depict random behavior that can be characterized in terms of a distribution. This distribution may either be

explicit, in terms of a large number of samples provided from fabrication line measurements, or implicit, in terms of a known probability density function (such as a Gaussian or a lognormal distribution) that has been fitted to the measurements. Random variations in some process or environmental parameters (such as those in the temperature, supply voltage, or  $L_{eff}$ ) can often show some degree of local *spatial correlation*, whereby variations in one transistor in a chip are remarkably similar in nature to those in spatially neighboring transistors, but may differ significantly from those that are far away. Other process parameters (such as  $t_{ox}$  and  $N_a$ ) do not show much spatial correlation at all, so that for all practical purposes, variations in neighboring transistors are uncorrelated.

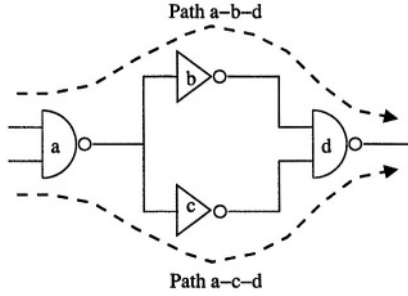
**Systematic variations** show predictable variational trends across a chip, and are caused by known physical phenomena during manufacturing. Strictly speaking, environmental changes are entirely predictable, but practically, due to the fact that these may change under a large number (potentially exponential in the number of inputs and internal states) of operating modes of a circuit, it is easier to capture them in terms of random variations. Examples of systematic variations include those due to (i) spatial intra-chip gate length variability, also known as across-chip linewidth variation (ACLV), which observes systematic changes in the value of  $L_{eff}$  across a reticle due to effects such as changes in the stepper-induced illumination and imaging nonuniformity due to lens aberrations [15], and (ii) ILD variations due to the effects of chemical-mechanical polishing (CMP) on metal density patterns: regions that have uniform metal densities tend to have more uniform ILD thicknesses than regions that have nonuniformities.

The existence of *correlations* between intra-die variations complicates the task of statistical analysis. These correlations are of two types:

**Spatial correlations.** To model the intra-die spatial correlations of parameters, the die region may be tessellated into  $n$  grids. Since devices or wires close to each other are more likely to have similar characteristics than those placed far away, it is reasonable to assume perfect correlations among the devices [wires] in the same grid, high correlations among those in close grids and low or zero correlations in far-away grids. Under this model, a parameter variation in a single grid at location  $(x, y)$  can be modeled using a single random variable  $p(x, y)$ . For each type of parameter,  $n$  random variables are needed, each representing the value of a parameter in one of the  $n$  grids.

**Structural correlations.** The structure of the circuit can also lead to correlations that must be incorporated in SSTA. Consider the reconvergent fanout structure shown in Figure 2. The circuit has two paths, a-b-d and a-c-d. If, for example, we assume that each gate delay is a Gaussian random variable, then the PDF of the delay of each path is easy to compute, since it is the sum of Gaussians, which admits a closed form. However, the circuit delay is the maximum of the delays of these two paths, and these are correlated since the delays of a and d contribute to both paths. It is important to take such structural correlations, which arise due to reconvergences in the circuit, into account while performing SSTA.





**Fig. 2.** An example to illustrate structural correlations in a circuit.

### 3 Analysis of Uncertainty

As an example, we will now illustrate the concepts involved in the statistical analysis of timing; similar techniques are being developed for power analysis.

The geometrical parameters associated with the gate and interconnect can reasonably be modeled as normally distributed random variables. Before we introduce how the distributions of gate and interconnect delays will be modeled, let us first consider an arbitrary function  $d = f(\mathbf{P})$  that is assumed to be a function on a set of parameters  $\mathbf{P}$ , where each  $p_i \in \mathbf{P}$  is a random variable with a normal distribution given by  $p_i \sim N(\mu_{p_i}, \sigma_{p_i})$ . We can approximate  $d$  linearly using a first order Taylor expansion:

$$d = d_0 + \sum_{\forall \text{ parameters } p_i} \left[ \frac{\partial f}{\partial p_i} \right]_0 \Delta p_i \quad (1)$$

where  $d_0$  is the nominal value of  $d$ , calculated at the nominal values of parameters in the set  $\mathbf{P}$ ,  $\left[ \frac{\partial f}{\partial p_i} \right]_0$  is computed at the nominal values of  $p_i$ ,  $\Delta p_i = p_i - \mu_{p_i}$  is a normally distributed random variable and  $\Delta p_i \sim N(0, \sigma_{p_i})$ . The delay function here is arbitrary, and may include, for example, the effects of the input transition time on the gate/wire delay.

If all of the parameter variations can be modeled by Gaussian distributions, this approximation implies that  $d$  is a linear combination of Gaussians, which is therefore Gaussian. Its mean  $\mu_d$ , and variance  $\sigma_d^2$  are:

$$\mu_d = d_0 \quad (2)$$

$$\sigma_d^2 = \sum_{\forall i} \left[ \frac{\partial f}{\partial p_i} \right]_0^2 \sigma_{p_i}^2 + 2 \sum_{\forall i \neq j} \left[ \frac{\partial f}{\partial p_i} \right]_0 \left[ \frac{\partial f}{\partial p_j} \right]_0 \text{cov}(p_i, p_j) \quad (3)$$

where  $\text{cov}(p_i, p_j)$  is the covariance of  $p_i$  and  $p_j$ .

This approximation is valid when  $\Delta p_i$  has relatively small variations, in which domain the first order Taylor expansion is adequate and the approximation is acceptable with little loss of accuracy. This is generally true of the impact of

intra-chip variations on delay, where the process parameter variations are relatively small in comparison with the nominal values, and the function changes by a small amount under this perturbation. For this reason, the gate and interconnect delays, as functions of the process parameters, can be approximated as a normal distributions when the parameter variations are assumed to be normal.

The existence of on-chip variations requires an extension of traditional STA techniques to move beyond their deterministic nature. The SSTA approach, which overcomes these problems, treats delays not as fixed numbers, but as probability density functions (PDF's), taking the statistical distribution of parametric variations into consideration while analyzing the circuit. The simplest way to achieve this, in terms of the complexity of implementation, may be through Monte Carlo analysis. While such an analysis can handle arbitrarily complex variations, its major disadvantage is in its extremely large run-times. Therefore, more efficient methods are called for.

The task of static timing analysis can be distilled into two types of operations:

- A gate is being processed in STA when the arrival times of all inputs are known, at which time the candidate delay values at the output are computed using the “sum” operation that adds the delay at each input with the input-to-output pin delay.
- Once these candidate delays have been found, the “max” operation is applied to determine the maximum arrival time at the output.

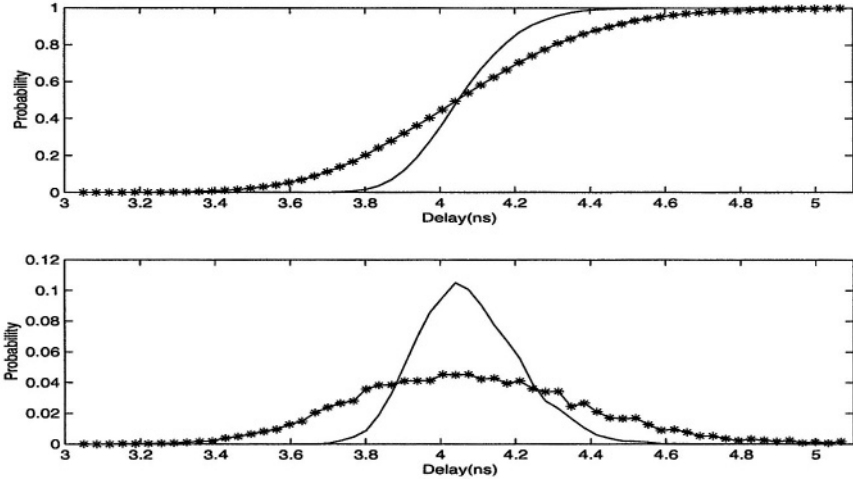
In SSTA, the operations are identical to STA; the difference is that the pin-to-pin delays and the arrival times are PDFs instead of single numbers.

The first method for statistical static timing analysis to successfully process large benchmarks under probabilistic delay models was proposed by Berkelaar in [4]. In the spirit of static timing analysis, this approach was purely topological, and ignored the Boolean structure of the circuit. It assumed that each gate in the circuit has a delay distribution that is described by a Gaussian PDF, and assumed that all process variations were uncorrelated.

The approach maintains an invariant that expresses all arrival times as Gaussians. As a consequence, since the gate delays are Gaussian, the “sum” operation is merely an addition of Gaussians, which is well known to be a Gaussian. The computation of the max function, however, poses greater problems. The set of candidate delays are all Gaussian, so that this function must find the maximum of Gaussians. In general, the maximum of two Gaussians is *not* a Gaussian. However, given the intuition that if  $a$  and  $b$  are Gaussian random variables, if  $a \gg b$ ,  $\max(a, b) = a$ , a Gaussian; if  $a = b$ ,  $\max(a, b) = a = b$ , a Gaussian, it may be reasonable to approximate this maximum using a Gaussian. In later work in [11], a precise closed-form approximation for the mean and variance was utilized.

Another class of methods includes the work in [3], which uses bounding techniques to arrive at the delay distribution of a circuit, but again, these ignore any spatial correlation effects, and it is important to take these into consideration.

Figure 3 shows a comparison of the PDF yielded by an SSTA technique that is unaware of spatial correlations, as compared with a Monte Carlo simulation that incorporates these spatial correlations. The clear difference between the



**Fig. 3.** A comparison of the results of SSTA when the random variables are spatially correlated. The line on which points are marked with stars represents the accurate results obtained by a lengthy Monte Carlo simulation, and the the solid curve shows the results when spatial correlations are entirely ignored. The upper plot shows the CDFs, and the lower plot, the PDFs [6].

curves demonstrates the need for developing methods that can handle these dependencies.

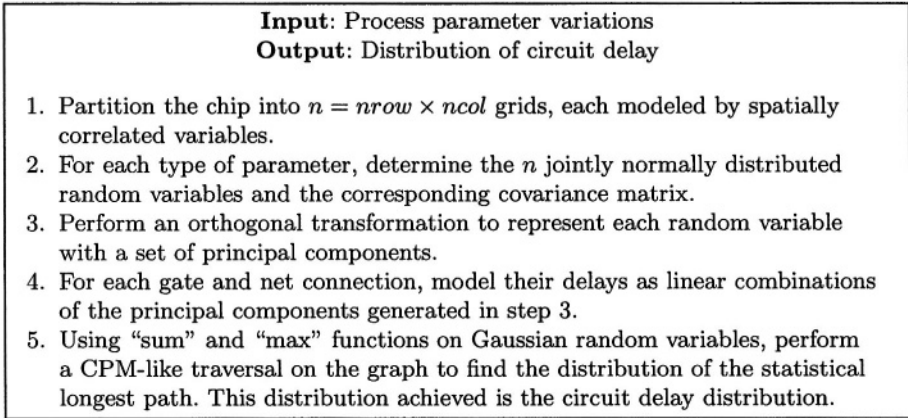
The approach in [6] presents a novel and simple method based on the application of principal component analysis (PCA) techniques [13] to convert a set of correlated random variables into a set of uncorrelated variables in a transformed space; the PCA step can be performed as a preprocessing step for a design. The overall idea is similar to that of Berkelaar's, but the use of PCA permits rapid and efficient processing of spatial correlations. In reality, some parameters may be spatially correlated and others (such as  $T_{ox}$  and  $N_d$ ) may be uncorrelated; this method is easily extended to handle these issues.

The overall flow of the algorithm is shown in Figure 4. The complexity of the method is  $p \cdot n$  times the complexity of CPM, where  $n$  is the number of squares in the grid and  $p$  is the number of correlated parameters, plus the complexity of finding the principal components, which requires very low runtimes in practice. The overall CPU times for this method have been shown to be low, and the method yields high accuracy results.

## 4 Uncertainty as a Virtue

### 4.1 Introduction

The concept of uncertainty can also be harnessed to advantage in providing efficient solutions to many difficult problems. Examples of such problems are as follows:



**Fig. 4.** Overall flow of the PCA-based statistical timing analysis method.

**Randomized algorithms** have been proposed in [14] for the solution of many combinatorial problems, including problems such as partitioning that arise in CAD. However, these have not been significantly developed in EDA.

**Monte Carlo methods** have been used very successfully to compute the average power dissipation of a circuit by applying a small fraction of the exponentially large space of possible input vectors to a circuit [5]. Such methods have also been employed for SSTA, as described earlier.

**Random walk methods** have been used to analyze large systems with localized behavior, such as in capacitance extraction [9], power grid analysis [16], and we are currently investigating their application to the analysis of electrostatic discharge (ESD) networks and to the problem of placement in physical design.

**Other miscellaneous applications** of random methods include techniques for crosstalk analysis [19] and in the probabilistic analysis of routing congestion [12,21].

All of these point to the fact that the use of statistical methods in design is a vibrant and growing field with many upcoming challenges, particularly as, when used in the right contexts (e.g., when the computation is localized), these methods can scale extremely well. We will illustrate one such method in the following section.

## 4.2 Case Study: Power Grid Analysis Using Random Walks

On-chip power grids play an important role in determining circuit performance, and it is critical to analyze them accurately and efficiently to check for signal integrity, increasingly so in nanometer technologies.

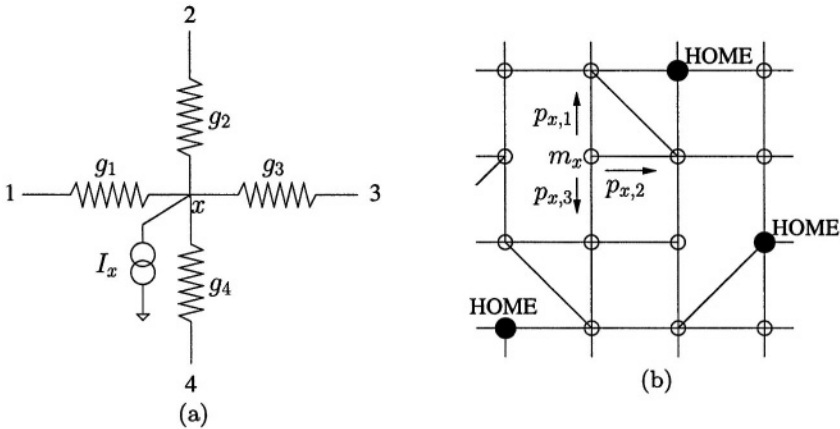
A typical power grid consists of wire resistances, wire inductances, wire capacitances, decoupling capacitors, VDD pads, and current sources that correspond

to the currents drawn by logic gates or functional blocks. There are two sub-problems to power grid analysis: *DC analysis* to find steady-state node voltages, and *transient analysis* which is concerned with finding voltage waveforms considering the effects of capacitors, inductors and time-varying current waveform patterns.

The DC analysis of a power grid is formulated as a problem of solving a system of linear equations:

$$G\mathbf{X} = \mathbf{E} \tag{4}$$

where  $G$  is the conductance matrix for the interconnected resistors,  $\mathbf{X}$  is the vector of node voltages, and  $\mathbf{E}$  is a vector of independent sources. Traditional approaches exploit the sparse and positive definite nature of  $G$  to solve this system of linear equations for  $\mathbf{X}$ . However, the cost of doing so can become prohibitive for a modern-day power grid with hundreds of millions of nodes, and this will only become worse as the circuit size is ever growing from one technology generation to the next.



**Fig. 5.** (a) A representative power grid node. (b) An instance of a random walk “game.”

For the DC analysis of a VDD grid, let us look at a single node  $x$  in the circuit, as illustrated in Figure 5(a). Applying Kirchoff’s Current Law, Kirchoff’s Voltage Law and the device equations for the conductances, we can write down the following equation:

$$\sum_{i=1}^{\text{degree}(x)} g_i(V_i - V_x) = I_x \tag{5}$$

where the nodes adjacent to  $x$  are labeled  $1, 2, \dots, \text{degree}(x)$ ,  $V_x$  is the voltage at node  $x$ ,  $V_i$  is the voltage at node  $i$ ,  $g_i$  is the conductance between node  $i$  and node  $x$ , and  $I_x$  is the current load connected to node  $x$ . Equation (5) can be reformulated as follows:

$$V_x = \sum_{i=1}^{\text{degree}(x)} \frac{g_i}{\sum_{j=1}^{\text{degree}(x)} g_j} V_i - \frac{I_x}{\sum_{j=1}^{\text{degree}(x)} g_j} \quad (6)$$

We can see that this implies that the voltage at any node is a linear function of the voltages at its neighbors. We also observe that the sum of the linear coefficients associated with the  $V_i$ 's is 1. For a power grid problem with  $N$  non-VDD nodes, we have  $N$  linear equations similar to the one above, one for each node. Solving this set of equations will give us the exact solution.

We will equivalence this problem to a random walk "game," for a given finite undirected connected graph (for example, Figure 5(b)) representing a street map. A walker starts from one of the nodes, and goes to an adjacent node  $i$  every day with probability  $p_{x,i}$  for  $i = 1, 2, \dots, \text{degree}(x)$ , where  $x$  is the current node, and  $\text{degree}(x)$  is the number of edges connected to node  $x$ . These probabilities satisfy the following relationship:

$$\sum_{i=1}^{\text{degree}(x)} p_{x,i} = 1 \quad (7)$$

The walker pays an amount  $m_x$  to a motel for lodging everyday, until he/she reaches one of the homes, which are a subset of the nodes. If the walker reaches home, he/she will stay there and be awarded a certain amount of money,  $m_0$ . We will consider the problem of calculating the expected amount of money that the walker has accumulated at the end of the walk, as a function of the starting node, assuming he/she starts with nothing.

The gain function for the walk is therefore defined as

$$f(x) = E[\text{total money earned} \mid \text{walk starts at node } x] \quad (8)$$

It is obvious that

$$f(\text{one of the homes}) = m_0 \quad (9)$$

For a non-home node  $x$ , assuming that the nodes adjacent to  $x$  are labeled  $1, 2, \dots, \text{degree}(x)$ , the  $f$  variables satisfy

$$f(x) = \sum_{i=1}^{\text{degree}(x)} p_{x,i} f(i) - m_x \quad (10)$$

For a random-walk problem with  $N$  non-home nodes, there are  $N$  linear equations similar to the one above, and the solution to this set of equations will give the exact values of  $f$  at all nodes.

It is easy to draw a parallel between this problem and that of power grid analysis. Equation (10) becomes identical to (6), and equation (9) reduces to the condition of perfect VDD nodes if

$$\begin{aligned} p_{x,i} &= \frac{g_i}{\sum_{j=1}^{\text{degree}(x)} g_j} & i &= 1, 2, \dots, \text{degree}(x) \\ m_x &= \frac{I_x}{\sum_{j=1}^{\text{degree}(x)} g_j} & m_0 &= V_{DD} & f(x) &= V_x \end{aligned} \quad (11)$$

A natural way to approach the random walk problem is to perform a certain number of experiments and use the average money left in those experiments as the approximated solution. If this amount is averaged over a sufficiently large number of walks by playing the “game” a sufficiently large number of times, then by the law of large numbers, an acceptably accurate solution can be obtained.

This is the idea behind the proposed *generic algorithm* that forms the most basic implementation. Numerous efficiency-enhancing techniques are employed in the implementation, and these have been described in [16,17]. The DC solution above has also been extended to solve the transient analysis problem, which can be handled similarly, and with greater efficiency.

## 5 Conclusion

The effects of variability and uncertainty are here to stay in nanometer VLSI designs, and CAD techniques must be found to overcome them. This paper has outlined the basics of how a CAD engineer will have to deal with randomness in the future: not only in terms of dealing with it during design, but also in the sense of exploiting it by using algorithms that exploit randomness. This paper only skims the very surface of this issue, and there is likely to be considerable work in this field in the future.

## References

1. A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula. Computation and refinement of statistical bounds on circuit delay. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 348–353, June 2003.
2. A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula. Statistical timing analysis using bounds. In *Proceedings of Design and Test in Europe*, pages 62–67, February 2003.
3. A. Agarwal, V. Zolotov, and D. T. Blaauw. Statistical timing analysis using bounds and selective enumeration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(9):1243–1260, September 2003.
4. M. Berkelaar. Statistical delay calculation, a linear time method. In *Proceedings of ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, pages 15–24, December 1997.
5. R. Burch, F. Najm, P. Yang, and T. N. Trick. A Monte Carlo approach for power estimation. *IEEE Transactions on VLSI Systems*, 1(1):63–71, March 1993.
6. H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 621–625, November 2003.
7. G. Chen and S. S. Sapatnekar. Partition-driven standard cell thermal placement. In *Proceedings of the ACM International Symposium on Physical Design*, pages 75–80, 2003.
8. Y. Cheng and S. M. Kang. A temperature-aware simulation environment for reliable ULSI chip design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(10):1211–1220, October 2000.

9. Y. L. Le Coz and R. B. Iverson. A stochastic algorithm for high-speed capacitance extraction in integrated circuits. *Solid-State Electronics*, 35:1005–1012, 1992.
10. B. Goplen and S. S. Sapatnekar. Efficient thermal placement of standard cells in 3D IC's using a force directed approach. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 86–89, 2003.
11. E. Jacobs and M. R. C. M. Berkelaar. Gate sizing using a statistical delay model. In *Proceedings of Design and Test in Europe*, pages 283–290, 2000.
12. J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng. Estimating routing congestion using probabilistic analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(1):32–41, January 2002.
13. D. F. Morrison. *Multivariate Statistical Methods*. McGraw-Hill, New York, NY, 1976.
14. R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, Cambridge, UK, 1995.
15. M. Orshansky, L. Milor, P. Chen, K. Keutzer, and C. Hu. Impact of spatial in-trachip gate length variability on the performance of high-speed digital circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(5):544–553, May 2002.
16. H. Qian, S. R. Nassif, and S. S. Sapatnekar. Random walks in a supply network. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 93–98, 2003.
17. H. Qian and S. S. Sapatnekar. Hierarchical random-walk algorithms for power grid analysis. In *Proceedings of the Asia/South Pacific Design Automation Conference*, pages 499–504, 2004.
18. S. S. Sapatnekar and H. Su. Analysis and optimization of power grids. *IEEE Design and Test*, 20(3):7–15, May-June 2002.
19. S. B. K. Vrudhula, D. Blaauw, and S. Sirichotiyakul. Estimation of the likelihood of capacitive coupling noise. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 653–658, 2002.
20. T.-Y. Wang and C. C.-P. Chen. 3-D thermal-ADI: A linear-time chip level transient thermal simulator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(12):1434–1445, December 2002.
21. J. Westra, C. Bartels, and P. Groeneveld. Probabilistic congestion prediction. In *Proceedings of the ACM International Symposium on Physical Design*, pages 204–209, 2004.



# Crosstalk Cancellation for Realistic PCB Buses

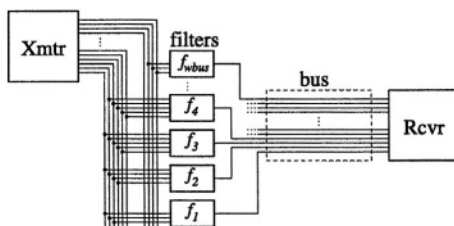
Jihong Ren and Mark R. Greenstreet\*

Department of Computer Science, University of British Columbia  
2366 Main Mall, Vancouver, BC, V6T 1Z4, Canada  
{jihong, mrg}@cs.ubc.ca

**Abstract.** We present a method for synthesizing crosstalk cancelling equalization filters for off-chip buses. Our approach is based on linear programming and allows direct optimization of the eye mask as well as controlling maximum filter output and maximum overshoot at the receiver. While the linear program formulation is flexible and straightforward, the resulting linear programs are quite large. We present an implementation of Mehrotra's interior point method that exploits the structure and sparsity of our problem, enabling the practical synthesis of filters for large buses. As an example, we show a 32-bit bus with tight wire spacings. Using our filters, we achieve 7.5 times the bandwidth of a bus with no equalization, and 5 times the bandwidth of a bus with single-line pre-emphasis.

## 1 Introduction

Exponentially increasing speed and integration levels of ICs have created a corresponding demand for high-bandwidth off-chip buses. However, off-chip electrical interconnects have limited bandwidth due to their dispersive and dielectric losses and crosstalk between wires. By integrating signal procession functions into I/O pads, designers can use the abundance and speed of on-chip transistors to compensate for the limitation of off-chip interconnect [1,2]. This paper presents a new approach for synthesizing equalizing filters for crosstalk cancellation.



**Fig. 1.** A typical channel with pre-equalizing filters for crosstalk cancellation.

Figure 1 shows the structure of a typical channel with a pre-equalization filter. A filter is assigned to each wire of the bus. Each filter takes as input a data bit and one

\* This research has been supported by BC-ASI, Intel, NSERC, and SUN Microsystems.

or more neighbouring bits in each direction and outputs a predistorted signal for one wire of the bus. Such filters can reduce cross talk and compensate for the high frequency attenuation.

For high speed digital transmission, equalization is commonly used to compensate for the high frequency attenuation arising from resistive losses of off-chip interconnect [3,4,5,6,7]. Most of these designs use independent pre-emphasis, where each wire is considered independently without regard to crosstalk. In [6], Zerbe et. al. describe a proprietary design with equalizers for crosstalk cancellation for nearest neighbours. They did not describe how they derived their filters. The work most closely resembling ours is [5], which models the distortions arising from an interleaved DAC as a multiple-input multiple-output (MIMO) response function. This paper generalizes [5] by looking at buses that are naturally modeled as MIMO channels when crosstalk is considered.

We present a practical method for synthesizing optimal filters based on linear programming (LP) along with results from using our synthesis procedure. In earlier work [8], we used a simplified, symmetric bus model where all wires were assumed to be identical. We now extend this work to allow different models for each wire. With this, parasitic effects of bonding wires, vias, connectors, terminators, etc. can be taken into account. The resulting optimization problems are much larger, and we present sparse matrix techniques that make their solution practical.

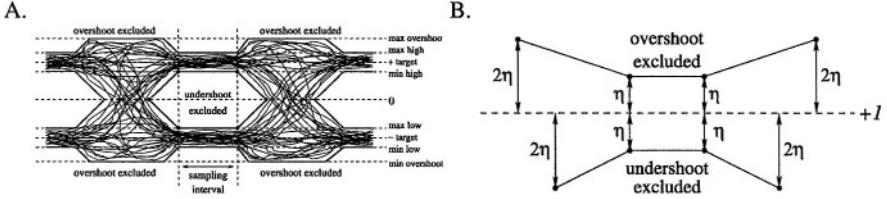
Section 2 gives mathematical programming formulations for optimizing signal integrity, and section 3 presents algorithms to solve these problems exploiting their sparsity. In section 4, we present the performance of filters designed with our methods showing that crosstalk cancellation can substantially improve the bandwidth achievable on buses with tight wire spacings.

## 2 Optimal Filter Synthesis

This section formulates filter synthesis problems in terms of mathematical programming. We describe how to optimize eye masks or minimize mean square error. Section 3 addresses practical issues that arise in solving these mathematical programs. To simplify the presentation, we make the following assumptions:

1. The response of the bus is linear and can be approximated accurately by a finite length impulse response.
2. Each wire is used to convey binary (i.e. two-level) data. The input high and low levels are +1 and -1. Likewise, the output target levels are +1 and -1.
3. The constraints on the high and low portions of the eye diagram are symmetric.
4. The same signal integrity criterion is used for every wire of the bus.

The first assumption pertains to the physical interconnect and should hold in most practical applications. The last three simplify the presentation. Extending our methods to multi-level signaling, other signaling levels, etc., is straightforward. Implementing a filter where every output depends on the values received on every input wire would consume unacceptable resources, especially for large buses. Noting that the largest contributions to crosstalk typically come from nearby wires, we consider filters where each output is computed from the input value for the wire itself and each of its  $w_{\text{filter}}$  closest



**Fig.2.** A. An Eye Mask with Overshoot Constraints; B. A Parameterized Eye Mask

neighbours in both directions. For example, the filter shown in figure 1 is a design with  $w_{\text{filter}} = 2$ .

To compensate for high-frequency losses, equalization filters typically have sample rates that are a small multiple of the data rate. Furthermore, the sample rate for our analysis is greater than the filter tap rate to prevent ringing. We write  $r_{\text{tap}}$  for the number of sample points used in the impulse response functions per filter tap time and  $r_{\text{bit}}$  for the number of sample points per data symbol transmitted. We write  $\delta_0$  for the delay from the input of the filter to the output of the bus. The examples in section 4 use a delay slightly greater than the LC delay of the bus for  $\delta_0$ .

### 2.1 Eye Diagrams and Masks

Eye diagrams and masks [9] are a worst-case measure of signal integrity, commonly used for digital communication. An eye diagram is formed by plotting traces from multiple bit periods on top of each other. During each bit period, there should be an interval during which each received value is either distinctly high or distinctly low according to the value of the transmitted datum. As shown in figure 2.A, an eye-mask specifies this interval as a region from which all traces are excluded. This excluded region is called an “eye”: the eye-height and eye-width are the maximum height and width of this region respectively. Moreover, we allow overshoot to be constrained by specifying excluded regions above and below the eye.

To formulate filter synthesis as an optimization problem, we create parameterized eye masks and maximize the height of the mask. By our symmetry assumption, it is sufficient to consider the upper half of the eye mask. Figure 2.B shows such an eye mask where the constraints for overshoot and undershoot are given in terms of the parameter  $\eta$ . More generally, we represent an eye mask as a set  $L$  of lower bounds and a set  $U$  of upper bounds. In particular,  $(s, \alpha)$  in  $L$  (resp.  $U$ ) has the interpretation that for every wire  $j$  and all inputs, the output from wire  $j$  must be at least  $1 - \alpha\eta$  (resp. at most  $1 + \alpha\eta$ ) at sample time  $s$ . Minimizing  $\eta$  optimizes the eye mask.

### 2.2 Optimizing Eye-Masks: Linear Programs

To find the worst-case inputs for the channel, we consider the response to each transmitted bit from each wire separately. For any fixed input and bus impulse response, the output from the bus is a linear function of the filter coefficients. Let  $\mathbf{f}$  be the vector of filter coefficients, and let  $w_{\text{bus}}$  denote the width of the bus. For any pair of wires,  $i, j \in$

$[1 \dots w_{\text{bus}}]$ , and any sampling time,  $s \in [0 \dots (n_{\text{bus}} + n_{\text{filter}} + 1)r_{\text{bit}}]$ , let  $g(i, j, s)$  be the vector such that  $g'(i, j, s)f$  is the response on wire  $j$  at time  $s$  to a one bit wide pulse on wire  $i$  starting at time 0 (with all other wires constant 0) given filter  $f$ . The  $g$  vectors are readily derived from the bus impulse response.

We now calculate the crosstalk and crosstalk-free components of the received signal. Focusing on the upper-half of the eye mask, we assume an input value of +1 on wire  $i$  at time 0. For each sample point  $s$  in the bit period, let  $u(i, s)$  denote the component of the received signal at sample time  $\delta_0 + s$  that is undisturbed by inputs on other wires or at other bit times:

$$u(i, s) = g'(i, i, \delta_0 + s) f \quad (1)$$

The disturbances are the contributions from all other wires, and from the wire itself at other bit times. Because the inputs are either +1 or -1, we can compute the responses for a +1 input bit on each wire at each bit time and sum the absolute values to get the maximal disturbance. Let  $d(i, s)$  denote the maximum total disturbance on wire  $i$  at sample time  $s$ :

$$d(i, s) = \left( \sum_{j=1}^{w_{\text{bus}}} \sum_{k=\mathbb{Z}} |g'(j, i, \delta_0 + k * r_{\text{bit}} + s) f| \right) - |u(i, s)| \quad (2)$$

The minimum value that wire  $i$  can have at time  $s$  is  $u(i, s) - d(i, s)$  and the maximum is  $u(i, s) + d(i, s)$ .

We can now write the linear program for eye mask optimization over a multi-wire bus:

$$\begin{aligned} \min_f \quad & \eta \quad \text{s.t.} \\ & \forall i \in [1 \dots w_{\text{bus}}]. \\ & \quad \forall (s, \alpha) \in L. \quad u(i, s) - d(i, s) \geq 1 - \alpha\eta \\ & \quad \wedge \forall (s, \alpha) \in U. \quad u(i, s) + d(i, s) \leq 1 + \alpha\eta \end{aligned} \quad (3)$$

### 2.3 Mean-Square-Error and Least-Squares Optimization

Mean square error (MSE) is an average-case measure of signal integrity. Typically, designers assume that the successive bits transmitted on each wire are independent, evenly weighted, Bernoulli random variables. We write  $Out(i, j, \tau)$  for the ideal output on wire  $j$  at time  $\tau$  in response to a one-bit pulse on wire  $i$  at time 0:

$$\begin{aligned} Out(i, j, \tau) \\ = 1, \text{ if } j = i \text{ and } \tau \in [\delta_0 \dots \delta_0 + r_{\text{bit}} - 1] \\ 0, \text{ otherwise} \end{aligned} \quad (4)$$

The value of the output within the measurement interval is usually much more critical than the values outside it. Accordingly, we assign a weight to each sample point in the bit period,  $\mathcal{Y}(s)$ , where  $\mathcal{Y}(s)$  is periodic with period  $r_{\text{bit}}$ . The resulting least-squares optimization problem for minimizing mean-square-error is:

$$\min_f \sum_{i,j=1}^{w_{\text{bus}}} \sum_{\tau \in \mathbb{Z}} \mathcal{Y}(\tau) (g(i, j, \tau) f - Out(i, j, \tau))^2 \quad (5)$$

### 3 Algorithms for Filter Design

We implemented both the LP and LSQ design methods using Matlab [10]. The LSQ problem is formulated directly according to the formulation presented in section 2.3. In this section, we present the implementation issues for the LP method and examine its time and space requirements. The size of the linear programs presents the greatest challenge in implementation, and developed sparse-matrix techniques that exploit structure in the filter design problem.

Given a bus impulse response and an eye mask specification, we set up a linear programming problem according to the formulation presented in section 2.2. In addition to the constraints from equation 3, we add constraints to limit the magnitude of the filter output on each wire at each tap time. By restricting the overdrive, we get filters that are much more practical to implement with little reduction in channel performance. Equation 3 becomes:

$$\min_{f,d,\eta,e} \eta \quad \text{s.t.} \quad \begin{bmatrix} G & I & 0 & 0 \\ -G & I & 0 & 0 \\ H & -X & \alpha & 0 \\ -H & -X & \alpha & 0 \\ I & 0 & 0 & I \\ -I & 0 & 0 & I \\ 0 & 0 & 0 & -M \end{bmatrix} * \begin{bmatrix} f \\ d \\ \eta \\ e \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ -\gamma \end{bmatrix} \quad (6)$$

Let  $A$  be the constraint matrix from equation 6. The rows with the  $G$  matrices compute  $d$ , the absolute values of the disturbances, the rows with  $H$  and  $-H$  compute the maximum undershoot and overshoot respectively:  $H$  computes the undisturbed response;  $X$  computes the total of the disturbance terms; and  $\alpha$  is vector of scaling terms for each measurement point of the eye mask. The rows with  $I$  in the leftmost column compute  $e$ , the absolute values of the filter coefficients. Finally, the row with  $M$  computes the maximum magnitude output for each wire at each sample time where  $\gamma$  is the output limit.

We now estimate the number of variables and constraints in the LP. The vector  $d$  includes one disturbance for each output wire for each input bit on each wire for each sample point in the eye mask. Typically,  $n_{\text{bus}} > n_{\text{filter}}$  and  $w_{\text{bus}} > w_{\text{filter}}$  in which case the number of filter coefficients  $k_{\text{filter}}$  is smaller than the number of disturbances  $k_{\text{disturb}}$ . Hence, the number of variables is dominated by  $k_{\text{disturb}}$ , the number of disturbances. This yields that the number of variables and the number of constraints are both  $O(w_{\text{bus}}^2 n_{\text{bus}} k_{\text{mask}})$ , and the total number of elements in the constraint matrix is  $O(w_{\text{bus}}^4 n_{\text{bus}}^2 k_{\text{mask}}^2)$ , where  $k_{\text{mask}}$  denotes the number of sample points for which the eye mask gives constraints. Fortunately, the matrix is quite sparse, and most of these elements are zero. Most of the nonzero elements of the matrix are from matrix  $G$  (see equation 6) whose rows are  $g(i, j, s)$  defined in section 2.2. Matrix  $G$  itself is sparse with approximately  $1/w_{\text{bus}}$  of its elements non-zero.

To solve these large LPs, we implemented Mehrotra's interior-point algorithm [11, 12] exploiting the sparsity and structure of our particular constraint matrix. Mehrotra's algorithm repeatedly solves for  $x$  in linear systems of the form:

$$A^T \Lambda^2 A x = y \quad (7)$$

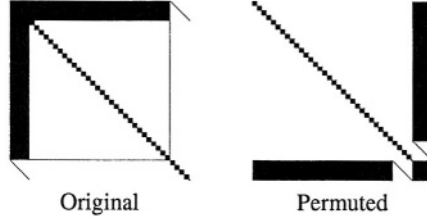


Fig. 3. Sparsity pattern of  $A^T A^2 A$ .

where  $A$  is the constraint matrix from equation 6, and  $\Lambda$  is a diagonal matrix whose elements are updated with each iteration of the algorithm. Equation 7 is called the “normal equation” for the LP. The  $A$  matrix is sparse, and, as shown in figure 3, so is  $A^T A^2 A$ . To simplify solving the system, we permute the columns of  $A$  so that the variable vector becomes  $[d, e, \eta, f]'$  which produces the sparsity pattern for  $A^T A^2 A$  shown on the right side of figure 3. Furthermore, we exploit the sparseness of  $G$  and the large blocks in  $A^T A^2 A$  by calculating the products for each block of the normal equation matrix separately, squeezing any remaining zeros out of the sparse representations, and then assembling the final matrix. We use Cholesky factorization to solve the permuted system. Computing the  $k_{\text{filter}} \times k_{\text{filter}}$  block in the lower right corner of the matrix dominates the time, and the total time requirement for solving the LP is

$$\text{Time(LP)} = O(w_{\text{bus}}^4 w_{\text{filter}}^2 m_{\text{filter}}^2 (n_{\text{filter}} + n_{\text{bus}}) k_{\text{mask}} k_{\text{iter}}) \quad (8)$$

where  $k_{\text{iter}}$  is the number of iterations required to converge to an optimal solution. The space requirement is

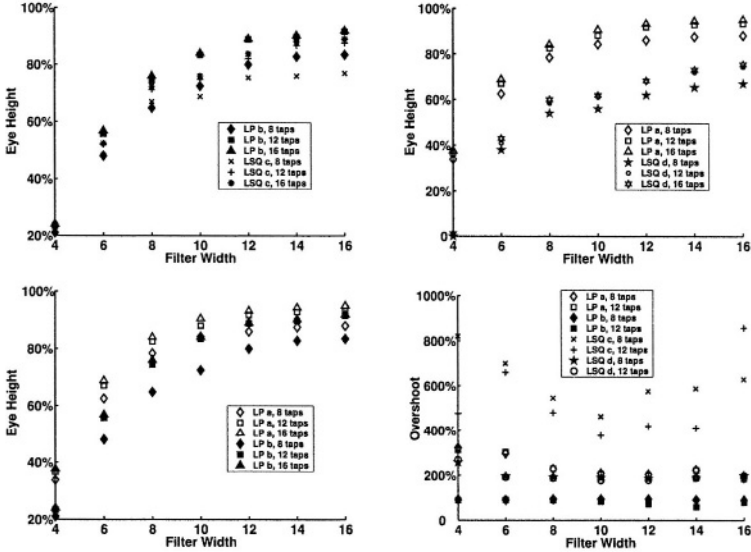
$$\begin{aligned} \text{Space(LP)} &= O(k_{\text{filter}} k_{\text{disturb}}) \\ &= O(w_{\text{bus}}^3 w_{\text{filter}} m_{\text{filter}} (n_{\text{filter}} + n_{\text{bus}}) k_{\text{mask}}) \end{aligned} \quad (9)$$

## 4 Results

To evaluate the filter design methods described in the previous sections, we implemented the optimization routines using Matlab [10]. All results were computed within Matlab’s 4Gbyte memory limit. For the least-squares based optimization, we used Matlab’s `lsq-mrdivide`. For the linear-programming based optimization, we wrote our own implementation of Mehrotra’s algorithm to exploit the sparsity described in section 3. We used Matlab’s `chol` function for sparse, Cholesky factorization.

### 4.1 A Realistic Bus Model

We used the field solver in HSPICE [13] to extract a model for a 32 bit bus with traces in 1 Oz copper, with 5 mil width and separation, running 10 mil above a ground plane and a dielectric constant of 4.5. Each line is resistively terminated. We then used HSPICE simulations to generate the bit responses and coupling terms. From these, our filter design procedure synthesized the optimal filters.



Filter design methods:

LP(a): The LP method using the eye mask from figure 2. There are no constraints for sample times corresponding to the transitions between bits. We constrained the magnitude of the filter output to be at most three times the target at all times.

LP(b): The same LP as above with further restrictions that the magnitude of the signals at the output of the bus must be less than twice the target level at all times.

LSQ(c): The LSQ method with weight  $\Upsilon(\tau) = 1$  for the four points of the measurement interval and unconstrained ( $\Upsilon(\tau) = 0$ ) elsewhere (see equation 5).

LSQ(d): The LSQ method with  $\Upsilon(\tau) = 1$  for the four points of the measurement interval and  $\Upsilon(\tau) = 0.4$  elsewhere. This prevents extreme overshoot during transitions.

Fig. 4. Performance of equalizing filters for a 32-bit bus

## 4.2 Simulation Results

All of the results reported in this section use four filter taps per bit-time and two sample times per tap time (i.e.  $r_{\text{tap}} = 2$  and  $r_{\text{bit}} = 8$ ). We write that a filter is  $m_{\text{filter}} \times w_{\text{filter}}$  to indicate that the filter has  $m_{\text{filter}}$  taps and a width (number of neighbours considered to each side) of  $w_{\text{filter}}$ . Unless otherwise noted, we use a bit-time of 500ps (i.e. 2Gbits/sec/wire). To test our filters, we generated the worst-case input sequences for each filter by determining the disturbance caused by each input wire and bit-time and then setting the sign of each input bit appropriately.

To obtain practical designs that achieve good crosstalk reduction, we must choose the width and length of the filter appropriately. In general, larger filters achieve better crosstalk cancellation at an increased cost for the hardware implementation. Figure 4 shows simulation results for filters with various lengths and widths and various synthesis methods. The time required for the several filter synthesis examples are shown in Figure 5.

Filter Size	$k_{\text{mask}}$	#variables	#constraints	Sparsity of $A^T A^2 A$	$k_{\text{iter}}$	Running Time (min)
$8 \times 8$	8	56449	105984	0.338%	15	38.88
$8 \times 16$	8	61441	110976	0.400%	17	71.45
$16 \times 8$	8	80385	146304	0.353%	17	112.83
$16 \times 16$	8	90369	156288	0.411%	17	214.82
$16 \times 16$	4	57729	90752	0.550%	16	116.88

Fig. 5. Statistics for various LP filter synthesis examples

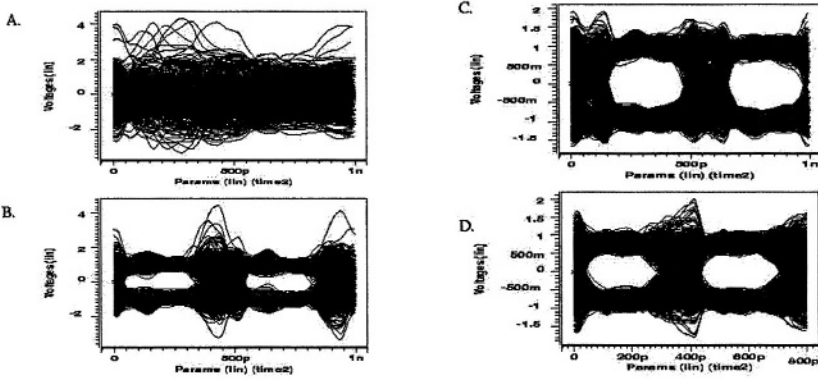


Fig. 6. Eye diagram at 2Gbits/sec/wire without filter (A) and with  $12 \times 10$  equalizing filters designed by the LSQ method without constraints on transition taps (B) and by the LP method with constraints on overshoot between measurement intervals (C). Panel D shows the eye diagram at 2.5Gbit/sec/wire with a  $12 \times 10$  equalizing filter designed by the LP method with constraints on overshoot between measurement intervals.

Figure 6 shows eye diagrams for the bus with no equalization and with  $12 \times 10$  filters synthesized by the LP and LSQ design methods.

We make several observations. First, the filters designed using the LP methods outperform their LSQ counterparts with the same width and length in nearly all cases. This shows the advantage of optimizing the eye mask directly rather than approximating it with mean-square error. Second, the LP filters have much lower overshoot than their LSQ counterparts. Furthermore, the overshoot for the LP filters can be greatly reduced with only a small penalty in the eye height, and this penalty is smaller for larger filters. The overshoot of the LSQ filters can be reduced by increasing the weight in the objective function of sample points outside of the measurement interval. However, LSQ eye height decreases significantly as overshoot is reduced.

Another way to evaluate the performance of an equalizing filter is its maximum operating bit rate. In this paper, we define the maximum operating bit rate as maximum bit rate at which the height of the eye is slightly greater than 50% and the eye width is over 25%. The maximum bit rate for the bus without a filter is 333MHz. Single line pre-emphasis using a 12 tap filter designed by the LP method increases the bandwidth to 500MHz. Nearest neighbour crosstalk cancellation ( $w_{\text{filter}} = 1$ ) raises the bandwidth to



800MHz. With a  $12 \times 10$  filter designed with the LP method, the channel has a maximum bit rate of approximately 2.5GHz; figure 6(D) shows the eye diagram. This is 7.5 times the bandwidth of the channel without a filter, and 5 times the bandwidth of the channel with independent pre-emphasis. With a  $12 \times 10$  filter designed by the LSQ method, the system has a maximum bit rate at approximately 2GHz; the eye is completely closed at 2.5GHz.

## 5 Hardware Implementation

We briefly consider hardware implementations of our filters based on a combination of look-up tables and adder trees. Using eight-bit data paths, 16 entry look-up tables and assuming 50 transistors for a one-bit adder or one-byte of SRAM, an  $8 \times 8$  filter can be implemented with about 50K transistors. Thus, we could provide 500 high-speed I/Os for a 500M transistor chip using about 5% of the total transistor budget. Looking to future technologies, increased integration densities and higher on-chip speeds will exacerbate the I/O gap. A  $12 \times 10$  filter requires about twice the hardware resources of an  $8 \times 8$  and could be very feasible for a 1 billion transistor chip. Our work thus far has focused on filter optimization, and our hardware estimates are based fairly simple, brute-force implementations. We plan to investigate hardware implementations further, including such issues as channel estimation and power consumption that must be considered in a real design.

## 6 Conclusions and Future Work

We have presented a method for synthesizing optimal filters for cross-talk cancellation for off-chip buses. Filters designed by our method can increase bus bandwidths by more than a factor of seven compared with a bus with no filters and more than a factor of three compared with a bus with only pre-emphasis and nearest neighbour crosstalk cancellation. We formulated the filter design problem as a linear programming (LP) problem, and showed how the problem could be solved using Mehrotra's interior point method by carefully permuting the normal equations to exploit the sparsity of the system. The ability to directly specify critical parameters such as eye masks, output magnitude, and overshoot is a clear advantage of our approach. This is in contrast with the commonly used least squares (LSQ) optimization techniques for filter design where error is minimized only in the average sense. Accordingly, filters designed with our method significantly outperform LSQ designed filters in terms of eye height, overshoot, and other eye mask parameters.

We are currently working on extending our work in several directions. First, our methods can be naturally applied to differential and multi-level signaling. With differential signaling, crosstalk drops off more rapidly with distance which should enable designs with smaller filters, but twice the number of wires and I/Os, than those described here. Because we constrain overshoot our methods extend directly to multi-level signaling.

**Acknowledgements.** Thanks to Robert Drost and John Poulton for helpful discussions of equalization and high-speed signaling. Thanks to Uri Ascher and Chen Greif for helpful discussions of large-scale optimization solvers.

## References

1. M. Horowitz, C-K.K. Yang, and S. Sidiropoulos. High speed electrical signalling: Overview and limitations. *IEEE Micro*, 18(1):12–24, Jan.–Feb. 1998.
2. M.–J.E. Lee, W. J. Dally, et al. CMOS high-speed I/Os - present and future. In *IEEE International Conference on Computer Design*, pages 454–461, 2003.
3. W.J. Dally and J.W. Poulton. Transmitter equalization for 4-GBPs signaling. *IEEE Micro*, 1:48–56, 1997.
4. A. Fiedler, R. Mactaggart, et al. A1.0625Gbps transceiver with 2x-oversampling and transmit signal pre-emphasis. In *Proc. of ISSCC97*, pages 238-239, 1997.
5. V. Stojanovic, G. Ginis, and M.A. Horowitz. Transmit pre-emphasis for high-speed time-division-multiplexed serial-link transceiver. *IEEE Transactions on Communications*, 38:551–558, 2001.
6. J.L. Zerbe, R.S. Chau, et al. A 2Gb/s/pin 4-PAM parallel bus interface with transmit crosstalk cancellation, equalization and integrating receivers. In *IEEE International Solid State Circuits Conference*, pages 430–432, 2001.
7. J.L. Zerbe, C.W. Werner, et al. Equalization and clock recovery for a 2.5-10 Gb/s 2-PAM/4-PAM backplane transceiver cell. *IEEE Journal of Solid-State Circuits*, pages 2121–2130, 2003.
8. J. Ren and M.R. Greenstreet. Synthesizing optimal filters for crosstalk-cancellation for high-speed buses. In *Proceedings of the 40th ACM/IEEE Design Automation Conference*, pages 592–597, 2003.
9. J. Chang. How to validate BLVDS SER/DES signal integrity using an eye mask. Technical Report AN200359, National Semiconductor Corporation, March 2002.
10. The Mathworks Inc. <http://www.mathworks.com>.
11. S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2:575–601, 1992.
12. J. Nocedal and S. Wright. *Numerical Optimization*, pages 395–417. Springer Series in Operations Research, Springer Press, 1999.
13. Avant! Software. *Star-Hspice Manual, Release 2001.4*, 2001.

# A Low-Power Encoding Scheme for GigaByte Video Interfaces

Sabino Salerno<sup>1</sup>, Enrico Macii<sup>1</sup>, and Massimo Poncino<sup>2</sup>

<sup>1</sup> Politecnico di Torino, Torino, ITALY

<sup>2</sup> Università di Verona, Verona, ITALY

**Abstract.** This work presents a low-power encoding techniques suitable for digital interface of Liquid Crystal Displays (LCD), in particular for a standard interface called Gigabyte Video Interface (GVIF), which is employed in many laptop computers.

The encoding is based on the spatial correlation that exists between consecutive pixels and, exploiting the serial nature of the transmission protocol, it encodes pixel differences with limited intra-word transition codewords so that to minimize the transition activity over serial links.

Application of this idea to GVIF-encoded data shows average transition savings of about 50% with respect to plain GVIF data, with the addition of relatively simple encoding architecture.

## 1 Introduction

Modern hand-held embedded systems host highly demanding multimedia applications that require high-end computational resources (high performance CPUs, large memories), but also high-quality Liquid Crystal Displays (LCDs). The latter requirement further complicates the issue of the energy demand of such devices; as a matter of fact, LCDs are taking an increasingly large share of the power budget: their consumption may easily exceed the 1 Watt mark. Power consumption of LCDs includes several sources of dissipation: the TFT panel, the frame buffer memory, the back-light inverter (for passive matrix displays) and the relative buses. For each component of display subsystem it is possible to find a proper low-power solution as shown in [1]. Off-chip parallel frame buffer buses consume a considerable amount of power due to the parasitic capacitances of wires that are orders of magnitude larger than that of on-chip buses.

RGB signals are sent from LCD controller to the LCD panel through serial links. Such links are flat cables with the largest capacitance (tens of pF per meters [15]). In this cables RGB and controls data travel continuously resulting in a large amount of power consumption. Encoding low-transition schemes have been proposed in [6] and [7] providing a significant saving in terms of transition activity. Both methods take advantage on strong correlation and on serial nature of adjacent pixels.

In this work we propose an extension of the encoding scheme in [7] called Limited Intra-Word Transition (LIWT) applied to not yet analyzed Gigabyte

Video Interface (GVIF) standard proposed by Sony. As it will be shown the LIWT encoding is suitable to minimize transition activity for GVIF because of its serial transmission nature.

The proposed encoding has been applied to a set of standard images, and has provided transitions savings of 58% on average with respect to a standard GVIF link.

The paper is organized as follows: Section 2 provides some background about digital visual interfaces for LCDs and the relative standards. Section 3 surveys previous work on low-power LCDs, and in particular solutions relying on bus encoding. Section 4 describes the limited intra-word transition (LIWT) codes, and how they can be applied to a GVIF interface. Section 5 shows the experimental results and Section 6 concludes the paper.

## 2 Digital Video Interface Standard

A typical digital LCD subsystem includes the following basic building blocks: the graphic controller, the *frame buffer* (i.e., a memory which stores an image, pixel by pixel and which is used to refresh a raster image), the *LCD controller*, which receives pixel data through a digital interface from the frame buffer, and the *LCD matrix*, usually made of active components such as a thin-film transistors (TFT).

Unlike its analog counterpart, digital LCD interfaces did not commit to a single official standard. A standard essentially defines (i) the electrical characteristics of the transmission (encoding of 0's and 1's, pixel clocks, etc.), (ii) the structure of the communication (how pixels are transmitted and encoded as symbols), and (iii) the physical shape of the connectors (number of pins, type of cable and connectors, etc.).

The various standards differ in each of these three features. However, there is a significant point in common concerning the second issue: *all standards rely on serial communication of pixel data*, mostly so as to minimize the electrical effects occurring during the transmission of data on a flat cable at frequencies in the range required by typical LCD displays (in the order of few hundred of MHz) as well as to keep the connectors' pin count low.

The following is a list of the most widely used digital LCD interface standards: Plug and Display [9] and Digital Flat Panel (DFP) [10], proposed by VESA (Video Electronics Standards Association); OpenLDI (Open LVDS Digital Interface), proposed by VICI (Visual Interface Consortium International) [11]; DVI (Digital Visual Interface) [12], proposed by DDWG (Digital Display Working Group), a working group driven by Intel; GVIF (Gigabyte Video Interface) [13], a proprietary interface proposed by Sony.

Each standard uses a different serial transmission protocol, which in principle is independent of the other characteristics of the standard. These are TMDS (Transition-Minimized Differential Signaling, used by Plug and Display, DFP and DVI), LVDS (Low-Voltage Differential Signaling, used by OpenLDI), and

GVIF (Gigabyte Video Interface, used in Sony’s proprietary interface). All signaling schemes include solutions to enforce *DC-balancing*, that is, to balance as much as possible the numbers of 0’s and 1’s transmitted, to reduce In most cases (e.g., short-length cables) DC-balancing is not an issue.

Some authors have devised solutions to improve the energy efficiency of these serial transmission schemes by properly reducing the number of transitions. This has been only done for TMDS [6,7], and LVDS [8]. In this work, we focus on the GVIF encoding, which has not been analyzed yet from this perspective.

### 2.1 Gigabyte Video Interface (GVIF) Standard Protocol

GVIF has been proposed by Sony to meet the need of a simpler hardware encoding scheme with thinner and longer cable drive capability suitable for modern multimedia system [13]. A simplified scheme of GVIF standard is shown on Figure 1. GVIF implements a PLL-based serialize/de-serialize (SER/DES) technique, using an encoder and a decoder linked by one differential pair flat cable.

The use of serial link to transmit XGA moving picture involves to have an high transmission rate (1.95 Gb/s). GVIF differs from TMDS and LVDS in the simpler required hardware, which does not require external components (EN-DEC, controller, reference clock source, cable equalizer), thus making transmission cost much cheaper. In fact, the receiver integrates a cable equalizer to allow displaying of XGA moving pictures along the only differential (up to 20 meters long) cable without requiring an optical fiber connection. The receiver also contains a data and clock recovery mechanism. Similarly to TMDS and LVDS, GVIF provides a DC balancing mechanism.

GVIF protocol encode 24-bit RGB signal onto 30-bit codeword. The code-word obtained can be divided into two sub-block: the first 24-bit sub-block packed the three 8-bit RGB color channel while the second 6-bit sub-block contain 1-3bit control signal (back light, DC-balancing), 1-bit for data enable (DE or blanking) and 2-bit for horizontal and vertical synchronism (H/V synch) operations.

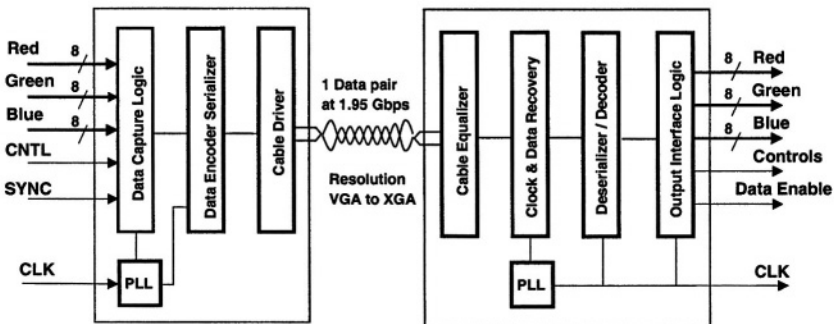


Fig. 1. Block Diagram of GVIF Digital Interface.

The main difference of the GVIF standard with respect to the TMDS and LVDS standards is the presence of only 1 serial link for transmitting RGB data. Redundant bit bits are needed to distinguish which of the three 8-bit color channel are a 2-LIWT differential code.

### 3 Previous Work

Techniques for reducing the energy consumption of the LCD sub-systems have been mostly limited to the design of individual component for the display systems [2,3,5] or to specific control mechanisms inside the LCD controller [1,4]. As a matter of fact, other common solutions such as shutdown techniques are not applicable to LCD, unless significant losses in performance and in image quality are tolerated.

Recently, some works [6,7,8] have addressed techniques to reduce the energy consumption during the transmission of pixel data over the serial links of the LCD bus. The main idea behind these works is that of exploiting the strong correlation between adjacent pixels, and transmitting reduced-transition codewords based on this fact. The encoding scheme exposed in [6], called *Chromatic Encoding*, has been applied to TMDS; it provides significant savings, but it presents several corner cases in which pixels cannot be encoded (overflow conditions, or situations in which the encoding must switch to plain TMDS). The approach of [7], applied to TMDS and to LVDS links [8], also exploits inter-pixel correlation but it uses the encoding conditionally, that is only for the most probable difference values.

The encoding proposed in this work, called *Limited Intra-Word Transition* (LIWT), generalizes the method of [7]. It uses the DBE as encoding basis scheme, but it extends so as to cover more cases to which encoding can be applied, and it applies it to the case of GVIF links. The following subsection presents the basics of the DBE encoding, whose principle is used by the techniques presented in this paper.

#### 3.1 Differential Bar Encoding

Differential Bar Encoding (DBE) [7] works by applying the principle of optimizing the *common case*; exploiting the inter-pixel locality, DBE encodes only a small subset of the possible inter-pixel difference values, and specifically *the most common ones*. All other difference values are not encoded, and the original pixel value (instead of its difference) is transmitted. These operations can be summarized by the pseudocode in Figure 2.

Consecutive pixels are represented with symbol  $w_t$  and  $w_{t-1}$  and  $\text{map}$  is the function mapping which translates differential pixel value on DBE code. *Range* represents the interval of differential value that can be encoded with a DBE code and it depends on the length of codeword allowed by the standard.

Transmission of pixel differences, however, does not guarantee by itself any reduction of the number of transitions; the issue is thus how to map values to

```

Encode( $w_t, w_{t-1}$ ) {
     $Diff = w_t - w_{t-1}$ ;
    if ( $|Diff| < Range$ )
         $Code_t = \text{map}(Diff)$ ;
    else
         $Code_t = w_t$ ;
    return  $Code_t$ 
}

```

**Fig. 2.** Encoding Algorithm.

low-transition codes. DBE simply takes pixel differences in order of occurrence probability and maps it to codewords having *only one intra-word difference*. For instance, TMDS allows 10-bit to encode an 8-bit RGB pixel value; using this 2-bit redundancy for encoding purposes, in [7] the DBE algorithm allows to cover the  $[-9,8]$  pixel difference range with codeword with at most 1 intra-word transition.

This guarantees that encoded pixel differences (which cover most of the transmitted pixels thanks to inter-pixel correlation) will be transmitted with *at most one transition per pixel value*.

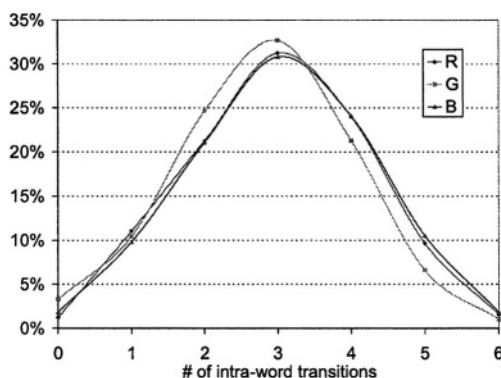
## 4 LIWT Encoding

LIWT extends the DBE scheme by allowing more than 1 intra-word transition per codeword. More precisely, we define *k-Limited Intra-Word Transition (k-LIWT)* codes as codewords with exactly  $\leq k$  intra-word transitions.

If we assume  $N$ -bit codewords, it can be shown that there are exactly  $2 \binom{N-1}{i}$   $N$ -bit words with  $i$  intra-word transitions [6]. The maximum value of  $k$  that can be tolerated depends on how intra-word transitions are distributed in typical images. Figure 3 shows the distribution of intra-word transitions, averaged over a set of 10 18-bit color images representing a wide variety of patterns, for each of the three RGB channels.

The plot shows a semi-Gaussian distribution, with an average of about 3 intra-word transitions (i.e., half the number of bits). The same behavior has been observed for 24-bit colors (with an average of about 4). Interestingly, this implies that  $k$ -LIWT codes with  $k$  strictly smaller than the average of the distribution will result in a saving of transitions. To be conservative, we will use 2-LIWT code for 24-bit colors as in GVIF.

Once the value of  $k$  is chosen, LIWT scheme simply works by mapping inter-pixels differences in decreasing order of occurrence probability to the codewords sorted in increasing order of intra-word transitions. This is similar to what is done in [6], except that LIWT uses codewords in a conditional fashion, as in DBE, and a pre-defined number of codes is used.



**Fig. 3.** Distribution of Intra-Word Transitions (18-bit Colors).

*Example 1.* Let's now derive the 2-LIWT code for an alphabet of  $N = 4$  bit words. The following is the code table, with patterns sorted in increasing order of intra-word transitions (right column):

Code	Intra- Word Transitions
0000	(0)
1111	(0)
1000	(1)
1100	(1)
1110	(1)
0001	(1)
0011	(1)
0111	(1)
1011	(2)
1001	(2)
1101	(2)
0100	(2)
0010	(2)
0110	(2)

There are a total of  $2 \binom{3}{0} + 2 \binom{3}{1} + 2 \binom{3}{2} = 2 + 6 + 6 = 14$  2-LIWT codes.

The above analysis describes the conceptual structure of a LIWT codeword, but it does not consider an important issue. Since the LIWT encoding may transmit either a pixel difference or a pixel value, we must somehow provide a way to tell which of the two has been transmitted. In other words, we need to also encode an information concerning the *interpretation* of the transmitted data. The simplest way to do this is to use an extra bit signal whether the bit pattern is encoded. This fact has two important consequences. First, the



encoding uses (some of) the available redundancy to signal what type data is sent, and, as a consequence, the “usable” redundancy is reduced, thus reducing the range of differences that can be encoded. Second, a careful analysis shows that the presence of this conditional bit may alter the “weight” of the encoding: a  $k$ -LIWT code may now exhibit more than  $k$  intra-word transitions because of this bit.

#### 4.1 2-LIWT Encoding on GVIF Standard

This section describes how a 2-LIWT encoding can be applied to a GVIF channel. As described in Section 2.1, GVIF encodes 24-bit RGB value onto a 30-bit word by adding six control bits. In principle, these bits are used by GVIF and reserved. However, half of them can be used as redundancy for implementing a 2-LIWT code. These bits (the *usable* bits) are the H-sync, the V-sync and the DE bits, which are not used at each clock cycle; conversely, the remaining three bits (the *unusable* bits) cannot be used because they are used by GVIF for DC balancing and clock and data recovery at each pixel transmission.

The 24-bit payload carries data about three pixel values, and, if they are LIWT-encoded, each one may be a pixel difference or a value, independent of the others. This means that there are  $2^3 = 8$  possibilities to describe the status of the three 8-bit values. The chosen encoding is that ‘1’ mean that 8-bit value is an 2-LIWT code, ‘0’ otherwise. Since we can play with exactly three usable redundancy bits the problem would be easily solved.

Unfortunately, things are not that simple. Any video encoding cannot work without taking synchronization operations into account. When horizontal or vertical synchronism (H-sync/V-sync) are requested, then a special, standard-specific word is sent over the cable instead of a pixel value (or difference). This implies that during those synchronization cycles, no encoding should be used, and this must be properly signaled.

Our energy-efficient GVIF protocol must then take synchronism and clock/data recovery into account. We achieve this by *discarding 2 out of the 8* combinations allowed by the three usable bits, and use them to signal H- and V-Sync operations. This choice still leaves six possibilities that can be used to identify which of the 8-bit data block are 2-LIWT code or the original RGB code. It is a good idea to assign this 6 possibilities to the most frequently combinations that actually occur.

This information has been obtained by analyzing the statistical behavior of a set of 10 images taken from a standard repository [14], and is summarized on table 1.

Column *RGB states* represents the eight possibilities of the RGB values (000 = all pixel values, 111 = all differences). The occurrence probability of each combination is shown in the right column. For instance, we observe that in 58.8% of the times differences can be transmitted for all the three RGB coordinates.

Since we have to choose six out of these eight possibilities, the choice is driven by two factors. First, we must maximize the probability of combinations covered by the chosen subset; second, we must also choose bit patterns with limited

**Table 1.** Analysis of Joint Probability.

<i>RGB states</i>	<i>Occurrence Probability %</i>
000	12.4
001	3.7
010	2.5
100	4.5
011	6.5
101	5.0
110	6.4
111	58.8

**Table 2.** Re-mapping table for Control Code.

<i>RGB states</i>	<i>Re-mapped Code</i>	<i>Function</i>
000	000	Data
001	101	H-synch
010	010	V-synch
100	100	Data
011	011	Data
101	001	Data
110	110	Data
111	111	Data

intra-weight transitions, since these patterns will be transmitted serially with the RGB codes.

These two objectives are met by assigning the configuration with lowest (highest) intra-word transition weight to the most (least) frequent configuration, and consider the occurrence of H- and V-synch signals as an event with small occurrence probability.

The mapping table that implements the above criteria is shown in Table 2. It can be noticed that the most frequent case (that is, all values can be encoded as differences) is encoded with the 111 patterns. Notice that the two least frequent cases (001 and 010) are discarded and are their pattern is used to represent H- and V-Synch data. Moreover, we can observe that the case 101 (representing R and B encoded), that would exhibit two intra-word transitions, has been remapped to 001.

## 5 Experimental Results

The proposed 2-LIWT/GVIF encoding scheme has been applied over a set of 10 24-bit RGB images taken from the SIPI database [14]. Table 3 reports the results in terms of total intra-word transition savings of the proposed 2-LIWT code with respect to the original GVIF. The savings are 58% on average, with maximum value around 68%.

**Table 3.** Transition Savings for 2-LIWT Code vs. GVIF standard.

<i>Bench. Image</i>	<i>GVIF</i>	<i>2-LIWT</i>	<i>Saving [%]</i>
couple	636715	303011	52.41
girl1	737606	322566	56.27
girl2	808481	320883	60.31
house	851053	300084	64.74
jellybeans	828634	263901	68.15
moonsurface	809739	368867	54.45
peppers	3241621	1385203	57.27
sailboat	3297321	1569274	52.41
splash	3052510	1118563	63.36
tree	840961	369503	56.06
<b>Average</b>			<b>58.54</b>

### 5.1 2-LIWT Encoder Architecture for GVIF

Figure 4 show a block diagram for the 2-LIWT/GVIF encoder. Each color channel is encoded through his own 2-LIWT encoder sub-block; the 8-bit Diff outputs are then latched in a register. The 2-LIWT encoder sub-block also generates a status bit that indicates the type (1 if a 2-LIWT code, 0 if a plain RGB value). All the status bits are sent, together with the control ones to a logic *Control Signal Block* to generate the actual six bits to be transmitted. The 30-bit data word generated are then sended to the serial link by a PISO (Parallel-In-Serial-Out) register. Due to the very high bit-rate (480-1776 Mb/s) over serial link, PISO register is builded with an appropriate double-polySi Bipolar technology as mentioned in [13].

The block diagram depicted in 5 shows the internal structure of a 2-LIWT encoder. A 2-LIWT code is obtained by two consecutive 8-bit data word  $w(t)$  and  $w(t - 1)$ . If differences fall into the  $[-29, 28]$  interval (the range actually represented), then the MUX controls (*set*) signal is set to 1 and the 9-bit arithmetic difference is translated in a 2-LIWT code. Otherwise the MUX select the plain RGB data value  $w(t)$ . When H-/V-sync operation are required, the *Control Signal Block* sets the six bits according to the GVIF standard.

The encoder has been implemented in VHDL and synthesized using Synopsys Design Compiler with a  $0.13\mu\text{m}$  technology by STMicroelectronics. The power consumption is about few hundreds of  $\mu\text{W}$  at a frequency of 200 MHz, using one of the images as a testbench.

## 6 Conclusions

Standards and currently used digital LCD interfaces don't provide an explicit mechanisms to contain energy consumption in spite of the impact on the overall power budget of the LCD subsystem.

We have shown a suitable encoding solution for reducing power consumption of such interfaces. This method exploit the correlation existing between consec-

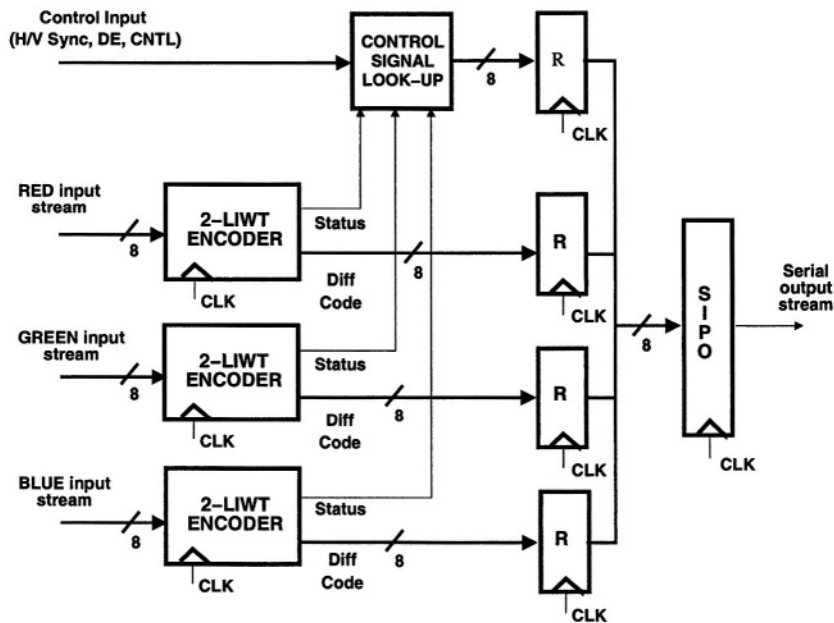


Fig. 4. 2-LIWT/GVIF Encoder Scheme.

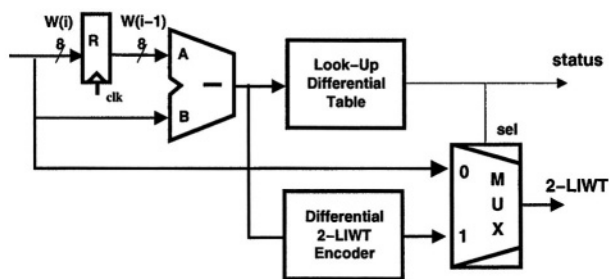


Fig. 5. 2-LIWT/GVIF Encoder Scheme.

utive pixel value and also based on the serial nature of transmission encode the most probable inter-pixels value with an appropriate low intra-word transition codeword. When we applied this encoding scheme to GVIF standard interface it possible to reach an average saving (term of intra-word transition activity) of over 50% with maximum value of 68%. The HW of this encoder is quite simple with power overhead less than power consumption over 1-pair link of order of magnitude.

## References

1. I. Choi, H. Shim, N. Chang, "Low-Power Color TFT LCD Display for Handheld Embedded Systems," *ISLPED'02: ACM/IEEE International Symposium on Low-Power Electronics and Design*, August 2002, pp. 112–117.
2. K. Meinstein. et al., "A Low-Power High-Voltage Column/Dot Inversion Drive System," *Society for Information Display International Symposium Digest of Technology*, Vol. 28. May 1997.
3. B.-D. Choi, O.-K. Kwon, "Stepwise Data Driving Method and Circuits for Low-Power TFT-LCDs," *IEEE Transactions on Consumer Electronics*, Vol. 46, No. 11, pp. 1155–1160, Nov. 2000.
4. F. Gatti, A. Acquaviva, L. Benini, B. Riccò, "Low Power Control Techniques For TFT LCD Displays," *CASES'02: International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pp. 218–224, Dec. 2002.
5. W.-C. Cheng, Yu Hou, M. Pedram, "Power Minimization in a Backlit TFT-LCD Display by Concurrent Brightness and Contrast Scaling" *DATE'04: Design, Automation and Test in Europe*, February 2004, pp. 252 - 257.
6. W.-C. Cheng, M. Pedram, "Chromatic Encoding: a Low Power Encoding Technique for Digital Visual Interface," *DATE'03: Design, Automation and Test in Europe*, March 2003, pp. 694–699.
7. E. Macii, M. Poncino, S. Salerno, A. Bocca, "Energy-Efficient Bus Encoding for LCD Displays" *GLSVLSI 2004 : Great Lakes Symposium on VLSI*, April 2004, to appear.
8. *Omitted for blind review.*
9. Plug and Display Standard, Version 1.0, VESA, June 11,1997.
10. Digital Flat Panel Standard, Version 1.0, VESA, February 14, 1998.
11. Open LVDS Display Interface (OpenLDI) Specification, Version 0.9, National Semiconductor, February 24, 1999.
12. Digital Display Working Group, Digital Visual Interface, V1.0, [www.ddwg.org](http://www.ddwg.org).
13. H. Kikuchi, T. Fukuzaki, R. Tamaki, T. Takeshita "Gigabit Video Interface : A Fully Serialized Data Transmission System for Digital Moving Pictures,"
14. A. G. Weber, "USC-SIPI Image Database Version 5," USC-SIPI Report #315, Oct. 1997. [sipi.usc.edu/services/database/Database.html](http://sipi.usc.edu/services/database/Database.html).
15. Extron Electronics, Cable Products Guide, <http://www.extron.com>.

# Dynamic Wire Delay and Slew Metrics for Integrated Bus Structures

Markus Tahedl and Hans-Jörg Pflleiderer

Department of Microelectronics, University of Ulm  
Albert-Einstein-Allee 43, 89081 Ulm, Germany  
markus.tahedl@e-technik.uni-ulm.de

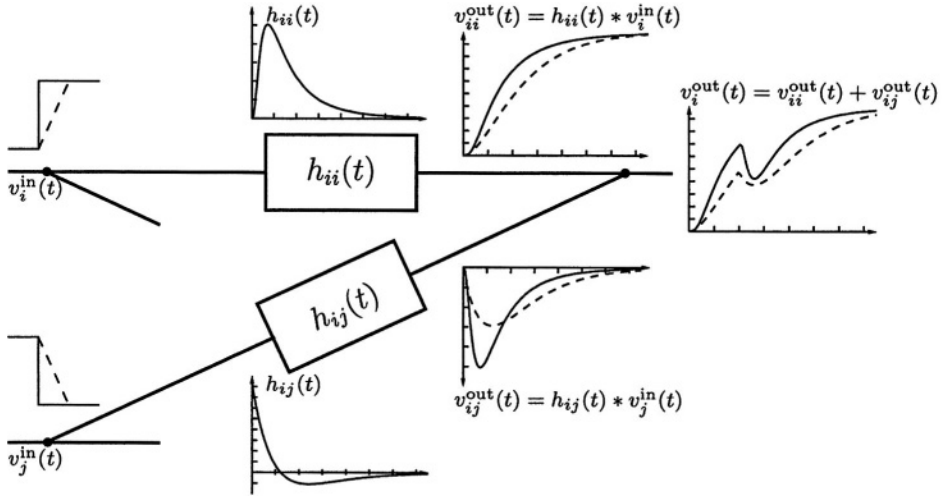
**Abstract.** In this paper we present a model for delay and slew calculation in on chip bus structures respectively parallel routed wires. Capacitive coupling is often neglected during circuit design although it can have significant influence on the wire delay. The model takes capacitive coupling effects into account. It is based on interpreting the impulse response of a linear circuit as a probability distribution function. Closed-form equations are derived for length dependent moment calculations in integrated bus structures as well as for the effects on the wire delay and output slew. The model is suitable for performance evaluation and optimization.

## 1 Introduction

Complex integrated systems on a single chip require communication between several components on the chip. Wires, buses or complex networks are used to transmit signals between subsystems. The physical representation of communication channels on a chip is a set of wires. While moving to deep-sub-micron and nano-technologies the bus performance is more and more determined by capacitive coupling effects. A lot of work is done reducing coupling effects. This spans from physical optimizations in wire routing over shielding and spacing techniques to coding schemes. Quite often a simple static model is used to account for coupling effects. The coupling capacitance is weighted with a switching factor and mapped to ground on the victim wire. This approach has some disadvantages. For the worst case a switching factor of two is often assumed. This is only true, when the victim and aggressor wires are switching at the same time and with the same transition time. The above assumption can lead to underestimations [1] in a real circuit. On the other hand worst case estimations can be too pessimistic. An accurate dynamic model, which enables the possibility to investigate many cases of switching pattern including different signal arrival times and transition times on each bus wire can help to determine the bus specific worst case delay and find novel methods to deal with crosstalk effects.

## 2 Modeling Concept

The modeling approach is based on decoupling of the non-linear driver and the linear wire behavior. The driver output signal can be determined using a



**Fig. 1.** Modeling concept

coupled  $\pi$ -model for the driver load [2]. It is an extension to [3] for coupled wires. Although the Elmore Delay [4] can result in excessive delay overestimation, it is widely used to determine the delay on a single wire. The Elmore Delay is based on the first moment of the impulse response. A metric which accounts for the first two moments of the impulse response was presented by the authors of [5]. In [6] a mathematical derivation for this metric is given. The first two moments of a lognormal distribution are matched to the first two moments of the impulse response. For delay calculation the method is extended to ramp inputs [7].

Crosstalk signals are induced by capacitive currents, which is basically the derivation of the aggressor voltage. Therefore, it is assumed that the crosstalk impulse response is the derivative of the lognormal distribution. This means the step response for crosstalk signal propagation can be matched to the lognormal distribution. With this assumption it is possible to use the formulas from [5] to compute a relative timing window where capacitive coupling occurs on a victim wire. To compute the delay change due to capacitive coupling, the approach presented by the authors of [8] is used. Therefore the maximum of the crosstalk signal within the timing window is derived by explicitly approximating the crosstalk signal waveform. Fig. 1 demonstrates the modeling concept for two bus wires. For the whole delay calculation approach the first two moments of the impulse response of each wire and the first three moments of the coupling impulse response of each victim-aggressor pair is required. For the derivation of the moments in a tree-structured bus a matrix representation is used.

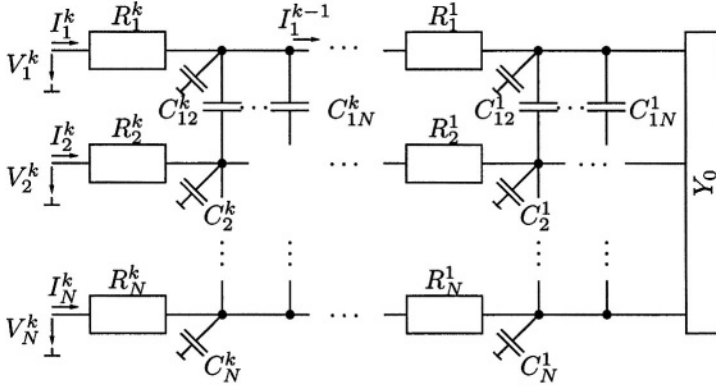


Fig. 2. Coupled bus representation with  $k$  RC elements

### 3 Transfer Function Moments

An on-chip bus structures can be modeled with distributed segments and lumped elements. A bus modeled by  $k$  lumped elements is depicted in Fig. 2. The input-output voltage relation of a lumped RC element can be expressed in matrix form as

$$V_k(s) = (s\mathbf{R}_k\mathbf{C}_k + \mathbf{R}_k\mathbf{Y}_{k-1}(s) + \mathbf{I}_N) \cdot V_{k-1}(s) . \quad (1)$$

$\mathbf{Y}_{k-1}(s)$  denotes the load and  $\mathbf{I}_N$  the  $(N \times N)$  identity matrix. The matrices  $\mathbf{R}_k$  and  $\mathbf{C}_k$  are the usual resistance and capacitance matrices similar to the matrices used in [2]. The transfer function for one single RC element is defined as  $V_{k-1}(s) = \hat{\mathbf{H}}_k^{-1}(s) \cdot V_k(s)$  and therefore  $\hat{\mathbf{H}}_k^{-1}(s) = s\mathbf{R}_k\mathbf{C}_k + \mathbf{R}_k\mathbf{Y}_{k-1}(s) + \mathbf{I}_N$ . With a concatenation of multiple transfer functions the input voltage can be expressed as

$$V_k(s) = \hat{\mathbf{H}}_k^{-1}(s) \cdot \dots \cdot \hat{\mathbf{H}}_1^{-1}(s) \cdot V_1(s) = \hat{\mathbf{H}}_k^{-1}(s) \cdot \mathbf{H}_{k-1}^{-1}(s) \cdot V_1(s) . \quad (2)$$

The first three moment matrices  $\mathbf{m}_1$ ,  $\mathbf{m}_2$  and  $\mathbf{m}_3$  of the transfer function  $\mathbf{H}(s)$  can be derived from the first three moments  $\tilde{\mathbf{m}}_1$ ,  $\tilde{\mathbf{m}}_2$  and  $\tilde{\mathbf{m}}_3$  of the inverse transfer function  $\mathbf{H}^{-1}(s)$  as

$$\mathbf{m}_1 = -\tilde{\mathbf{m}}_1 \quad (3)$$

$$\mathbf{m}_2 = -\frac{1}{2}\tilde{\mathbf{m}}_2 + \tilde{\mathbf{m}}_1^2 \quad (4)$$

$$\mathbf{m}_3 = -\frac{1}{6}\tilde{\mathbf{m}}_3 + \frac{1}{2}\tilde{\mathbf{m}}_2\tilde{\mathbf{m}}_1 + \frac{1}{2}\tilde{\mathbf{m}}_1\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1^3 . \quad (5)$$

Therefore it is enough to derive formulas for the moments of the inverse transfer function. For one single RC-Element the moment matrices are derived as

$$\hat{\mathbf{m}}_{k,1} = \mathbf{R}_k\mathbf{C}_k + \mathbf{R}_k\mathbf{y}_{k-1,1} \quad (6)$$

$$\hat{\mathbf{m}}_{k,2} = \mathbf{R}_k\mathbf{y}_{k-1,2} \quad (7)$$

$$\hat{\mathbf{m}}_{k,3} = \mathbf{R}_k\mathbf{y}_{k-1,3} \quad (8)$$



with  $\mathbf{y}_{k-1,i}$  the  $i$ -the admittance moment of the load to the element. A method for admittance moment derivation was already presented in [2]. In a recursive approach the moments of the inverse transfer function from stage  $k$  to stage 1 can be derived with (2) as

$$\tilde{\mathbf{m}}_{k,0} = \mathbf{I}_N \quad (9)$$

$$\tilde{\mathbf{m}}_{k,1} = \hat{\mathbf{m}}_{k,1} + \tilde{\mathbf{m}}_{k-1,1} \quad (10)$$

$$\tilde{\mathbf{m}}_{k,2} = \hat{\mathbf{m}}_{k,2} + \tilde{\mathbf{m}}_{k-1,2} + \hat{\mathbf{m}}_{k,1} \cdot \tilde{\mathbf{m}}_{k-1,1} \quad (11)$$

$$\tilde{\mathbf{m}}_{k,3} = \hat{\mathbf{m}}_{k,3} + \tilde{\mathbf{m}}_{k-1,3} + 3 \cdot \hat{\mathbf{m}}_{k,2} \cdot \tilde{\mathbf{m}}_{k-1,1} + 3 \cdot \hat{\mathbf{m}}_{k,1} \cdot \tilde{\mathbf{m}}_{k-1,2} \quad (12)$$

Further a distributed segment can be expressed as a concatenation of infinity lumped elements with  $\mathbf{R}_i = \mathbf{R}$  and  $\mathbf{C}_i = \mathbf{C}$  and length  $z$ . The lumped Elements are expressed as  $\mathbf{R} = \frac{\mathbf{r} \cdot z}{k}$  and  $\mathbf{C} = \frac{\mathbf{c} \cdot z}{k}$  with  $\mathbf{r}$  and  $\mathbf{c}$  the per unit length resistance and capacitance matrices respectively. For the limit  $k \rightarrow \infty$  the moments for a segment are derived as

$$\hat{\mathbf{m}}_1 = \frac{1}{2} \mathbf{rc} \cdot z^2 + \mathbf{ry}_1 \cdot z \quad (13)$$

$$\hat{\mathbf{m}}_2 = \frac{1}{12} \mathbf{rcrc} \cdot z^4 + \frac{1}{3} \mathbf{rcry}_1 \cdot z^3 + \mathbf{ry}_2 \cdot z \quad (14)$$

$$\hat{\mathbf{m}}_3 = \frac{1}{120} \mathbf{rcrcrc} \cdot z^6 + \frac{1}{20} \mathbf{rcrcry}_1 \cdot z^5 + \frac{1}{2} \mathbf{rcry}_2 \cdot z^3 + \mathbf{ry}_3 \cdot z \quad (15)$$

With (9) to (15) the first three moments of the inverse transfer function can be calculated recursively starting at the far ends of each branch of a tree structured bus. The transfer function moments depend on the first three inverse transfer function moments of the downstream element and they are computed using (3) to (5).

## 4 Crosstalk Evaluation

The delay and slew metrics for single wires from [6] are based on the assumption that the impulse response of a wire is similar to a lognormal distribution function. For slew calculation we use the simpler form in [6] which is based on the first two moments.  $LnD$  denotes the delay and  $LnS$  the slew at the far end of a wire.  $LnD$  and  $LnS$  are calculated with the transfer function moments  $\mathbf{m}_i$  where the elements on the main diagonals describes the wire behavior without crosstalk and the elements outside the main diagonals describes the coupling effects.

### 4.1 Crosstalk Timing Window

A similar approach like in [6] can be applied to determine a crosstalk timing window. A crosstalk impulse response  $h_{ij}(t)$  as illustrated in Fig. 1 is assumed. The coupling signal on wire  $i$  induced by a transition on wire  $j$  is given as  $v_{ij}^{\text{out}}(t) = h_{ij}(t) * v_j^{\text{in}}(t)$ . Furthermore we assume that the coupling step response

is similar to the impulse response of a single wire. Thus, integrating the coupling output voltage results in  $\int_0^t v_{ij}^{\text{out}}(\tau) d\tau = \int_0^t h_{ij}(\tau) d\tau * v_j^{\text{in}}(t)$ . Substituting  $v_{ij}'(t) = c'_{ij} \cdot \int_0^t v_{ij}^{\text{out}}(\tau) d\tau$  and  $h'_{ij}(t) = c'_{ij} \cdot \int_0^t h_{ij}(\tau) d\tau$  and normalizing with  $c'_{ij}$  to  $\int_0^\infty h'_{ij}(t) dt = 1$  results in a input output voltage relation similar to a single wire. The virtual transfer function  $h'_{ij}(t)$  is represented by a lognormal distribution function and the methods from [6] can be applied. In the  $s$ -domain this means that the moments of  $H'_{ij}(s)$  are derived as  $m'_{k,ij} = \frac{m_{k+1,ij}}{m_{1,ij}}$  and the normalizing factor is determined by the reciprocal first moment  $c'_{ij} = \frac{1}{m_{1,ij}}$ . The resulting delay  $LnD_{ij}$  and slew  $LnS_{ij}$  values describe the integral of the normalized crosstalk signal. Thus, they spawn a timing window when coupling effects can be significant.

## 4.2 Crosstalk Waveform and Peak Value

To derive a delay change curve (DCC) like in [8] explicit waveform estimations are necessary. The worst case delay as well as the parameters of the DCC's coefficients are determined by the crosstalk peak voltage and by the relative point of time when the crosstalk signal has its maximum. To determine these values, the normalized waveform of the coupling signal is assumed to be

$$v_{ij}(t) = \frac{e^{-\frac{t-t_s}{\alpha}} - e^{-\frac{t-t_s}{\beta}}}{\alpha - \beta}, \quad t \geq t_s, \quad \alpha > \beta \geq 0. \quad (16)$$

The parameter  $t_s$  represents the start time of the crosstalk signal on the victim wire whereas  $t = 0$  represents the start time of the input transition on the aggressor wire. For the case of better readability the indices  $i$  and  $j$  are omitted in the parameters. The waveforms and several parameters are illustrated in Fig. 3(a). One drawback of modeling the crosstalk waveform with (16) is that it does not match the assumed step response of the wire asymptotically. But it is very similar to the step response and provides analytical resolvability. The coefficients  $\alpha$  and  $\beta$  are derived by matching the first two central moments  $\mu$  and  $\sigma^2$  of (16) to the moments of the actual crosstalk signal. The central moments of the crosstalk signal are derived by adding the moments of the modified transfer function and the moments of the input signal as  $\mu = -m'_{1,ij} + \mu_{\text{in}}$  and  $\sigma^2 = 2 \cdot m'_{2,ij} - (m'_{1,ij})^2 + \sigma_{\text{in}}^2$ . We use  $\mu_{\text{in}} = \frac{T_a}{2}$  and  $\sigma_{\text{in}}^2 = \frac{T_a^2}{12}$  from [7] assuming an ramp input on the aggressor wire with  $T_a$  the input transition time. Hence, the parameters  $\alpha$  and  $\beta$  in (16) are derived as

$$\alpha = \frac{\mu - t_s}{2} + \frac{\sqrt{-(\mu - t_s)^2 + 2 \cdot \sigma^2}}{2} \quad (17)$$

$$\beta = \frac{\mu - t_s}{2} - \frac{\sqrt{-(\mu - t_s)^2 + 2 \cdot \sigma^2}}{2}. \quad (18)$$

The relative point of time and the value of the maximum are computed by setting the derivative of (16)  $\frac{d}{dt} v_{ij}(t)|_{t=t_m} = 0$ . This leads to

$$t_m = \ln\left(\frac{\alpha}{\beta}\right) \frac{\alpha \cdot \beta}{\alpha - \beta} + t_s \quad (19)$$

$$v_m^* = v_{ij}(t_m) . \quad (20)$$

The actual peak voltage value is  $v_m = V_{dd} \cdot \frac{v_m^*}{c_{ij}^*}$ . One problem occurs while calculating  $\alpha$  and  $\beta$ . The time  $t_s$  which represents the starting time point of the crosstalk signal is not exactly known. It will be in the range of  $\mu - \sigma$  and  $\mu - \sqrt{2}\sigma$  which results from the conditions  $\beta > 0$  and  $\alpha, \beta$  real values. An intuitive analytical solution for  $t_s$  follows from a worst case observation. The maximum crosstalk peak voltage occurs in the extreme case of  $\beta = 0$ . In this case the crosstalk waveform equation (16) reduces to

$$v_{ij}(t) = \frac{e^{-\frac{t-t_s}{\alpha}}}{\alpha}, t \geq t_s, \alpha > 0 . \quad (21)$$

The transition time  $LnS_c = LnS_{ij}$  of the virtual output voltage  $\int_{t_s}^t v_{ij}(\tau) d\tau = \left(1 - e^{-\frac{t-t_s}{\alpha}}\right)$  can be derived as  $LnS_c = \alpha \cdot \ln(9)$ . The virtual 50% voltage is reached at  $t_{50} = LnD_c + \frac{T_a}{2} = \alpha \cdot \ln(2) + t_s$ . Hence, the maximum of  $t_s$  is

$$t_{s,max}^{slope} = LnD_c + \frac{T_a}{2} - LnS_c \cdot \frac{\ln(2)}{\ln(9)} . \quad (22)$$

Another way to derive the maximum of  $t_s$  results from parameter analysis in (18). The maximum  $t_s$  can be derived from the case  $\beta = 0$  and results in  $t_{s,max}^{curve} = \mu - \sigma$ . If  $t_s > t_{s,max}^{curve}$  is chosen  $\beta < 0$  results from (18) and (16) doesn't describe a crosstalk waveform any longer. Therefore  $t_{s,max} = \min\{t_{s,max}^{slope}, t_{s,max}^{curve}\}$  is chosen. This approach will result in overestimation of the crosstalk peak value. A slightly better but still overestimating value for  $t_s$  was found by calculating the 90% point of the above described virtual output voltage and extrapolate a linear ramp back to zero voltage through the 10% point. The 90% voltage point is reached at the time  $t_{90} = t_{50} + \alpha \cdot \ln(5)$ . With this approach we believe we can partly correct the error induced by worst case estimations. No theoretical proof for this approach exists. But a good agreement with spice simulations is shown. Hence, the empirical value for the start time is computed with

$$t_{s,emp} = \min\left\{\mu - \sigma, LnD_c + \frac{T_a}{2} + LnS_c \left(\frac{\ln(5)}{\ln(9)} - \frac{9}{8}\right)\right\} . \quad (23)$$

## 5 Delay and Slew Metrics Including Coupling Effects

The step response and ramp response of one wire of a capacitive coupled bus structure can be determined by superimposing estimated waveforms. For calculating a delay change curve (DCC) the methods from [8] are used in a slightly modified manner.

## 5.1 Bus Delay Derivation

The approach in [8] computes a DCC which determines the wire delay depending on the relative signal arrival time on two coupled wires. Also an approach to enhance this to multiple aggressors was proposed by the authors of [8]. The waveforms for deriving a DCC are illustrated in Fig. 3(b) and described with

$$f(t, k) = \begin{cases} 0 & \text{for } t - k < 0 \\ \frac{v_m}{\tau_a} (t - k) & \text{for } 0 \leq t - k < \tau_a \\ v_m e^{-(t - \tau_a - k)/\tau_d} & \text{for } \tau_a \leq t - k \end{cases} \quad (24)$$

$$g(t) = \begin{cases} 0 & \text{for } t < 0 \\ V_{dd} (1 - e^{-t/\tau_r}) & \text{for } t \geq 0 \end{cases} \quad (25)$$

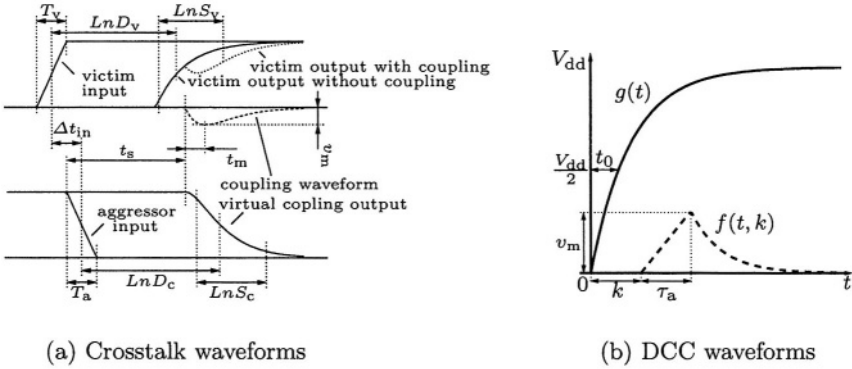
Whereas  $g(t)$  represents the self induced signal and  $f(t)$  the crosstalk signal. To obtain the parameters of these waveforms, the characteristic values are matched to the ones derived in section 4. We can establish the parameters as  $\tau_a = t_m - t_s$ ,  $\tau_d = \alpha$ ,  $\tau_r = \frac{LnS}{ln(9)}$ ,  $t_0 = \tau_r \cdot \ln(2)$ ,  $k_0 \simeq t_0 - \tau_a - \tau_r \cdot \ln(2\nu + 1)$  and  $k = \Delta t_{in} + t_s - \frac{T_a}{2} - LnD_v + t_0$  where  $\Delta t_{in} = t_v^{in} - t_a^{in}$  is the relative signal arrival time with  $t_v^{in}$  and  $t_a^{in}$  the absolute switching time points on the victim and aggressor wires respectively.  $T_a$  represents the aggressor transition time and  $LnD_v$  denotes the delay on the victim wire without coupling. A linear approximations for the delay change  $\Delta t(k)$  as proposed in [8] shows good results for  $k > k_0$ . For  $k \leq k_0$  a linear approximation can result in significant errors. Also higher order approximations doesn't gain much accuracy but the analytical complexity rises. Our approach is to use the first two Taylor coefficients two match a exponential function  $u_{sp}(t) = 1 - e^{-(t-t_i)/\tau}$  for  $k \leq k_0$ . Furthermore the linear approximation for  $k_0 < k \leq t_0$  is fitted to match  $\Delta t(k_0)$  and  $\Delta t(t_0) = 0$ . With this description and  $\nu = \frac{v_m}{V_{dd}}$  the DCC is summarized as

$$\Delta t(k) = \begin{cases} 0 & \text{for } k_0 < k, \nu \leq 0 \\ 0 & \text{for } t_0 < k, \nu > 0 \\ \frac{\tau_r \cdot \tau_d \ln(1 - B) \cdot (1 - B)}{(\tau_r \cdot B - \tau_a) \cdot (t_0 - k_0)} (k - t_0) & \text{for } k_0 < k \leq t_0, \nu > 0 \\ \frac{\tau_r \cdot \tau_d \cdot \ln(1 - 2 \cdot \nu \cdot A(k)) \cdot (2 \cdot \nu \cdot A(k) - 1)}{2 \cdot \nu \cdot A(k) \cdot \tau_r - \tau_d} & \text{for } k \leq k_0 \end{cases} \quad (26)$$

The coefficients are  $A(k) = \exp\left(-\frac{t_0 - \tau_a - k}{\tau_d}\right)$  and  $B = 2 \cdot \nu \cdot A(k_0) = 2 \cdot \nu \cdot (2\nu + 1)^{-\tau_r/\tau_d}$ . An error is induced by choosing  $\tau_d = \alpha$  but it asymptotically matches the crosstalk signal (16) for  $t = t_m$  and  $t \rightarrow \infty$ .

## 5.2 Effective Slew Derivation

Beyond the delay on wires the transition time or the slew is important for timing analysis. The output slew is used as input slew to the next gate. This is necessary



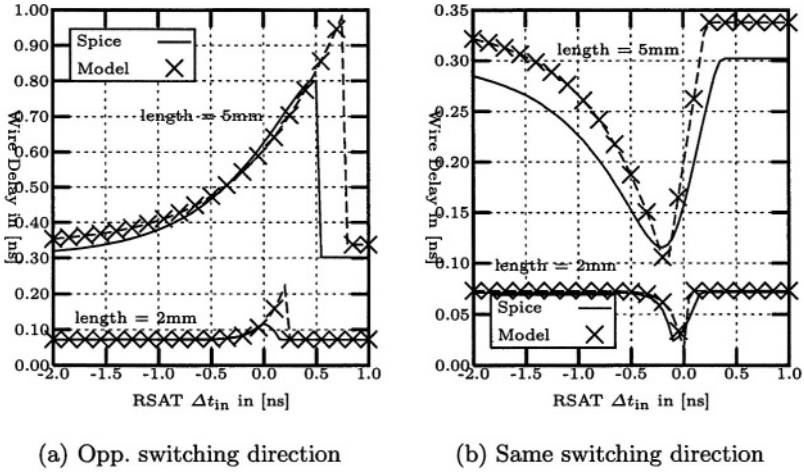
**Fig. 3.** Variable definition for waveform approximations

for delay calculations. The slew time normally is defined as the time the output signal takes to travel from 10% to 90% of  $V_{dd}$ . In presence of crosstalk signals this definition may not make much sense. In modern technologies the gain of a logic gate is much higher than one and therefore the derivative of the input signal at  $\frac{V_{dd}}{2}$  will define the delay. In [9] the output slew rate of CMOS gates was assumed to be 70% of the slew rate at  $\frac{V_{dd}}{2}$ . In our model we use the waveform which results from superposition for DCC derivation as the output waveform. It matches the real waveform accurately in some range of  $\frac{V_{dd}}{2}$ . Unlike in [9] we use the time from 10% to 90%  $V_{dd}$  of the superimposed output waveform. Similar to the delay change metric we can define a metric for the output slew dependant on the relative signal arrival time, where  $LnS_v$  represents the slew without coupling.

$$S_{out}(k) = \begin{cases} LnS_v & \text{for } k_0 < k, \nu \leq 0 \\ LnS_v & \text{for } t_0 < k, \nu > 0 \\ \frac{8\tau_a\tau_r}{5\tau_a \cdot e^{-\Delta t(k)/\tau_r} + 10\nu\tau_r} & \text{for } k_0 < k \leq t_0, \nu > 0 \\ \frac{2 \cdot \tau_r \cdot \tau_d \cdot (2 \cdot \nu \cdot A(k) - 1) \cdot \ln(3)}{2 \cdot \nu \cdot A(k) \cdot \tau_r - \tau_d} & \text{for } k \leq k_0 \end{cases} \quad (27)$$

## 6 Simulation Results

The data used for simulation is based on a  $0.18\mu m$  process. In Fig. 4 the delay on a victim wire is plotted versus the relative signal arrival time (RSAT) in a two-wire-bus. As a parameter the wire length is varied. The calculated delay is in good agreement with spice simulations. An exception is a delay overestimation near the delay peak. This overestimation has a positive effect on delay estimation and RSAT optimization. Some additional noise can force the delay to jump to the large estimated delay values. Or the crosstalk peak may propagate through the connected logic gates and causes logical hazards. For this reason the case where crosstalk just doesn't affect the delay should be avoided while circuit design.



**Fig. 4.** Wire delay versus relative signal arrival time (RSAT)  $\Delta t_{in} = t_v^{in} - t_a^{in}$  in a two-wire-bus.  $t_v^{in}$  and  $t_a^{in}$  are the absolute switching time points of the victim and aggressor wires respectively

Overestimation of the delay in this range provides a metric which accounts for the possible larger delay and tends to avoid this case in an optimization approach. All presented simulation results are based on the empirical start time estimation described in (23). For a worst case estimation of the crosstalk start time the curves in Fig. 4 would be similar but with an larger range of overestimation and larger overestimation values.

Due to the large amount of parameters which can be varied, a Monte Carlo simulation was applied. Independent uniform distributions were used for bus length, resistance per unit length, capacitance per unit length, switching direction, relative signal arrival time and input transition time. Bus lengths up to  $> 5$  mm has been used. The results are compared to spice simulations. The mean  $\mu_{err}$  and standard deviation  $\sigma_{err}$  of the delay error and the delay error induced by an slew error to the following gate are listed in Table 1. For simulations a balanced buffer with driver strength 1 and fan-out 10 was used as following gate. The second line of Table 1 shows the error distribution when all errors with an absolute value of  $\geq 100\%$  are neglected. This can be done because we believe that all errors  $\geq 100\%$  occur due to the large overestimation around the delay peak.

**Table 1.** Error distribution for delay and slew

Error Distribution in %	Delay		Slew	
	$\mu_{err}$	$\sigma_{err}$	$\mu_{err}$	$\sigma_{err}$
Full Distribution	5.13	18.71	3.17	4.04
Truncated Distribution	3.52	7.72	3.22	3.88

## 7 Conclusion

We have presented closed form equations for dynamic delay and slew modeling in integrated bus structures. Capacitive coupling can have a significant influence on the delay. Therefore crosstalk effects are included in the model. It is based on a moment matching method, where length dependent transfer function moments are used. The output slew is given as an effective slew for delay calculations in connected gates. The model is suitable for timing analysis and for investigation on timing optimization. Fast explorations of the influence of capacitive coupling effects on the bus delay is enabled and the model is in good agreement with spice simulations.

**Acknowledgment.** This work was supported by Atmel Germany GmbH and by the German ‘Bundesministerium für Bildung und Forschung’ with the indicator 01M3060C. The authors are responsible for the content.

## References

1. F. Dartu, L. T. Pileggi: Calculating Worst-Case Delays Due to Dominant Capacitive Coupling. in *Proc. IEEE/ACM Design Automation Conference*, pp. 46-51, 1997.
2. M. Tahedl, H.-J. Pflleiderer: A Driver Load Model for Capacitive Coupled On-Chip Interconnect Busses. in *Proc. Int. Symposium on System-on-Chip*, pp. 101-104, 2003.
3. P. R. O’Brien, T. L. Savarino: Modeling the Driving-Point Characteristic of Resistive Interconnect for Accurate Delay Estimation. in *Proc. IEEE Int. Conference on Computer-Aided Design*, pp. 512-515, 1989.
4. W. C. Elmore: The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. *Journal of Applied Physics*, vol. 19, pp. 55-63, 1948.
5. C. J. Alpert, A. Devgan, C. V. Kashyap: RC Delay Metrics for Performance Optimization. *IEEE Trans. on Computer-Aided Design*, vol. 20, pp. 571-582, 2001.
6. C. J. Alpert, F. Liu, C. Kashyap, A. Devgan: Delay and Slew Metrics Using the Lognormal Distribution. in *Proc. IEEE/ACM Design Automation Conference*, pp. 382-385, 2003.
7. C. V. Kashyap, C. J. Alpert, F. Liu, A. Devgan: Closed Form Expressions for Extending Step Delay and Slew Metrics to Ramp Inputs. in *Proc. Int. Symposium on Physical Design*, pp. 24-31, 2003.
8. T. Sato, Y. Cao, K. Agarwal, D. Sylvester, C. Hu: Bidirectional Closed-Form Transformation Between On-Chip Coupling Noise Waveforms and Interconnect Delay-Change Curves. *IEEE Trans. on Computer-Aided Design*, vol. 22, pp 560-572, 2003.
9. J. L. Rosello, J. Segura: A Compact Propagation Delay Model for Deep-Submicron CMOS Gates including Crosstalk. in *Proc. IEEE/ACM Design, Automation and Test in Europe*, 2004.

# Perfect 3-Limited-Weight Code for Low Power I/O

Mircea R. Stan and Yan Zhang

University of Virginia, ECE Department  
Charlottesville, VA 22904, USA

**Abstract.** Codes for low power I/O reduce the number of  $1$ 's transmitted over the bus at the expense of overhead (encoding and decoding) and an increase in the number of bus lines and/or bus cycles. We propose treating special cases of Limited-Weight Codes (LWC) as duals of block Error Correcting Codes (ECC). This enables the *algorithmic* generation of a LWC based on its dual ECC and is a fundamentally novel way of encoding for low power I/O. We also propose a perfect 3-LWC as dual to the Golay binary ECC which shows good power savings for heavily loaded buses.

## 1 Introduction

The power dissipated on an unterminated bus is dominated by the number of transitions on the bus, while the power on a terminated bus is dominated by the number of  $1$ 's for each transfer over the bus [1,2]. *Encoding* the data can reduce this power while adding redundancy in *space* in the form of extra bus lines or in *time* in the form of extra bus cycles [3]. An unterminated bus using *transition signaling* becomes equivalent to a terminated bus in the sense that only  $1$ 's (transitions) are dissipative, while  $0$ 's (no transition) are not. In this paper we assume that unterminated buses use transition signaling which means that we can use similar coding strategies for low-power I/O (*i.e.* reduce the number of  $1$ 's) for both terminated and unterminated buses [3].

The loads associated with buses are typically large, hence the savings at the I/O translate into overall power savings as long as there is little overhead for encoding/decoding. This requires in general that the coding be done *algorithmically* [3]. The resulting code is a mapping:  $C : U \rightarrow V$  where  $U$  has dimension  $2^K$  (uncoded  $K$ -bit data) and the codewords are  $N$ -bit wide, with  $N > K$ .

There has been a significant body of work in the area of bus encoding in the past ten years: encodings for address buses [4], encodings for multiplexed buses [5], custom encodings for special purpose buses [4], general purpose encodings for data buses [3], limits to power reduction efficiency [6], encoding for buses with coupling [7], and many others. More recently though many results in this field have been incremental and derivative of earlier work. In contrast, we present here a totally new approach that has a strong theoretical foundation and also leads to practical implementations.

### 1.1 Limited-Weight Codes

In general, both self- and coupling-capacitances affect power dissipation on buses, with coupling being the dominant factor for on-chip buses in deep submicron technologies [8].



Special purpose applications can lead to correlation on buses which can benefit significantly from custom encodings [9]. This paper addresses highly loaded *off-chip* general purpose data buses. These are typically terminated buses on which there is little data correlation, and for which coupling is not dominant. In such cases *general purpose* encoding schemes (e.g. Bus-Invert [1]) are an appropriate low power solution.

Limited-Weight Codes (LWC) represent such a general framework for encoding such data buses [3]. In this paper we use capital letters for LWC parameters,  $K$  for unencoded symbol length,  $N$  for codeword length, and  $N - K$  for the extra redundancy; and the usual  $k$  for information bits,  $n$  for code length, and  $n - k$  for check bits, in the case of Error Correcting Code (ECC) parameters. An  $M$ -LWC [3] will use  $2^K$   $N$ -bit codewords with at most  $M$  1's (Hamming  $\text{weight}[v(t)] \leq M$ ). Because the entropy [10] of the source must remain the same with or without coding (we need to be able to distinguish between codewords) it is necessary that:

$$\binom{N}{0} + \binom{N}{1} + \binom{N}{2} + \dots + \binom{N}{M} \geq 2^K \quad (1)$$

A simple-minded method of generating an  $M$ -LWC would be to choose  $2^K$  of all possible  $N$ -bit codewords with  $\text{weight} \leq M$  and arbitrarily map the  $2^K$  unencoded symbols to the chosen codewords. A  $2^K \times N$  look-up table (LUT) would then be needed for coding and a  $2^N \times K$  LUT for decoding [11]. This is rarely practical due to the large LUTs needed. Ideally we should be able to “design” new codes starting from  $M$ ,  $K$  and  $N$ , which define the statistical properties of a LWC. In this paper we provide such a practical method based on a duality between LWCs and ECCs [3].

## 2 The LWC-ECC Duality

We link LWCs to ECCs by observing that the *codewords* of a LWC, when viewed as patterns of 1's and 0's, are similar to the *error patterns* corrected by a standard ECC [3]. As opposed to other types of “duality” which usually appear in the Euclidean space [12], the duality between LWCs and block ECCs is along the Hamming distance. An  $m$ -ECC is *designed* to correct all patterns with at most  $m$  1's; this is “dual” to an  $M$ -LWC which *contains* codewords with at most  $M$  1's. *Decoding* an ECC requires finding the error pattern and, since the error pattern is dual to a LW codeword, this is similar to *encoding* for a LWC. *Decoding* an LWC will be dual to finding the ECC syndrome for a particular error pattern which is intrinsically a very simple operation.

### 2.1 Perfect and Semi-perfect EC and LW Codes

There are only a few type of combinations of  $N$ ,  $M$  and  $K$  for which (1) becomes an equality. We call the corresponding codes LW *perfect* codes as they are dual to EC perfect codes. LW perfect codes use *all* possible patterns with  $M$  1's and from this point of view are somewhat “optimal” [3]. Since there exist only three types of perfect ECCs (Hamming codes, repetition codes and the Golay codes), there can be only three types of perfect dual LWCs. An ECC that is not perfect, if *completely* decoded, will correct some words with more than  $m$  bits in error. The dual LWC will then have some codewords

with a larger number of 1's (corresponding to the error patterns with  $> m$  bits in error) which will reduce its low-power effectiveness. Even worse, in practice efficient complete decoding methods for non-perfect EC codes are generally lacking.

Several conclusions can be drawn from the ECC/LWC duality:

- using an ECC for generating a LWC requires a *complete* decoding algorithm,
- the duals of *perfect* or semi-perfect ECCs are desirable for LWC generation,

The previously proposed Bus Invert [1] can be viewed as  $N/2$ -LWCs dual to ECC repetition codes, and one-hot codes as 1-LWCs dual to Hamming codes. The only remaining type of EC perfect code is the Golay code. In this paper we investigate its dual LWC and show that it has good low power characteristics.

### 3 Perfect 3-LWC Dual to the Golay Binary ECC

The binary Golay code is a (23,12) ECC that can correct up to 3 bits in error. We can obtain the dual 3-LWC by any complete decoding algorithm for the Golay code. The 3-LWC will be a mapping from  $2^{11}$  unencoded symbols to the  $2^{11}$  patterns of length 23 with at most 3 1's that represent the error patterns of the Golay code. Here are some definitions needed to understand the algorithm [13]:

#### Definitions

Let  $I_{12}$  be the  $12 \times 12$  identity matrix.

Let  $I_{11}$  be the  $11 \times 11$  identity matrix.

Let  $B$  be the  $12 \times 12$  matrix (and let  $A$  be the the  $12 \times 11$  matrix  $A$  obtained by removing the last column of  $B$ ):

$$B = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Let  $G = [I_{12}B]$  be the generator matrix for the *extended* (24, 12) Golay code.

Let  $H = [A^T I_{11}]$  be the parity-check matrix for the *nonextended* (23, 11) code.

Let  $l_1, l_2, l_3, \dots, l_{11}$  be the 12-bit columns of  $A$  (same as first 11 columns of  $B$ ).

Let  $l_{12}$  the last 12-bit column of  $B$ .

Let  $r_1, r_2, r_3, \dots, r_{12}$  the 11-bit rows of  $A$ .

Let  $x_{12}^{(i)}$  denote the binary 12-tuple in which only bit  $i$  is non-zero.

Let  $y_{11}^{(i)}$  denote the binary 11-tuple in which only bit  $i$  is non-zero.

Let  $0_{11}$  denote the 11-tuple in which all bits are zero.

Let  $0_{12}$  denote the 12-tuple in which all bits are zero.

Let  $s$  be the unencoded 11-bit data.

Let  $e$  be the encoded 23-bit 3-LWC code.

### 3.1 3-LWC Encoding

Starting from a decoding algorithm for the (24, 12) binary extended Golay ECC [13] we propose the following efficient encoding for a perfect 3-LWC (see also Fig. 1):

1. Start with the arbitrary 11-bit  $s$  (uncoded data)
  - a) If  $\text{weight}(s) \leq 3$ , then set  $e = (0_{12}, s)$  (block A in Fig. 1)
  - b) If  $\text{weight}(s \oplus r_i) \leq 2$  for some  $1 \leq i \leq 12$ , then  $e = (x_{12}^{(i)}, s \oplus r_i)$  (block B)
2. Compute 12-bit  $t = (A \cdot s^T)^T$  (block C)
  - a) If  $\text{weight}(t) \leq 3$ , then set  $e = (t, 0_{11})$  (block D)
  - b) If  $\text{weight}(t \oplus l_i^T) \leq 2$  for some  $1 \leq i \leq 11$ , then  $e = (t \oplus l_i^T, y_{11}^{(i)})$  (block E)
3. Compute  $u = t \oplus l_{12}^T$  (block F)
  - a) If  $\text{weight}(u) \leq 3$ , then set  $e = (u, 0_{12})$  (block G)
  - b) If  $\text{weight}(u \oplus l_i^T) \leq 2$  for some  $l_i, 1 \leq i \leq 11$ , then  $e = (u \oplus l_i^T, y_{11}^{(i)})$  (block H)
4. Output 23-bit 3-LWC code  $e$

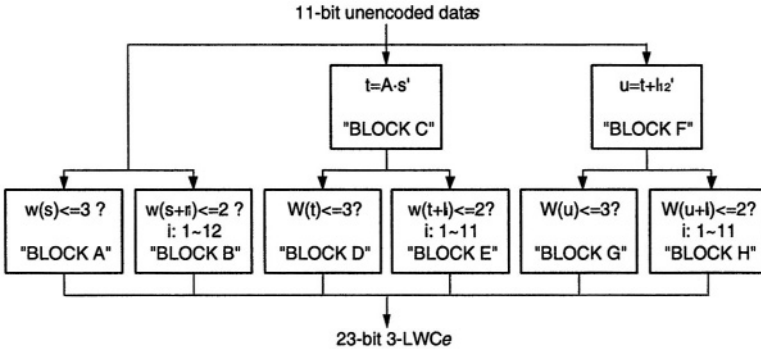


Fig. 1. Block diagram of 3-LWC parallel encoding algorithm.

A key aspect of the algorithm is that (due to the reliance on coding theory concepts) in almost all cases only one of the “If” statements above will be true. The few cases where more than one “If” is true are trivial, and the resulting output pattern  $e$  is still unique. For example, if  $s = 0_{11}$ , then both block A and block D will be true, yet the output is the same  $e = 0_{23}$  either way. This important property is due to the duality with the ECC and permits a simple combining of all the outputs either sequentially (as in the step-by-step description) or concurrently (as in Fig. 1). Although the parallel calculation can increase the power overhead, it will also reduce the latency of the encoder and is more appropriate for a hardware implementation. Here are a few encoding examples to help understand the procedure:

*Example 1.* Let the uncoded data  $s = (0001100000)$ . Since  $\text{weight}(s) < 3$  (case 1(a)) it means the coded data  $e = (0_{12}, s) = (0000000000000001100000)$ .

*Example 2.* Let the uncoded data  $s = (10111110101)$ . Since  $\text{weight}(s) > 3$  we first try to find rows of  $A$   $r_i$  such that  $\text{weight}(s \oplus r_i) \leq 2$ . We see that  $s \oplus r_4 = (00000110000)$ . This has weight 2 (case 1(b)), so  $e = (x_{12}^{(4)}, s \oplus r_4) = (00010000000000000110000)$ .

*Example 3.* Let the uncoded data  $s = (11100111111)$ . Since  $\text{weight}(s) > 3$  we first try to find  $r_i$  such that  $\text{weight}(s \oplus r_i) \leq 2$ . There is no such row. Hence we compute  $t = (A \cdot s^T)^T = (000000010011)$ . This has weight 3 (case 2(a)), so  $e = (t, 0_{11}) = (0000000100110000000000)$ .

*Example 4.* Let the uncoded data  $s = (00110110010)$ . Since  $\text{weight}(s) > 3$  we try to find rows of  $A$   $r_i$  such that  $\text{weight}(s \oplus r_i) \leq 2$ . There is no such row. Hence we compute  $t = (A \cdot s^T)^T = (111011100100)$ . Since  $\text{weight}(t) > 3$ , we then check columns of  $A$   $l_i$  such that  $\text{weight}(t \oplus l_i^T) \leq 2$ . We see that for  $i = 2$ ,  $t \oplus l_2^T = (000000000110)$ . This has weight 2 (case 2(b)), so  $e = (t \oplus l_2^T, y_{11}^{(2)}) = (000000000110010000000000)$ .

*Example 5.* Let the uncoded data  $s = (11001010101)$ . Since  $\text{weight}(s) > 3$  we try to find rows of  $A$   $r_i$  such that  $\text{weight}(s \oplus r_i) \leq 2$ . There is no such row. Hence we compute  $t = (A \cdot s^T)^T = (111010000110)$ . Since  $\text{weight}(t) > 3$ , we then check columns of  $A$   $l_i$  such that  $\text{weight}(t \oplus l_i^T) \leq 2$ . There is no such column. Hence we compute  $u = t + l_{12}^T = (010111110111)$ . Since  $\text{weight}(u) > 3$ , we then check columns of  $A$   $l_i$  such that  $\text{weight}(u \oplus l_i^T) \leq 2$ . We see that for  $i = 1$ ,  $u \oplus l_1^T = (001000001000)$ . This has weight 2 (case 3(b)), so  $e = (u \oplus l_1^T, y_{11}^{(1)}) = (001000001000100000000000)$ .

### 3.2 3-LWC Decoding

Based on the ECC duality, the decoding algorithm for the 3-LWC is equivalent to finding the ECC syndrome from the ECC error pattern. This is much simpler than the encoding and consists of only one computation:

1. Start with the received 23-bit  $e$  (coded data)
2. Compute and output the 11-bit uncoded data  $s = (H \cdot e^T)^T$

It is clear that the 3-LWC decoding can be implemented with just a few gates and does not require a look-up table. Here is an example:

*Example 6.* Let the encoded data  $e = (001000001000100000000000)$ . Then we compute  $s = (H \cdot e^T)^T = (11001010101)$  which is exactly the uncoded vector of Example 5.

### 3.3 Savings for the Proposed 3-LWC

Assuming random data uniformly distributed on the bus, the switching activity for a 11-bit uncoded bus is 5.5 transitions/cycle. With the perfect 3-LWC the resulting 23-bit coded bus will have a switching activity of:  $(3 \cdot \binom{23}{3} + 2 \cdot \binom{23}{2} + \binom{23}{1}) / 2048 = 5842 / 2048 = 2.85$  transitions/cycle. This represents 48% theoretical savings compared with the uncoded case, which of course comes as a trade-off with significant redundancy (12 extra bus lines or 1.1 extra bus cycles). In order to achieve a practical implementation we need to keep the encoding/decoding overhead to a minimum and to choose applications for which the savings are worth the extra redundancy.

In order to verify the algorithm on non-random data we have also tested it on various files as shown in table 1. As expected, the savings are always quite close to the theoretical values for the random case.

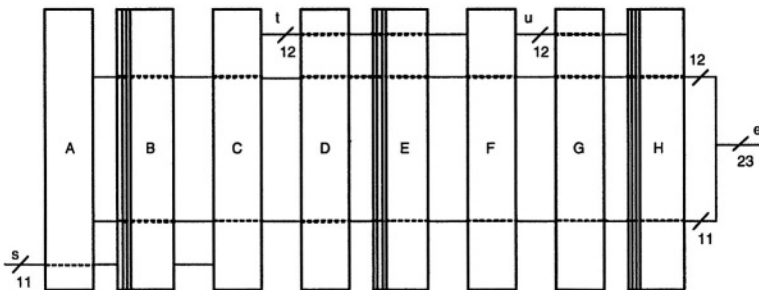
**Table 1.** Savings in number of 1's for different nonrandom data transferred over the bus.

file name	size (bytes)	1's uncoded	1's coded	savings
ntvdm.exe	395024	1292912	713347	44%
setuplog.txt	214511	717889	430630	40%
kernel32.dll	731920	2014425	1218219	40%
pic.jpg	69050	272070	140783	48%
video.mpg	653071	2251026	1307255	42%
audio.wav	904190	3599951	1857568	48%
win.zip	472940	1929448	981468	49%

### 4 3-LWC Hardware Implementation

The proposed LWC encoding algorithm includes three basic operations: GF(2) bitwise add (XOR), GF(2) matrix multiply (AND and bitwise XOR) and threshold gate (with threshold 2 or 3) to determine the Hamming weight.

Since the vectors and matrices involved are all known and fixed, we can simplify the addition and matrix multiplication according to the specific inputs. For example XORs become buffers or inverters depending on the fixed vector bits being 0 or 1, respectively. Similarly the AND gates can be replaced by wires (when the bit is 1) or by lack of a connection (when the bit is 0) for the fixed matrix multiplications. The entire hardware becomes quite simple and can be efficiently implemented as a bit-sliced 23-bit wide datapath with 11-bit input and 23-bit output as in Fig. 2. The only complex gates remaining are the threshold gates.

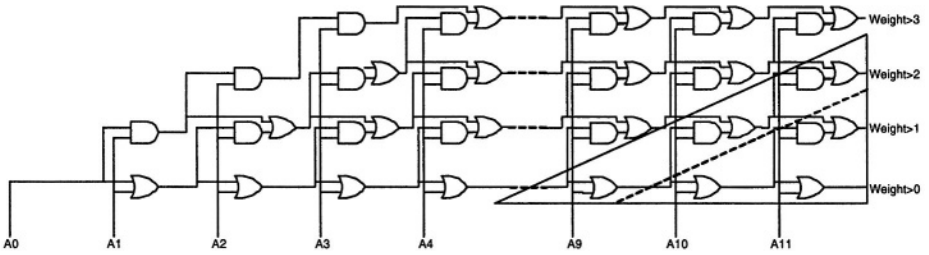


**Fig. 2.** Block-level floorplan of the hardware bit-sliced encoder. The letters in every block (from A to H) correspond to those in the step-by-step algorithm. All computations are done in parallel as a large bit-sliced combinational logic.

#### 4.1 Array Threshold Gate Circuit

In the above algorithm threshold functions appear in almost every step. The threshold gates in this algorithm are somewhat related to the majority gates necessary for the original Bus Invert [1], a majority gate being just a special case of a threshold gate. As

such we could implement threshold gates using a tree of full-adders and a comparator, but this would be an inefficient circuit. Fig. 3 shows the proposed implementation of an *array* threshold gate. For the 3-LWC the thresholds are small (2 or 3), instead of  $N/2$  as for Bus Invert, and we can take advantage of this aspect. Each row of the array computes an increasing threshold for the  $N$  inputs (12 inputs in the figure). The output of the first row will be 1 if the weight is  $>0$ , the second will be 1 if the weight is  $>1$ , and so on. The area complexity of such a gate is  $O(N \times (W + 1))$ , where  $W$  is the desired threshold. The delay is  $O(N)$ , both the area and delay comparing favorably to other alternatives for small thresholds. We use the gate in Fig. 3 as a universal threshold gate for the proposed 3-LWC algorithm. When only 11 inputs are needed we connect an input to 0, and for a threshold of 2 we ignore the most significant output bit as needed.



**Fig. 3.** Schematic of the proposed array 3-threshold gate. The rightmost triangular area can be omitted if only the most significant outputs are needed.

## 4.2 Practical Applications

We have implemented the entire encoding/decoding LWC circuit in Fig. 2 at the transistor level in a generic  $0.18\mu\text{m}$  CMOS at 2 V supply voltage. Due to the complexity of the encoding procedure, the 3-LWC is suitable mainly for off-chip or otherwise heavily loaded buses. We have considered first the straightforward case where we compare an 11-bit uncoded bus with a 23-bit coded bus (with transition signaling) driving full-swing an I/O load capacitance of 100pF. Spice simulations using uniform random data show the 3-LWC encoding circuit achieving 45% actual power savings, the overhead being only 3% for the encoding/decoding circuits. The delay for encoding/decoding is less than 15% of the total delay on the bus. Assuming a pipelined bus transfer this adds only one extra cycle to the overall latency which should be negligible. For smaller load capacitance the savings are reduced since the overhead of the encoder stays almost the same. For a load capacitance of 5pF the overhead is the same as the savings and the LWC technique becomes ineffective.

Although the setup above demonstrates savings with an actual implementation, the 11-bit and 23-bit formats for the uncoded and coded buses are hardly practical. We have chosen a terminated SCSI bus as a more realistic test case for our encoding scheme. For a terminated bus the 3-LWC code can be used directly to encode the data with no need for transition signaling [2].

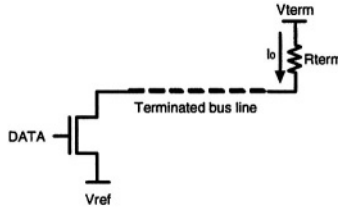


Fig. 4. Terminated bus line as used on a SCSI bus.

Fig. 4 shows a simple model for a parallel-terminated bus line using a passive resistor to match the characteristic impedance of the line. When Data is 1, the voltage on the bus line is pulled low and the current consumption is large. When Data is 0, the voltage on the bus line stays high and the current consumption is minimal.

The SCSI bus is a terminated bus widely used in computer system to control hard drives, CD-ROMs, scanner, tape drives and other peripherals, several variants of the bus being standardized and widely used in computer systems. Modern SCSI buses in general use active terminator circuits instead of simple passive resistors, but the power consumption aspects are quite similar. When an output is active, it is at a logic low (typically 0.2V) and consumes 24mA (maximum). When inactive, the line is at a logic high (minimum 2.85V) and the output consumption is only 0.5mA (maximum). Under these conditions, with a Vdd of 5.5V (worst case) the power dissipation for a 1 is  $Pd(1) = (5.5V - 0.2V) \times (24mA) = 127.mW$  and for a 0 is  $Pd(0) = (5.5V - 2.85V) \times (0.5mA) = 1.325mW$  [14].

We propose an implementation of 3-LWC for a SCSI bus as shown in Fig. 5. Shift register FIFOs are used to match throughput for different bus widths (e.g. shift in 11 bits uncoded data and shift out 23 bits codeword). The sources of overhead are the encoder/decoder and shift registers, and the extra redundancy on the bus (extra cycles).

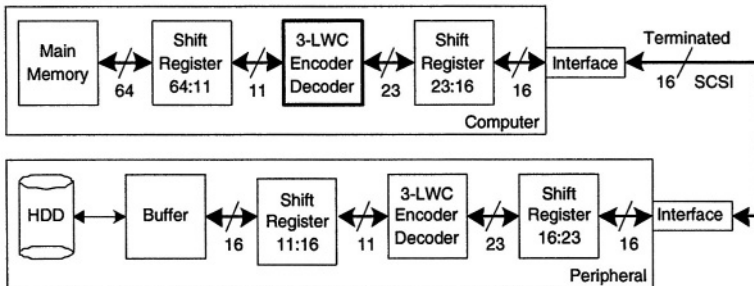


Fig. 5. Block diagram of 3-LWC for 16-bit SCSI.

Assuming uniformly distributed random data (on average half 0's and half 1's) the energy to transmit 1000 bytes of data will be:

$$E_{uncoded} = 1000 \cdot (4 \cdot Pd(1) + 4 \cdot Pd(0)) = 514.1J$$

In the 3-LWC coded case we will need to transmit 23/11 more data (2091 bytes), but the number of 1's is going to be significantly less. The energy to transmit the same amount of data in the coded case will be:

$$E_{coded} = 2091 \cdot (0.99 \cdot Pd(1) + 7.01 \cdot Pd(0)) = 282.7J$$

This represents a respectable 45% savings without the encoding/decoding overhead. The overhead of the encoder is only  $0.2mW$  which is truly negligible compared to the power consumed on the bus and will add only  $0.4J$  to the total energy consumed. We expect the same kind of savings for non-random data as suggested by the results in section 3.3. Of course these savings come at the expense extra latency for the transfer and may not be a convenient trade-off for high performance applications. Better figures of merit for such applications are the energy-delay product or the energy-delay-squared products. The 3-LWC does not improve the energy-delay product, however, encoding methods are orthogonal to other methods to reduce power (*e.g.* voltage scaling) and can be used in conjunction with such methods for low-power applications for which performance is not a first order concern.

## 5 Conclusion

The duality between Limited Weight codes and linear block Error Correction codes described in this paper gives a theoretical method for the algorithmic generation of codes for low-power I/O. This observation can also offer new insights on previously proposed low-power encoding methods, like Bus Invert and one-hot encoding. The 3-LWC dual to the Golay (23, 12) binary ECC proposed here shows significant power savings for heavily loaded buses. On a SCSI bus the circuit implementation of the encoder/decoder achieves  $\approx 45\%$  savings in energy at the expense of extra latency.

The fact that ECC decoding is “hard” means that LWC encoding and decoding will be hard for arbitrary parameters. Any future advances in the efficient complete decoding of block EC codes will be directly applicable to the algorithmic generation of their dual LWCs.

## References

1. Stan, M.R., Burleson, W.P.: Bus-invert coding for low power I/O. IEEE Transactions on VLSI Systems **3** (1995) 49–58
2. Stan, M.R., Burleson, W.P.: Coding a terminated bus for low power. In: Great Lakes Symposium on VLSI, Buffalo, NY (1995) 70–73
3. Stan, M.R., Burleson, W.P.: Low-power encodings for global communication in CMOS VLSI. IEEE Transactions on VLSI Systems (1997)
4. Benini, L., De Micheli, G., Liroy, A., Macii, E., Odasso, G., Poncino, M.: Power optimization of core-based systems by address bus encoding. IEEE Transactions on VLSI Systems **6** (1998) 554–562
5. Cheng, W., Pedram, M.: Power-optimal encoding for a DRAM address bus. IEEE Transactions on VLSI Systems **10** (2002) 109–118



6. Ramprasad, S., Shanbhag, N.R., Hajj, I.N.: A coding framework for low-power address and data busses. *IEEE Transactions on VLSI Systems* **7** (1999) 212–221
7. Sotiriadis, P.P., Chandrakasan, A.P.: Bus energy reduction by transition pattern coding using a detailed deep submicrometer bus model. *IEEE Transactions on Circuits and Systems I* **50** (2003) 1280–1295
8. Sotiriadis, P.P., Chandrakasan, A.P.: A bus energy model for deep submicron technology. *IEEE Transactions on VLSI Systems* **10** (2002) 341–350
9. Benini, L., De Micheli, G., Macii, E., Poncino, M., Quez, S.: System-level power optimization of special purpose applications: the Beach Solution. In: *International Symposium on Low Power Electronics and Design*. (1997) 24–29
10. Blahut, R.E.: *Theory and Practice of Error Control Codes*. Addison Wesley, Reading, MA (1983)
11. Tabor, J.F.: Noise reduction using low weight and constant weight coding techniques. Master's thesis, Massachusetts Institute of Technology (1990)
12. Conway, J.H., Sloane, N.J.A.: *Sphere Packings, Lattices and Groups*. Springer Verlag, New York, NY (1993)
13. Vanstone, S.A., van Oorschot, P.C.: *An Introduction to Error Correcting Codes with Applications*. Kluwer Academic Publishers, Norwell, MA (1989)
14. Fairchild Semiconductor: Active SCSI terminator - important issues, capacitance and power dissipation (1994) Application Note 42027, [www.fairchildsemi.com/an/AN/AN-42027.pdf](http://www.fairchildsemi.com/an/AN/AN-42027.pdf).

## A Appendix: Matlab Code for Encoding and Decoding

Matrices A and B are the ones described in section 3.

```

%LWC encoding, DATA is 11-bit input vector, CODE is 23-bit output
%Block A
VECT11 = DATA;
if gfweight(VECT11)<=3,
    CODE = [zeros(1,12) VECT11];
end
%Block B
for i = 1:12,
    VECT11 = gfadd(DATA,A(i,:));
    if gfweight(VECT11)<=2,
        ROW = zeros(1,12);
        ROW(i) = 1;
        CODE = [ROW, VECT11];
    end
end
%Block C
DATAC = [encode(DATA, 12 , 11, 'linear', A)]];
%Block D
VECT12 = DATAC;
if gfweight(VECT12)<=3,
    CODE = [VECT12, zeros(1,11)];
end
%Block E
for i = 1:11,
    VECT12 = gfadd(DATAC,B(:,i)');
    if gfweight(VECT12)<=2,
        COL = zeros(11,1);
        COL(i) = 1;
        CODE = [VECT12, COL'];
    end
end
%block F
DATAF = gfadd(DATAC,B(:,12)');
%Block G
VECT12 = DATAF;
if gfweight(VECT12)<=3,
    CODE = [VECT12, zeros(1,11)];
end
%Block H
for i = 1:11,
    VECT12 = gfadd(DATAF,B(:,i)');
    if gfweight(VECT12)<=2,
        COL = zeros(11,1);
        COL(i) = 1;
        CODE = [VECT12, COL'];
    end
end
end

%LWC decoding, CODE is encoded 23-bit LWC pattern
%DATADEC should be the same as original transmitted DATA
DATADEC = [encode(CODE, 11 , 23, 'linear', [A',eye(11)])]];

```

# A High-Level DSM Bus Model for Accurate Exploration of Transmission Behaviour and Power Estimation of Global System Buses\*

Claudia Kretzschmar, Torsten Bitterlich, and Dietmar Müller

Dpt. of Systems and Circuit Design  
Chemnitz University of Technology  
09126 Chemnitz, Germany  
ckkre@infotech.tu-chemnitz.de

**Abstract.** Deep sub-micron effects influence the signal transmission especially on global wires. In order to select communication architectures, minimize power dissipation and improve signal integrity it is necessary to explore DSM effects already at high levels of design abstraction. Therefore in this paper we present a parameterized high-level simulation model which is based on SPICE simulations and evaluates therefore signal integrity and power dissipation very accurately. The maximum error of our model was 1.3%. In comparison to a full SPICE simulation the required computation time could be reduced by a factor of up to 900.

## 1 Introduction

Technology scaling in the deep sub-micron (DSM) range allows a higher functionality on the same area since the integration densities of systems on chip (SoC) increase. These advantages are bought by a number of parasitic effects whose relevance grows inverse proportional to the structure size. Within current technologies the DSM effects can be alleviated by designs in a tile-like structure [3] of modules with 50 to 100 kgates. Due to the complexity power dissipation of integrated circuits is becoming a limiting factor for SoCs [1]. A major portion of overall power dissipation is consumed by global wires [2]. Reasons are the high coupling capacitances between adjacent wires which together with mutual inductance influence the signal integrity. The generated crosstalk causes signal over- and undershoots and glitches on adjacent wires as well as delay variations. For opposite switches on neighbouring wires due to the miller effect twice the coupling capacitance has to be charged. Since the parasitics increase with wire length especially global and semi-global wires are concerned. The power dissipation of wires is significantly increased by the coupling capacitances  $C_c$  [4] [5], which inserts a data dependency in power consumption, and the insertion of repeaters used to control the wire delay [6] [1]. The authors in [7] predict an even higher portion of power consumed by interconnects in the future.

The impact of DSM effects on signal integrity and power dissipation depends on the data to be transmitted and on the bus configuration which includes wire distance, repeater

\* This work is funded by the German Science Foundation DFG within the joint project VIVA under MU 1024/5-3.

insertion scheme, frequency and so on. Unfortunately the bus configuration is not fixed until the layout of the circuit. Therefore neither power dissipation nor signal integrity can be easily evaluated on high levels of abstraction. But on high levels of design abstraction the module structure and the communication architecture are chosen. Knowledge of the communication channel could be very helpful for such decisions. Therefore we present in this paper a parameterized, high-level HDL simulation model in order to evaluate signal integrity and power dissipation very early in the design cycle. Our model is based on SPICE simulations. The underlying SPICE model is parameterized as well and can be configured and executed by a set of scripts which allows a fast adaptation to varying technologies. A data base containing a set of relevant parameters is created and annotated to the simulation. The model delivers very accurate results for the high-level simulation (maximum error of 1.31%) while accelerating the simulation up to a factor of 900.

The remainder of the paper is structured as follows: Section 2 gives an overview of related work. In Sect. 3 the model for SPICE simulations is presented. After giving the structure the configuration and stimulation will be explained in more detail. Section 4 gives an overview over the high-level VHDL model. Experimental results are presented in Sect. 5. Section 6 will summarize the paper.

## 2 Related Work

A number of publications deal with transmission behaviour and power dissipation of global interconnects. A very extensive survey of power analysis and optimization techniques can be found in [8]. Starting from that a lot of work has been done in order to extend these models for DSM technologies. Some authors present closed form expressions to estimate coupling power [4] [9]. Tools such as presented in [5] are developed which analyze interconnect requirements and estimate power consumption. These techniques provide quite accurate results but restrict the analysis to a subset of sources of power consumption. While several techniques perform low-level estimations we refer to accurate high-level approaches since their computation time generally differs by several orders of magnitude.

A detailed model for global DSM interconnect is developed in [10]. The authors present an analytical model for energy dissipation which is based on lumped capacitors. It considers drivers and receivers but neglects power dissipation of repeaters which significantly contribute to total power dissipation. By comparing the results to SPICE simulations the necessity to take DSM parasitics into account and the accuracy of the model could be shown for the selected configuration.

A high-level power model for interconnect energy dissipation is presented in [11]. It is based on precomputing a look up table which contains power estimation values calculated by a SPICE simulation. The estimated power dissipation has a very small error. However, the simulations are restricted to a single bus configuration since the look up table is parameterized with respect to a single value: wire length. The exploration of different wire widths, wire distances and repeater insertion schemes is not considered. The closed form expressions presented in [12] consider repeaters and vias. Average wire lengths are estimated from previous designs and from a high-level RTL floor plan. The model concentrates on transition types instead of single transitions why the charge of

coupling capacitances is considered. Thereby the activity (self and coupling activity) is estimated instead of counted. This approach reduces computation time but also accuracy of the results because glitches are not considered and according to our investigations coupling activity depends very strongly on the application if data is not sent by an independent, identically distributed (i.i.d.) source.

The models mentioned so far estimate power dissipation quite accurately but do not explore the transmission behaviour of interconnects. In [13] this aspect of global DSM interconnect is addressed. The derived parameter file is based on low-level simulation lumping capacitance values.

The models described so far explore either power dissipation or transmission behaviour but not both. Therefore it is our intention to develop a high-level model which can be used to simultaneously evaluate signal integrity and power dissipation of global wires. In addition the results should be accurate while requiring a short computation time. The high-level model to be developed is based on SPICE simulations since these models are characterized by the highest accuracy. In order to explore a wide range of bus implementations the underlying SPICE model will be strongly parameterized and controlled by scripts in order to provide flexibility with respect to technology parameters.

### 3 SPICE Model

#### 3.1 Structure

The structure of our model of DSM wires is depicted in Fig. 1. Since capacitive coupling is restricted to directly adjacent wires a 3 wire model is sufficient for investigations in DSM effects [12]. We used distributed instead of lumped parasitics since only for very small distances  $\Delta x$  the wire is considered to behave homogeneously. Such a piece of wire with length  $\Delta x$  is referred to as a *segment*. It consists of a resistor  $R$ , an inductor  $L$ , a capacitor  $C$  to ground and a coupling capacitor  $C_c$  to each of the adjacent wires. We neglected the mutual inductance since it still has a minor influence on the transmission characteristics. A global wire is modeled by a series connection of segments. Therefore a segment was defined as sub circuit.

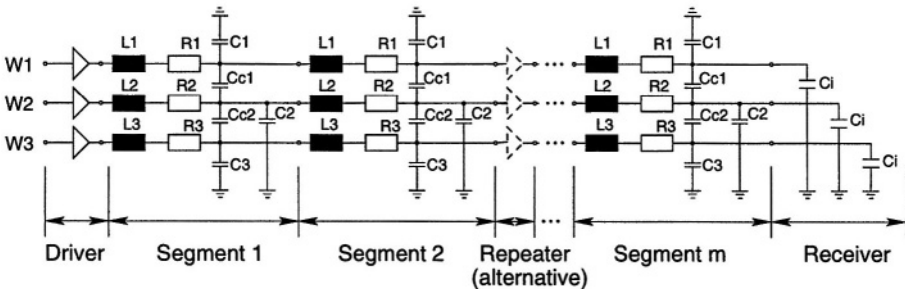


Fig. 1. Structure of the SPICE model for a DSM bus

Apart from the wire parasitics described so far also active elements such as driver and receiver have to be considered. In order to provide a signal with suitable edges at the end of long wires, repeaters with a parameterized distance can alternatively be inserted in the wire model. For both, drivers and repeaters strong buffers composed of PMOS and NMOS transistors are modeled. Their channel widths are dimensioned according to a 120 run UMC library such that the required saturation current is achieved. In order to provide balanced switching behaviour different values for NMOS and PMOS transistor were chosen which compensate for the different mobility of electrons and holes:  $\beta_n = \frac{4.31\mu m}{0.12\mu m}$  and  $\beta_p = \frac{10\mu m}{0.12\mu m}$ . The model of the receiver is very simple. From the electrical point of view only the capacitance which has to be charged is relevant. Therefore we restricted the receiver to its input capacitance  $CI$ .

### 3.2 Configuration and Selection of Input Stimuli

The SPICE model described so far allows the exploration of a single bus implementation with respect to signal integrity, transmission delay and power dissipation. However, parameters such as capacitance values, wire length, repeater distance and bus frequency have a large bandwidth. Our intention was to build a general model which covers a wide range of bus implementations. Therefore the structure of the model as well as the frequency range are controlled with a configuration file for the SPICE model. The configuration file contains all relevant parameters such as the number of segments per  $\mu m$  wire, whether to include repeaters and their distance, wire resistance, wire inductance, self capacitance and coupling capacitance and the frequencies to investigate in. For each parameter combination specified a transient analysis is conducted. If the parameters influence the SPICE netlist, a new netlist is constructed from the sub circuits which is followed by a simulation. Although we currently configured the netlist using typical parameters of a  $0.12\mu m$  technology, our approach allows a fast adaption to different processes.

The SPICE model is used as a vehicle to explore the response of a wire to an input transition. Therefore no extensive simulations are required. The investigation in a few relevant input transition vectors is sufficient. For the middle wire two possible scenarios exist: stable input ( $0 \rightarrow 0$  or  $1 \rightarrow 1$ ) or a transition ( $0 \rightarrow 1$  or  $1 \rightarrow 0$ ). These four states are combined with all ten transition possibilities of the outer wires which are depicted in Fig. 2. As a result the model is stimulated with 40 input vectors which minimizes the required simulation time.

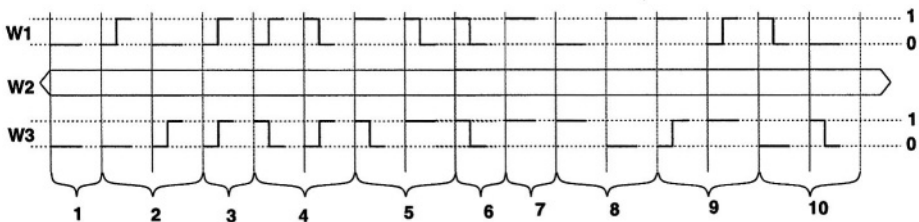


Fig.2. Stimuli for SPICE model simulation

## 4 VHDL Model

### 4.1 Principle

The purpose of our approach is to perform timing-efficient investigations in the transmission characteristics and power dissipation of global DSM communication channels on high levels of design abstraction with a very high accuracy. Therefore the SPICE model is explored for configurations of interest. Relevant parameters are extracted from SPICE simulations and stored in the *data base* before the high-level simulation. During high-level simulation these parameters are annotated to the VHDL model. The principle is shown in Fig. 3 and subsequently explained in more detail.

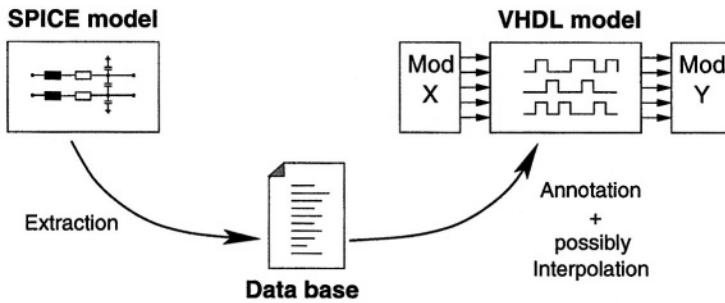


Fig.3. Principle of DSM annotation to VHDL model

### 4.2 The Data Base: Transfer of Relevant Parameters to High-Level Simulation

The *data base* plays a central role in our approach by transferring the high accuracy of the SPICE model to our high-level VHDL simulation model. The data base is created only once per technology. Therefore the time-intensive SPICE simulations are decoupled from the VHDL simulations. While stimulating the SPICE model with the set of input transition vectors the middle wire is observed with respect to a number of parameters such as the voltage level over the clock cycle, the timing of the signal at the end of the wire and the power dissipation. For each simulation run an entry is created in the data base which stores the extracted parameters. In order to recover the corresponding parameters all entries are classified with respect to wire length, frequency and input transition vector.

One of the parameters to annotate to the high-level model is the timing behaviour at the end of the wire during an input transition. Due to capacitive and inductive parasitics signal over- and undershoots can occur. However, the VHDL model is restricted to logic values: '0', '1' or Unknown ('X'). For a  $0 \rightarrow 1$  transition the voltage level continuously increases from ground to  $V_{dd}$ . The receiving buffer detects a logic value '0' as long as the level is below the threshold voltage  $V_{th,l}$ . For voltages above  $V_{th,l}$  the buffer starts to switch, both PMOS and NMOS transistor are short-circuited. The logic value is indeterminate until the threshold voltage  $V_{th,h}$  is achieved and a logic '1' can be detected.

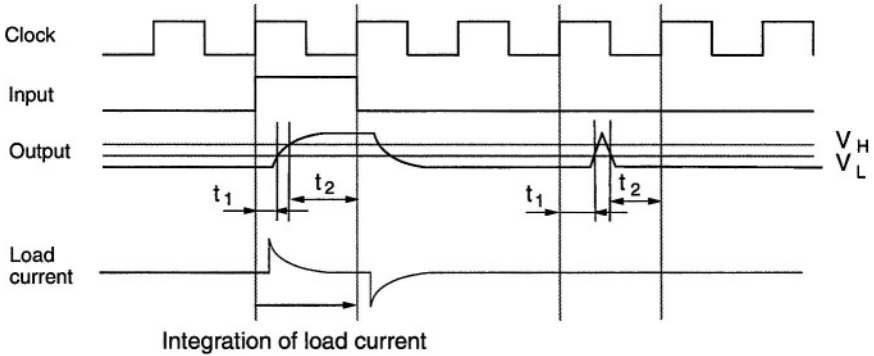


Fig. 4. Extraction of relevant parameters from SPICE simulation

We define two times  $t_1$  and  $t_2$  which describe relative to the clock edges the duration the signal has a stable, determinable logic value. As Fig. 4 shows, also spurious transitions due to cross coupling are captured.

A different parameter we are interested in is the power dissipation for each input transition vector. The power dissipated by the middle wire can be calculated from voltage and current. Since the current flow varies depending on the load state of the capacitors the current is integrated over the whole clock cycle. Thereby all sources of current flow are considered: load of mutual and self capacitances, load of input and internal capacitances of active elements, short circuit currents and leakage current. Even the power dissipated while causing glitches on adjacent wires is taken into account.

### 4.3 Parameter Annotation

During high-level simulation the SPICE information of the data base is annotated to the VHDL model. In order to select the corresponding set of parameters from the data base the entity of the VHDL model is parameterized regarding wire length, frequency and repeater distance. The required flexibility is provided by generics. The “ports”-section of the entity consists of a data input which is connected to the data to be transmitted over the bus, and a data output. The latter is delayed according to the times  $t_1$  and  $t_2$ .

The VHDL model comprises several packages which select the parameters from the data base and implement global signals for evaluation purposes. At simulation start the data base is read in. According to the generics specified for the simulation run the corresponding timing and power parameters for all input transitions are selected and stored in an array. In case no corresponding entry exists in the data base which means that the requested configuration was not explored by a SPICE simulation, the missing values are interpolated using the Inverse Distance Method.

During the simulation each wire is regarded in connection with its two adjacent wires. In each cycle the transitions of each three wires are detected and the corresponding entry is selected from the data base array. The timing of the middle wire is annotated to the output of the bus model whereby the indeterminable signal value between  $t_1$  and  $t_2$  is represented by the signal value ‘X’. Additionally the total power dissipation is increased



by the value for the occurred transition. For the evaluation of outer wires we assume that the bus is shielded by wires connected to ground which is quite common for global buses. The total power dissipation for the simulated data is displayed at the end of the simulation.

## 5 Experimental Results

We conducted a number of experiments in order to evaluate the accuracy of the VHDL model with the SPICE model. In a first experiment we compared the signal levels. Due to coupling effects the voltage level of the SPICE model over- and undershoots the supply voltage  $V_{dd}$  while the VHDL model can only represent logic '0' and '1'. However, as long as the voltage level is above  $V_{th,h}$  or below  $V_{th,l}$  the circuitry can unequivocally assign a logic value. Therefore we normalized the voltage levels recorded during SPICE simulations to their logic levels as depicted in Fig. 5(a). This normalized curve is shown in Fig. 5(b) together with the curve of the VHDL simulation while being stimulated with the same input patterns. As the figure indicates the signal levels of both models correspond quite good. Also the delay of the signals are quite similar. The annotation of the delay to the output signal of the VHDL model is shown as simulator plot in Fig. 6. The logic level 'X' is annotated to the output of the model between  $t_1$  and  $t_2$ . Both DSM effects are presented by the waveforms: the expected delay variation as a function of transitions on adjacent wires and glitches due to crosstalk. These effects would not be observable in a conventional high-level simulation.

In order to evaluate the accuracy of the power dissipation computed with the VHDL model we compared the figures with the results of the SPICE simulation. The VHDL model was configured for different bus implementations including such that have no corresponding parameters in the data base for which the values had to be determined by interpolation. The such prepared VHDL model was stimulated with a test data stream. The power dissipated by each wire is added over each clock cycle of the simulation

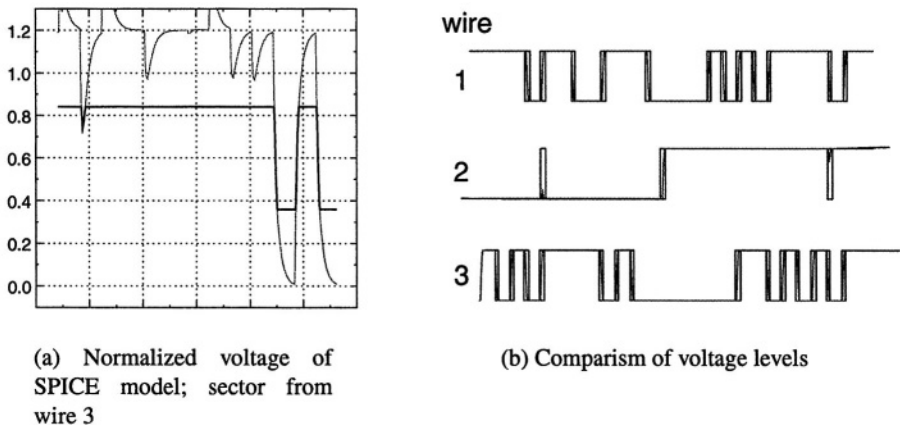


Fig. 5. Comparison of voltage levels: SPICE vs. VHDL

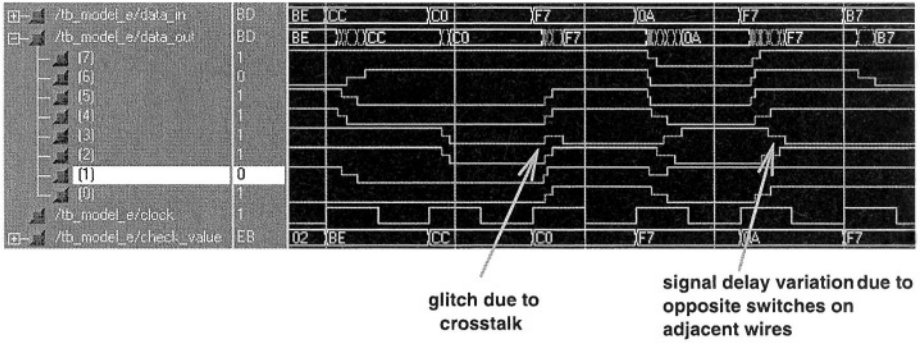


Fig. 6. Waveform with timing annotation extracted from VHDL simulator

Table 1. Comparism of dissipated energy

Configuration	Dissipated energy of		Calculated Error
	VHDL model [J]	SPICE model [J]	
$l=1000\mu\text{m}$ , $f=100\text{MHz}$	2.388942e-11	2.3938e-11	0.20%
$l=2000\mu\text{m}$ , $f=500\text{MHz}$	4.218873e-11	4.2164e-11	0.06%
$l=3500\mu\text{m}$ , $f=500\text{MHz}$ (interpolated)	6.918810e-11	6.9900e-11	1.02%
$l=4000\mu\text{m}$ , $f=500\text{MHz}$	7.738243e-11	7.8409e-11	1.31%
$l=10000\mu\text{m}$ , $f=500\text{MHz}$ , Repeater	2.621546e-10	2.6185e-10	0.12%

ran. At the end of the simulation the total energy consumption and the average power dissipation per clock cycle are displayed. For the comparism the SPICE model was prepared in analog fashion and stimulated with the same test data stream. During the simulation the total dissipated energy is determined by integrating the current flowing from the voltage source through all elements of the netlist. Table 1 summarizes the energy dissipation computed with both models for a number of bus configurations. The figures for VHDL and SPICE model correspond very good with each other. They show only minor variations even in the case of the interpolated values for the VHDL model. The maximum error found was only 1.31%.

In a next experiment we investigated in the simulation times required by both models. In general SPICE is considered to be very accurate but the more elements a netlist contains the longer the transient analysis lasts. In contrast to that the VHDL simulation is much faster although our model also requires a number of calculations to annotate timing and power consumption. For comparism purposes both models were constructed for a 32bit bus with different wire lengths. As presented in Fig. 7(a) the simulation time of the SPICE model increases quadratically with bus length while the simulation time of the VHDL model does not depend on wire length. For the VHDL model in each case a single annotation is done, the different lengths are realized by annotating figures from different parameter sets of the data base. In contrast to that the SPICE netlist increases due to the higher number of segments each of which realizes a piece of wire. Therefore the simulation of a fixed time period lasts longer for longer wires. The longer the wire

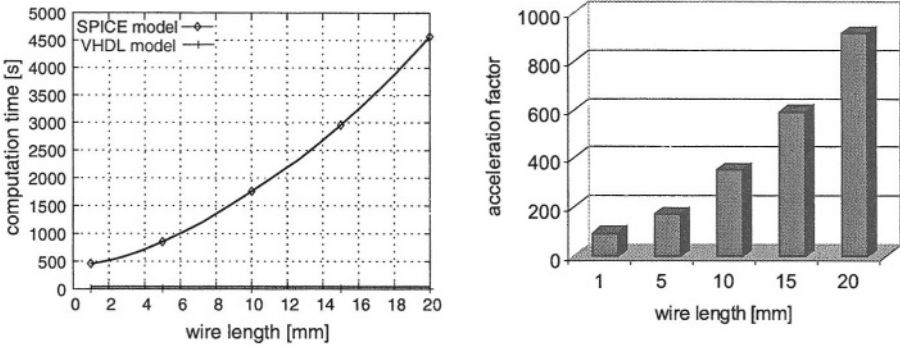


Fig.7. Simulation time: SPICE model vs. VHDL model

the more the simulation time is reduced by the VHDL model. As depicted in Fig. 7(b) the simulation can be accelerated by a hundred times for short wires of 1 mm and up to a factor of 900 for very long buses of 20 mm length.

In a last experiment we applied the model to a Businvert encoded system bus [14] and compared the power reduction for selected data streams (Figure 8). We calculated the power reduction using the self activity (first column of bars) and both: coupling and self activity (second column). For all streams the figures of the VHDL model correspond quite well with these considering self and coupling activity. For an i.i.d. source as *gauss* the figures with coupling activity only slightly differ from that without. In contrast to that for the other streams the figures differ strongly which indicates that the coupling activity is very data dependent.

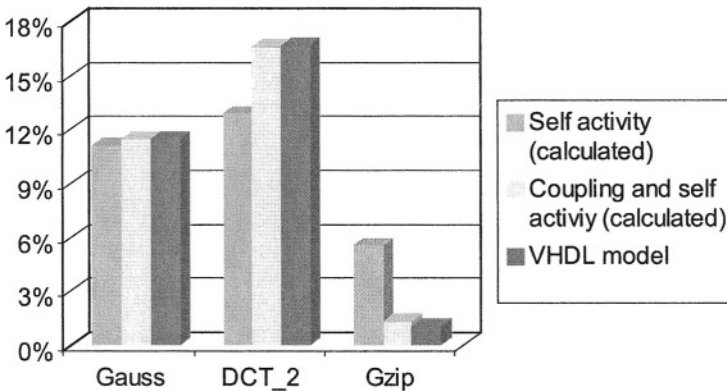


Fig. 8. Reduction in power dissipation of a BI transition-reduced system bus

## 6 Conclusions

The parameterized VHDL model presented in this paper allows the exploration of global DSM buses with respect to timing behaviour and power dissipation on high levels of design abstraction. A number of different DSM bus configurations can be explored in order to select the most suited one for a certain design. Since the model can even be combined with higher levels of design abstraction as long as they can be simulated a very early power and signal integrity estimation can be done.

The results confirm the high accuracy of the developed VHDL model which is based on figures extracted from SPICE simulations. The results achieved for both models exploring a wide number of bus configurations and test data streams correspond very good with each other. The maximum error is 1.31%. Due to the high level of abstraction the VHDL model accelerates the simulation time by several orders of magnitude while providing results with SPICE accuracy. Furthermore our approach can be very easily adapted to different technologies since the configuration of the SPICE model and the generation of the data base is controlled by a number of scripts. The simulation time for the SPICE model is thereby short due to the restriction to a very small number of input transition vectors for the triple of wires.

## References

1. D. Sylvester and K. Keutzer. Getting to the Bottom of Deep Submicron. In *ICCAD*, 1998.
2. T. Sakurai. Design challenges for  $0.1 \mu\text{m}$  and beyond. In *ASP-DAC*, 2000.
3. D. Sylvester and K. Keutzer. Impact of Small Process Geometries on Microarchitectures in Systems on a Chip. *Proceedings of the IEEE*, 89(4):467–489, April 2001 2001.
4. M. Kuhlmann and S. S. Sapatnekar. Exact and efficient crosstalk estimation. *IEEE Transactions on Computer-Aided Design*, 20(7):858–866, July 2001.
5. T. Uchino and J. Cong. An interconnect energy model considering coupling effects. *IEEE Transactions on Computer-Aided Design*, 21(7):763–776, July 2002.
6. P. Kapur, G. Chandra, and K. C. Saraswat. Power estimation in global interconnects and its reduction using a novel repeater optimization methodology. In *Design Automation Conference*, pages 461–466, June 2002.
7. D. Sylvester and C. Hu. Analytical modeling and characterization of deep-submicrometer interconnect. *Proc. IEEE*, 89(5):634–664, May 2001.
8. E. Macii, M. Pedram, and F. Somenzi. High-level power modeling, estimation, and optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(11):1061–1079, November 1998.
9. P. Heydari and M. Pedram. Interconnect energy dissipation modeling in high-speed ULSI circuits. In *ASP DAC*, pages 132–140, January 2002.
10. P. Sotiriadis and A. Chandrakasan. A bus energy model for deep sub-micron technology. *IEEE Transactions on VLSI*, 2002.
11. C. Taylor, S. Dey, and Y. Zhao. Modeling and minimization of interconnect energy dissipation in nanometer technologies. In *DAC*, 2001.
12. Pallav Gupta, Lin Zhong, and Niraj K. Jha. A high-level interconnect power model for design space exploration. In *ICCAD*, 2003.
13. X. Bai and S. Dey. High-level crosstalk defect simulation for system-on-chip interconnects. In *IEEE VLSI Test Symposium*, pages 169–175, March 2001.
14. Mircea R. Stan and Wayne P. Burlison. Bus-Invert Coding for Low-Power I/O. In *Transactions on VLSI Systems*, volume 3, pages 49–58, March 1995.

# Performance Metric Based Optimization Protocol

X. Michel, A. Verle, P. Maurine, N. Azémard, and D. Auvergne

LIRMM, UMR CNRS/Université de Montpellier II, (C5506),  
161 rue Ada, 34392 Montpellier, France  
{azemard, pmaurine, auvergne}@lirmm.fr

**Abstract.** Optimizing digital designs implies a selection of circuit implementation based on different cost criteria. Post-processing methods such as transistor sizing, buffer insertion or logic transformation can be used for optimizing critical paths to satisfy timing constraints. However most optimization tools are not able to select between the different optimization alternatives and have high CPU execution time.

In this paper, we propose an optimization protocol based on metrics allowing to characterize a path and to select the best optimization alternative. We define a way to characterize the design space of any circuit implementation. Then we propose a constraint distribution method allowing constraint satisfaction at nearly minimum area. This quasi optimal tool is implemented in an optimization tool (POPS) and validated by comparing the area necessary to satisfy delay constraints applied to various benchmarks (ISCAS'85) to that resulting from an industrial tool.

## 1 Introduction

Trade-off between speed, power and area can be achieved with circuit simulators and critical path analysis tools to modify iteratively the size of the transistors until complete constraint satisfaction [1-4]. More general speed-up techniques involve buffer insertion [5-6] and logic transformation [7]. If these techniques may be found efficient for speeding-up combinational paths they may have different impacts in the resulting power dissipation or area. Gate sizing is area (power) expensive and, due to the resulting capacitive loading effects, may slow down adjacent upward paths. This implies complex and iterative timing verifications. Buffer insertion preserves path interaction but is only efficient for relatively highly loaded nodes. To manage these alternatives it is necessary to evaluate and compare the performance of the different implementations. Without using any robust indicator, selecting between all these different techniques for the various gates of a library is NP complex and induces more iterative attempts which are processing time explosive.

A reasonable selection of speed-up technique must be based on a characterization of the available speed on a critical path, on the determination of the critical nodes and the characterization of the gate sensitivity to the sizing or buffering alternatives.

The main contribution of this paper is to define different metrics for path characterization, transistor sizing and buffer insertion, to be used as efficient indicators for characterizing the logic gates in terms of sensitivity to the sizing and buffering techniques.

Section 2 presents the elements used to define the optimization protocol. The optimization alternative with structure conservation is presented and validated in section 3. The proposed optimization method with buffer insertion is detailed and validated in section 4, in which the resulting optimization protocol is presented, before to conclude in section 5.

## 2 Optimization Protocol

Current path optimization tools [8] require large CPU times and too significant calculation computer resources to manage the complexity of nowadays developed circuits [9]. The uncertainty in parasitic capacitance estimation imposes to use many iterations or to consider very large safety margin resulting in oversized circuits.

### 2.1 Optimization Tool

As a solution to these drawbacks, we have developed an analysis and performance optimization tool based on an accurate representation of the physical abstraction of the layout (POPS: Performance Optimization by Path Selection) [10]. It gives facilities in analyzing and optimizing combinatorial circuit paths in submicronic technologies.

This tool allows to consider an user specified limited number of paths [11-12], for easy application and validation of the different path optimization criteria. The delay model implemented in this tool is based on an analytical representation of the timing performance, allowing to obtain for any logic gate, in its environment, an accurate evaluation of its switching delay and output transition time.

### 2.2 Delay Model

Real delay computation must consider finite input transition and I/O coupling [13]. We capture the effect of the input-to-output coupling and the input slope effect in the delay as

$$\begin{aligned} t_{HL}(i) &= \frac{v_{TN}}{2} \tau_{I_{NLH}}(i-1) + \left(1 + \frac{2C_M}{C_M + C_L}\right) \frac{\tau_{outHL}}{2}(i) \\ t_{LH}(i) &= \frac{v_{TP}}{2} \tau_{I_{NHL}}(i-1) + \left(1 + \frac{2C_M}{C_M + C_L}\right) \frac{\tau_{outLH}}{2}(i) \end{aligned} \quad (1)$$

where  $\tau_{I_{NHL,LH}}$ ,  $\tau_{outHL,LH}$  are the input and output transition time duration, respectively.  $C_M$  is the coupling capacitance between the input and output nodes, that can be evaluated as one half the input capacitance of the P(N) transistor for input rising (falling) edge, respectively or directly calibrated from SPICE simulation.

The general expression of the transition time has been developed in [14] as

$$\begin{aligned} \tau_{outHL} &= \tau \cdot S_{HL} \cdot \frac{C_L}{C_{IN}} & S_{HL} &= (1+k) \cdot DW_{HL} \\ \tau_{outLH} &= \tau \cdot S_{LH} \cdot \frac{C_L}{C_{IN}} & S_{LH} &= R \cdot \frac{(1+k)}{k} \cdot DW_{LH} \end{aligned} \quad (2) \quad (3)$$

where  $\tau$  is a time unit that characterizes the process.  $C_L$  and  $C_{IN}$  represent, respectively, the output load and the gate input capacitance.  $S_{HL,LH}$  represent the symmetry factor of the falling, rising edges.  $R$  represents, for identical load and drive capacitance, the ratio of the current value available in N and P transistors,  $k$  is the P/N configuration ratio and  $D_{wHL,LH}$  the gate logical weight defined by the ratio of the current available in an inverter to that of a serial array of transistors [14].

If eq.2,3 are quite similar to the logical effort expressions [4], they only represent the transition time expression. The delay is given by (1) that completely captures the input-to-output coupling and the input transition time effect on the delay. Using these expressions to define metrics for optimization, we always consider that the resulting implementation is in the fast input control range [14].

As shown from eq. (1-3) the delay on a bounded combinatorial path is a convex function and these expressions can easily be used to determine the best condition for path optimization under delay constraint.

By bounded combinatorial path we signify that the path input gate capacitance is fixed by the load constraint imposed on the latch supplying the path. This implies that the path terminal load is completely determined by the total input capacitance of the gates or registers controlled by this path. This guarantees the convexity of the delay on this path.

### 3 Optimization with Structure Conservation

The goal of gate sizing is to determine the optimum size for path delay constraint satisfaction at the minimum area/power cost. For that an essential parameter to be considered is the feasibility of the constraint imposed on the path. The target of this section is twofold: defining the delay bounds of a given path and determining a way for distributing a delay constraint on this path with the minimum area/power cost.

#### 3.1 Constraint Feasibility

This is the important section of this approach. Without indication on the feasibility of a constraint any iterative method may infinitely loop with no chance to reach a solution. For that, in order to verify the feasibility of a constraint, we explore the path optimization space by defining the max and min delay bounds ( $T_{max}$ ,  $T_{min}$ ) of this path. It is clear that if the delay constraint value is lower than the minimum delay achievable on this path, whatever is the optimization procedure, there is no way to satisfy the constraint without path modification. These bounds are of great importance in first defining the optimization alternative.

Theoretically and without gate size limitation, no upper delay bound can be defined a path. To define a pseudo-upper bound we just consider a realistic configuration in which all the gates are implemented with the minimum available drive.

The definition of the lower bound has been the subject of numerous proposals. For ideal inverters without parasitic loading the minimum is reached when all the inverters have an equal tapering factor that can be easily calculated from a first order delay representation [7,15]. Applying the explicit representation given in (1) to a bounded combinatorial path, the inferior delay bound is easily obtained by canceling the de-

rivative of the path delay with respect to the input capacitance of the gates. This results in a set of link equations

$$C_{IN}^2(i) = \frac{A_i}{A_{i-1}} \cdot (C_{IN}(i+1) + C_{par}(i)) \cdot C_{IN}(i-1) \quad (4)$$

where (i) specifies the rank of the gate,  $C_{par}(i)$  is the gate (i) output parasitic capacitance and the  $A_i$  correspond to the design parameters involved in (1,2).

As shown, the size of gate (i) depends on that of (i+1) and (i-1). This is exactly what we are looking for. Instead to solve the corresponding set of equations we prefer to use an iterative approach starting from a local solution defined with  $C_{IN}(i-1)$  equal to the minimum available drive ( $C_{REF}$ ). Then processing backward from the output, where the terminal load is known, to the input, we can easily determine an initial solution. Then by applying this solution in (4) we can reach, after few iterations, the minimum of delay achievable on the path. An illustration of the evolution of these iterations is given in Fig.1. We can easily verify that the final value,  $t_{min}$  is conserved whatever is the initial solution, ie the  $C_{REF}$  value.

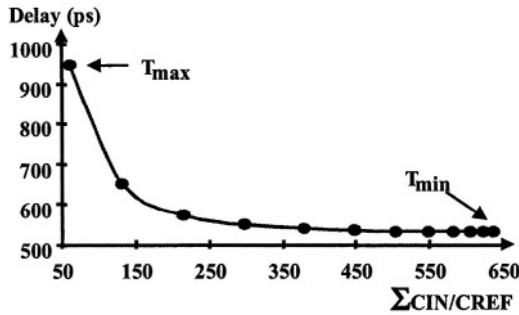


Fig. 1. Illustration of the sensitivity of the path delay to the gate sizing

This method has been implemented in POPS. Validation has been obtained by comparing on the longest path of different ISCAS'85 benchmarks (process CMOS,  $0.25\mu\text{m}$ ) the minimum delay value, obtained from the proposed method, to that reached by an industrial tool (AMPS from Synopsys). Fig.2 illustrates the resulting comparison that demonstrates the accuracy of the proposed method.

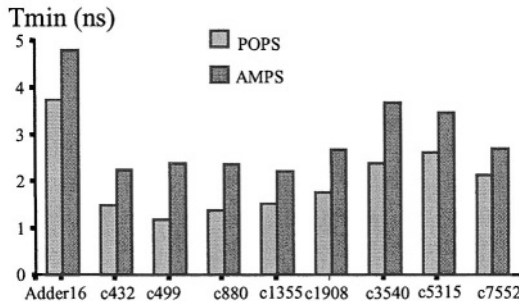


Fig. 2. Comparison of the minimum delay value (Tmin) determined with POPS and AMPS.



For any path the determination of the delay bounds gives facilities in verifying the feasibility of the constraint. For a delay constraint value higher than the minimum bound, the optimization alternative to be chosen is transistor sizing with structure conservation. Next step is to develop a fast technique allowing to efficiently distribute the constraint on the path.

### 3.2 Constraint Distribution: Constant Sensitivity Method

Several methods can be used. The simplest method is the Sutherland method [4], directly deduced from the Mead’s optimization rule of an ideal inverter array [15]: the same delay constraint is imposed on each element of the path. If this supplies a very fast method for distributing the constraint, this is at the cost of an over sizing of the gates with an important logical weight value

We propose a new method based on the gate sensitivity to the sizing, that can be directly deduced from (4), as illustrated in Fig.3.

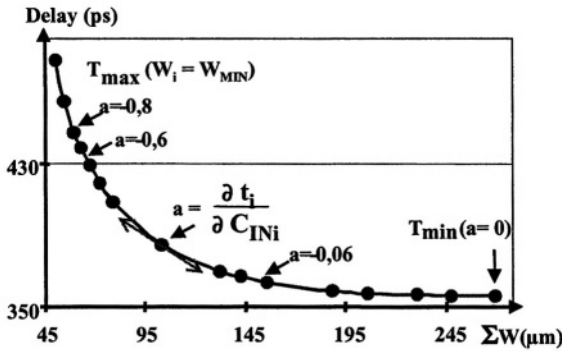


Fig. 3. Example of design space exploration on a 11 gate path, using the constant sensitivity method.

This Figure represents the variation of the path delay to the gate sizing. Each point has been obtained by imposing the same value of each partial derivative:

$$\frac{\partial T}{\partial C_{IN}(i)} = a \tag{5}$$

“a” = 0 corresponds to the minimum, varying the value of this coefficient from 0 to a high negative value allows the exploration of the full design space

Solving:

$$A_{i-1} \cdot \frac{1}{C_{i-1}} - A_i \cdot \frac{C_{i+1} + C_{Pi}}{C_i^2} = a \tag{6}$$

$$A_i \cdot \frac{1}{C_i} - A_{i+1} \cdot \frac{C_{i+2} + C_{Pi+1}}{C_{i+1}^2} = a \dots\dots$$

supplies a sizing solution for each value of the sensitivity coefficient “a”, at which corresponds a value of the path delay. Few iterations on the “a” value allows a quick satisfaction of the delay constraint.

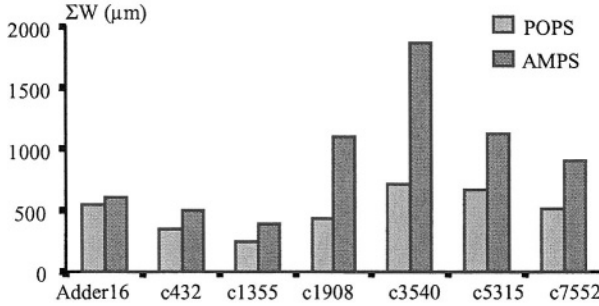


Fig. 4. Comparison of the constraint distribution methods on different ISCAS circuits.

Table 1. CPU time comparison in satisfying path delay constraint.

Circuits	Gate nb	POPS (ms)	AMPS (ms)
Adder16	99	159	23700
fp	14	19	6120
c432	29	29	9950
c499	29	30	9050
c880	28	29	9850
c1355	30	49	11400
c1908	44	49	11760
c3540	58	69	15890
c5315	60	90	19400
c6288	116	210	21920
c7552	47	69	16400

This method has been implemented in POPS and validated on different ISCAS circuits. In Fig.4 we compare the final area, given as the sum of the transistor widths ( $\Sigma W$ ), necessary to implement the critical path of each circuit under an identical hard constraint ( $T_c = 1.2T_{min}$ ), using POPS and AMPS. As shown the equal sensitivity method results in a smaller area/low power implementation.

In Table 1 we compare the CPU time necessary for AMPS and POPS in sizing under delay constraint different benchmarks. As illustrated the use of a deterministic approach in POPS, results in a two order speed up of the constraint distribution step, compared to the random approach used in standard tools (AMPS).

When the delay constraint has a smaller value than the minimum delay available, the only alternative is to modify the structure of the path.

## 4 Optimization with Buffer Insertion

The goal of this part is to define a way to select between sizing and buffer insertion. We just focus here on the buffer insertion method, that can be easily extended to the logic path modification. The problem is to determine, at minimum area cost, the best location to insert a buffer and the minimal sizing satisfying the delay constraint.

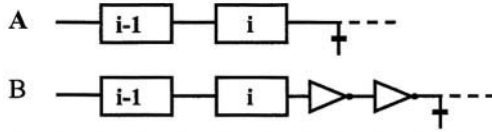


Fig. 5. Local buffer insertion

In Fig 5 we represent a path general situation where an overloaded node is guessed to be sped up by buffer insertion. The problem is to remove the guess by a metric directly determining for what level of load a gate switching speed can be improved. For that we compare the delay (1) of the A and B structures for determining at what fan out value ( $F = C_L/C_N$ ) the B structure becomes faster than A. This defines the “load buffer insertion limit” (Flimit). In a first step we use a local insertion method in which we conserve the size of gates (i-1) and (i) and just size the buffer (4), for minimizing the delay between the output of (i) and the terminal load.

The values of these Flimit are listed in Table 2. In the configuration of Fig.5, (i-1) is an inverter and we have considered the evolution of the limit with the gate (i). A complete characterization must involve all possibility of (i-1) gates and can be done easily following the same procedure. Validation of these limits has been obtained through Hspice simulations. As expected, greater is the logical weight of the gate, lower is the limit that may constitute a measure of the gate efficiency.

In fact the buffer insertion acts as a load dilution for the initial gate. In this case the size of this gate can be decreased. A complete consideration of the method involves using the predefined limits for critical nodes identification and then to distribute the delay constraint on the full path using the constant sensitivity method in order to preserve an area efficient gate sizing.

Validation of this approach is given in Table 3 where we compare the minimum delay obtained, from POPS, on the different ISCAS circuits using sizing and buffer insertion techniques. As shown, depending on the path structure significant minimum delay value improvement can be obtained with buffer insertion. Note that considering the delay sensitivity to the gate sizing (Fig.4), any minimum delay improvement on a path will result in a delay constraint satisfaction with smaller area.

This is illustrated in Fig.6 where we compare, on a 13 gate array, the path delay versus the area for the two methods: gate sizing (full line) and buffer insertion with global gate sizing (dotted line).

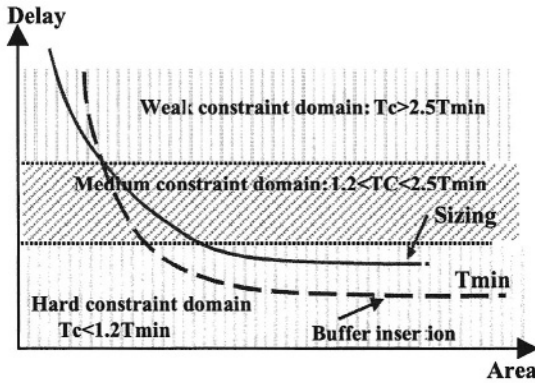
Three regions can be defined, a weak constraint domain where sizing is the best solution ( $T_c > 2.5T_{min}$ ), a medium constraint domain where buffer insertion is not necessary, but allows path implementation with area reduction ( $1.2T_{min} < T_c < 2.5T_{min}$ ) and a hard constraint domain ( $T_c < 1.2T_{min}$ ), where buffer insertion is the

Table 2. Fan out limit (Flimit) for a gate (i) controlled by an inverter.

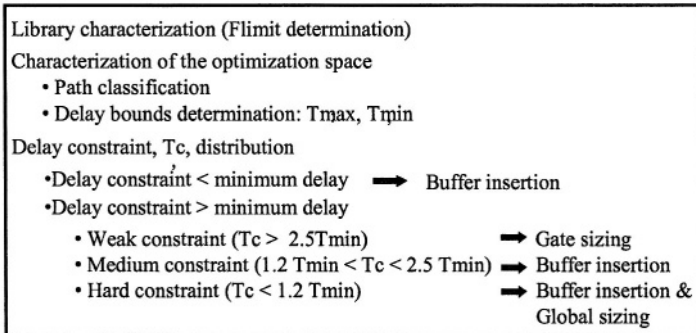
Gate i-1	Gate i	Calcul.	Simulation
inv	inv	5,7	5,9
inv	nand2	4,9	5,4
inv	nand3	4,5	5,2
inv	nor2	3,8	3,5
inv	nor3	2,7	2,5

**Table 3.** Comparison of sizing and buffer insertion techniques.

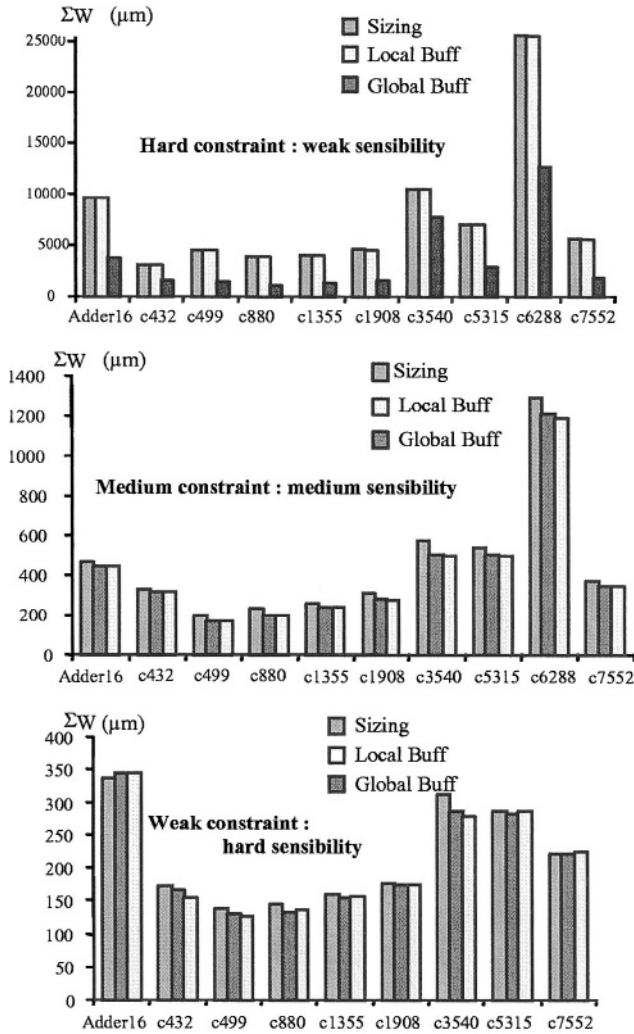
Circuits	Method	Tmin(ns)	Circuits	Method	Tmin(ns)
Adder	sizing	4,53	c1908	sizing	2,66
	buff	4,39		buff	2,32
	gain	3%		gain	15%
c432	sizing	2,22	c354	sizing	3,29
	buff	1,97		buff	3,21
	gain	13%		gain	2%
c499	sizing	1,79	c5315	sizing	3,57
	buff	1,64		buff	3,20
	gain	9%		gain	12%
c880	sizing	2,09	c6288	sizing	7,98
	buff	1,71		buff	7,74
	gain	22%		gain	3%
c1355	sizing	2,16	c7552	sizing	3,08
	buff	1,89		buff	2,60
	gain	14%		gain	18%



**Fig. 6.** Constraint domain definition.



**Fig. 7.** Optimization protocol



**Fig. 8.** Area saving in the different constraint domains for different optimization methods.

most efficient alternative. Note that these conditions defined with respect to the lower bound are circuit independent.

The resulting optimization protocol, (Fig.7), has been implemented in POPS for validation on the different ISCAS benchmarks. The comparison of the different steps is illustrated in Fig.8 where for three different delay constraint values (weak, medium, hard) we compare the path implementation area on the ISCAS circuits.

As shown, if for weak and medium constraints the different optimization methods are quite equivalent in terms of area, for hard constraint the buffer insertion with global sizing always results in important area saving.

## 5 Conclusion

Based on a realistic model for gate timing performance, we have defined metrics for selecting path optimization alternatives. We have proposed a method for determining the minimum delay,  $T_{min}$ , achievable on a path. Then we have defined, at gate level, the fan out limit for buffer insertion,  $Flimit$ .  $Flimit$  has been used to determine the path critical nodes and  $T_{min}$ , to select between sizing and buffer insertion alternatives. We have defined a gate sensitivity factor “ $a$ ”, to distribute the delay constraint, allowing path optimization at provably minimum area cost. These metrics have been used to define a general path optimization protocol that has been implemented in an optimization tool.

Validation on various benchmark circuits has demonstrated the validity of the defined boundaries for selecting between the different optimization alternatives.

## References

- [1] J. M. Shyu, A. Sangiovanni-Vincentelli, J. Fishburn, A. Dunlop, “Optimization-based transistor sizing” *IEEE J. Solid State Circuits*, vol.23, n°2, pp.400-409, 1988.
- [2] J.P. Fishburn, A.E. Dunlop, “Tilos: a posynomial programming approach to transistor sizing”, *Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 326-328, 1985.
- [3] S.S. Sapatnekar, V.B. Rao, P.M. Vaidya, S.M. Kang, “An exact solution to the transistor sizing problem for CMOS circuits using convex functions”, *IEEE trans. CAD*, pp. 1621-1634, November 1993.
- [4] I.E. Sutherland, B. Sproull, D. Harris, “Logical effort, designing fast cmos circuits”, Morgan Kaufmann Publishers, Inc., 1999.
- [5] S.R. Vemuru, A.R. Thorbjornsen, A.A. Tuszynski, “ CMOS tapered buffer”, *IEEE J. Solid State Circuits*, vol.26, n°9, pp.1265-1269, 1991.
- [6] Y. Jiang, S.S. Sapatnekar, C. Bamji, J. Kim, “Interleaving buffer insertion and transistor sizing into a single optimization”, *IEEE Trans. on VLSI*, pp. 625-633, December 1998.
- [7] P.G. Paulin, F. J. Poirot, “Logic decomposition algorithm for the timing optimization of multilevel logic”, *Proc. ICCD 89*, pp.329-333.
- [8] D. Singh, J.M. Rabaey, M. Pedram, F. Cathoor, S. Rajgopal, N. Sehgal, T.J. Mozden, “Power Conscious CAD tools and Methodologies: a Perspective”, *Proc. IEEE*, vol.83, n°4, pp.570-593, April 1995.
- [9] H.C. Chen, D.H.C. Du and L.R. Liu, “Critical Path Selection for Performance Optimization”, *IEEE trans. On CAD of Integrated Circuits and Systems*, vol. 12, n°2, pp. 185-195, February 1995
- [10] N. Azemard, D. Auvergne, “POPS : A tool for delay/power performance optimization ”, *Journal of Systems Architecture*, Elsevier, n°47, pp375-382, 2001.
- [11] S. Yen, D. Du and S. Ghanta., “Efficient Algorithms for Extracting the k Most Critical Paths in Timing Analysis”, *Design Automation Conference*, pp. 649-654, June 1989.
- [12] S. Cremoux, N. Azemard, D. Auvergne, “Path resizing based on incremental technique”, *Proc. ISCAS*, Monterey, USA, 1998.
- [13] K.O. Jeppson, “Modeling the Influence of the Transistor Gain Ratio and the Input-to-Output Coupling Capacitance on the CMOS Inverter Delay”, *IEEE JSSC*, Vol. 29, pp. 646-654, 1994.
- [14] P. Maurine, M. Rezzoug, N. Azémard, D. Auvergne “Transition time modeling in deep submicron CMOS” *IEEE trans. on Computer Aided Design*, Vol.21, n°1 1, pp.1352-1363, nov. 2002.
- [15] Mead, M. Rem, “Minimum propagation delays in VLSI” “, *IEEE J. Solid State Circuits*, Vol.SC17, n°4, pp.773-775, 1982.

# Temperature Dependence in Low Power CMOS UDSM Process

B. Lasbouygues<sup>1</sup>, R. Wilson<sup>1</sup>, P. Maurine<sup>2</sup>, N. Azémard<sup>2</sup>, and D. Auvergne<sup>2</sup>

<sup>1</sup> STMicroelectronics Design Department, 850 rue J. Monnet, 38926, Crolles, France  
{benoit.lasbouygues, robin.wilson}@st.com

<sup>2</sup> LIRMM, UMR CNRS/Université de Montpellier II, (C5506),  
161 rue Ada, 34392 Montpellier, France  
{pmaurine, azemard, auvergne}@lirmm.fr

**Abstract.** In low power UDSM process the combined use of reduced value of the supply voltage and high threshold voltage value may greatly modify the temperature sensitivity of designs, which becomes structure and transition edge dependent. In this paper we propose a model for determining the temperature coefficient of CMOS structures and defining the worst Process, Voltage and Temperature condition to be verified for qualifying a design. This model is validated on two 0.13 $\mu\text{m}$  processes by comparing the calculated values of the temperature coefficient of the performance parameters to values deduced from electrical simulations (Eldo). Application to combinatorial path gives evidence of the occurrence of temperature inversion that is structure and control condition dependent and must carefully be considered for robust design validation.

## 1 Introduction

With the shrinking of technologies, the contribution of the leakage current to the total power consumption has increased dramatically, and is becoming a critical problem in CMOS UDSM technologies. In low power applications, one possible solution to reduce the static to dynamic power ratio is to use higher threshold voltage ( $V_t$ ) devices. As expected this approach results in a design speed reduction. Moreover, the temperature sensitivity of the mobility and the threshold voltage [1,2] may reverse the temperature coefficient of the N and P transistor current. This completely modifies the critical characterization corners of a design (temperature inversion phenomenon). This effect is expected when the threshold voltage value approaches half  $V_{dd}$ . Unfortunately this configuration is becoming a standard for low power design in UDSM technologies (130nm and beyond).

In order to validate a design in a static timing analysis (STA) flow, it is necessary to validate the worst and the best case timing conditions. Without any temperature inversion phenomenon, it is well known that the worst case timings are observed for the worst case Process, the lowest Voltage value and the highest Temperature condition (PVT conditions). In low power design, due to the use of high threshold voltage devices, the worst case timing conditions are no more guaranteed at the highest temperature operating point.

Consequently to guarantee the correct behavior of a design it is necessary to verify various PVT conditions. This implies a standard cell library characterization for several (Three worst case for three different temperature values) PVT conditions, resulting in a huge amount of simulations for determining the standard tabular or polynomial representation of performance [3].

The main contribution of this paper is to propose a new representation of the timing performance of a CMOS library [4], to characterize the PVT conditions to be used for validating a design. The target of the proposed method is to deduce from a calibration of the model on one PVT condition, the evolution of each library cell performance at all the other PVT corners. This gives facility to the designer in predicting and estimating the evolution of the design performance with a minimum of simulation time.

The rest of the paper is organized as the following. In section 2, the analytical model generating the new representation of the timing performances is briefly introduced. Section 3 characterizes the temperature inversion phenomenon and defines the corresponding model. Section 4 applies this model to a data path of a circuit, designed in two different  $0.13\mu\text{m}$  technologies, in order to demonstrate the influence of the process conditions to the temperature inversion phenomenon.

## 2 Physical Timing Model

### A. Transition time modeling

Considering the transistor as a current generator [5], the output transition time of CMOS structures can be directly obtained from the modeling of the charging (discharging) current that flows during the switching process of the structure and from the amount of charge ( $C_L \cdot V_{DD}$ ) to be exchanged with the output node as:

$$\tau_{\text{outHL}} = \frac{C_L \cdot V_{DD}}{I_{N\text{Max}}} \quad \tau_{\text{outLH}} = \frac{C_L \cdot V_{DD}}{I_{P\text{Max}}} \quad (1)$$

where  $C_L$  represents the total output load (parasitic and active capacitance),  $V_{DD}$  the supply voltage value, and  $I_{N/P\text{Max}}$  the maximum current available in the structure. The key point here is to evaluate this maximum current, which depends strongly on the input controlling condition. For that, two main domains have to be considered: the fast input and the slow input range.

In the fast input range, the driving condition imposes a constant and maximum current value in the structure. The current expression can then be directly obtained from the Sakurai's representation [6]:

$$I_{N,P}^{\text{Fast}} = K_{N,P} \cdot W_{N,P} \cdot (V_{DD} - V_{TN,P})^{\alpha_{N/P}} \quad (2)$$

Where  $K_{N/P}$  is an equivalent conduction coefficient,  $V_{TN,P}$  are the threshold voltages of N/P transistors, and  $\alpha$  the velocity saturation index to be calibrated on the process.

Combining eq. 1 and 2 finally leads to the following expressions of the output transition time of a CMOS structure in the fast input range:

$$\tau_{\text{outHL}}^{\text{Fast}} = \tau_N \cdot S_{HL} \frac{C_L}{C_{IN}} \quad \tau_{\text{outLH}}^{\text{Fast}} = \tau_P \cdot S_{LH} \frac{C_L}{C_{IN}} \quad (3)$$

where  $S_{HL/LH}$  are the symmetry factors of the considered CMOS structure,  $C_L$  and  $C_{IN}$  its load and input capacitance respectively. The parameters:



$$\tau_N = \frac{C_{ox} \cdot L \cdot V_{DD}}{K_N \cdot (V_{DD} - V_{TN})^{\alpha_N}} \quad (4)$$

$$\tau_P = \frac{C_{ox} \cdot L \cdot V_{DD}}{K_P \cdot (V_{DD} - V_{TP})^{\alpha_P}}$$

are the process metrics for the transition time, since these parameters capture the sensitivity of the output transition time to both the supply and threshold voltage values.

### B. Propagation delay modeling

The delay of a CMOS logic gate is load, gate size and input slew dependent. Extending the work of [7] by using a velocity saturation index different from unity, the input slope and the input-to-output coupling effects can be introduced in the propagation delay as:

$$t_{HL} = \frac{\tau_{in}}{\alpha_N + 1} \left( \frac{\alpha_N - 1}{2} + v_{TN} \right) + \left( 1 + \frac{2C_M}{C_M + C_L} \right) \frac{\tau_{outHL}}{2} \quad (5)$$

## 3 Temperature Inversion: Characterization

The threshold voltage and the carrier mobility values are temperature dependent [1,2]. If both threshold voltage and carrier mobility values monotonically decrease when the temperature increases, the resulting impact on the timing performance of a design depends on the range of operating supply voltage values.

Considering (3,4), for an increase of temperature, it is obvious that a decrease of the  $V_{TNP}$  values results in a decrease of both the output transition time and the propagation delay, while a decrease of the carrier mobility induces an opposite variation of the timing performance.

It has been experimentally observed [8] that for a specific range of supply voltage value the temperature coefficient of the transition time becomes supply voltage dependent and may have negative values for  $V_{dd} < 2V_T$ . This is the temperature inversion phenomenon that could appear in low power processes, using high threshold voltage and reduced Vdd values.

Due to the resulting non-monotonic temperature impact on the design performance, it is clear, from eq. 3,5,6, that the STA flow must be performed with great care, in order to properly capture the worst performance of a design

**Table 1.** Threshold Voltage value at nominal conditions.

$V_{DD}=1.2V$	A Process	B Process
$V_{TN} (V)$	0.56	0.64
$V_{TP}(V)$	0.47	0.56

In order to illustrate this effect, let us study two different  $0.13\mu\text{m}$  processes dedicated to general purpose design (A) and to low power applications (B). The only difference between these two processes is on the threshold voltage values, as given in Table 1.

#### A. Transistor current temperature dependence.

Let us analyze the drain source current temperature sensitivity of an NMOS transistor under different temperature conditions. In Fig 1 we report the NMOS drain source current (process B) evolution, with the Vdd supply voltage value, for three different temperature values. We consider the maximum current available by imposing  $V_{GS}=V_{DD}$ .

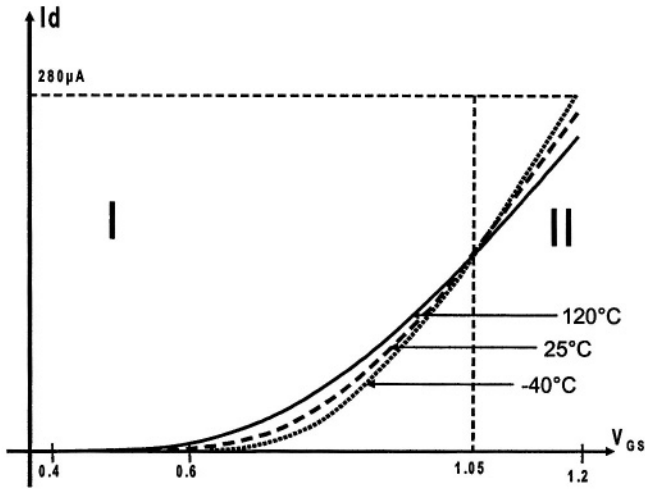


Fig. 1.  $I_d = f(V_{GS}=V_{DD})$  for NMOS (B process)

As shown, it is possible to define two supply voltage domains: I and II defined by the crossing points ( $V_{GS}=1.08\text{V}$ ) of the three  $I_d$  versus  $V_{GS}$  curves.

In domain I, the NMOS drain source current has a greater value at high temperature. In the domain II, the situation is reversed, the greatest value of  $I_d$  occurs at low temperature ( $-40^\circ\text{C}$ ). This demonstrates that depending on the supply voltage value the worst performance operating point can be obtained for high or low temperature values. The temperature coefficient of the current is supply voltage dependent and is cancelled, in this case, for the crossing point value  $V_{DD}=1.05\text{V}$ .

This evolution of the worst case PVT condition from high to low temperatures can be explained by the temperature variation of the  $V_{TN}$  and  $K_N$  values. Both parameters have a negative temperature coefficient with an opposite effect on the evolution of the transistor current value. At high supply voltage value the sensitivity of the K term (2) dominates, but for low Vdd value the variation of the Vdd-VT term becomes preponderant, reversing the global  $I_D$  current temperature coefficient.

One interesting point here is to note that at a supply voltage equal to the crossing point value the effect of  $V_{TN}$  variation on the current is balanced by the effect of  $K_N$  variation, meaning that around this voltage, the performance of a design is quasi temperature independent.

As a result, the inversion voltage value of N and P transistors are the key parameters for characterizing design temperature sensitivity or predicting possible temperature inversion. As an example, in Table 2 we give the inversion voltage obtained from simulated values of the N, P transistor current, on both A and B processes. As shown, considering the voltage corners defined for B process (1.08V and 1.32V), it appears that the NMOS device is not subject to T inversion phenomenon (negative current temperature coefficient), while the PMOS is always in inversion (positive current temperature coefficient).

**Table 2.** Inversion voltage and nominal threshold voltages for A and B process.

	A Process		B Process	
	NMOS	PMOS	NMOS	PMOS
<b>Inversion point</b>	0.92 V	1.08 V	1.05 V	1.38 V
<b>Vt nom.</b>	0.56 V	0.47 V	0.64 V	0.56 V

### B. Transition time temperature coefficient

In order to model and characterize a process with respect to the temperature inversion phenomenon, the  $\delta$  threshold voltage and Xk conduction factor temperature coefficients must be considered [9, 10] for determining, respectively, the sensitivities of  $V_T$  and K to the temperature:

$$Vt = Vtnom - \delta \cdot (\theta - \theta_{nom})$$

$$K = Knom \cdot \left( \frac{\theta_{nom}}{\theta} \right)^{Xk} \quad (6)$$

where the nominal values are chosen at 25°C and at (1.20V). Including these parameters into (2), supplies a derating coefficient *Der* that allows estimating the timing performance (4,5) of any cell operating under any PVT condition.

$$Der = \frac{\tau(V_{DD}, \theta)}{\tau_{nom}} = \left( \frac{\theta}{\theta_{nom}} \right)^{Xk} \cdot \left( \frac{V_{DD}}{V_{DDnom}} \right) \cdot \left( \frac{V_{DDnom} - V_{Tnom}}{V_{DD} - V_{Tnom} + \delta \cdot (\theta - \theta_{nom})} \right)^\alpha \quad (7)$$

In fact, as previously discussed this temperature sensitivity is directly obtained from the  $\tau_{NP}$  parameters which capture all the sensitivity of the output transition time and propagation delay to both the supply and threshold voltage values.

In order to validate this approach, we have made an application to the PMOS and NMOS transistor of B process. We first have extracted  $\tau_{NP}$  values from Eldo simulations for all PVT corners. Then, using the determined  $\tau_{NP}$  values, at the nominal point, as a reference we have calculated the  $\tau_{NP}$  values at the different PVT corners by applying the derating coefficient (7) with  $\delta_N=1.3mV/C$  and  $Xk_N=1.6$ ,  $\delta_P=1.6mV/C$  and  $Xk_P=1.2$  to the  $\tau$  expression (4). Results are reported in Table 3 and 4.

**Table 3.**  $\tau_p$  for each PVT corners and errors between calculated and simulated values.

B	$\tau_p$ (ps) Simulation			Error (%) Model vs. Simulation		
	1.08V	1.20V	1.32V	1.08V	1.20V	1.32V
233°K	36.6	26.9	21.9	-0.7	-2.2	-6.1
298°K	33.2	26.2	21.8	0.3	0	-1.1
398°K	31.2	25.6	21.9	-3.2	-0.7	0.9

**Table 4.**  $\tau_n$  for all PVT corners and errors between calculated and simulated values.

B	$\tau_n$ (ps) Simulation			Error (%) Model vs. Simulation		
	1.08V	1.20V	1.32V	1.08V	1.20V	1.32V
233°K	13.9	9.6	7.4	0.1	-4.4	-7.2
298°K	14.2	10.2	8.0	-1.0	0	0.1
398°K	14.5	11.1	9.1	-2.5	1.4	3.1

**Table 5.** Tau\_N and Tau\_P for A process.

A	Tau_P (ps) Simulation			Tau_N (ps) Simulation		
	1.08V	1.20V	1.32V	1.08V	1.20V	1.32V
233°K	27.3	22.0	18.6	9.5	7.6	6.5
298°K	27.6	23.1	20.0	10.9	8.2	7.0
398°K	27.9	24.1	21.4	11.1	9.3	8.2

As shown the calculated values are in good agreement with the simulated ones. The PMOS transistor of the B process is inverted over the whole supply voltage range and is temperature independent at 1.32V. At the contrary, the NMOS transistor is not impacted by the temperature inversion phenomenon as expected from Table 2 that gives its inversion voltage (1.05V) smaller than the worst-case supply voltage.

Finally, following the transition time and the delay equations (4,5), we can conclude that timing values (rising edge only) will suffer from temperature inversion. The worst case PVT must occur at the worst-case process defined by small supply voltage and low temperature values.

The main result obtained on this process, for the  $\tau_n$  values sensitivity, is that the falling edge is not impacted by the temperature inversion. At 1.08V the sensitivity to the temperature is very small but the worst case operating mode corresponds to the standard definition with high temperature and low voltage values.

As a summary, for the B process, both edges vary in an opposite way. The temperature coefficient of the transition time is negative (-0.15ps/°C) for the rising edge. The falling edge exhibits a small but positive temperature coefficient (+0.04ps/°C). In that condition the critical operating mode will be defined by the rising edge.

Considering the A process with its small threshold voltage values, this technology is supposed to be less influenced by the temperature inversion phenomenon.

As shown in Table 5, the worst case for the current is clearly at high temperature and low supply voltage for both edges. The PMOS study gives an inversion point around 1.08V. At 1.08V  $\tau_p$  exhibits a small variation between -40°C and 125°C (+0.021ps/°C). Following these results, the worst case for the drain current is at high temperature.

The main conclusion of this part is that the  $\tau_{N,P}$  coefficients can be used to characterize the drain current variations taking account all the PVT parameters. We have shown that the model differentiates both edges separately to take care of the N and P MOS dissymmetry. With respect to the supply voltage value, the threshold voltage appears to be the most sensitive parameter to control the temperature coefficient. A 5% variation of  $V_{T_0}$  can completely reverse the transistor temperature sensitivity.

## 4 Application to Path Timing Performance Sensitivity

In this part we study the temperature inversion phenomenon for cells and for a path designed in the two A and B technologies. We highlight the behavior of the propagation delay and the transition time.

### A. Cell level

It is clear from (4) that the temperature sensitivity of the transition time (3) is completely defined from that of the  $\tau_{N,P}$  parameter. Following this conclusion, only the rising edge for process B will exhibit a temperature inversion (Table 3). In Table 6 we have verified this result with Eldo for inverters implemented in A and B processes.

**Table 6.** Transition time at  $-40^{\circ}\text{C}$  and  $125^{\circ}\text{C}$ .

Vdd 1.08V	INV Process A		INV Process B	
	Rising	Falling	Rising	Falling
$-40^{\circ}\text{C}$	52ps	48ps	78ps	55ps
$125^{\circ}\text{C}$	65ps	58ps	70ps	58ps

Considering the delay expression (5), two separate coefficients have to be considered. The first parameter is input slope and threshold voltage dependent. The second one is the output transition time, which is directly connected to  $\tau_{N,P}$ .

For the A process and the rising edge,  $\tau_P$  is not in temperature inversion, so the second term of the delay is not impacted. But if we consider the first part of the formula, and the fact that the threshold voltage increases when the temperature decreases (Table 7), for a given input slope, this part of the delay has a greater contribution for low temperature and can cause temperature inversion. And the greater the input transition time value is, the more important the phenomenon is. The main result is that the delay can be in temperature inversion even if the drain current and transition time sensitivity are not modified.

**Table 7.**  $V_t$  for PMOS in A process.

Temp.	$V_t_P(\text{V})$
$-40^{\circ}\text{C}$	0.61
$25^{\circ}\text{C}$	0.56
$125^{\circ}\text{C}$	0.48

### B. Path level

Validation is obtained by evaluating the temperature sensitivity of a data path designed in the two A and B technologies, and comparing the results with electrical

simulations (Eldo's). The data path is constituted of inverters, nand, nor, AO, buffer and multiplexer for a total of 10 cells and an input slope of 800ps. The characterizations are done for a worst-case process at 1.08V for  $-40^{\circ}\text{C}$ ,  $25^{\circ}\text{C}$  and  $125^{\circ}\text{C}$ .

**Table 8.** Path delay at several temperatures.

Temp. ( $^{\circ}\text{C}$ )	Delay (ns) Simulation	Delay (ns) & error model
-40	2.178	2.261 (+4%)
25	2.263	2.231 (-2%)
125	2.178	2.073 (-5%)

From the simulated values, given in Table 8, we observe on this path an almost temperature independent delay, the worst case occurring around  $25^{\circ}\text{C}$ , with a very small temperature coefficient ( $0.01\text{ps}/^{\circ}\text{C}$ ). This is a consequence of the reverse temperature coefficient of the edges resulting in an overall compensation.

Let us focus on the first cell (inverter) of the path (controlled by the slower input ramp). We can see in Table 9 that, as generally expected, the transition time has a positive temperature coefficient. However, due to the threshold voltage temperature sensitivity ( $V_t, \tau_{\text{IN}}$ , eq.5), the delay exhibits a negative coefficient.

**Table 9.** Timing of the first cell, controlled by a 800ps input ramp.

Temp. ( $^{\circ}\text{C}$ )	Delay (rising)	Slope (rising)
-40	378 ps	316 ps
25	362 ps	341 ps
125	333 ps	372 ps

This highlight the rule defines at the cell level study.

Considering the second process (B) with a greater threshold voltage value, on the same path (same cells, same drives), this technology appears deeper inverted. The transistor current study demonstrates that the rising edge is fully inverted (delay and transition time) and the falling edge is almost temperature independent. On a path, where falling and rising delays add up, the worst case is for a worst process, at low supply voltage and temperature values. The results are summarized in the Table 10. As shown the calculated values of delay (5) are in good agreement with the simulated one. This demonstrates the ability of the model in characterizing the temperature inversion phenomenon.

**Table 10.** Path delay at several temperatures.

Temp. ( $^{\circ}\text{C}$ )	Delay (ns) Simulation	Delay (ns) & error model
-40	3.197	3.381 (+5.7%)
25	3.032	3.129 (+3.2%)
125	2.868	2.794 (-2.6%)

To summarize, for the B process that is fully inverted, the worst case is obtained for a PVT defined at the lowest temperature. For the A process, the delay temperature

dependence is input slope dependent. Despite the nearly temperature independence of the process, the path temperature sensitivity depends on its structure and must be carefully considered by controlling slope for defining the critical configuration.

## 5 Conclusion

The determination of robust PVT corners is of fundamental importance in actual low power UDSM process, for validating the critical configuration of designs. Using a physical and analytical model of the timing performance of CMOS structures, we have demonstrated that it was possible to accurately model the temperature sensitivity of the transition time and delay. We have clearly identified a process parameter,  $\tau_{NP}$ , that can be used as a robust metric to define the temperature sensitivity of CMOS structures.

Validations have been given on two specific  $0.13\mu\text{m}$  processes by comparing the temperature sensitivity of timing performance, calculated with the model to values obtained from electrical simulations.

We have shown the possibility on a specific process to get transition edges with opposite temperature coefficient (temperature inversion) resulting in a complete modification of the definition of the design validation critical conditions. This temperature inversion sensitivity has been shown to be input slope and path structure dependent.

This model appears as a powerful help for designers in characterizing, with few simulations, the timing performance of designs for any PVT corner.

## References

- [1] Changhae Park et al, "Reversal of temperature dependence of integrated circuits operating at very low voltages", Proc. IEDM conference, pp.71-74, 1995.
- [2] S.M. Sze, "Physics of semiconductor devices", Wiley ed. 1983.
- [3] Synopsys Inc., "Scalable Polynomial Delay And Power Model", Rev 5, October 2002.
- [4] B. Lasbouygues, J. Schindler, S. Engels, P. Maurine, N. Azemard, D. Auvergne, "Continuous representation of the performance of a CMOS library", European Solid-State Circuits, ESSIRC'03 Conf. on 16-18 Sept 2003.
- [5] P. Maurine, M. Rezzoug, N. Azemard, D. Auvergne, "Transition time modeling in deep submicron CMOS" IEEE Trans. on Computer Aided Design, vol.21, n11, pp.1352-1363, nov.2002.
- [6] T.Sakurai and A.R. Newton, "Alpha-power model, and its application to CMOS inverter delay and other formulas", J. Solid State Circuits vol. 25, pp.584-594, April 1990.
- [7] K.O. Jeppson, "Modeling the Influence of the Transistor Gain Ratio and the Input-to-Output Coupling Capacitance on the CMOS Inverter Delay", IEEE JSSC, Vol. 29, pp. 646-654, 1994.
- [8] Daga JM, Ottaviano E.; Auvergne D, "Temperature effect on delay for low voltage applications", Design, Automation and Test in Europe, Proc., pp. 680-685, 23-26 Feb. 1998.
- [9] J.A. Power et al, "An Investigation of MOSFET Statistical and Temperature Effects", Proc. IEEE 1992 Int. Conference on Microelectronic Test Structures, Vol. 5, March 1992.
- [10] A. Osman et al, "An Extended Tanh Law MOSFET Model for High Temperature Circuit Simulation", IEEE JSSC, Vol. 30, No2, Feb. 1995.

# Yield Optimization by Means of Process Parameters Estimation: Comparison Between ABB and ASV Techniques

Mauro Olivieri, Mirko Scarana, Giuseppe Scotti, and Alessandro Trifiletti

Dept. of Electronic Eng, "La Sapienza" University of Rome,  
Via Eudossiana 18, I-00184, Roma, ITALY.  
Phone: +39 06 44585679  
trifiletti@die.uniroma1.it

**Abstract.** In this work a novel approach to optimize digital integrated circuits yield with regards to speed, dynamic power and leakage power constraints is presented. The method is based on process parameter estimation circuits and adaptive body bias (ABB) and/or adaptive supply voltage (ASV) performed by an on-chip digital controller. The associated design flow allows to quantitatively predict the impact of the method on the expected yield in a specific design. We present the architecture scheme, the estimation circuits used, the proposed design flow. An application case study, referring to an industrial  $0.13\ \mu\text{m}$  CMOS process is used to compare ABB and ASV techniques. It is shown that ABB technique is particularly effective at high working temperatures and allows a stronger yield improvement with respect to the ASV technique. In the presented study, yields originally below 18% are improved to 83 % by means of ABB alone and to 97% using ABB and ASV jointly.

## 1 Introduction

Traditionally, the goal of digital circuit designers has always been to obtain the best trade off between speed (in terms of achievable clock frequency) and power consumption. Recently, the requirement of low power consumption has become the dominant constraint for microprocessors design in mobile environments. Moreover, as technologies continue to scale, power density has become an even more significant concern due to its impact on packaging costs. At circuit level, the most important components of power dissipation are dynamic power – due to CMOS switching activity – and leakage power – due to parasitic currents in switched-off CMOS devices – and it is expected that leakage power will become comparable to dynamic power consumption in a near future[1].

Process parameters variations at the different production levels (i.e. chip, die, wafer, lot and run) can have a strong impact on circuit performance – in terms of speed, dynamic power and leakage power – possibly compromising the production of most innovative digital IC designs due to an unacceptably low yield. To achieve a satisfying yield, the designer often refers to process corner values for assessing expected performance figures. However, extreme worst-case designs cannot meet the performance and power requirements of VLSI chips and therefore cannot sustain the trend



predicted by the International Technology Roadmap for Semiconductors (ITRS)[2]. A better trade-off between performance and yield can be obtained using a design approach based on statistical tools such as Monte Carlo analysis, instead of a corner analysis based one.

The yield optimization problem has been addressed in [3], and various approaches, such as design centering [3] or DOE [4], have been proposed to find CAD-oriented solutions.

Previous research works have targeted controlling body bias voltage to maximize specific system-level macro parameters (such as the sustainable clock frequency) or minimizing the leakage power [5], [6], [7]. In [8], compensation of parameters variations across different dies, is achieved applying the body bias that maximize the die frequency, subject to a power constraint.

Dynamic and continuous adjustable supply voltage schemes have been proposed in [9] and [10]. Recent research works in digitally controlled variable supply regulation circuits ([11], [12]) allow to practically apply ASV schemes in a dynamic and continuous fashion. In [14], the authors show that the simultaneous use of adaptive body bias and adaptive supply voltage is much more effective in reducing power consumption in high performance processors than using one of the above method alone. In [14] adaptive  $V_{DD}$  and body bias are jointly used to reduce the impact of parameter variations on frequency, dynamic power and leakage power of microprocessors.

Almost all the existing research in ASV schemes was focused on optimizing a design in terms of its power consumption under a set of timing constraints; in [2] the impact and applicability of ASV schemes on post silicon tuning, and thus, yield improvement under the presence of process variations is discussed.

In this work we focus on a control architecture scheme based on process parameter estimation circuits and active control of body bias and/or supply voltage and compare the use of ABB and ASV in optimizing yield by using such approach.

In the following, section II introduces the control architecture and the yield oriented design flow. In section III the proposed estimation circuits are discussed. Section IV reports the application of the proposed approach to a case study involving the yield oriented design of a 15 stage microprocessor critical path, with the aim of comparing the use of ABB, ASV and ABB+ASV techniques within the proposed design flow. Finally in section V some conclusions are reported.

## 2 Approach to Yield Prediction and Optimization

Our approach consists in correcting the effects of die-to-die parameter variations estimating their actual values (by means of estimation circuits placed very close to the critical part of the circuit) and using active control of body bias and/or of supply voltage in order to control circuit merit figures, as detailed in the following. The proposed control architecture is depicted in Fig. 1a: a set of sensing circuits allows us to estimate the current value of critical process parameters such as threshold voltage and working temperature; information provided by the sensing circuits are sent to the ADC and then to the digital controller. For a given circuit, the values of the optimal body bias and/or of the supply voltage that maximise performance for a given set of process parameters/temperature values are stored into the controller PROM. The procedure to find the values of the optimal body bias and/or of supply voltage starts from

the assigned constraints for clock frequency, dynamic power and leakage power and requires finding the optimal values for different values of the threshold voltages ( $V_{TN}$ ,  $V_{TP}$ ) and for the operating temperature. This can be done in the design phase by means of a spice simulation of the critical path: a process/temperature grid is generated (as in figure 1b), each cell containing the body bias and/or power supply which best meets the given speed and power constraints.

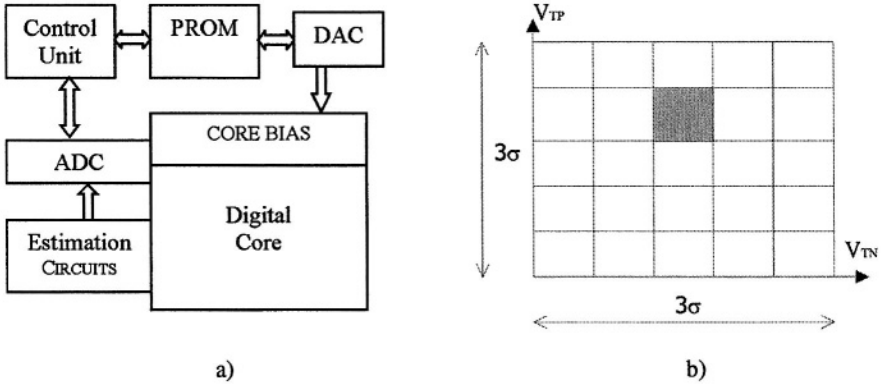


Fig. 1. Scheme of the control architecture (a); Grid of cases for optimization (b).

### 3 Estimation Circuits

#### 3.1 Estimation Circuits for Process Parameter Variations

Here we discuss the circuits for the estimation of process parameters variation, in particular die-to-die variations of the threshold voltage in both n-channel ( $V_{TN}$ ) and p-channel ( $V_{TP}$ ) devices. The topology of the circuit which allows the estimation of the threshold voltage  $V_{TN}$  of n-channel MOSFETs is reported in Fig. 2a.

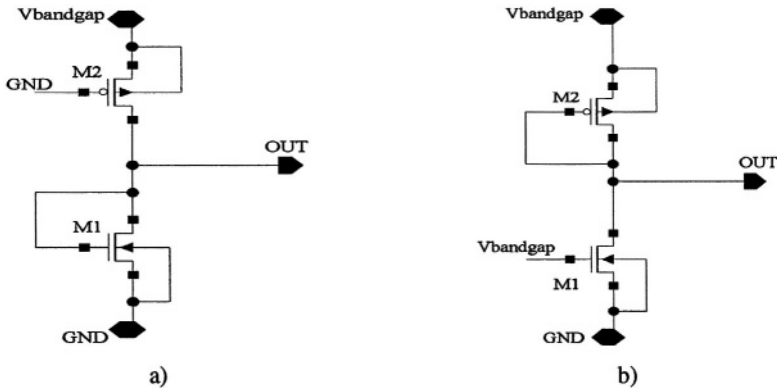


Fig. 2. a) Topology of  $V_{TN}$  estimator, b) topology of  $V_{TP}$  estimator

Using a quadratic model for the drain current [and denoting  $\Delta V_{TN}$  and  $\Delta V_{TP}$  the variations of the threshold voltage of p-channel and n-channel devices respectively and  $\Delta V_{BG}$  the variations of the bandgap reference voltage,] and the hypothesis  $K_N \gg K_P$ , it can be shown that

$$\Delta V_{OUT} \cong \Delta V_{TN} \quad (1)$$

The topology of the circuit which allows the estimation of the threshold voltage  $V_{TP}$  of p-channel MOSFETs is reported in Fig. 2b. It can be shown that, in the hypothesis  $K_P \gg K_N$ :

$$\Delta V_{OUT} \cong \Delta V_{TP} \quad (2)$$

Table 1 reports the threshold voltages  $V_{TN}$  and  $V_{TP}$  for different process corners and the corresponding output voltages  $V_{OUTN}$  (circuit in Fig. 2a) and  $V_{OUTP}$  (circuit in Fig. 2b) (normalized with respect to the nominal values) at  $T=60^\circ\text{C}$  temperature.

**Table 1.**  $V_{out}$  vs.  $V_T$ .

Process corner	$V_{TN}/V_{TN} (TT)$	$V_{OUTN}/V_{OUTN} (TT)$	$V_{TP}/V_{TP} (TT)$	$V_{OUTP}/V_{OUTP} (TT)$
FS	0.85	0.88	0.7	0.74
FF	0.92	0.95	0.85	0.88
TT	1	1	1	1
SS	1.07	1.05	1.14	1.11
SF	1.15	1.13	1.3	1.26

### 3.2 Temperature Sensor

The temperature sensor is based on a PTAT current reference generator; the adopted topology is reported in Fig. 3a: the delta  $V_{GS}$  generator performs the temperature sensing and the output resistor converts the current into a voltage which can be sent to the ADC. The voltage at the node  $V_{ptat}$  for the circuit in Fig. 3b can be expressed as

$$V_{ptat} = \frac{\Delta V_{GS}}{R_1} R_2 = \left( \sqrt{\frac{K_2}{K_1}} - 1 \right) \frac{R_2}{R_1} V_{GS2}(T) \quad (3)$$

Equation (3) shows that the sensitivity of the circuit depends on the  $R_2/R_1$  ratio and is therefore insensitive to die-to-die process resistance value variations.

The simulated sensor response is depicted in Fig. 3b and shows a good linearity in the  $-10^\circ\text{C}$  to  $120^\circ\text{C}$  temperature range. The output dynamic range for the discussed sensing circuits is about 0.2 V, therefore a 5 bit resolution [5] and an accuracy of few mV for both the ADC and the DAC are sufficient for the proposed application.

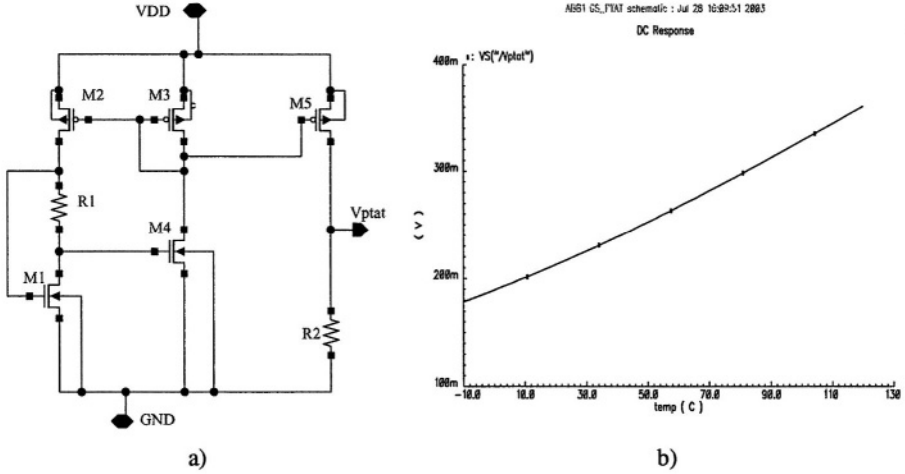


Fig. 3. Schematic (a) and response (b) of the PTAT temperature sensor

## 4 Design Case Study and Comparisons

### 4.1 Die-to-Die Parameter Variation Compensation

In order to compare the effectiveness of ABB and ASV in optimizing circuit yield by means of the proposed approach, we have applied it to a circuit block extracted from a microprocessor critical path, meant to model the effects of the proposed control scheme and design flow on the yield of a real microprocessor design. Our circuit under test (CUT) consists of 15 CMOS gates; similar test circuits have been used in [8]; the figures of merit considered in the experiment were the oscillation frequency and the dynamic power consumption of the CUT configured as a ring oscillator, and the leakage power consumption of the CUT in static conditions.

The circuit was designed in the Cadence IC 4.4.6 environment referring to a 0.13  $\mu\text{m}$  CMOS process with a 1.3 V breakdown voltage: transistor level simulations (backannotated with parasitic capacitances extracted from the layout) were used to compute the desired figures of merit. In order to perform yield estimation and optimization we used the statistical model of MOS devices which is available in the design kit provided by the foundry. This model is able to take into account both die-to-die and within-die process parameter variations at different operating temperatures.

The yield specifications used in the yield optimization are reported in table 2: the operating frequency target has been set according to system specifications, while the power constraints have been derived assuming the power density limit of  $10 \text{ W/cm}^2$  and a standby leakage power of  $0.5 \text{ W/cm}^2$ , which are typical values for low power microprocessors in mobile systems [14]. Since the threshold voltage is a critical parameter because of its impact on both frequency and power and since the operating temperature strongly affects the leakage power, we have used the  $V_{TN}$ ,  $V_{TP}$  estimators and the PTAT sensor discussed in section III in order to detect process parameters and temperature variations.

**Table 2.** Yield specifications

Fck	> 1100 MHz
Average Dynamic power	<1.4 $\mu$ W
Leakage power	< 49 $\mu$ W

A grid of cases (see figure 1b) has been considered in the study: the  $3\sigma$  variability range of  $V_{TP}$  and  $V_{TN}$  has been divided into 5 bins according to the process corners of the technology; the variability range for operating junction temperature has been assumed to be 0-120 °C and 0°C, 60°C and 120 °C have been the considered temperatures for a total of 75 regions.

In the ABB scheme, reverse body bias is applied in order to reduce power consumption (and the oscillation frequency), while forward body bias is applied to increase oscillation frequency (and power consumption as well). The maximum allowed reverse body bias voltage in the target technology is limited by breakdown considerations and is set to -0.5 V. Maximum allowed forward body bias voltage is limited by the forward biasing of the drain-bulk junction and is set to 0.25 V in our design. In the ASV scheme, maximum supply voltage is limited by breakdown considerations and has been set to be 1.2 V in our design, while the minimum supply has been set to 0.8 V.

A procedure has been developed and implemented in Cadence IC 4.4.6 environment (by means of an ocean script), which allows to automatically perform a set of simulations for different values of process parameters and operating temperatures and for different values of body bias and supply voltage. Simulation data have been used to find the optimal values for the controlling variables for each of the 75 regions, using an ABB scheme, an ASV scheme and a ABB+ASV scheme; the optimal values of the controlling variables have been stored in the digital controller PROM.

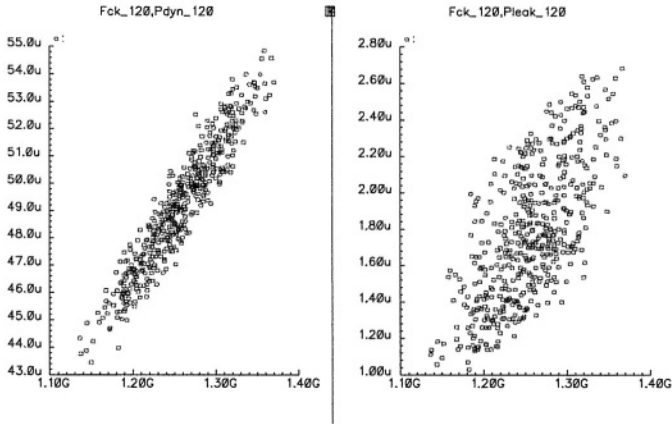
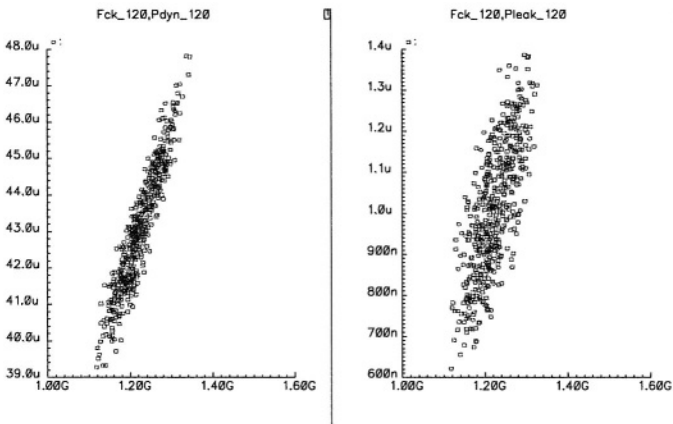
Circuit yield was then estimated by means of MonteCarlo simulations for each of the three considered control schemes; as a first step, only die-to-die parameter variations were taken into account,. Table 3 reports the partial and cumulative yield for the circuit designed without any control, and for the circuit designed with the proposed approach, using an ABB, an ASV and an ABB+ASV control scheme. Fig. 4 and 5 report the scatterplots of Dynamic power and leakage power as a function of operating frequency, for the circuit without the controller and for the circuit designed with the proposed approach and the ABB+ASV scheme, both at 120 °C operating temperature.

## 4.2 Effects of Within-Die Parameter Variations

The effects of within-die fluctuations increase as the channel length is shortened, so that circuit design methodologies aimed at suppressing the effects of both die-to-die and within-die parameter variations are required for the future technology generations [1]. The relative impact of die-to-die and within-die parameter variations depends on the number of critical paths per die and on the number of gate stages in the critical path [5]. If a controller and a set of estimation circuits are used for each critical path, the effects of within-die parameter variations are minimized and the results reported in section 4.1 are a good approximation of the effective yield enhancement.

**Table 3.** Yield estimated at 120°C considering only die-to-die parameter variations.

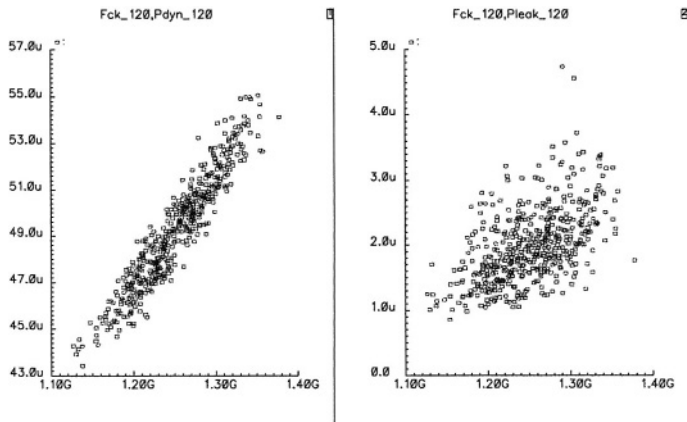
	No Control	ASV	ABB	ABB+ASV
Y_Fck	100 %	97.60 %	100 %	100 %
Y_Pdyn	46.80 %	76.80 %	98.80 %	100 %
Y_Pleak	17.60 %	18.40 %	98.60 %	100 %
Y_tot	17.40 %	18.20 %	98.40 %	100 %

**Fig. 4.** Scatterplots without the controller at  $T=120^{\circ}\text{C}$ **Fig. 5.** Scatterplots with the ABB+ASV control scheme at  $T=120^{\circ}\text{C}$ 

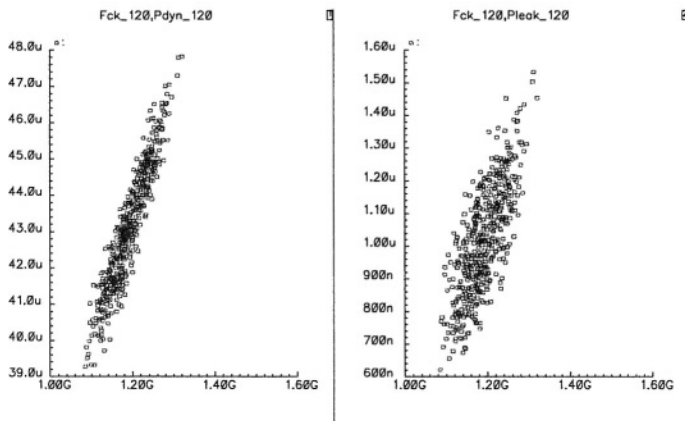
In order to validate the control architecture using only one set of estimation circuits and one controller, we present the results of the yield analysis, taking into account both die-to-die and within-die parameter variations. In Table 4 the partial and cumulative yields are reported for the circuit designed without any control, and for the circuit designed with the proposed approach using an ABB, an ASV and an ABB+ASV control scheme; within-die parameter variations have also been taken into account.

**Table 4.** Yield estimated at 120°C considering die-to-die and within die parameter variations.

	No Control	ASV	ABB	ABB+ASV
Y_Fck	100 %	97.40 %	99.20 %	98.40 %
Y_Pdyn	46.00 %	74.40 %	98.60 %	100 %
Y_Pleak	14.20 %	16.20 %	83.20 %	98.60 %
Y_tot	12.80 %	14.40 %	83.00 %	97 %



**Fig. 6.** Scatterplots without the controller considering die-to-die and within die parameter variations



**Fig. 7.** Scatterplots with the controller considering die-to-die and within die parameter variations

Fig. 6 and 7 reports the scatter plots of Dynamic power and leakage power as functions of operating frequency, for the circuit without the controller and for the circuit designed with the proposed approach and the ABB+ASV scheme, both at 120 °C operating temperature; both die-to-die and within-die parameter variations are consid-

ered. As we can see from table 4, a 12.8 to 83% yield enhancement is achieved using ABB technique. It must be noted that the current drawn from additional on chip power distribution networks for the body voltage is very small, therefore minimum metal width can be used to distribute bias voltage: this results in small additional silicon area. Further area increase is due to a 5 bit ADC, a 5 bit DAC, one set of estimation circuits, a 75 5 bit words PROM and a small control logic.

## 5 Conclusion

A control architecture and a design flow have been proposed, which allow to enhance the yield of CMOS digital integrated circuits. The method is based on the use of specific circuits to sense die-to-die process parameter and operating temperature variations, and of a digital controller which is able to set the body bias and/or supply voltage of MOS devices as a function of sensing circuits outputs. The body bias and/or supply voltage are therefore a function of the actual values of process parameters and operating temperature, allowing to optimise the trade off between speed and power consumption imposed by yield specifications. For its inherent characteristics the approach is more flexible than previously proposed ABB and ASV techniques in yield optimisation. The proposed method, applied to the design of a microprocessor critical path, has shown that ABB technique is more effective than ASV technique, especially at high temperatures. A strong yield improvement, in comparison to the non-controlled design, has been achieved using the ABB and ABB+ASV schemes. Using the ABB scheme with only one set of estimation circuits and one controller, a 12.8 to 83% yield enhancement is achieved with small silicon area footprint.

## References

1. S. Narendra, S. Borkar, V. De, D. Antoniadis, A. Chandrakasan, "Scaling of stack effect and its application for leakage reduction" International Symposium on Low Power Electronics and Design 2001, pp. 195-200.
2. T. Chen, S. Naffziger, "Comparison of Adaptive Body Bias (ABB) and Adaptive Supply Voltage (ASV) for Improving Delay and Leakage Under the Presence of Process Variation" IEEE Trans. on VLSI Systems, vol. 11, no. 5, 2003, pp. 888-899.
3. S. W. Director, P. Feldman, K. Krishna: "Statistical Integrated Circuit Design", IEEE Journal of Solid-State Circuits, vol. 28, no. 3, 1993, pp. 193-202.
4. Q. Zhang, J. Liou, J. McMacken, J. Thomson, and P. Payman, "Development of robust interconnect model based on design of experiments and multiobjective optimization," *IEEE Trans. Electron Devices*, vol. 48, pp. 1885-1891, Sept. 2001.
5. J.W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage" IEEE Journal of Solid-State Circuits, vol. 37, no. 11, 2002, pp. 1396-1402.
6. A. Keshavarzi, S. Ma, S. Narendra, B. Bloechel, K. Mistry, T. Ghani, S. Borkar, V. De, "Effectiveness of reverse body bias for leakage control in scaled dual Vt CMOS Ics" International Symposium on Low Power Electronics and Design 2001 pp 207-212.



7. A. Keshavarzi, S. Narendra, B. Bloechel, S. Borkar, V. De, "Forward Body Bias for Microprocessors in 130 nm Technology Generation and Beyond," Symposium on VLSI Circuits Digest of Technical Papers, 13-15 June 2002 pp. 312-315.
8. S. Narendra, D. Antoniadis, V. De, "Impact of using adaptive body bias to compensate die-to-die Vt variation on within-die Vt variation" International Symposium on Low Power Electronics and Design, 1999 pp. 229-232.
9. O. Y.-H. Leung, C.-W. Yue, C.-Y. Tsui, and R. S. Cheng, "Reducing power consumption of turbo code decoder using adaptive iteration with variable supply voltage," in *Proc. 1999 Int. Symp. Low-Power Electronics and Design*, Aug. 1999, pp. 36-41.
10. M. T. Schmitz and B. M. Al-Hashimi, "Energy minimization for processor cores using variable supply voltages," in *Proc. Institute of Electronic Engineers Workshop Systems on a Chip*, Sept. 2000.
11. G.-Y. Wei and M. Horowitz, "A fully digital, energy-efficient, adaptive power-supply regulator," *IEEE J. Solid-State Circuits*, vol. 34, pp. 520-528, Apr. 1999.
12. J. Kim and R. Horowitz, "An efficient digital sliding controller for adaptive power supply regulation," in *Proc. 2001 Symp. VLSI Circuits Dig. Tech. Paper*, June 2001, pp. 133-136.
13. S. M. Martin, K. Flautner, T. Mudge, D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads" IEEE/ACM International Conference on Computer Aided Design, 2002, pp. 721-725.
14. J. Tschanz, S. Narendra, R. Nair, V. De, "Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors," Symposium on VLSI Circuits Digest of Technical Papers, 13-15 June 2002 pp. 310-311.

# High Yield Standard Cell Libraries: Optimization and Modeling

Nicola Dragone, Michele Quarantelli, Massimo Bertoletti, and Carlo Guardiani

PDF Solutions  
25015 Desenzano, Italy  
(nicolad, micheleq, massimob, carlo}@pdf.com  
<http://www.pdf.com>

**Abstract.** In this paper, we present a flow for architecting standard cell libraries in nanometer technologies. The proposed approach relies on a Yield Characterization Environment to evaluate a set of manufacturability metrics and analyze multiple design trade-offs. We have identified four classes of manufacturability objectives that we addressed in our standard cell architectures: Average Functional Yield, Functional Yield Consistency, Performance Consistency and Leakage. Cells optimized for different manufacturability objectives present different trade-offs and require different layout architectures. We will show examples of such different trade-offs and architectural requirements and show the impact of adopting high-manufacturability standard cells on product Yields.

## 1 Introduction

Maximum packing density and performance  $\times$  power trade-off optimization have always been the key objectives of IP library physical design. As technology scales in the nanometer regime the manufacturability aspects of different layout architectures are increasingly becoming a critical factor of consideration for IC designers. In fact as minimum feature size scales to 130nm and below, the introduction of new materials and process flows on one hand and the increase of chip complexity and consequent dramatic explosion of the number of elements that must be correctly manufactured on a single chip on the other hand, has significantly changed the landscape. In fact, while in the past IC products Yield loss has been dominated by random defects (Random Mechanisms Limited Yield or RMLY) at 130nm and below an increasingly significant amount of issues is caused by Systematic Mechanism Limited Yield (SMLY). SMLY is defined as any type of Yield Loss that have either a spatial or temporal signature, or that is dependent on a specific attribute of the design, like for example a Via or contact open failure rate that is considerably higher for a feature that is laid out in a dense vs. isolated context. A significant fraction of SMLY occurs in standard cells. Logic limited Yield associated with SMLY may become the most critical factor in designs, such as graphic applications, where the logic content dominates over analog and memory content.

Composition of standard cell libraries, both in terms of logic functionalities and driving strengths, has been often recognized by the design community as one of the most critical aspects to achieve high quality synthesis [1][2]. In fact semiconductor

manufacturers have always reserved great care and large amount of resources to design and optimize standard cells and implement library modules that support their entire product portfolio at each technology node. A number of actions can be taken to improve standard cells manufacturability, like for example adding feature redundancy, over-sizing, spacing, uniformity or density. However, the applications of such techniques often conflicts with the optimization of other design parameters such as area, speed and power. Therefore, the ability of quantifying the contribution of different effects and ranking them in order of importance is of utmost importance in a yield-aware design flow. The problem of IC yield modeling, analysis and optimization has been explored in a number of papers in the literature [3],[4],[5],[6]. In these methods, yield is simulated directly on the final layout. However, if we want to take advantage from a DFM approach early in the design flow we need a hierarchical and modular approach to yield modeling. In this paper we propose a methodology for characterizing and modeling yield at standard cell level that can be used throughout the design flow from floorplan exploration, through logic synthesis, down to place & route.

The implementation of high yield standard cells and of the library views that are required to make them usable is a process that should span process characterization, yield modeling and characterization, and design optimization. In the following sections we will present a methodology that supports such manufacturability aware library design flow.

## 2 Yield Strength Standard Cell Architectures

General-purpose standard cell libraries should support a differentiated portfolio of IC products, that are characterized by different performance specifications and volume production requirements. We have identified three significantly different scenarios from manufacturing point of view:

1. High volume products at mid to mature phase of process ramp.
2. Mid-to-low volume products and engineering samples typically manufactured early in the process ramp phase.
3. High speed/high performance products.

A product with relatively long life cycle usually starts with the requirement of engineering samples for product qualification and SW development; it transitions to product ramp typically in the middle of the process Yield ramp and is finally produced with high volumes as the process matures. Obviously most products would also require careful optimization of timings in order to meet performance specs with reasonable margins and achieve good parametric Yields.

We have translated the requirements of products for each of the above scenarios into design objectives for the standard cell library as follows:

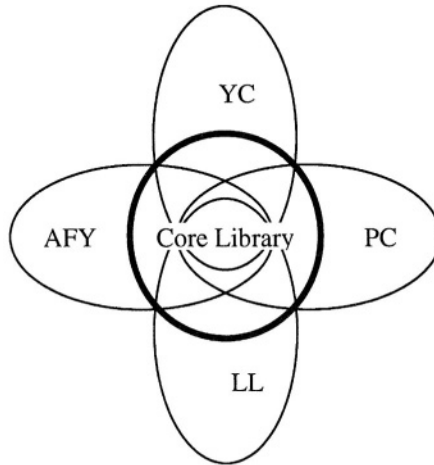
- Max average functional yield (AFY). Typically this is the most important manufacturing objective for high-volume parts.
- Max functional yield consistency (YC), i.e. minimize yield excursions (lot-to-lot, wafer-to-wafer and within wafer). This is a key objective early in ramp phase and for low volume products or engineering samples.

- Max performance consistency (PC). Minimize timing variability.
- Minimum leakage and power consumption (LL).

These four manufacturability optimization objectives are in general characterized by different and actually often conflicting process-layout interaction mechanisms, therefore we will call them orthogonal manufacturing objectives. The manufacturing aware standard cell library should provide tolerance for all of the orthogonal classes of objectives; therefore it should contain cell modules that are optimized for each of the key process-design interaction mechanisms that are relevant for a given class. An exemplary and non-exhaustive set of orthogonal effects that are addressed in each class is the following:

- AFY: hole (contact & Via) opens, critical area for opens and shorts, etc ...
- YC: hole coverage by the upper/lower layer, metal islands, field transitions, active leakage paths, etc ...
- PC: poly flaring, STI stress, etc ...
- LL: effective width, effective length, STI stress, poly end-cap, etc ...

As already stated, any optimization of the above process-design interaction mechanisms comes at the cost of some cell area increase, performance degradation and/or power consumption increase. Therefore it is of utmost importance to be able to quantify precisely the positive effect on each cell and eventually on the product Yield in order to take optimal design decisions.



**Fig. 1.** DFM library modules

The picture in Fig. 1 shows the high-level architecture of the proposed DFM standard cell library. A core library including a rich function set of high-density cells plus all required drive strength variants is complemented by additional cells that address one or more orthogonal objectives. Within each class and for each logic function/drive strength combination represented in the class, multiple *Yield Strength* variants are designed that implement different levels of Yield-area trade-off optimization. The cell variant that optimizes 100% of all the process-design interaction mechanisms

for a given orthogonal manufacturing class is called the Maximum Yield Strength variant for that class. Fractional Yield Strength cells optimize a subset of the process-design interaction mechanisms and/or they only achieve a fraction (i.e. 50%) of the optimization potential for a given interaction mechanism.

**Table 1. ELY improvements**

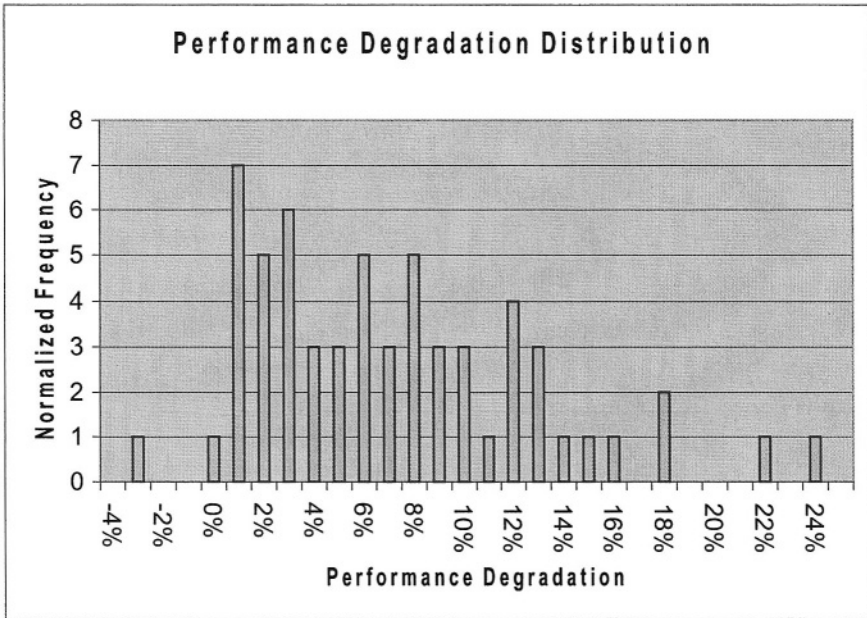
	Equivalent NAND2 gates	ELY Improvement
Process 1	1 million	2.08%
	2 millions	4.03%
	3 millions	5.84%
Process 2	1 million	4.96%
	2 millions	9.27%
	3 millions	12.99%

In Table 1 we report example statistics from application of high-yield standard cell library design in a 0.13 $\mu$ m process. We use here the concept of Equivalent Limited Yield (ELY), which expresses the Yield of an equivalent area covered by one type of cell. The equivalent area in our definition is equivalent to the area occupied by 1, 2 and 3 million NAND2 gates respectively. We show the ELY improvements obtained on two very similar 0.13 $\mu$ m processes but at a different stage of maturity. On a million-gate design, the average functional Yield improvement is 2% and 5% respectively. In Fig. 2 we also show the distribution of performance degradation: most of the cells show a performance degradation between 1% and 10% but we also have standard cells with worst case timing arcs with more than 20% of degradation. In Fig.3 we show the histogram of area degradation. The average standard cell area increase for this implementation is 15% with peaks close to 50%. The impact of a rigorous DFM approach on cell area and performance degradation clearly demonstrates the need of an accurate characterization of the Yield benefit and of the separate contribution of all the different effects.

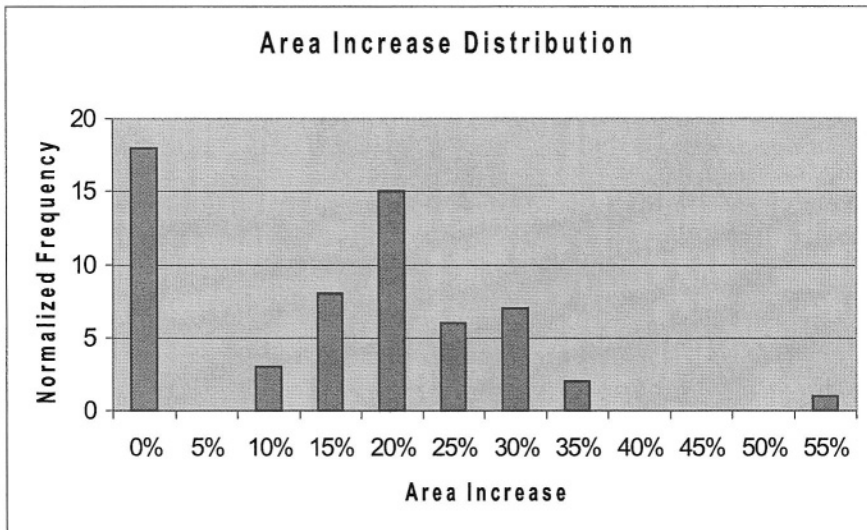
In the following section we will present our proven methodology to characterize standard-cell Yield and Yield model abstractions.

### 3 Std Cell Characterization and Modeling

As we discussed in previous sections, standard cell Yield modeling and characterization is a key pre-requisite in order to include Yield in the synthesis cost function. In fact, logic design DFM involves optimizing the trade-off among parameters such as area, speed or power and manufacturability. Therefore Yield must be characterized and a proper DFM library view created to represent cell library Yield data like standard timing and power views are created to evaluate speed or power during design synthesis. This DFM view can be utilized during analysis and optimization of digital circuits throughout the entire design flow, from logic synthesis to place & route and finally for design sign-off. In Fig. 4 we illustrate the methodology implemented to generate a DFM view. A Yield library compiler is used to generate the DFM view from the cell design attributes and process attributes. Design attributes, such as feature counts, critical areas and many other pattern specific configurations, are extracted



**Fig. 2.** Performance degradation of a DFM library



**Fig. 3.** Area degradation of a DFM library

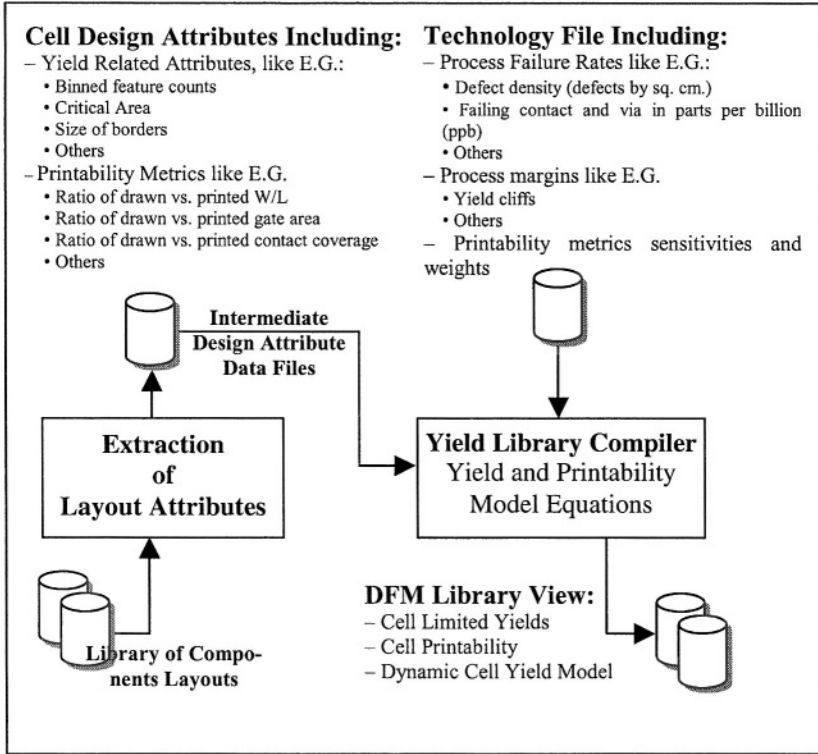


Fig. 4. Standard Cell DFM View Description and Creation Flow

from the layout using SiCat, the layout engine of YRS [7]. In the technology file we place information describing the maturity of the process, as far as defect density, attribute dependent feature failure rates, etc. that are used to weight the relevance of each Yield detractor at a specific point in time.

The technology file is generated using the flow depicted in Fig. 5. From an analysis of the library components and technology design rules we define a DOE (Design Of Experiments) that we use to implement the Characterization Vehicles (CV™). CVs are test chips that serve multiple purposes, like for example random defects characterization, systematic characterization and calibration structures for OPC and lithography simulations. After CVs are fabricated and tested, a very large amount of data is generated and needs to be processed and analyzed. The technology file represents a summary of the results of this analysis.

## 4 Cell Design Flow

An efficient implementation of the DFM library modules creation flow is shown in Fig. 6. After an initial study of the process characteristics, design rules and tool requirements, we define a set of architectures that are potentially relevant for the technology of interest. Each architecture template is applied to every standard cell for

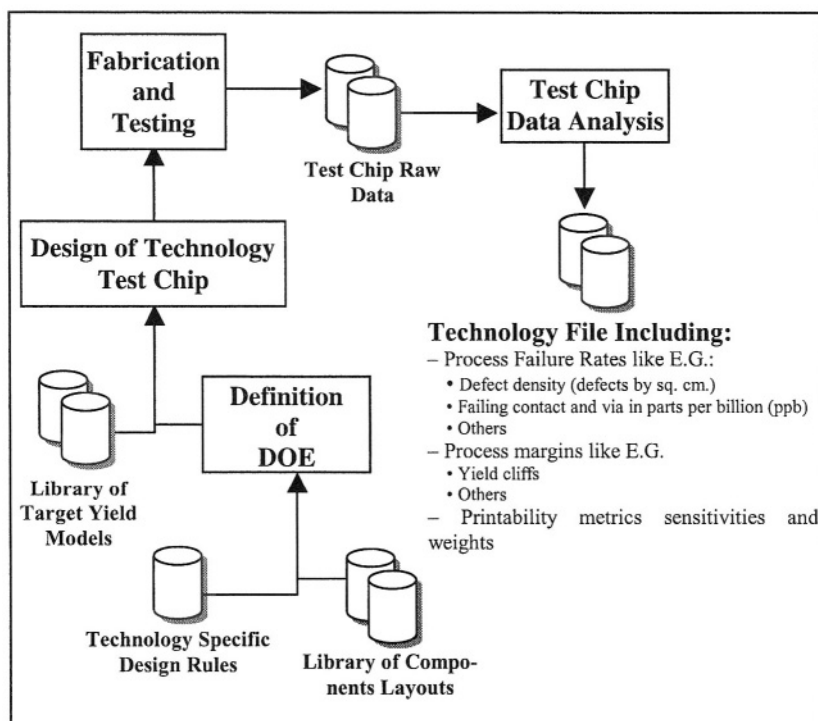


Fig. 5. Technology File Description and Creation Flow

manual or automated layout generation. A post-layout hot spot localization analysis is performed to identify potential sources of Yield loss and a set of new constraints or directives is generated to further improve manufacturability and RET related issues. After a few iterations, the final cell layout is generated and a pruning procedure is applied to eliminate cells that do not meet specifications or overlap with other cells from different modules. This is done in order to minimize the total number of cell variants in each library because it may happen quite often that different Yield strength variants of different manufacturability classes have quite similar Yield properties, hence they would be redundant. The pruned library is then characterized for Yield and the corresponding DFM view is generated.

## 5 Experimental Results

We have implemented the High-yield library concept described in this paper in a 0.13 $\mu$ m technology and tested its validity by using the DFM optimization flow on a suite of industrial examples. The DFM flow has been implemented using the Yield-aware physical synthesis environment, called pDfx [8], which can exploit the high Yield cells by intelligently trading off manufacturability for local area and timing slack. The first design example is the 8051DW micro-controller from the Synopsys Design Ware Library; the second example is the public domain processor core pico-



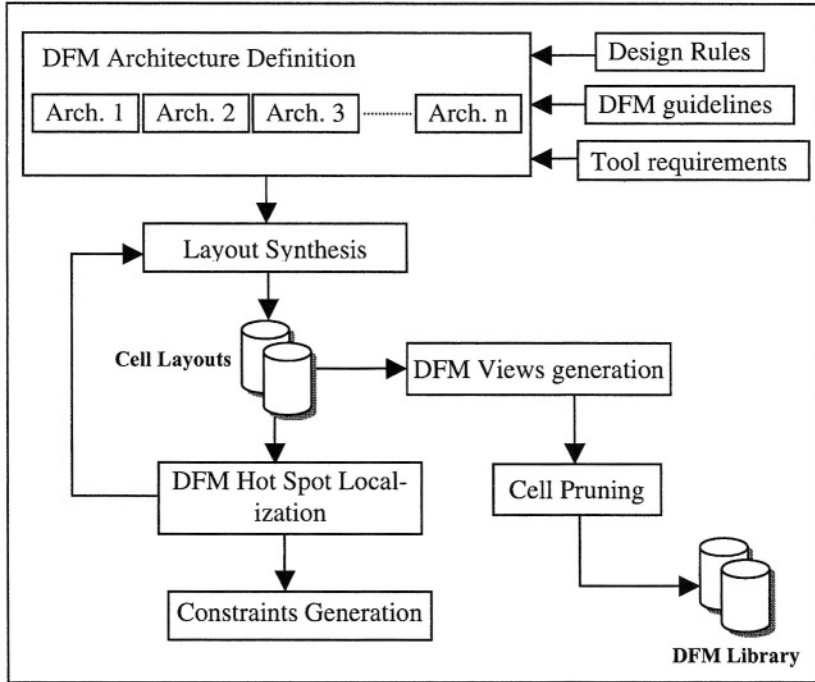


Fig. 6. DFM library design flow

Table 2. Results for the 8051 testcase

Layout(DW8051)	Area(um <sup>2</sup> )	YIELD	ELY	Delta ELY	ELY	Delta LY	Util
			1 million gates	1 million gates	2 million gates	2 million gates	
CLK_3_UTIL_60	1.17E+05	99.9188%	94.29%		88.91%		0.60
CLK_3_UTIL_60_OPT	1.17E+05	99.9587%	97.06%	2.77%	94.20%	5.29%	0.73
CLK_3_UTIL_85	8.26E+04	99.9183%	94.25%		88.82%		0.85
CLK_3_UTIL_85_OPT	8.26E+04	99.9326%	95.93%	1.68%	92.03%	3.20%	0.85
CLK_3_UTIL_100_OPT	8.26E+04	99.9557%	96.92%	2.68%	93.94%	5.12%	0.99
CLK_5_UTIL_60	1.17E+05	99.9212%	94.45%		89.22%		0.60
CLK_5_UTIL_60_OPT	1.17E+05	99.9607%	97.19%	2.74%	94.47%	5.25%	0.73
CLK_5_UTIL_85	8.26E+04	99.9181%	94.23%		88.80%		0.85
CLK_5_UTIL_85_OPT	8.26E+04	99.9350%	96.07%	1.84%	92.30%	3.50%	0.85
CLK_5_UTIL_100_OPT	8.26E+04	99.9597%	97.24%	3.01%	94.56%	5.76%	0.99

Table 3. Results for the picoJava testcase

Layout(Picojava)	Area(um <sup>2</sup> )	YIELD	ELY	Delta ELY	ELY	Delta ELY
			1 Mgate	1 Mgate	2 Mgate	2 Mgate
CLK_7_UTIL_30	581948.4978	99.32%	94.19%		88.72%	
CLK_7_UTIL_30_OPT	669315.3732	99.54%	96.52%	2.33%	93.16%	4.44%
CLK_6_UTIL_30	586684.2438	99.31%	94.16%		88.65%	
CLK_6_UTIL_30_OPT	675044.0982	99.52%	96.46%	2.31%	93.05%	4.40%
CLK_10_UTIL_30	580999.6512	99.31%	94.09%		0.8852	
CLK_10_UTIL_30_OPT	669629.3922	99.53%	96.45%	2.36%	0.9302	4.50%

Java II. Both designs have been implemented under multiple different conditions of clock cycle and utilization ratio. Results are summarized in Table 2 and Table 3. We report the actual Yield of the blocks and the scaled Yield gain on a million and 2 million equivalent gate design.

The main results can be summarized as follows.

1. Significant Yield improvement can be achieved by using a DFM library extension concept and the DFM design flow (over 5% Yield points on a 2 million gate design).
2. A constraint on the utilization ratio limits the potential Yield gain, due to the fact that fewer cells with wider area can be employed.
3. The impact of tighter clock cycle is less obvious. Theoretically, smaller clock period constraints should lead to reduced timing slack hence smaller Yield improvement. However since the synthesized netlist is structurally quite different and only a few paths are affected, the difference in Yield may be related to particular features of a few relevant cells.

## 6 Conclusions

In this paper we have presented a methodology to design high-yield cell libraries that can be used by yield-aware logic synthesis tools to optimize the manufacturability of logic designs. The trade-off between Yield and speed/area/power has been carefully analyzed and the need for accurate cell limited Yield characterization has been discussed. Finally we have presented the results from using a high-yield standard cell library in a 0.13um technology showing that significant improvement can be achieved by the proposed methodology when used in conjunction with a systematic DFM environment.

## References

1. K. Keutzer, K. Kolwicz, and M. Lega, "Impact of library size on the quality of automated synthesis," *Proc. of ICCAD*, 120-123, 1987.
2. Ken Scott and Kurt Keutzer, "Improving cell libraries for synthesis," *Proc. of CICC*, 128-131, 1994.
3. A.J. Strojwas, S.W. Director, "VLSI: linking design and manufacturing," *IEEE Spectrum*, 24-28, Issue 10, Oct. 1988.
4. P.K. Nag, W. Maly, "Hierarchical extraction of critical area for shorts in very large ICs," *Proceedings of International Workshop on Defect and Fault Tolerance in VLSI Systems*, 19-27, Nov. 1995.
5. H.T. Heineken, W. Maly, "Manufacturability analysis environment - MAPEX," *Proceedings of Custom Integrated Circuits Conference*, May 1994.
6. D.J. Ciplickas, X. Li, A.J. Strojwas, "Predictive yield modeling of VLSIC's," *5<sup>th</sup> International Workshop on Statistical Metrology*, 28-37, 2000.
7. YRS User's manual", PDF Solutions, Inc.
8. PDF Solutions, "Successful Integration of DFM Nanometer Era Design Flows," *CICC 2004, to be published*, 2004.

# A Study of Crosstalk Through Bonding and Package Parasitics in CMOS Mixed Analog-Digital Circuits

Gabriella Trucco, Giorgio Boselli, and Valentino Liberali\*

University of Milano,  
Department of Information Technologies,  
Via Bramante 65, 26013 Crema, Italy  
{trucco,liberali}@dti.unimi.it, gboselli@crema.unimi.it

**Abstract.** This paper presents an approach for simulation of mixed analog-digital CMOS integrated circuits, aiming at estimating crosstalk effects due to current pulses drawn from voltage supplies. A simple expression of voltage and current in the pull-up and the pull-down of a CMOS logic gate is derived, and a representation of digital switching noise in time domain can be easily calculated through a dedicated computer program. This representation is used to perform an analog simulation using SPICE, to evaluate the propagation of the switching noise through the parasitic elements of the package and of the bonding wires. Simulation results for two case studies are presented.

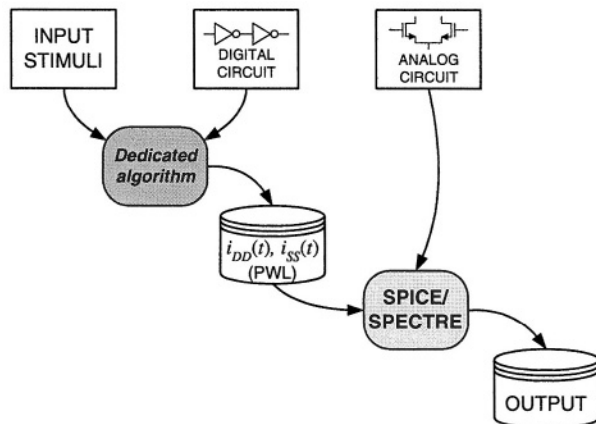
## 1 Introduction

In mixed analog-digital integrated circuits, effects of digital switching noise on the analog section are often the limiting factor affecting the overall system performance [1]. Therefore, analog designers must choose optimum circuit architectures taking robustness and crosstalk immunity into account. Hence, a correct design methodology should account for digital switching noise from early stages of the design process.

Simulation tools, capable of accurate analysis of current consumption during logic transitions, are required [2]. Digital simulation tools are mostly optimized for simulation speed and for “average” power consumption. On the other hand, analog simulators are quite inefficient for the analysis of large digital circuits. In our work, we propose a simulation method to analyze the current consumption and switching noise due to the digital part of the circuit. To this end, suitable simplified models of active devices are required: in particular, we need to consider all relevant aspects to obtain sufficiently accurate results, and we wish to neglect all issues that do not contribute to a remarkable improvement, in order to keep simulation time reasonably small [3].

---

\* This research has been partially supported by the Italian national program FIRB, contract no. RBNE01F582.



**Fig. 1.** Flow diagram of the proposed approach

It is well known that circuit-level (SPICE/SPECTRE) simulation for large digital circuits is very time-consuming. Dedicated algorithms can be more efficient, although they can be applied only to a specific class of circuits. In the case of mixed analog-digital circuits, we can speed up simulation by analyzing the digital and the analog sections separately, as illustrated in Fig. 1. A dedicated algorithm, described in Sect. 2, evaluates the supply currents  $i_{DD}$  and  $i_{SS}$  drawn by the digital part, and saves a piece-wise linear (PWL) description of current waveforms. In Sect. 3, the current waveforms are used as an input for subsequent simulation of the analog section of a mixed-signal circuit. This approach is applied to two blocks used in radio-frequency front-end ICs. The first circuit is a pipeline analog-to-digital converter (ADC), which includes a two phase clock generator acting as digital noise source, and a flash ADC based on resistive string affected by digital disturbances. The second circuit is a voltage-controlled oscillator (VCO), also affected by the disturbances coming from the clock generator.

## 2 Simulation Algorithm Based on Continuous Functions

The dedicated simulation algorithm, written in C++, uses time-continuous functions, instead of sample sequences, to represent signals. The advantages of this approach are: better accuracy in the calculation of derivatives, pre-layout switching noise estimation, and faster simulation speed compared to a circuit-level simulator. Accurate calculation of derivatives is important, as voltage drops due to bonding inductances represent a major source of crosstalk in mixed-signal systems.

Let us assume that the bonding and package parasitics between the digital supply  $V_{DD}$  and an analog signal  $v_s$  can be modeled as illustrated in Fig. 2 [4]. The instantaneous current  $i_{DD}$  due to digital switching of logic gates produces a voltage drop, which propagates to the adjacent wires through both capacitive coupling due to the capacitance between wires  $C_{1,2}$  and inductive coupling due

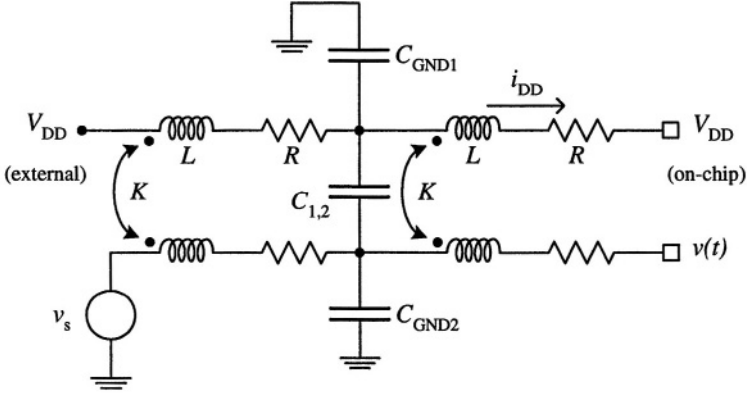


Fig. 2. Equivalent circuit for digital switching noise

to the mutual inductance represented by  $K$ . The analog on-chip voltage  $v(t)$  is a function of the external voltage  $v_s$  and of the digital switching current  $i_{DD}$  and its derivative:

$$v(t) = f\left(v_s, i_{DD}, \frac{di_{DD}}{dt}\right). \quad (1)$$

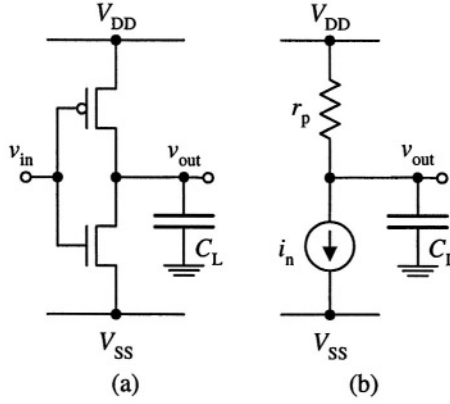
In our analysis of the digital switching in CMOS logic cells, the pull-up and the pull-down branches of logic gates are modeled either as current generators (MOS transistors in the saturation region) or as resistive switches (MOS transistors in the triode region), in order to keep the complexity of the network at an acceptable level.

The analysis method is similar to the one described in [5]. The overall simulation time is divided into time slices, depending upon input signal variations and the operating region of pull-up and pull-down MOS devices. Within each time slice, time-continuous functions describing the output voltage  $v_{out}(t)$  and the supply currents  $i_p(t)$  (through the pull-up branch) and  $i_n(t)$  (through the pull-down branch) are calculated. When the operation region of a digital cell changes, an event is generated and all the digital cells are analyzed again. At any time slice, the operating condition of the pull-up and the pull-down branch of each cell is determined. The output voltage  $v_{out}(t)$  and the currents drawn from the positive and the negative supply ( $i_{DD}(t)$  and  $i_{SS}(t)$ , respectively) are calculated as a function of the input voltage  $v_{in}(t)$ . The use of a simple model for active elements ensures that a closed-form expression exists.

Under the above assumptions for the models of MOS transistors, when piecewise linear external inputs are applied, all signals at the output of digital gates can be expressed in the form:

$$v(t) = \sum_i p_i(t) \cdot e^{-\alpha_i t} \quad (2)$$

where  $p_i(t)$  is a real polynomial function in  $t$  and  $\alpha_i$  is a real quantity (the inverse of a time constant).



**Fig. 3.** CMOS inverter: (a) schematic diagram; (b) simplified model with NMOS in saturation and PMOS in triode

As an example, let us consider the case of a CMOS inverter having the NMOS transistor working in the saturation region and the PMOS transistor in the triode region; this occurs when  $V_{SS} + V_{th,n} < v_{in}(0) < V_{DD} + V_{th,p}$  and  $v_{in}(0) - v_{out}(0) < V_{th,p}$ . Fig. 3 illustrates the simplified model used for the analysis. The current through the NMOS transistor is:

$$i_n(t) = K_n \cdot (v_{in}(t) - V_{SS} - V_{th,n})^2 \quad (3)$$

where  $K_n = \frac{1}{2}\mu_n C_{ox} \frac{W}{L}$ , with usual meaning of symbols.

Assuming that the PMOS transistor in the triode region can be approximated by a resistance  $r_p$  constant over the time interval  $[0, t_{max}]$ , the output voltage can be obtained from the differential equation:

$$r_p C_L \frac{dv_{out}(t)}{dt} + v_{out}(t) - V_{DD} + r_p i_n(t) = 0 \quad (4)$$

where  $C_L$  is the capacitive load. The solution is:

$$v_{out}(t) = v_0 e^{-\frac{t}{r_p C_L}} + e^{-\frac{t}{r_p C_L}} \frac{1}{C_L} \int_0^t \left( \frac{V_{DD}}{r_p} - i_n(t) \right) \cdot e^{\frac{t}{r_p C_L}} dt \quad (5)$$

The current through the PMOS transistor is:

$$i_p(t) = \frac{V_{DD} - v_{out}(t)}{r_p} \quad (6)$$

If the input voltage  $v_{in}(t)$  is expressed in the form (2), a closed form solution for voltages and currents can be calculated. For the cascade configuration of logic cells, the output of any gate is used as the input for the next one.

After calculating the output voltage, we also need to determine the maximum time  $t_{max}$  for which the computed solution is valid:  $t_{max}$  is the time instant until which no change occurs in the operation region of any device in the circuit.

The average on-resistance of a MOS transistor in triode region,  $r$ , depends on its gate-to-source voltage  $v_{GS}$  and, therefore, on its input voltage, through the relationship:

$$r = \frac{1}{K |v_{GS} - V_{th}|} \quad (7)$$

where the drain-to-source voltage  $v_{DS}$  has been neglected with respect to the gate-to-source voltage  $v_{GS}$ .

A more detailed description of the algorithm can be found in [6].

The proposed algorithm has been used to simulate the two phase clock generator illustrated in Fig. 4. Currents and switching noise due to series inductances are shown in Figs. 5 and 6, respectively. The time-domain current waveforms in Fig. 5 are in good agreement with the result obtained with a SPECTRE simulation [7]. To obtain the switching noise waveform shown in Fig. 6, the currents

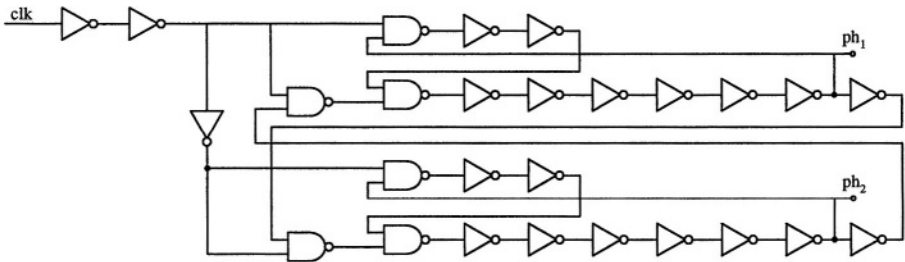


Fig. 4. Circuit of the two phase clock generator

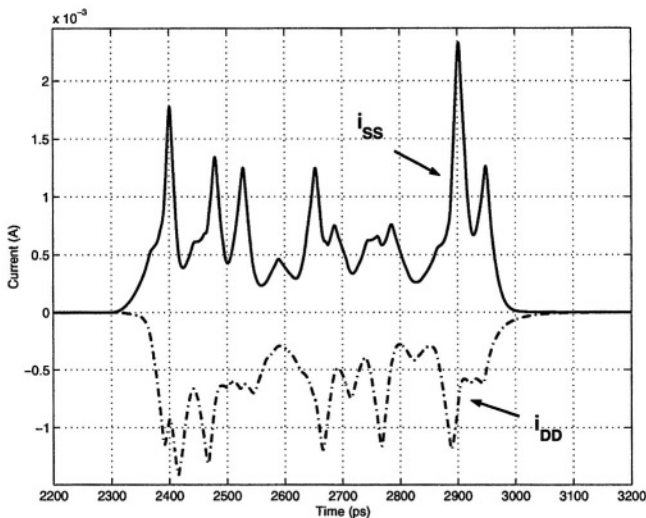
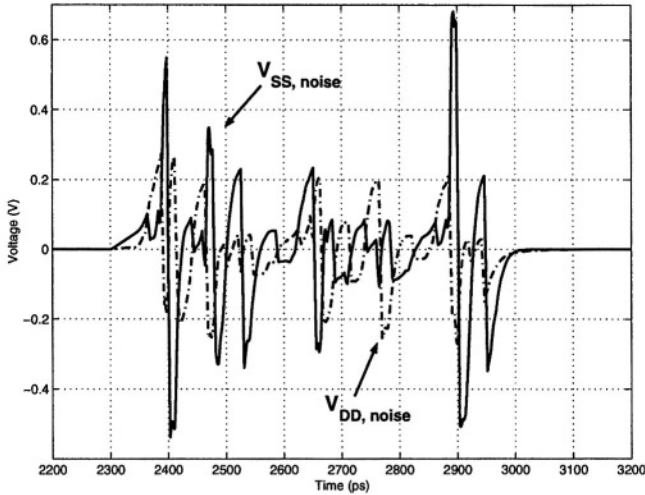


Fig. 5. Current consumption of the two phase clock generator



**Fig. 6.** Switching noise due to inductances on power supply lines

calculated assuming ideal supply lines (Fig. 5) were injected into the respective inductances ( $L_{SS} = L_{DD} = 5$  nH).

### 3 Analysis of Crosstalk Through Parasitics

To evaluate the effects of switching currents on analog circuits, we have considered two case studies: a pipeline ADC and a VCO. Both circuits have been simulated with SPICE at transistor level, including the model for parasitics illustrated in Fig. 2 for all bonding wires. The digital section has been modeled with PWL current generators; Fig. 5 shows the currents in time domain.

#### 3.1 Effects of Crosstalk on an ADC

The first circuit is a pipeline ADC, whose architecture is shown in Fig. 7 [8].

Each pipeline stage contains the flash ADC shown in Fig. 8. The sample-and-hold (S/H), the digital-to-analog converter (DAC) and the residue amplifier are synchronized with a two phase clock generated by the circuit in Fig. 4.

We have simulated the flash ADC including the effects of the switching currents through all parasitics associated to bonding and package. Each wire has series inductance and resistance, capacitance to ground, and both capacitive and inductive couplings towards other wires.

Fig. 9 shows the on-chip supply voltages waveforms of the analog section (nodes  $V_{DD}$  and  $V_{SS}$  in Fig. 8). Values of parasitic elements used in simulation are: inductance  $L = 2.5$  nH, resistance  $R = 50$  m $\Omega$ , ground capacitance  $C_{GND} = 15$  fF, capacitance between wires  $C_{i,j} = 5$  fF, mutual inductance coupling factor  $K = 0.2$ . From the waveforms in time domain, we can see that on-chip supplies



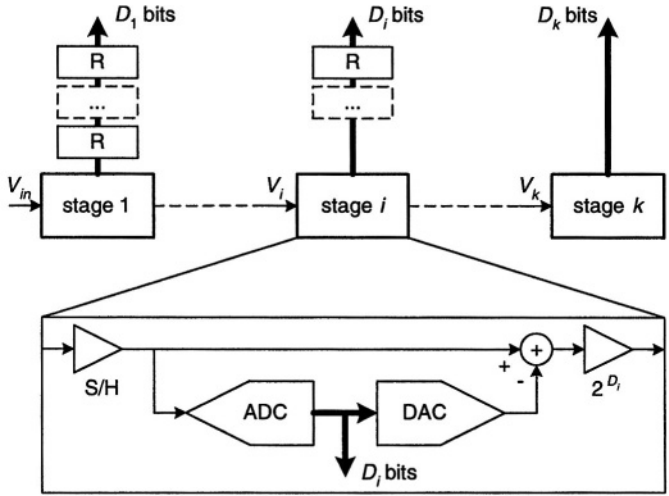


Fig. 7. Architecture of the pipeline A/D converter

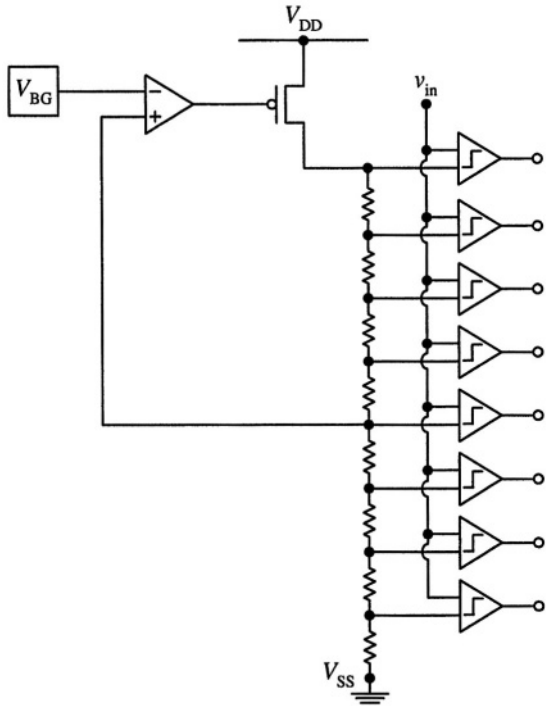
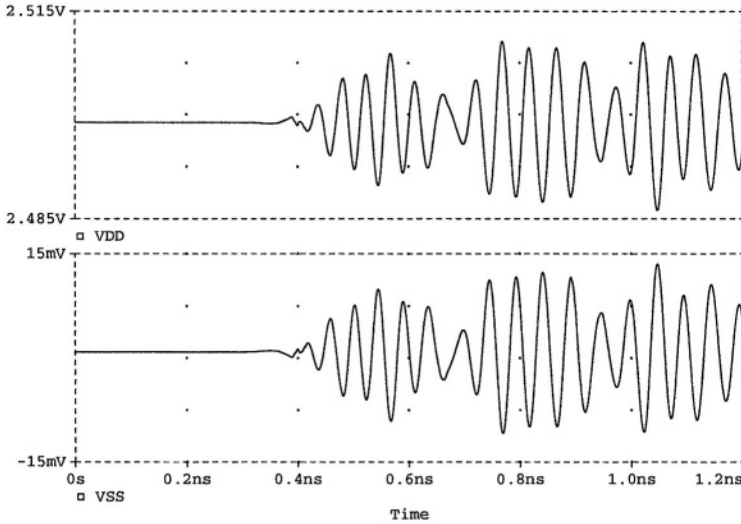


Fig. 8. Schematic diagram of the flash ADC



**Fig. 9.** Simulated on-chip values of  $V_{DD}$  and  $V_{SS}$  in the flash ADC

are affected by digital noise and display a “bouncing” effect with a peak value larger than 10 mV. This effect limits the resolution of the pipeline ADC to 8 bits.

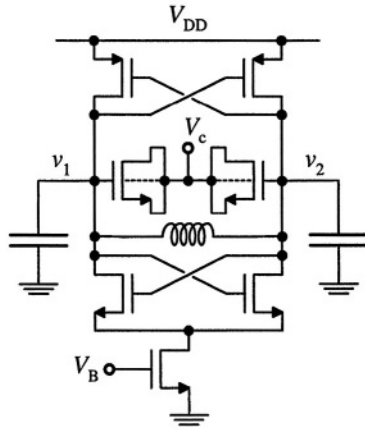
### 3.2 Effects of Crosstalk on a VCO

We have investigated the effects of crosstalk on a VCO, which is the most critical block in a PLL-based clocking system [9].

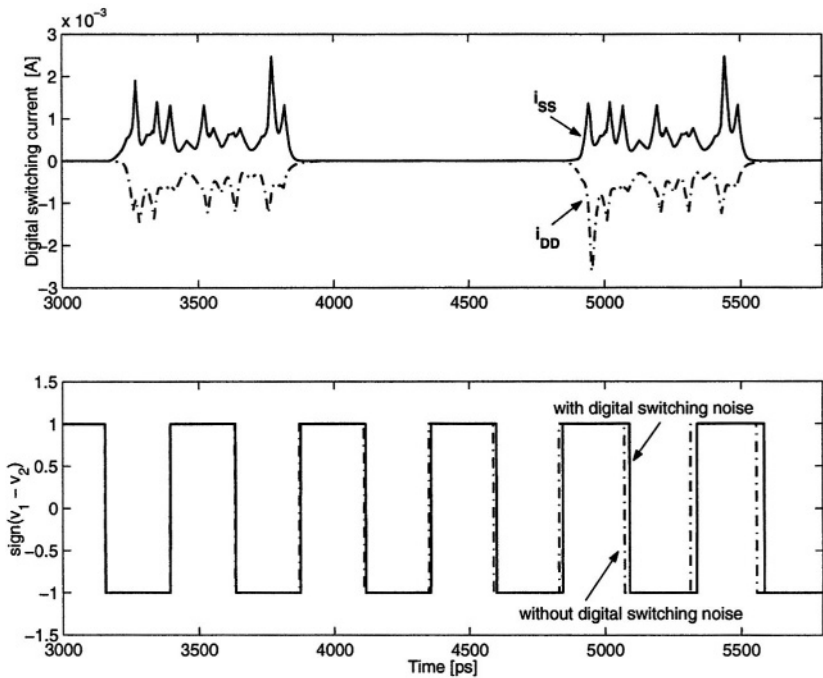
The schematic diagram of the VCO is illustrated in Fig. 10 [10]. The VCO frequency is tuned by adjusting the voltage  $V_c$ , which controls the gate-to-channel capacitance of the two MOS transistors. To reduce the effect of digital disturbances, the VCO has a fully-differential structure and the output signal is differential:  $v_1 - v_2$ . Since the digital switching noise is a common mode signal, the differential output should not be affected, provided that the differential structure is perfectly matched.

However, the voltage  $V_c$  is affected by switching noise, which propagates through interconnection parasitics and through the substrate. Indeed, coupling through bonding and package parasitics affects the substrate bias voltage, which is no more held at a constant value. Since the substrate is capacitively coupled with the well of the MOS transistors used for tuning, part of the high-frequency digital switching noise is injected into  $V_c$ . Therefore, the VCO output signal is affected by phase noise.

Fig. 11 illustrates the polarity of the differential VCO output  $v_1 - v_2$ , with and without the noise due to the digital switching current. The effect of switching noise is apparent: it delays the zero-crossing instants of the VCO signal, thus modulating the VCO phase.



**Fig. 10.** Schematic diagram of the VCO



**Fig. 11.** Sign of the differential VCO output voltage with and without digital switching noise

## 4 Conclusion

This paper has presented a method for the analysis of the effects of digital switching noise on analog circuits.

A dedicated algorithm for the calculation of digital switching current in time domain has been illustrated. Current waveforms can be used as an input for SPICE analysis of the analog section, to evaluate the performance of analog circuits in presence of bonding and package parasitics.

Disturbances generated by switching currents in digital blocks propagate through interconnection parasitics, and affect analog voltages. Simulation results on an ADC and on a VCO demonstrate that switching noise can severely degrade circuit performance. Therefore, digital switching effects must be carefully considered when designing accurate analog circuits.

## References

1. Donnay, S., Gielen, G., eds.: *Substrate Noise Coupling in Mixed-Signal ASICs*. Kluwer Academic Publishers, Boston, MA, USA (2003)
2. van Heijningen, M., Badaroglu, M., Donnay, S., Engels, M., Bolsens, I.: High-level simulation of substrate noise generation including power supply noise coupling. In: *Proc. Design Automation Conf. (DAC)*, Los Angeles, CA, USA (2000) 446–451
3. Gerez, S.H.: *Algorithms for VLSI Design Automation*. John Wiley & Sons, Chichester, UK (1999)
4. Liberali, V., Pettazzi, S., Rossi, R., Torelli, G.: Analysis and simulation of substrate noise coupling in mixed-signal CMOS ICs. *Transworld Research Network – Recent Res. Devel. Electronics* **1** (2002) 145–163
5. Armaroli, D., Liberali, V., Vacchi, C.: Behavioural analysis of charge-pump PLLs. In: *Proc. Midwest Symp. on Circ. and Syst.*, Rio de Janeiro, Brazil (1995) 893–896
6. Trucco, G., Boselli, G., Liberali, V.: An analysis of current waveforms in CMOS logic cells for RF mixed circuits. In: *Proc. IEEE Int. Conf. on Microelectronics (MIEL)*, Niš, Serbia (2004) 563–566
7. Trucco, G., Boselli, G., Liberali, V.: Simulation of crosstalk through bonding and package in mixed-signal CMOS ICs. In: *Proc. Midwest Symp. on Circ. and Syst.*, Hiroshima, Japan (2004) ...
8. Rodríguez-Vázquez, A., Medeiro, F., Janssens, E., eds.: *CMOS Telecom Data Converters*. Kluwer Academic Publishers, Boston, MA, USA (2003)
9. Heydari, P.: Characterizing the effects of clock jitter due to substrate noise in discrete-time  $\Delta\Sigma$  modulators. In: *Proc. Design Automation Conf. (DAC)*, Anaheim, CA, USA (2003) 532–537
10. Liao, H., Rustagi, S.C., Shi, J., Xiong, Y.Z.: Characterization and modeling of the substrate noise and its impact on the phase noise of VCO. In: *Proc. Radio Frequency Integr. Circ. Symp. (RFIC)*, Philadelphia, PA, USA (2003) 247–250

# Sleepy Stack Reduction of Leakage Power

Jun Cheol Park, Vincent J. Mooney III, and Philipp Pfeiffenberger

Center for Research on Embedded Systems and Technology  
School of Electrical and Computer Engineering, Georgia Institute of Technology  
Atlanta, GA 30332-0250 USA

{jcpark, mooney}@ece.gatech.edu, gte125y@mail.gatech.edu

**Abstract.** Leakage power consumption of current CMOS technology is already a great challenge. ITRS projects that leakage power consumption may come to dominate total chip power consumption as the technology feature size shrinks. We propose a novel leakage reduction technique, named “sleepy stack,” which can be applied to general logic design. Our sleepy stack approach retains exact logic state – making it better than traditional sleep and zigzag techniques – while saving leakage power consumption. Unlike the stack approach (which saves state), the sleepy stack approach can work well with dual- $V_{th}$  technologies, reducing leakage by several orders of magnitude over the stack approach in single- $V_{th}$  technology. Unfortunately, the sleepy stack approach does have a area penalty (roughly 50~120%) as compared to stack technology; nonetheless, the sleepy stack approach occupies a niche where state-saving and extra low leakage is desired at a (potentially small) cost in terms of increased delay and area.

## 1 Introduction

The advent of a mobile computing era has become a major motivation for low power design because the operation time of a mobile device is heavily restricted by its battery life. The growing complexity of mobile devices, such as a cell phone with a digital camera or a personal digital assistant (PDA) with global positioning system (GPS), makes the power problem more challenging.

Dynamic power consumption was previously a major concern for chip designers since dynamic power accounted for 99% or more of the total chip power. However, as the feature size shrinks, static power, which consists mainly of sub-threshold and gate-oxide leakage power, has become a great challenge for current and future technologies. The main reason is that leakage current increases exponentially as the feature size shrinks. Based on the International Technology Roadmap for Semiconductors (ITRS), Kim et al. report that subthreshold leakage power dissipation of a chip will exceed dynamic power dissipation at the 65nm feature size [1][2].

Techniques for leakage power reduction can be grouped in two categories: state-preserving techniques where circuit state (present value) is retained and state-destructive techniques where the current boolean output value of the circuit

might be lost [1]. A state-preserving technique has an advantage over a state-destructive technique in that with a state-preserving technique the circuitry can resume operation at a point much later in time without having to somehow regenerate state. Our new design technique, which we call the “sleepy stack” technique, retains data during sleep mode while providing reduced leakage power consumption at a cost of slightly increased delay. Furthermore, the sleepy stack approach can be applicable to single- and dual-threshold voltage technologies. The sleepy stack approach delivers a new choice to designers to implement low-leakage-power circuits that retains state.

The rest of the paper is organized as follows: Section 2 discusses sources of leakage power and previous low-leakage approaches. Section 3 describes the proposed sleepy stack approach with comparisons to previous approaches. Section 4 presents the simulation methodology, and Section 5 explains the results. Section 6 concludes the paper.

## 2 Previous Work

In today’s technology, the main contributor to static power consumption of a CMOS circuit is subthreshold leakage, i.e., source to drain current when gate voltage is smaller than the transistor threshold voltage. The subthreshold leakage current can be expressed as follows:

$$I_{sub} = K_1 W e^{-V_{th}/nV_\theta} (1 - e^{-V/N_\theta}) \quad (1)$$

where  $K_1$  and  $n$  are experimental values,  $W$  is the width of the transistor,  $V_{th}$  is the threshold voltage and  $V_\theta$  is the thermal voltage [1]. Since subthreshold current increases exponentially as the threshold voltage decreases, deep sub-micron technologies with scaled down threshold voltages will severely suffer from subthreshold leakage power consumption. In addition to subthreshold leakage, another contributor to leakage power is gate-oxide leakage power due to the tunneling current through the gate-oxide insulator. Since gate-oxide thickness will be reduced as the technology decreases, deep sub-micron technology will also suffer from gate-oxide leakage power. However, previously proposed work for leakage power reduction at the circuit level has focused on subthreshold leakage power because gate-oxide leakage is relatively small compared to subthreshold leakage [1]. Although gate-oxide leakage will also increase exponentially as technology feature size decreases, a solution for gate-oxide leakage may lie in the development of high dielectric constant (high- $k$ ) gate insulators. In this paper, we focus exclusively on reducing subthreshold leakage power.

As introduced in Section 1, previously proposed work can be divided into techniques that either (i) preserve state or (ii) destroy state. State-destructive techniques (ii) cut off transistor (pull-up or pull-down or both) networks from supply voltage or ground using sleep transistors [5]. These types of techniques are also called gated- $V_{DD}$  (note that a gated clock is generally used for dynamic power reduction). Motoh et al. propose a technique called multi-threshold-voltage CMOS (MTCMOS), which adds high  $V_{th}$  sleep transistors between pull-up networks and  $V_{DD}$  and between pull-down networks and ground while logic

circuits use low  $V_{th}$  to maintain logic performance [5]. The sleep transistor is turned off when the logic circuits are not in use. Powell et al. propose a gated- $V_{DD}$  cache technique called DRI cache, which dynamically changes cache size with gated- $V_{DD}$  transistors of dual or single  $V_{th}$  [4]. Since the gated- $V_{DD}$  technique cuts off logic blocks from  $V_{DD}$  and  $Gnd$ , time and energy for waking up are significant. The zigzag technique proposes the reduction of wake-up overhead by choosing a particular circuit state (e.g., corresponding to a “reset”) and then, for the exact circuit state chosen, cutting off the pull-down network for each gate whose output is high while conversely cutting off the pull-up network for each gate whose output is low [9]. Although the zigzag technique can retain a particular state chosen prior to chip fabrication, any other arbitrary state during regular operation is lost in power-down mode. Another technique to reduce leakage power is transistor stacking that exploits the stack effect, i.e., it turns out that two stacked and turned off transistors reduce subthreshold leakage current substantially due mainly to a reverse bias between the gate and source [6]. Narendra et al. study the effectiveness of the stack effect including effects from increasing the channel length [7]. Since forced stacking of what previously was a single transistor increases delay, Johnson et al. propose an algorithm that finds circuit input vectors that maximize stacked transistors of existing complex logic [8]. Flautner et al. propose the “drowsy cache” technique that switches supply voltage instead of gating  $V_{DD}$  [10]. The effect of drowsy caches in terms of leakage power reduction is smaller than gated- $V_{DD}$  techniques, but the drowsy cache technique can retain the original state thus can be used for designing memories.

### 3 Sleepy Stack

In this section, our new low-leakage-power design, named “sleepy stack,” is described and compared with well-known previous approaches, i.e., the sleep, zigzag and stack techniques explained in Section 2. The base case, shown in Fig. 1, consists of three stages each with a pull-up network and a pull-down network and with the final stage connected to a load capacitance. Fig. 2, 3 and 4 show previous low-leakage approaches applied to the base case. The sleep approach in Fig. 2

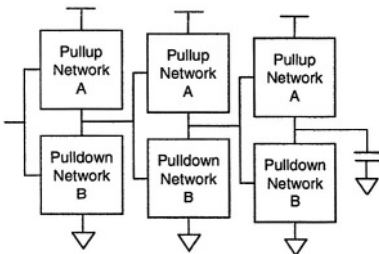


Fig. 1. Base case

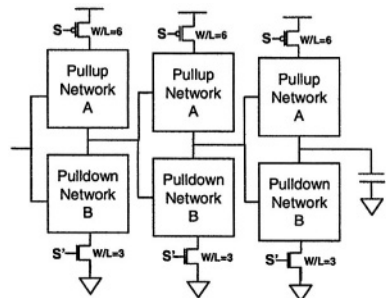


Fig. 2. Sleep

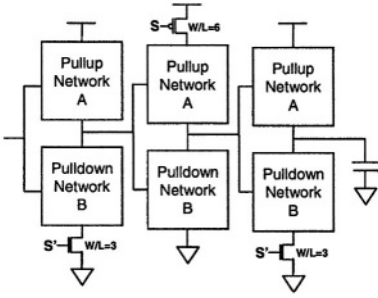


Fig. 3. Zigzag

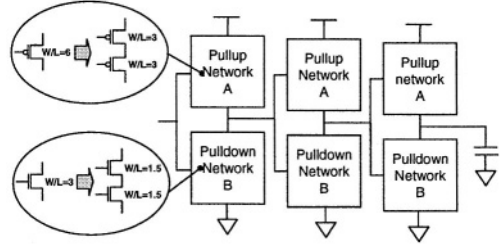


Fig. 4. Stack

uses sleep transistors between both  $V_{DD}$  and the pull-up network as well as between  $Gnd$  and the pull-down network. Generally, the width/length ( $W/L$ ) ratio is sized based on a trade-off between area, leakage reduction and delay. During sleep mode, sleep transistors are turned off and leakage current is suppressed. However, the additional sleep transistors increase area and delay. Furthermore, the pull-up and pull-down network will have floating values and lose state during sleep mode. The zigzag approach in Fig. 3 first analyzes each gate for a particular input assumed during sleep mode and either assigns a sleep transistor to the pull-down network if the output is “1” or else assigns a sleep transistor to the pull-up network if the output is “0.” In Fig. 3, we assume that, in the sleep mode, the input of the logic is “0” and each logic stage reverses its input signal, i.e., the output is “1” if the input is “0,” and the output is “0” if the input is “1.” Accordingly, the pull-down network of the first stage is off, and so a sleep transistor is added. Thus, the zigzag approach uses fewer sleep transistors than the original sleep approach. However, analyzing logical state and finding input vectors of complex logic blocks are NP-hard problems which need to be solved in order to apply the zigzag approach [8].

Fig. 4 shows the stack approach, which forces a stack effect by breaking down an existing transistor into two half size transistors. The forced stack approach can achieve huge leakage power saving while retaining the logic state. However, the divided transistors increase delay significantly and may restrict the usefulness of the forced stack approach.

The key idea of the sleepy stack technique is to combine the sleep transistor approach during active mode with the stack approach during sleep mode. The structure of the sleepy stack approach is shown in Fig. 5. The sleepy stack technique divides existing transistors into two transistors each typically with the same width  $W_1$  half the size of the original single transistor’s width  $W_2$  (i.e.,  $W_1 = W_2/2$ ). Then sleep transistors are added in parallel to one of the transistors in each set of two stacked transistors; see Fig. 6 for an example. The divided transistors reduce leakage power using the stack effect while retaining state. The added sleep transistors operate similar to the sleep transistors used in the sleep technique in which sleep transistors are turned on during active mode and turned off during sleep mode. Fig. 6 depicts the sleepy stack operation



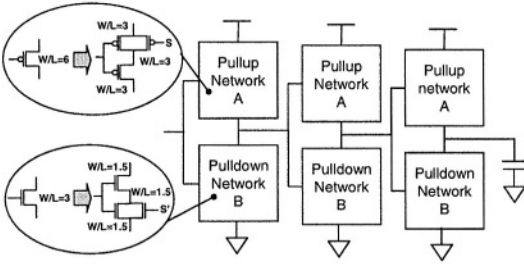


Fig. 5. Sleepy stack

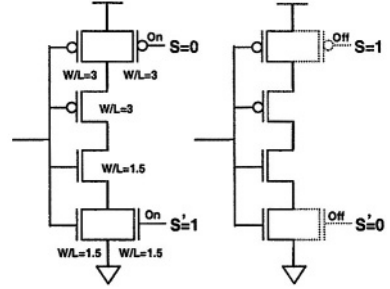


Fig. 6. Sleepy stack active mode (left) and sleep mode (right)

using an inverter. During active mode,  $S=0$  and  $S'=1$  are asserted, and thus all sleep transistors are turned on. Due to the added sleep transistor, the resistance through the activated (i.e., “on”) path decreases, and the propagation delay decreases (compared to not adding sleep transistors while leaving the rest of the circuitry the same, i.e., with stacked transistors). During the sleep mode,  $S=1$  and  $S'=0$  are asserted, and so both of the sleep transistors are turned off. The stacked transistors in the sleepy stack approach suppress leakage current.

## 4 Experimental Methodology

We compare the proposed sleepy stack approach to a base case (based on Fig. 1) and three of the previous approaches explained earlier, namely zigzag, sleep and stack. Thus, we compare five design approaches in terms of power consumption (dynamic and static), delay and area.

To show that the sleepy stack approach is applicable to general logic design, we choose three generic circuits: (i) a chain of 3 inverters, (ii) a 4-input multiplexer and (iii) a 4-bit adder. The logic diagram of the 4-input multiplexer used for (ii) is shown in Fig. 7. One bit of the 4-bit adder used for (iii) is shown in Fig. 8 of which is sized as noted in the same figure.

The inverter chain uses three inverters each with  $W/L=6$  for PMOS and  $W/L=3$  for NMOS for the base case. Sleep transistors in the sleep approach (Fig. 2) and the zigzag approach (Fig. 3) are sized such that any sleep transistor between  $V_{DD}$  and a pull-up network takes the size of the largest transistor in the pull-up network, and any sleep transistor between  $Gnd$  and a pull-down network takes the size of the largest transistor in the pull-down network. For example, sleep transistors used in the pull-up and pull-down networks of the base case inverter chain have  $W/L=6$  and  $W/L=3$ , respectively as shown in Fig. 2. Transistors in the stack approach are sized to half of the size of the base case transistors, e.g., transistors used in pull-up and pull-down of the base case inverter chain have  $W/L=3$  and  $W/L=1.5$ , respectively, as shown in Fig. 4. Similarly, transistors, including sleep transistors, in the sleepy stack approach are sized to half of the size of the base case transistors as shown in Fig. 5. The

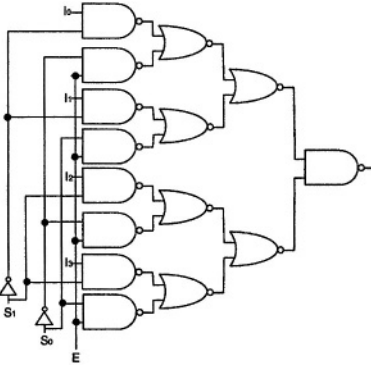


Fig. 7. 4-input multiplexer

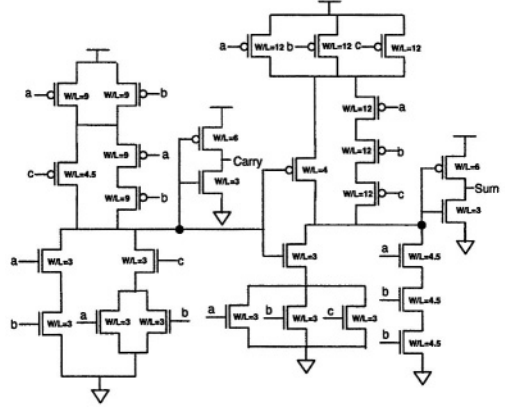


Fig. 8. 1-bit full adder

load capacitance is set to  $3C_{inv}$  (where  $C_{inv}$  is the capacitance of an inverter with pull-up  $W/L=3$ , pull-down  $W/L=1.5$ ).

To estimate area, delay and power, we first design target benchmark circuits using Cadence Virtuoso, a custom layout tool [12], and North Carolina State University Cadence design kit targeting TSMC  $0.18\mu$  technology [13]. Then we extract schematics from layout to obtain transistor circuit netlists. we use HSPICE simulation to estimate delay and power of the benchmarks, for which we use Avant! Stat-HSPICE [11].

Since we don't have access to the layout design proprietary below  $0.18\mu$  technology, we use the Berkeley Predictive Technology Model (BPTM) parameters to estimate delay and power for technologies below  $0.18\mu$  [14,15]. We choose four different technologies from BPTM to observe changes of power and delay as technology shrinks. The chosen technologies, i.e.,  $0.07\mu$ ,  $0.10\mu$ ,  $0.13\mu$  and  $0.18\mu$ , use supply voltages of 1.0V, 1.3V, 1.6V and 1.8V, respectively. We assume that only a single supply voltage is available in each technology. We do consider both single- and dual- $V_{th}$  technology for the zigzag, sleep and sleepy stack approaches (note that for the straightforward stack approach, no transistors exist which could be made high- $V_{th}$  without a dramatic increase in delay). With dual- $V_{th}$  technology, the zigzag and sleep approaches use high- $V_{th}$  sleep transistors. Similarly, in the sleepy stack approach with dual- $V_{th}$  technology, high- $V_{th}$  transistors are used for sleep transistors and transistors that are parallel to the sleep transistors. We set all high- $V_{th}$  transistors to have 2.5 times higher  $V_{th}$  than the  $V_{th}$  of a normal transistor just like [5]. More details about the experimental methodology can be found in a technical report [16].

## 5 Experimental Results

We measure worst case propagation delay and power for the five design approaches, which are the base case, zigzag, sleep, stack and sleepy stack ap-

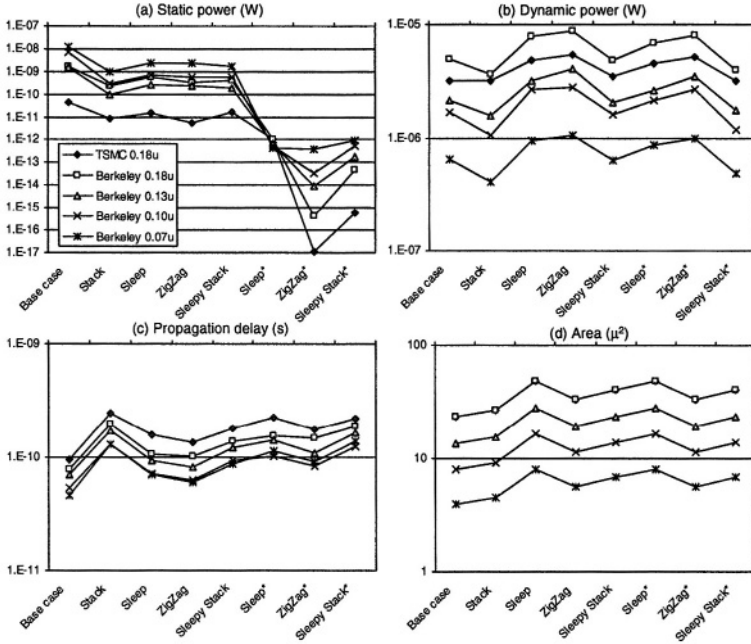


Fig. 9. Results of chain of 3 inverters (\*dual  $V_{th}$ )

proaches. Dynamic power is measured with a random input vector changing every clock cycle, i.e., 4ns; static power is measured using a sampling of input vectors and averaged. The area of the benchmarks below  $0.18\mu$  technology is estimated by scaling area of benchmark layout using TSMC  $0.18\mu$  technology. We add 10% of area overhead considering non-linear scaling layers, e.g., a metal layer.

Fig. 9, 10 and 11 show the experimental results. Focusing on the single  $V_{th}$   $0.07\mu$  technology implementation of each benchmark shown in Table 1, we see that our sleepy stack approach results in leakage power roughly equivalent to the other three leakage-reduction approaches in the same technology. Compared to the sleep and zigzag approaches, which do not save state, the sleepy stack approach results in up to 67% delay increase and up to 69% area increase. Thus, we do not recommend the sleepy stack approach with single- $V_{th}$  when state-preservation is not needed. Compared to the stack approach, which saves state, the sleepy stack approach results in up to 120% area increase, but the sleepy stack is up to 32% faster. Note that if we increase stack widths (thus decreasing transistor resistances), we obtain the results shown in Table 2. In this case, rather than improve, the stack approach shows increased leakage with a slight improvement in delay!

As explained in Section 4, the zigzag, sleep and sleepy stack approaches are also implemented using dual- $V_{th}$  technology. The main advantage of the sleepy stack approach over the stack approach is that dual- $V_{th}$  technology can be effec-

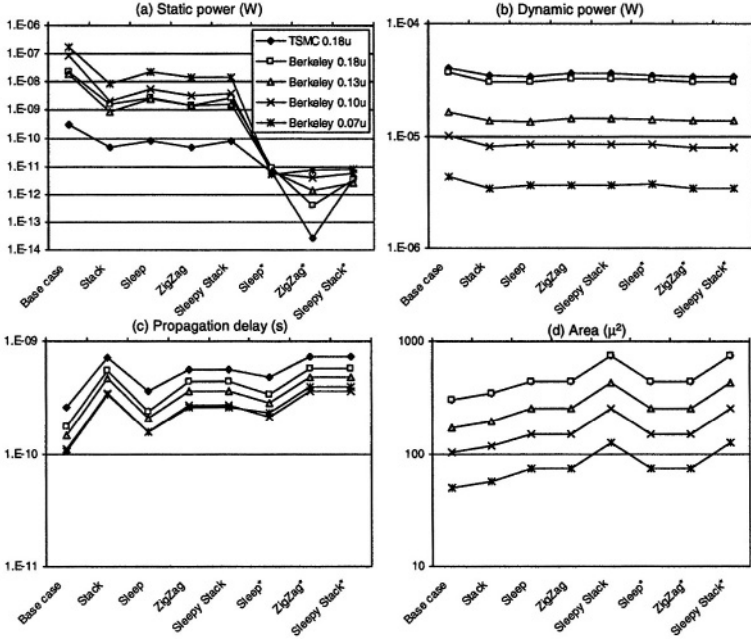


Fig. 10. Results of chain of 4-input multiplexers (\*dual  $V_{th}$ )

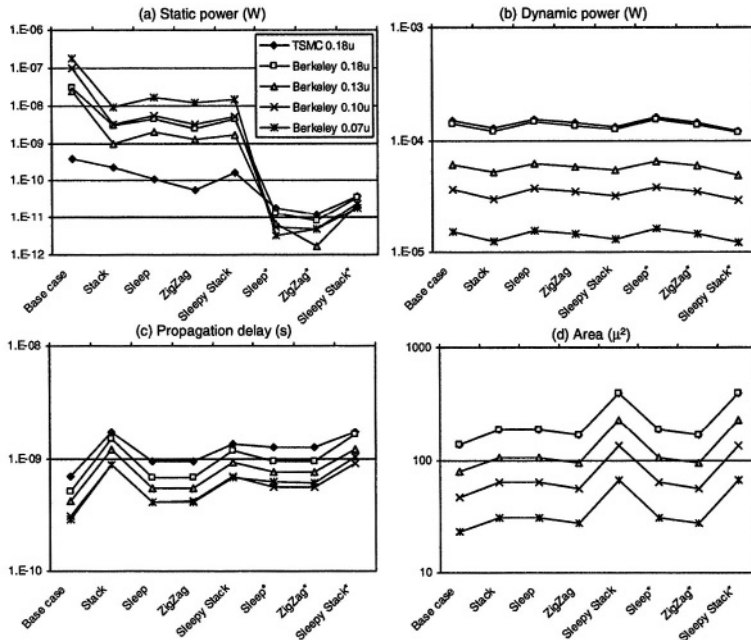


Fig. 11. Results of chain of 4-bit adders (\*dual  $V_{th}$ )

**Table 1.** Area, delay and power estimation (0.07 $\mu$ )

A chain of 3 inverters	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area ( $\mu^2$ )
Base case	4.61E-11	1.24E-08	6.56E-07	3.92
Stack	1.28E-10	9.89E-10	4.08E-07	4.48
Sleep	6.98E-11	2.40E-09	9.49E-07	8.00
ZigZag	5.99E-11	2.27E-09	1.05E-06	5.54
Sleepy Stack	8.75E-11	1.77E-09	6.35E-07	6.78
Sleep (dual Vth)	1.14E-10	4.32E-13	8.58E-07	8.00
ZigZag (dual Vth)	9.03E-11	3.84E-13	9.87E-07	5.54
Sleepy Stack (dual Vth)	1.38E-10	9.88E-13	4.88E-07	6.78

4-input multiplexer	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area ( $\mu^2$ )
Base case	1.05E-10	1.72E-07	4.35E-06	50.17
Stack	3.39E-10	8.63E-09	3.43E-06	57.40
Sleep	1.56E-10	2.24E-08	3.66E-06	74.11
ZigZag	2.58E-10	1.41E-08	3.64E-06	74.36
Sleepy Stack	2.58E-10	1.51E-08	3.64E-06	125.33
Sleep (dual Vth)	2.35E-10	5.03E-12	3.73E-06	74.11
ZigZag (dual Vth)	3.97E-10	7.54E-12	3.43E-06	74.36
Sleepy Stack (dual Vth)	3.97E-10	8.19E-12	3.43E-06	125.33

4-bit adder	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area ( $\mu^2$ )
Base case	2.91E-10	1.81E-07	1.52E-05	22.96
Stack	8.89E-10	9.25E-09	1.24E-05	30.94
Sleep	4.11E-10	1.69E-08	1.54E-05	30.94
ZigZag	4.06E-10	1.20E-08	1.47E-05	27.62
Sleepy Stack	6.79E-10	1.50E-08	1.31E-05	65.88
Sleep (dual Vth)	6.20E-10	3.31E-12	1.61E-05	30.94
ZigZag (dual Vth)	6.15E-10	4.92E-12	1.47E-05	27.62
Sleepy Stack (dual Vth)	1.03E-09	1.88E-11	1.22E-05	65.88

**Table 2.** Area, delay and leakage power with various stack width (a chain of 3 inverters, 0.07 $\mu$ )

	Base case	Stack (1X)	Stack (2X)	Stack (3X)	Stack (4X)	Sleepy stack (single Vth)	Sleepy stack (dual Vth)
Area ( $\mu^2$ )	3.92	4.48	5.37	6.27	7.17	6.78	6.78
Delay (S)	4.61E-11	1.28E-10	1.14E-10	1.10E-10	1.07E-10	8.75E-11	1.38E-10
Leakage (W)	1.24E-08	9.89E-10	1.98E-09	2.97E-09	3.96E-09	1.77E-09	9.88E-13

tively applied to the sleepy stack, resulting in three orders of magnitude reduction in leakage when compared to the stack approach as seen in Figs. 9(a), 10(a) and 11(a) with small (7~17%) associated increases in delay. Not surprisingly, the sleepy stack approach has 50~120% larger area as compared to the stack approach. Therefore, our sleepy stack approach with dual- $V_{th}$  can be used where state-preservation and ultra-low leakage power consumption are needed and are judged to be worth the area overhead.

One observation we notice from the results is that none of the approaches shows the best result in all criteria. Designers need to choose an appropriate technique for a given technology and a particular chip. Our sleepy stack approach provides a new low-power VLSI design technique to achieve significant leakage power reduction in deep sub-micron while achieving either (i) saving of state (unlike sleep and zigzag) or (ii) lower delay than a straightforward stack approach.

## 6 Conclusions

In nanometer scale CMOS technology, subthreshold leakage power consumption is a great challenge. Although previous approaches are effective in some ways, no perfect solution for reducing leakage power consumption is yet known. Therefore, designers choose techniques base upon technology and design criteria. Our novel sleepy stack, which combines the sleep and the stack approaches, is proposed as a new choice for logic designers. Furthermore, the sleepy stack is applicable to single and multiple threshold voltages. In conclusion, the sleepy stack combine some of the advantages of sleep transistors – most notably the effective use of dual- $V_{th}$  technology – with some of the advantages of the stack approach – most notably the ability to save state. As such, the sleepy stack approach represents a new weapon in the VLSI designer’s repertoire.

## References

1. N. S. Kim et al., “Leakage Current: Moore’s Law Meets Static Power,” *IEEE Computer*, Vol. 36, Issue 12, pp. 68-75, December 2003.
2. International Technology Roadmap for Semiconductors by Semiconductor Industry Association, <http://public.itrs.net>, 2002.
3. L. T. Clark et al., “An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications,” *IEEE Journal of Solid-State Circuits*, Vol. 36, No. 11, pp. 1599-1608, November 2001.
4. M. Powell, S.-H. Yang, B. Falsafi, K. Roy and T. N. Vijaykumar, “Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-submicron Cache Memories,” *International Symposium on Low Power Electronics and Design*, pp. 90-95, July 2000.
5. S. Mutoh et al., “1-V Power Supply High-speed Digital Circuit Technology with Multithreshold-Voltage CMOS,” *IEEE Journal of Solis-State Circuits*, Vol. 30, No. 8, pp. 847-854, August 1995.
6. Z. Chen, M. Johnson, L. Wei and K. Roy, “Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks,” *International Symposium on Low Power Electronics and Design*, pp. 239-244, 1998.
7. S. Narendra, S. Borkar, V. De, D. Antoniadis and A. Chandrakasan, “Scaling of Stack Effect and its Application for Leakage Reduction,” *International Symposium on Low Power Electronics and Design*, pp. 195-200, August 2001.
8. M. Johnson, D. Somasekhar, L-Y. Chiou and K. Roy, “Leakage Control with Efficient Use of Transistor Stacks in Single Threshold CMOS,” *IEEE Transactions on VLSI Systems*, Vol. 10, No. 1, pp. 1-5, February 2002.
9. K.-S. Min, H. Kawaguchi and T. Sakurai, “Zigzag Super Cut-off CMOS (ZSC-CMOS) Block Activation with Self-Adaptive Voltage Level Controller: An Alternative to Clock-gating Scheme in Leakage Dominant Era,” *IEEE International Solid-State Circuits Conference*, Vol. 1, pp. 400-401, February 2003.
10. K. Flautner, N. S. Kim, S. Martin, D. Blaauw and T. Mudge, “Drowsy Caches: Simple Techniques for Reducing Leakage Power,” *International Symposium on Computer Architecture*, pp. 148-157, May 2002.
11. Avant! Corporation, <http://www.avanticorp.com>.
12. Cadence Design Systems, <http://www.cadence.com>.

13. NC State University Cadence Tool Information, <http://www.cadence.ncsu.edu>.
14. Berkeley Predictive Technology Model (BPTM), <http://www-device.eecs.berkeley.edu/~ptm/>.
15. Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," *Proc. of IEEE CICC*, pp. 201-204, June 2000.
16. P. Pfeiffenberger, J. Park and V. Mooney, "Some Layouts Using the Sleepy Stack Approach," Technical Report GIT-CC-04-05, Georgia Institute of Technology, June 2004. <http://www.cc.gatech.edu/tech.reports/index.04.html>

# A Cycle-Accurate Energy Estimator for CMOS Digital Circuits

Eunseok Song, Young-Kil Park, Soon Kwon, and Soo-Ik Chae

School of Electrical Engineering & Inter-university Semiconductor Research Center,  
Seoul National University, Kwanak P.O. Box 34, Seoul 151-742, KOREA  
{dooly, zeroad, soonyk, chae}@sdgroup.snu.ac.kr  
<http://sdgroup.snu.ac.kr>

**Abstract.** Cycle-accurate energy estimation is very useful in reducing the energy consumption in the CMOS digital circuits, which was reported in [1]–[3]. This paper introduces a measurement-based energy estimator, describes a new energy calculation method for the CMOS circuits. We found that instead of representing all load capacitance as a capacitor to ground, we should separate the capacitance to ground and that to supply for the precise energy estimation. The verification of the measurement system and calculation method was done through the experiment for the inverter with a known load capacitance. Then, we measured the base cost of the instructions of an ARM7 processor, and 4-bit counters implemented with an Altera's APEX20K FPGA.

## 1 Introduction

There have been several methods to estimate per-cycle energy or power consumed in CMOS digital circuits [4]–[7]. Simulator-based energy estimation methods are simple and popular. Low-level energy estimators are accurate but slow for a complex circuit. On the other hand high-level simulators are fast but inaccurate.

We can measure the energy consumed in CMOS digital circuits by integrating the instantaneous current value for a given period when the supply voltage is constant. However, what we can measure with an ammeter is not an instantaneous current but its average or RMS current for the circuit under measurement. A simple method to measure the consumed energy using capacitors and switches was recently proposed by Chang [1]–[3]. By using a measurement-based energy estimator it is possible to make fast and accurate energy estimation for a complex CMOS circuit.

For the precise estimation of consumed energy it is very important to establish the proper energy calculation model for the CMOS circuits. In this paper we present a new energy calculation method that is appropriate to the capacitor-based energy estimator. This paper organized as follows. In section 2, we explain the capacitor-base energy estimator. In section 3, we describe a new cycle-accurate energy calculation method. The experimental results using the energy measurement system are described in section 4. Finally the conclusion is given in section 5.



## 2 Capacitor-Based Energy Estimator

A simple solution to measure the energy consumed in CMOS circuits is measuring the current with an ammeter. However, most of the available ammeters today can measure only the RMS or average value for a certain time interval due to their bandwidth limitation. Therefore, the ammeters are not appropriate to cycle-accurate energy measurement of the CMOS VLSI circuits.

Another method is measuring the voltage drop by using the capacitors and switches. It can integrate high frequency components inherently, as shown in Fig. 1. First, the capacitor  $C_M$  is charged to a certain reset level before arriving a clock edge. Although the current is spiky when the energy is consumed in the chip, we can calculate the consumed energy by measuring the voltage drop of  $V_M$ . The energy transferred into the chip can be written as follows,

$$E = \frac{1}{2} C_M V_1^2 - \frac{1}{2} C_M V_2^2$$

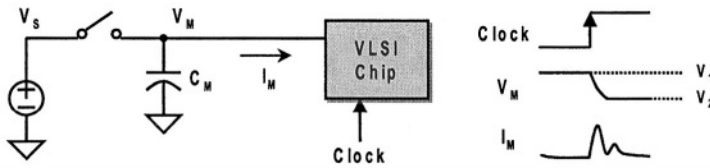


Fig. 1. Energy measurement using a capacitor and a switch.

The transferred energy is consumed or stored in the chip under measurement. A typical block diagram of the energy measurement system using a capacitor and two switches is shown in Fig. 2. Capacitor  $C_L$  represents the internal load capacitance, which varies after every clock edge. Capacitor  $C_B$  represents the total capacitance connected to the internal power line, including the on-chip decoupling capacitance, which also varies after every clock edge. At first, switch  $S_1$  is closed before arriving a clock edge, which charges capacitor  $C_M$  to the supply voltage level  $V_s$ . Switch  $S_2$  is open during charging capacitor  $C_M$ , which separates the chip from the supply  $V_s$ . After  $C_M$  is charged, switch  $S_1$  is open and switch  $S_2$  is closed. Then charge sharing between  $C_M$  and  $C_B$  will occur and we can calculate the value of  $C_B$  because  $C_M$  is the capacitor with a known value. After charge sharing, a clock edge is applied to the test digital IC introducing the voltage drop of node  $V_M$  according to the amount of current consumption in the test IC. An amplifier is employed to amplify the voltage drop of  $V_M$  to meet the ADC input range of 1 or 2 volts. The voltage drop of  $V_M$  should be limited not to disturb the proper operation of the chip. Consequently, the voltage drop of  $V_M$  must be limited, which can be controlled by adjusting the size of capacitor  $C_M$ . For example, for the supply voltage of 1 to 2 volt, it should be within 100~200mV. The digital data from the ADC is stored in real time in memory. Later, the host computer calculates the consumed energy using the ADC data.

Fig. 3 shows a timing diagram of the control signals in the measurement system in Fig. 2. When  $S_1$  is closed,  $V_M$  is charged as  $V_s$ . When  $S_2$  is closed, charge sharing between  $C_M$  and  $C_B$  occurs, which induces a small voltage drop of  $V_M$ . Then, when a clock edge arrives, the chip draws a relatively large current, which makes another

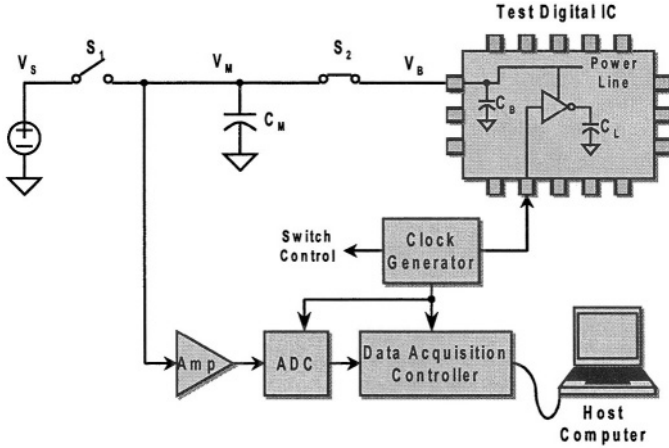


Fig. 2. Cycle-accurate energy measurement system using a capacitor and two switches.

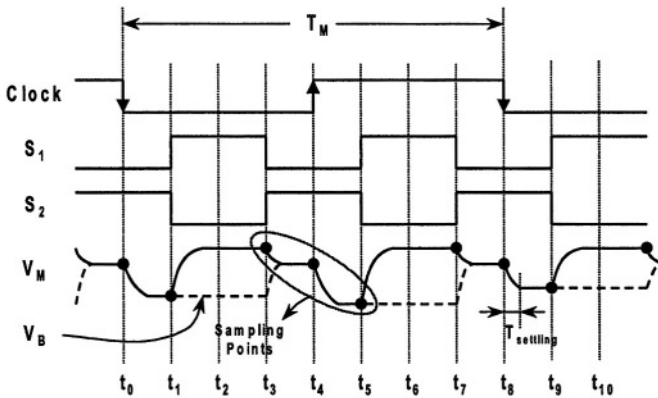


Fig. 3. Timing diagram of cycle-accurate energy measurement system.

voltage drop of  $V_M$ . Three sampling points for a clock edge are necessary for estimating the consumed energy. Therefore, the clock frequency should be lowered to ensure enough settling time for both recharging and sampling  $V_M$  for each clock edge.

### 3 Cycle-Accurate Energy Calculation

#### 3.1 Previous Method

In the previous method [1]–[3], the energy transferred from the capacitors  $C_M$  and  $C_B$  into the load capacitor  $C_L$  is calculated. If the sampled values of  $V_M$  before and after a clock edge are represented as  $V_1$  and  $V_2$ , respectively, then we can calculate the consumed energy,

$$E_1 = \frac{1}{2}(C_M + C_B)(V_1^2 - V_2^2)$$

assuming that the capacitors  $C_M$  and  $C_B$  are constant. However, note that  $C_B$  can change for every clock edge.

To calculate the consumed energy for the rising edge at  $t=t_4$ , capacitor  $C_M$  is charged to the reset level  $V_s$  after  $S_1$  is closed at  $t=t_1$ . Then, when  $S_2$  is closed at  $t=t_3$ , charge sharing occurs between  $C_M$  and  $C_B$ . From the charge conservation law (CCL),

$$C_M V_M(t_3) + C_B(t_3) V_B(t_3) = (C_M + C_B(t_4)) V_M(t_4)$$

Assuming that no leakage current exist in the device under measurement, it is clear that  $V_B(t_3)=V_B(t_1)=V_M(t_1)$ . Furthermore,  $C_B(t_3) = C_B(t_4)$  because  $C_B$  does not change before arriving a clock edge. Therefore,

$$C_M V_M(t_3) + C_B(t_4) V_M(t_1) = (C_M + C_B(t_4)) V_M(t_4)$$

Because  $C_M$  is a known constant, capacitor  $C_B$  can be written as

$$C_B(t_4) = \frac{V_M(t_3) - V_M(t_4)}{V_M(t_4) - V_M(t_1)} C_M$$

Similarly,

$$C_B(t_5) = \frac{V_M(t_7) - V_M(t_8)}{V_M(t_8) - V_M(t_5)} C_M$$

Consequently, the consumed energy after a clock edge at  $t=t_3$  can be written as

$$E_1 = \frac{1}{2}(C_M + C_B(t_4)) V_M^2(t_4) - \frac{1}{2}(C_M + C_B(t_5)) V_M^2(t_5)$$

Note that because it includes the measurement distortion due to the measurement setup, it is not exactly equal to the energy consumed in the chip that is in the real operation setup.

Another method is to calculate the energy transferred into the load capacitor after a clock edge. We first calculate the load capacitance  $C_L$  per clock edge, which does not include any distortion due to the measurement setup, assuming that the load capacitance  $C_L$  does not depend on the voltage. Then, we can calculate the energy transferred after the edge as follows,

$$E_2 = C_L V_{DD}^2 \quad (1)$$

$V_{DD}$  in Eq. (1) does not mean the  $V_s$  in the measurement setup but the supply voltage used in the real application setup. A half of the transferred energy is consumed as a heat in the MOS transistors and the rest of it is stored in many load capacitors in a complex CMOS VLSI circuit. However, the stored energy of 50% is consumed partially when the logic state of an output, where the energy is stored in its load capacitance, is changed.

To find  $C_L$ , we first calculate  $C_B$ , which is the same with  $C_B$  in model 1. After that, we can write CCL for the rising edge at  $t=t_4$ ,

$$(C_M + C_B(t_4))V_M(t_4) = (C_M + C_B(t_5) + C_L(t_5))V_M(t_5)$$

Then,  $C_L(t_5)$  can be written as,

$$C_L(t_5) = \frac{(C_M + C_B(t_4))V_M(t_4) - (C_M + C_B(t_5))V_M(t_5)}{V_M(t_5)} \quad (2)$$

With Eq. (1) and Eq. (2), we can calculate the energy transferred to the chip after a clock edge.

### 3.2 Proposed Method

In Eq. (1), all load capacitance is represented as a capacitor to ground, which is ok only for timing analysis. However, for exact energy analysis, we should model the load capacitance that includes both capacitors to ground and capacitors to supply in a CMOS VLSI circuit. There are four cases for the state transitions of the load capacitance, which are modeled as the circuit shown in Fig. 4, where  $n$  denotes to the transition for the  $n$ -th clock edge.

Because the capacitors in groups 1 and 2 in Fig. 4 are static for the  $n$ -th clock edge and they can be included in the internal bypass capacitor  $C_B$ . We can define all the static capacitors as  $C_H(n)$  for simplicity. In Fig. 4, the energy transferred to the chip at the  $n$ -th clock edge is written as follows,

$$E_{transferred} = (C_6(n) + C_7(n)) \cdot V_{DD}^2$$

The timing diagram for the energy measurement system is redrawn in Fig. 5. After  $S_2$  is closed, the  $n$ -th clock edge arrives. We can employ the CCL for the  $n$ -th clock edge as follows,

$$(C_M + C_H(n))V_2(n) = (C_M + C_{H67}(n))V_3(n) + \frac{I_{leak}T_M}{8} \quad (3)$$

where  $C_{H67}(n) = C_H(n) + C_6(n) + C_7(n)$ . In Eq. (3),  $I_{leak}$  is the leakage current and  $T_M$  is the clock period under measurement. Then,

$$C_H(n) = \frac{(C_M + C_{H67}(n))V_3(n) + \frac{I_{leak}T_M}{8}}{V_2(n)} - C_M \quad (4)$$

Here,  $C_{H67}(n)$  can be obtained from the measurements for the  $(n+1)$ -th edge. When  $S_2$  is closed prior to the  $(n+1)$ -th edge, by employing the CCL,

$$C_M V_1(n+1) + C_{H67}(n)V_3(n) = (C_M + C_{H67}(n))V_2(n+1) + \frac{3I_{leak}T_M}{8}$$

Therefore,

$$C_{H67}(n) = \frac{[V_1(n+1) - V_2(n+1)]C_M - \frac{3I_{leak}T_M}{8}}{V_2(n+1) - V_3(n)} \quad (5)$$

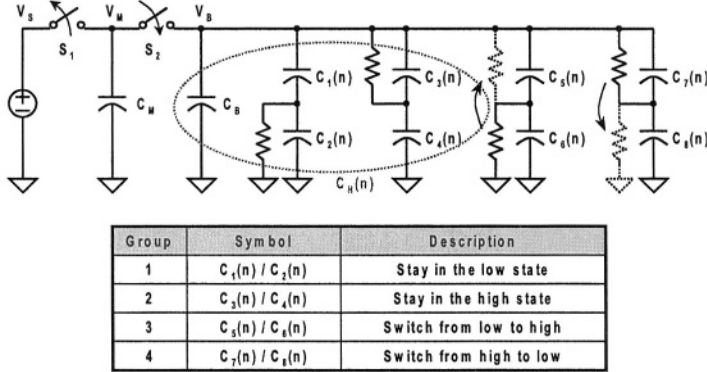


Fig. 4. Circuit modeling and four cases according to the output transition at the nth clock edge.

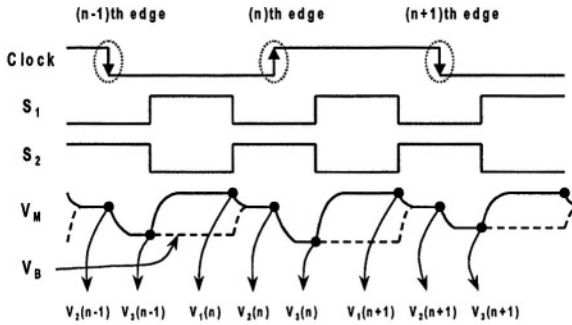


Fig. 5. Timing diagram for the energy measurement system.

From Eqs. (4) and (5), we can write  $C_{67}(n)$

$$\begin{aligned}
 C_{67}(n) &= C_{H67}(n) - C_H(n) \\
 &= \left(1 - \frac{V_3(n)}{V_2(n)}\right) \left(\frac{V_1(n+1) - V_3(n)}{V_2(n+1) - V_3(n)}\right) C_M \\
 &\quad - \frac{V_3(n)[V_2(n+1) - V_3(n) - 3] + 3V_2(n)}{V_2(n)[V_2(n+1) - V_3(n)]} \cdot \frac{I_{leak} T_M}{8}
 \end{aligned}$$

Consequently, the energy transferred to the chip at the n-th clock edge can be written as

$$E_{transferred}(n) = C_{67}(n) V_{DD}^2$$

## 4 Experimental Results

### 4.1 Inverter with a Known Load Capacitance $C_L$

First, we tested the proposed method by estimating the load capacitance for an inverter with a known load capacitance  $C_L$ . Two load capacitors of 430pF are used in this experiment: one to ground and the other to supply line. We used the large  $C_L$  in order to minimize the effect of the parasitic capacitance. And the clock frequency we used is 625kHz, which is slow enough to satisfy the settling time constraints.

Experimental results are summarized in Table 1. The data in Table 1 are the average value of 1000 samples for each  $C_M$ . The error in the previous method varies as the  $C_M$ . It is because the smaller  $C_M$  results in the larger voltage swing of  $V_M$  node. If the large  $C_M$  is used, the error of previous method can be reduced but the SNR of the measure system becomes small.

**Table 1.** Experimental result: % error vs.  $C_M$

$C_M$ [nF]	% error			
	Previous method		Proposed method	
	$C_L$	Energy	$C_L$	Energy
5.3	-	-6.1	0.2	0.2
6.8	-	-5.3	-0.5	-0.5
8.2	-	-4.5	-0.3	-0.3
10	-	-3.8	-0.5	-0.5
22	-	-2.6	-0.5	-0.5

The error in the proposed method was much smaller than those in the previous method because the proposed method uses the energy estimation based on the load capacitor calculation. Note that the error of energy in Table 1 is the same with that of  $C_L$  because we use the energy calculated from  $C_L V_{DD}^2$  as a reference.

### 4.2 APEX20K FPGA: Binary Counter

Next we measured the energy profile for a 4-bit binary counter implemented in an APEX20K FPGA chip, as shown in Fig. 6, where the x-axis represents the count output sequence. We found that about 90% of the consumed energy was used by the clock tree in the APEX20K FPGA chip. Therefore each energy value was obtained by averaging for the 100 measurements after subtracting the energy consumed by the clock trees in this experiment. Note that the energy consumed by the clock tree was estimated with the energy measured at the falling edges because only the activities of the clock tree exist at the falling edges in a synchronous circuit with positive edge triggering if its clock is slow enough. With the microscopic view of the energy consumption in Fig. 6, we confirmed that the consumed energy is roughly proportional to the Hamming distance of the two consecutive codes in the output sequence of the counter.

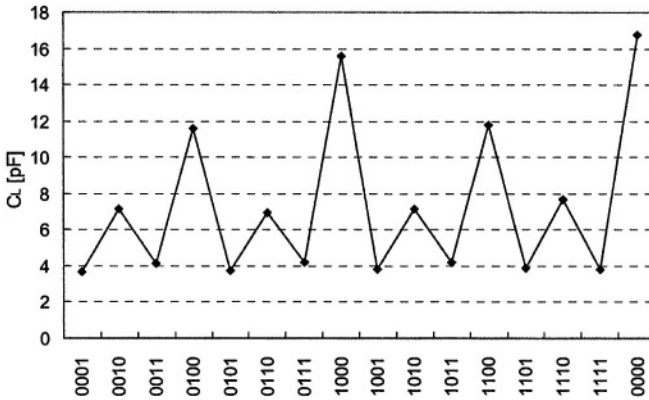


Fig. 6. Experimental results for a 4-bit binary counter.

### 4.3 ARM7: Base Energy Cost of Instruction

In order to apply our measure system to the more complex digital circuit, we measured the energy consumption of an instruction in an ARM7 processor. The histograms of the energy consumed by ADD and LDR instructions, which are constructed with 1000 measured data for each distribution, are shown in Fig. 7. Note that in Fig. 7, the standard deviation of the distribution for an instruction with random operands is significantly larger than that for the instruction with a fixed operand. We conjectured that the standard deviation for a fixed operand is mainly due to the noise of the measurement system, which corresponds to an error of about 1% [3].

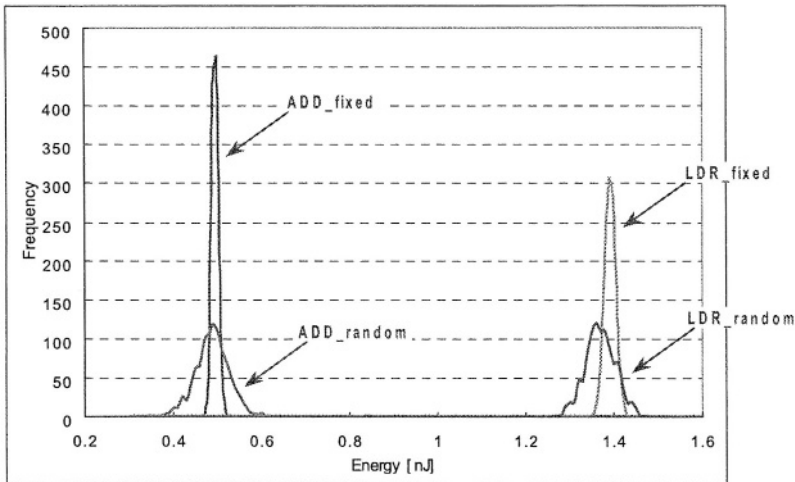


Fig. 7. Base cost of ADD: random data vs. fixed data.

**Table 2.** Base energy cost and inter-instruction overhead of several typical instructions of the ARM7TDMI chip

Instruction	Pipeline stage	Base energy cost [pJ] ( $M / \sigma$ )	Inter-instruction overhead [pJ] ( $M / \sigma$ )					
			ADD	SUB	MOV	AND	STR	LDR
ADD	3	482.6 / 38.9		41.2 / 3.5	69.3 / 5.4	26.0 / 1.6	17.7 / 0.8	24.5 / 0.9
SUB	3	469.2 / 40.1	41.2 / 3.5		56.5 / 4.4	21.4 / 1.8	29.1 / 1.8	36.4 / 1.3
MOV	3	480.7 / 36.6	69.3 / 5.4	56.5 / 4.4		58.1 / 5.9	45.5 / 4.6	35.5 / 1.8
AND	3	476.0 / 37.9	26.0 / 1.6	21.4 / 1.8	58.1 / 5.9		16.3 / 0.8	32.9 / 1.6
STR	4	915.9 / 38.0	17.7 / 0.8	29.1 / 1.8	45.5 / 4.6	16.3 / 0.8		7.8 / 0.3
LDR	5	1364.1 / 38.0	24.5 / 0.9	36.4 / 1.3	35.5 / 1.8	32.9 / 1.6	7.8 / 0.3	

Table 2 shows the base energy cost and inter-instruction energy overhead of the instructions of an ARM7TDMI chip. The mean values and standard deviations in Table 2 are obtained from 1000 measurements for the random operands. The average base energy cost of an instruction in the ARM7 chip is significantly higher if the number of its pipeline stage is large. The inter-instruction energy overhead in Table 2 is the additional energy consumption incurred from changing the instructions. For simple calculation of the inter-instruction energy overhead, we assumed that the inter-instruction overhead for a pair of instructions is independent of their order although it may not be true in general. In Table 2, the inter-instruction energy overheads for an instruction are less than about 15% of its base energy cost.

## 5 Conclusion

Cycle-accurate energy estimator is useful for energy optimization in both the embedded software development and low-power SOC design. For the precise energy estimation, correct circuit modeling and analysis are essential. Therefore, in this paper we proposed a new calculation method that can be used for cycle-accurate energy measurement systems and derived their energy equations for a clock edge.

After comparing their errors through the experiments for an inverter, we found that simple load capacitance model to ground is not good enough for energy analysis. Instead, we should separate the capacitance to ground and that to supply for accurate energy analysis in implementing an accurate joulemeter.

In this paper we presented that the capacitor-based energy estimator can be useful for energy optimization of CMOS VLSI circuits through the experiments with FPGA and ARM7 processor.



## References

1. Naehyuck Chang, Kwan-Ho Kim, and Hyung Gyu Lee, "Cycle-accurate energy consumption measurement and analysis: case study of ARM7TDMI," *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 185-190, July 2000.
2. Naehyuck Chang, and Kwan-Ho Kim, "Real-time per-cycle energy consumption measurement of digital systems," *IEE Electronics Letters 22nd June 2000*, vol. 36, pp. 1169-1171.
3. Naehyuck Chang, Kwan-Ho Kim, and Hyung Gyu Lee, "Cycle-accurate energy measurement and characterization with a case study of the ARM7TDMI," *IEEE Transactions on VLSI Systems*, vol. 10, pp. 146-154, April 2002.
4. Davide Sarta, Dario Trifone, and Giuseppe Ascia, "A data dependent approach to instruction level power estimation," *Proceedings of IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, pp. 182-190, 1999.
5. R. Yu Chen, R. M. Owens, M. J. Irwin, and R. S. Bajwa, "Validation of an architectural level power analysis technique," *Proceedings of 35th Design Automation Conference*, pp. 242-245, June 1998.
6. Tajana Simunic, Luca Benini, and Giovanni De Micheli, "Cycle-accurate simulation of energy consumption in embedded systems," *Proceedings of 36th Design Automation Conference*, pp. 867-872, June 1999.
7. Eunseok Song, In-Chan Choi, Young-Kil Par, and Soo-Ik Chae, "A cycle-accurate joulemeter for CMOS VLSI circuits," *European Solid-State Circuits Conference*, pp. 619-622, Sep. 2003.

# Leakage Reduction at the Architectural Level and Its Application to 16 Bit Multiplier Architectures

Christian Schuster<sup>1</sup>, Jean-Luc Nagel<sup>1</sup>, Christian Piguet<sup>2</sup>, and Pierre-André Farine<sup>1</sup>

<sup>1</sup>IMT, University of Neuchâtel, Switzerland

{Christian.Schuster, jean-luc.nagel, pierre-andre.farine}@unine.ch

<sup>2</sup>CSEM, Neuchâtel, Switzerland

Christian.piguet@csem.ch

**Abstract.** In very deep submicron technologies ( $< 0.13 \mu\text{m}$ ) the leakage power consumption becomes an important contribution to total power consumption. Consequently a new low-power design methodology will be required at circuit, architectural and system levels. This paper focuses on architecture comparison and aims at selecting the one with the minimum total power consumption by simultaneously optimizing static and dynamic power dissipation. This optimal power has been estimated for eleven 16-bit multiplier architectures.

## 1 Introduction

Predictions of the SIA Roadmap [1] forecast supply voltage ( $V_{dd}$ ) as low as 0.8 to 0.5 V in year 2018. Since a reduction of  $V_{dd}$  requires an additional reduction of the transistor threshold voltage ( $V_{th}$ ) to maintain speed, this will result in an exponential increase of the static power consumption. As a consequence the static power on its own may already exceed the required maximum total power consumption in stand-by mode for certain applications. When the activity is very low, i.e. when very few gates are switching during a clock cycle, the static power will become a significant contributor to the total power consumption.

Table 1 summarizes predictions of the main parameters required for the next generations of Bulk MOSFET devices in 2018, related to High Performance (HP), Low Operating Power (LOP) and Low Standby Power (LSTP) devices [1][2]. LSTP transistors have high  $V_{th}$  and present thus reduced leakage compared to other devices. Conversely, HP transistors present the largest drive current for maximum speed but suffer from a very large leakage.

**Table 1.** Predicted (year 2018) MOS characteristics for High Performance (HP), Low Operating Power (LOP) and Low Standby Power (LSTP) devices (2003 SIA Roadmap)

MOS Type	Physical length [nm]	Supply Voltage [V]	Threshold Voltage [V]	Ion [ $\mu\text{A}/\mu\text{m}$ ]	Ioff [ $\text{nA}/\mu\text{m}$ ]	Ion/Ioff
HP	7	0.7	0.11	2190	500	4380
LOP	9	0.5	0.17	950	30	31'667
LSTP	10	0.8	0.4	990	0.1	9'900'000

New design methodologies taking into account the contribution of leakage will be *required* for the conception of innovative System-on-Chips realized in future very deep sub-micron technologies (e.g. 18 nm in 2018), noticing that leakage gains already in importance for state-of-art technologies (e.g. 90 nm in 2004). Similarly to previous design methodologies, solutions can be proposed at different levels, namely at *circuit*, *architecture*, and *system* levels.

At the circuit level many techniques have been proposed:

- Multi  $V_{th}$  technology, with fast low  $V_{th}$  transistors on critical paths and slow high  $V_{th}$  transistors outside critical paths (MTCMOS, Multiple Threshold) [3][4][5][6]
- Electrical regulation of  $V_{th}$  (VTCMOS, Variable Threshold, SATS) [7] [8] [9]
- Gated-Vdd, with  $V_{dd}$  switched off in sleeping mode [11][12][13].
- DTMOS (Dynamic  $V_{th}$ ) with transistor bodies connected to MOS gates [10][11]

Only few publications can be found on the reduction of leakage at architectural level. It is however obvious that some digital architectures are better than others in terms of leakage. The next two sections describe trade-offs between dynamic and static power from a theoretical point of view with the aim of understanding the influence of architectural parameters on this trade-off and on total power consumption. The last part of the paper compares several implementations of 16 bit multipliers and how they compare in terms of total power consumption.

## 2 Optimal Total Power Consumption

The total power dissipation is defined as the sum of dynamic and static contributions:

$$\begin{aligned} P_{tot} &= P_{dyn} + P_{stat} = a \cdot N \cdot C \cdot V_{dd}^2 \cdot f + V_{dd} \cdot N \cdot I_s \cdot \exp(-V_{th} / nU_t) = \\ &= V_{dd} \cdot N \cdot (a \cdot C \cdot f \cdot V_{dd} + I_s \cdot \exp(-V_{th} / nU_t)) \end{aligned} \quad (1)$$

with  $N$  number of cells;  $a$  average cell activity (i.e. the number of switching cells in a clock cycle over the total number of cells);  $C$  equivalent cell capacitance;  $f$  operating frequency;  $I_s$  average transistor current per cell for  $V_{gs} = V_{th}$ ;  $n$  slope in weak inversion;  $U_t = kT/q$  thermal voltage.

Note that power analysis was performed at the library cell level. For this reason, short circuit power was not distinguishable from capacitive switching power. Consequently, our model considers dynamic power as the sum of switching and short circuit power, but this total power is considered as switching power. Subthreshold leakage was considered in static power, while gate and junction leakage were neglected in this first part of the project. Considering the delay of a particular cell to be constant, the parameters  $V_{dd}$  and  $V_{th}$  can be related:

$$CellDelay \propto \frac{C_L V_{dd}}{(V_{dd} - V_{th})^\alpha} = cst \quad (2)$$

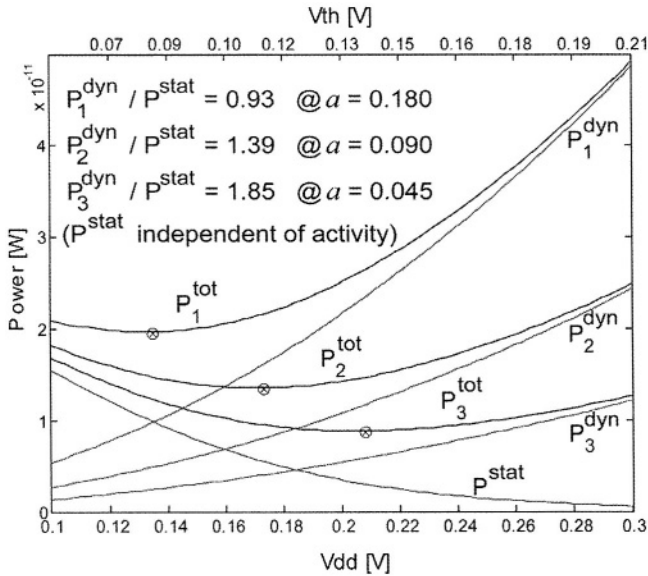
with  $C_L$  cell load capacitance;  $\alpha$  velocity saturation factor.

Therefore for every value of  $Vdd$ , there is one unique corresponding  $Vth$ , depending on speed requirements, which can be expressed as a function of  $Vdd$ :

$$Vth(Vdd) = Vdd - \chi Vdd^{1/\alpha} \quad (3)$$

where  $\chi$  is a parameter depending on  $C_L$ , logical depth (LD), frequency and technology parameters.

For any given architecture it is therefore possible to reduce  $Vdd$  and  $Vth$  while maintaining a constant throughput. Whereas  $P_{dyn}$  decreases in a quadratic manner with the reduction of  $Vdd$ ,  $P_{stat}$  increases instead (due to reduced  $Vth$ ). The total power presents thus a minimum i.e. an *optimal* power consumption (marked with a  $\otimes$  symbol in Fig. 1).



**Fig. 1.** Total Power ( $P_{tot}$ ), dynamic Power ( $P_{dyn}$ ) and static Power ( $P_{stat}$ ) of a single square transistor, as a function of  $Vdd$  and  $Vth$  for three different circuit activities  $a$

### 3 Dynamic over Static Power Ratio at the Optimum $Vdd$ and $Vth$

As can be observed in Fig. 1 the ratio of dynamic ( $P_{dyn}$ ) over static ( $P_{stat}$ ) power at the minimum of total power consumption is not constant and depends on some parameters e.g. the activity as illustrated in Fig. 1, but also on the logical depth, working frequency, etc. Without loss of generality we can write:

$$P_{dyn}^{opt} = k_1 \cdot P_{stat}^{opt} \quad (4)$$

where  $k_1$  is expected to depend in a complex fashion on architectural parameters.

At the optimal point, the circuit works at its maximal frequency, i.e. the entire clock period is used and the last logical cell on the critical path switches just before the end of the clock cycle. This case is the optimal one, otherwise  $Vdd$  could be further reduced to improve power saving. Consequently, the frequency can be rewritten as:

$$f = f_{\max} = \frac{I_{on}}{LD \cdot C \cdot Vdd} \quad (5)$$

So the ratio between  $P_{dyn}$  and  $P_{stat}$  (4) becomes:

$$a \cdot N \cdot C \cdot Vdd^2 \frac{I_{on}^{opt}}{LD \cdot C \cdot Vdd} = k_1 \cdot N \cdot Vdd \cdot I_{off}^{opt} \quad (6)$$

$$\frac{I_{on}^{opt}}{I_{off}^{opt}} = k_1 \cdot \frac{LD}{a}$$

A first approximation [16, 17] was obtained by considering that the optimum of the total power consumption is about the same amount of static and dynamic power ( $k_1=1$ ). From the result shown in Fig. 1 this approximation is clearly too rough.

The parameter  $k_1$  can be obtained by searching the minimum of  $P_{tot}(Vdd)$  as:

$$\frac{\partial P_{tot}(Vdd)}{\partial Vdd} = \frac{\partial P_{dyn}(Vdd)}{\partial Vdd} + \frac{\partial P_{stat}(Vdd)}{\partial Vdd} = 0 \quad (7)$$

Combining (3) and (7) leads to:

$$k_1 = \frac{P_{dyn}^{opt}}{P_{stat}^{opt}} = \frac{(\alpha - 1)(\eta + 1)Vdd^{opt} + Vth^{opt}}{2nUt\alpha} - \frac{1}{2} \quad (8)$$

with  $\alpha$  velocity saturation factor of the alpha power law;  $\eta$  drain induced barrier lowering (DIBL) coefficient;  $n$  slope in weak inversion.

Introducing the numerical values of a  $0.18\mu\text{m}$  technology (UMC L180 process) in (8), the expression of  $k_1$  (which has a scalar size) becomes (with  $\alpha=1.5$ ,  $n=1.3$ ,  $T=300\text{K}$ ,  $\eta=0.033$ ):

$$k_1 \approx 5 \cdot Vdd^{opt} + 10 \cdot Vth^{opt} - 0.5 \quad (9)$$

Equation (9) shows that  $k_1$  linearly depends on  $Vdd^{opt}$  and on  $Vth^{opt}$  that in turn depend on LD and  $a$ . Consequently  $k_1$  is as well depending on  $a$  and LD. This can be observed in Fig. 2 (left) where  $k_1$  is obtained from Eq. 9 and is compared to a  $k_1$  obtained from simulation using  $P_{dyn}$  over  $P_{stat}$ . Section 0 describes in more details how  $P_{dyn}$  and  $P_{stat}$  are extrapolated from power analysis at nominal conditions, but it can already be observed that simulation and calculation are in good agreement.

Interestingly, for high activity ( $a > 0.1$ ),  $k_1$  decreases quasi linearly with  $a$ , while for low activity ( $a < 0.02$ ),  $k_1$  decreases exponentially with  $a$ . The dependency of  $k_1$  vs. LD can be similarly plotted (right part of Fig.2). It can be observed that  $k_1$  is quasi linearly proportional to the logical depth. When combining both dependencies (i.e.

left and right parts of Fig.2), it can be concluded that a little  $LD/a$  ratio (e.g. small  $LD$  and large  $a$ ) results in a small  $k_1$ , and hence, referring to (6), an even smaller  $I_{on}/I_{off}$ .

The  $a$  and  $LD$  parameters can be altered by adjusting the degree of pipelining and parallelism of a given architecture. Every different architecture will thus exhibit a certain ratio  $I_{on}/I_{off}$  at the optimum of the total power consumption. This ratio  $I_{on}/I_{off}$  is generally much smaller than the values reported in Table 1 and more likely around 100 to 1'000 for the best architecture. According to Eq. 6, this would correspond to a high activity  $a$  and a small logical depth  $LD$ . Parallelizing a circuit reduces  $LD$  and the activity, so that  $LD/a$  remains approximately constant, but the behavior of  $k_1$  depends on the working point on the curves illustrated in Fig. 2. For example, reducing the activity from 0.16 to 0.08 has a much smaller impact on  $k_1$  than reducing the activity from 0.02 to 0.01. This tends to confirm that having architectures with very low activities (i.e. heavily parallel) is probably not a good choice. Pipelining an architecture reduces the  $LD$  but does not modify  $a$ . The  $LD/a$  ratio is consequently reduced, while  $k_1$  is similarly reduced as can be seen in Fig. 2. Indeed, pipelining should be preferred to parallel structures as long as register overhead remains small.

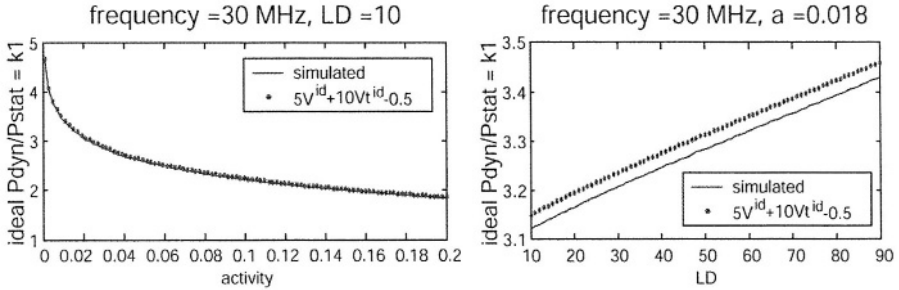


Fig. 2. Example of  $k_1 = P_{dyn}/P_{stat}$ . Left:  $k_1$  vs. activity factor; right:  $k_1$  vs. logical depth ( $LD$ )

## 4 Comparison of a Variety of Multiplier Architectures

Provided that different architectures present different minima of total power consumption, there is presumably one optimal architecture with the smallest total power consumption. This optimal architecture will be characterized by a reduced leakage so that the dynamic power consumption can be reduced by decreasing  $V_{dd}$  and  $V_{th}$ .

The same logical function (arithmetic, filter banks, microprocessors, data paths, etc.) can be implemented using different architectures varying in their degree of parallelization and pipelining, length of critical path, number of transistors, etc. The following design parameters are proposed to compare digital architectures regarding total power and leakage: activity factor ( $a$ ), logical depth ( $LD$ ), the number of cells ( $N$ ), the equivalent cell capacitance ( $C$ ) and the ratio  $I_{on}/I_{off}$ .

Eleven 16 bit multipliers architectures have been described and synthesized. Their power consumption was analyzed using the same stimuli for all circuits and the optimal  $V_{dd}$  and  $V_{th}$  were estimated.

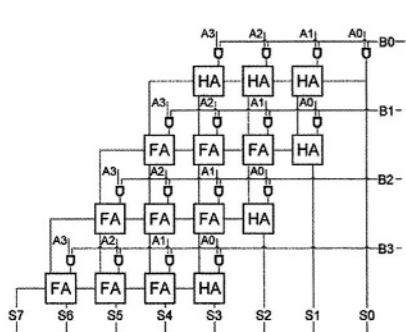
The test circuits properly populate the space spanned by the above parameters, and can be classified as:

- *Ripple Carry Array* or RCA (basic, pipelined 2, pipelined 4, parallel 2, parallel 4): the basic implementation is constructed as an array of 1-bit adders (see for instance Fig. 3(a) for a 4x4 multiplier). The speed of this structure is limited by the carry propagation. The two pipelined versions (2 or 4 stages) use registers in the critical path so that the logical depth is directly shortened by the number of stages, as shown in Fig. 3(b). Finally, both parallelized versions (factor of 2 or 4) are obtained by replicating the basic multiplier and multiplexing data at each clock across different multipliers (see Fig. 3(c)). This way, the multiplier has additional clock cycles at its disposal relaxing timing constraints.
- *Wallace Tree* (basic, parallel 2, parallel 4): the Wallace Tree structure adds the partial products (P0-P7 in Fig. 3(d)) using Carry Select Adders in parallel. Path delays are better balanced than in RCA, resulting in an overall faster architecture. Parallelized versions use a circuit replica as for the RCA structure.
- *Sequential* (basic, 4x16 Wallace, parallel): the basic implementation computes the multiplication with a sequence of “add and shift” resulting in a very compact circuit. The intermediate result is shifted and added to the next partial product, and stored in a register (see Fig. 3(e)). This type of structure needs as many clock cycles to complete as the operand width, but only one 16-bit adder is necessary. The architecture called 4x16 Wallace reduces the number of clock cycles per multiplication from 16 to 4 by using a 4x16 Wallace tree multiplier i.e. by adding 4 partial products in parallel (see Fig. 3(f)). The parallelized version is also a simple replica of the basic version.

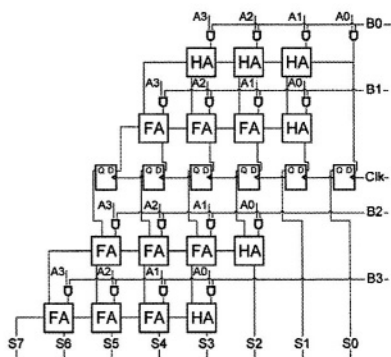
For all the architectures, the throughput has been fixed to 31.25 MHz. All architectures have been synthesized with Synopsys using Virtual Silicon Technology library for the UMC 0.18 $\mu$ m 1.8V process. Static and dynamic power consumption were then estimated by back-annotating the cell activities with a test bench generating the same random stimuli for all circuits. The power consumption was then extrapolated at different  $V_{dd}$  and  $V_{th}$  using Eq. 1 and Eq. 3. The  $V_{dd}$  and  $V_{th}$  providing the minimum total power consumption were finally found numerically.

Results are summarized on Table 2. The architecture providing the lowest total power consumption is the structure based on the Wallace tree, operating at very low  $V_{dd}$  and  $V_{th}$ . As could be expected at these low  $V_{dd}$  and  $V_{th}$ , the  $I_{on}/I_{off}$  ratio for optimal total power consumption is rather small, approximately from 100 to 1'000 (compared to that of LOP devices reported in Table 1). Moreover, the percentage of static power over the total power is between 15% and 30%, i.e.  $k_1$  between 3 and 6, far from  $k_1=1$ .  $I_{on}/I_{off}$  being very close to  $k_1*LD/a$ , Eq. 6 seems to be a good approximation.

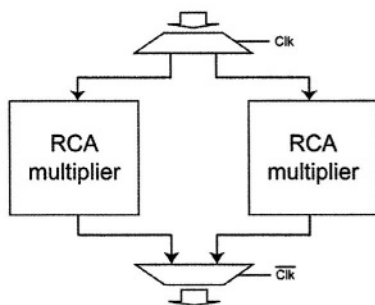
However, it is difficult to link optimal total power consumption (last column) with  $I_{on}/I_{off}$  (or  $k_1*LD/a$ ) because similar  $k_1*LD/a$  result in quite different total power consumptions. It is the case for sequential architectures that present a very high total power consumption but average  $k_1*LD/a$ . Indeed LD of the basic sequential architecture is very important (192, i.e. 12 times that of the basic Wallace structure) but since the activity is also huge (220%) the ratio  $LD/a$  stays roughly the same. The constant  $k_1$  is similarly constant since it is increased due to the large LD, but simultaneously reduced due to the large  $a$ . Besides,  $P_{dyn}$  is large due to the high  $V_{dd}$  (relatively to other architectures) and  $P_{stat}$  is also very large due to the very low  $V_{th}$ .



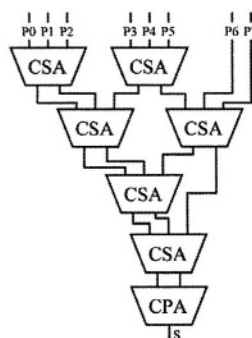
(a) RCA basic



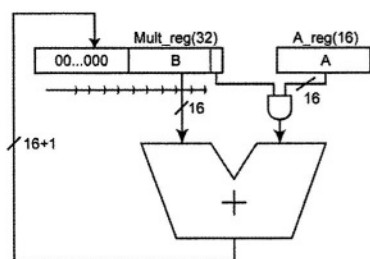
(b) RCA pipelined 2 stages



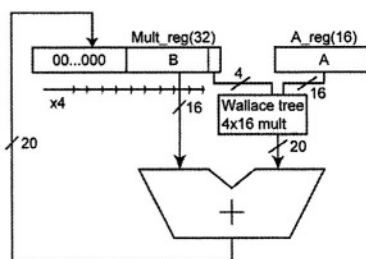
(c) RCA parallel 2 times



(d) Wallace Tree



(e) Sequential



(f) Sequential 4x16

Fig. 3. Various architectures implementing 16-bit multipliers



**Table 2.** Main parameters characterizing the 16 bit multiplier architectures

	Area [*10 <sup>3</sup> μm <sup>2</sup> ]	# of cells	# of cell transitions	a	LD	k1	k <sub>1</sub> LD/a	I <sub>out</sub> /I <sub>off</sub>	V <sub>dd</sub> [V]	V <sub>th</sub> [V]	P <sub>dyn</sub> [μW]	P <sub>stat</sub> [μW]	P <sub>tot</sub> [μW]
RCA	23	577	171	0.20	46	3.8	856	919	0.43	0.21	92	24	116
RCA parallel 2	47	1156	181	0.11	23	3.6	761	740	0.35	0.24	76	19	94
RCA parallel 4	92	2316	208	0.06	12	3.5	660	452	0.31	0.25	70	22	92
RCA pipeline 2	27	641	187	0.21	31	3.3	497	448	0.35	0.21	73	25	98
RCA pipeline 4	33	769	220	0.21	24	3.1	350	278	0.31	0.21	75	29	104
Wallace	25	755	199	0.20	16	3.1	254	226	0.29	0.21	43	15	58
Wallace parallel 2	50	1506	208	0.11	8	3.1	237	147	0.27	0.23	45	18	63
Wallace parallel 4	97	3017	238	0.06	4	3.3	218	112	0.27	0.25	54	21	75
Sequential	10	257	654	2.23	192	6.2	530	439	1.06	0.14	3269	522	3792
Sequential 4x16 wallace	14	355	255	0.59	100	3.8	646	432	0.53	0.17	302	84	386
Sequential parallel 2	16	325	502	1.28	160	5.6	701	634	0.91	0.16	2053	356	2409

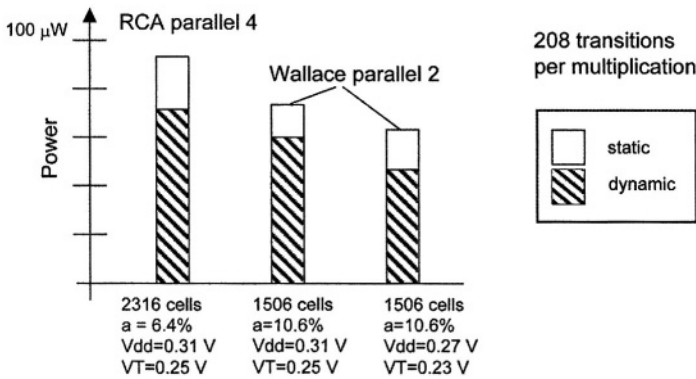
These  $V_{dd}$  and  $V_{th}$  values are necessary to maintain the throughput of the architecture. Nevertheless the ratio  $P_{dyn}$  to  $P_{stat}$  is roughly similar to the ratio of the more parallel architectures.

Parallelizing the RCA (i.e. dividing its  $LD$  and  $a$ ) shows improvements in total power. Similarly, using a two-stage pipeline RCA (i.e. dividing its  $LD$ , but keeping the same  $a$ ) reduces the total power as compared to the basic RCA, as could be expected [16]. However, the register overhead becoming too important using a four-stage pipeline leads to a poorer solution. It should be noted that  $LD$  is not perfectly divided by the number of stages, and that the area is increased due to this overhead. As a result, the optimization of the number of pipeline stages should clearly take into account the register overhead.

Although increasing the parallelism of the RCA was beneficial, increasing the parallelism of the Wallace structure does not lead to a better solution. Even though  $k_1 * LD/a$  is reduced with the increase of parallelism, the static power consumption becomes too large due to the increased number of cells, which are inactive and leaky. The parallel versions of the Wallace architecture can no longer be operated at significantly lower  $V_{dd}$  because the static power is already important, resulting in larger dynamic and total power compared to the basic Wallace architecture.

Comparing the four-time parallel RCA and twice parallel Wallace in Table 2, it can be observed that the first is composed of 2316 cells, while the second has only

1506 cells, the ratio of areas being over 1.8. Still, both architectures present 208 cell transitions for executing a 16x16 multiply, showing that various architectures with a quite different number of cells can achieve their function with the same number of transitions. The optimum  $V_{dd}$  and  $V_{th}$  conditions used in the table are not the same for both architectures. Fig. 4 on the other hand report the same architecture in the same conditions ( $V_{dd} = 0.31$  V and  $V_{th} = 0.25$  V). The architecture containing fewer cells still presents a larger activity, a smaller leakage but also a smaller dynamic power (second bar in Fig. 4, at same  $V_{dd}$  and  $V_{th}$ ). However, this architecture can be supplied with smaller  $V_{dd}$  and  $V_{th}$  thanks to its smaller LD (third bar in Fig. 4) to meet the same speed performances than the RCA parallel 4, with better results in total power consumption. Accordingly, an attractive goal to reduce the total power consumption would be to have fewer transistors to perform the same logic function, i.e. with the same number of transitions but more activity.



**Fig. 4.** Comparison of two multiplier architectures (RCA parallel 4 and Wallace parallel 2) showing the same number of transitions per multiplication but different total power

## 5 Conclusions

Total power consumption resulting from the sum of the static and dynamic contribution shows an optimum corresponding to very low  $V_{dd}$  and  $V_{th}$ . As illustrated in this paper, this minimum is characterized by a dynamic power somewhat larger than the static power (between 3.1 and 6.2 times in the considered multiplier architectures). The optimal ratio  $I_{on}/I_{off}$  is also discussed and reported to be related to the architectural parameters  $LD$  and  $a$ . This relationship has been demonstrated for eleven 16 bit multipliers showing optimal values of  $I_{on}/I_{off}$  between 100 and 1000.

Parallelization and pipelining can improve the power consumption when this leads to a reduction of  $V_{dd}$  and an increase of  $V_{th}$  (e.g. with RCA multiplier) but are not interesting for structures where the increase in static power does not allow this reduction (e.g. Wallace structure).

The results in this paper as well as supplementary experiments will lead us to develop a new methodology able to optimize existing architectures in order to achieve a better tradeoff between dynamic and static power, resulting in a total power reduction.

## References

1. <http://public.itrs.net/>
2. Belleville, M., Faynot, O.: Evolution of Deep Submicron Bulk and SOI Technologies, Chapter 2 in Low Power Electronics Design. C. Piguet, CRC Press, (will be published in 2004)
3. Kang, S.-M., Leblebici, Y.: CMOS Digital Integrated Circuits. 2nd edn. McGraw-Hill. (1999) 466-469
4. Anis, M., Elmasry, M.: Multi-Threshold CMOS Digital Circuits. Kluwer Academic Publishers. (2003)
5. Usami, K., Kawabe, N., Koizumi, M., Seta, K., Furusawa, T.: Automated Selective Multi-Threshold Design For Ultra-Low Standby Applications. Proc. Int'l Symp. on Low Power Electronics and Design. (2002)
6. Kao, J., Chandrakasan, A.: MTCMOS Sequential Circuits. Proc. European Solid-State Circuits Conf. Villach, Austria. (2001, Sep.)
7. Von Kaenel, V., et al.: Automatic Adjustment of Threshold & Supply Voltage Minimum Power Consumption in CMOS Digital Circuits. Proc. Int'l Symp. on Low Power Electronics and Design. San Diego. (1994, Oct.) 78-79
8. Kim, C.H., Roy, K.: Dynamic Vt SRAM: A leakage Tolerant Cache Memory for Low Voltage Microprocessor. Proc. Int'l Symp. on Low Power Electronics and Design. (2002)
9. Mizuno, H., et al.: An  $18\mu\text{A}$  Stand-by Current 1.8V,200 MHz Microprocessor with Self-Substrate-Biased Data-Retention Mode. IEEE J. Solid-State Circuits, Vol. 34, No. 11. (1999, Nov.) 1492-1500
10. Assaderaghi, F., et al.: A Dynamic Threshold Voltage (DTMOS) for Ultra-Low Voltage Operation. IEEE IEDM Tech. Dig., Conf. 33.1.1. (1994) 809-812
11. Soeleman, H., Roy, K., Paul, B.: Robust Ultra-Low Power Sub-threshold DTMOS Logic. Proc. Int'l Symp. on Low Power Electronics and Design. (2000)
12. Enomoto, T., Oka, Y., Shikano, H., Harada, T.: A Self-Controllable-Voltage-Level (SVL) Circuit for Low-Power High-Speed CMOS Circuits. Proc European Solid-State Circuits Conf. (2002)411-414
13. Cserveny, S., Masgonty, J.-M., Piguet, C.: Stand-by Power Reduction for Storage Circuits. Proc. Int'l Workshop on Power and Timing Modeling, Optimization and Simulation. Torino, Italy. (2003, Sep.)
14. Clark, L.T., Deutscher, N., Ricci, F., Demmons, S.: Standby Power Management for 0.18mm Microprocessor. Proc. Int'l Symp. on Low Power Electronics and Design. (2002)
15. Piguet, C., Narayanan, V.: Guest Editorial. IEEE Trans. on VLSI Systems, Vol. 12, No 2, (2004, Feb.)
16. Brodersen, R.W., et al.: Methods for True Power Minimization. Proc. Int'l Conf. on Computer Aided Design. San Jose, California. (2002, Nov.) 35-42
17. Heer, C., et al.: Designing Low-Power Circuits: An Industrial Point of View", Proc. Int'l Workshop on Power and Timing Modeling, Optimization and Simulation. Yverdon, Switzerland. (2001, Sep.)
18. Piguet, C., et al.: Techniques de Circuits et Méthodes de Conception pour Réduire la Consommation Statique dans les Technologies Profondément Submicroniques", Proc. FTFC. Paris, France. (2003, May) 21-29

# Reducing Cross-Talk Induced Power Consumption and Delay\*

André K. Nieuwland, Atul Katoch, and Maurice Meijer

Philips Research, Prof. Holstlaan 4, 5656 AA, Eindhoven, Netherlands  
andre.nieuwland@philips.com

**Abstract.** Coupling capacitances between on-chip wires dominate the power consumption of deep submicron busses. Opposite switching signals on adjacent lines cause additional power to be consumed and cause a crosstalk induced delay. This paper proposes a transition dependent skew on all the wires, such that opposite transitions do not occur simultaneously, but transitions in the same direction still do. An example is given how this can be realized with only simple coding circuitry. By inserting an intermediate state, this technique offers almost 25% savings of power on the bus lines, and a delay reduction of at least 5%. Larger reductions of the delay (upto 20%) are possible using a more aggressive repeater insertion scheme.

## 1 Introduction

The scaling towards deep submicron regions of CMOS technology led to an increase in the lateral capacitances between wires. The wires get a higher aspect ratio while the spacing between them decrease. From the ITRS roadmap [1] it becomes clear that for 65 nm technology the lateral capacitance will be more than 3 times higher than the vertical capacitance. This increase leads to increasing cross-talk induced power dissipation, which results in an increased heat production and to increasing cross-talk induced delay and noise. Opposite transitions on adjacent wires effectively double the physical capacitance between the wires due to the reversing of the polarity. This is sometimes referred to as the em Miller effect. Since bus lines run in parallel for a long distance, these lines are particularly sensitive for these effects.

A staggered repeater insertion scheme [2], is a way to average the cross coupling effect by limiting opposite transitions to half the bus length. As a drawback, transitions on adjacent wires in the same direction, are converted into an opposite transition for half the bus length. This reduces the delay spread but the power consumption remains high. Hirose et.al [3] suggested to skew transitions between adjacent wires. Although a reduction of the worst case delay was obtained upto 20%, this method has the drawback that adjacent transitions in the same direction are split in time. This causes extra power consumption due

---

\* This work is partly sponsored by the European community within FET-open, IST-2001-38930.

to the temporal charging of the inter-wire (coupling) capacitance. Kim et.al [4] presented a coupling driven transition minimisation scheme, where transitions on wires are coded with respect to the transitions on adjacent wires. Although savings in power of upto 30% are claimed, this scheme requires complex codec circuitry and introduces a serial dependency between all the bits on a bus. This dependency causes a significant additional bus delay. Other techniques, such as transition pattern coding [5], require a complex codec as well, which cause additional delay and power consumption. In many cases, complex codec circuitries consume more power than what they save on the (mutual) switching activity [6], which renders coding ineffective.

In this paper, an analytical model is introduced which describes the average charge and power required for a transition. This model prevents the use of transition look-up tables as is done in [7] and allows for simple extension to unequal switching activities per wire and larger buswidths. This model is introduced in section 2 together with definitions and notations. This model is then extended to multiple wires in section 3. In section 4, the circuit (codec) for preventing opposite transitions is introduced. Thereafter, the results are discussed in section 5 followed by the conclusions in section 6.

## 2 Notation, Definitions, and Modelling

A single wire without can be modeled as a capacitance to ground as indicated by  $C_b$  in Fig. 1. The charge ( $Q$ ) on the wire capacitance is determined by  $Q = C_b V$ , where  $V$  is the voltage across the capacitance. It is important to realise that this charge exist as positive charge (+) at one side of the capacitor, and as negative charge (-) on the other side (See Fig. 1). Charging of this capacitance will draw a current from the power supply. In this paper we define energy consumption' as the energy delivered by the power supply. That implies that no power is consumed when a capacitance is discharged to ground. Although this might seem strange at first, this discharging does not draw any current from the power supply and therefore, according to our definiton, does not *consume* energy.



**Fig. 1.** Wire capacitance model

In a digital system, a wire is normally either in a '0' (ground) or in a '1' ( $V_{dd}$ ) state. Transitions between these two states generally happen fast and are of transient nature. Table 1 indicates all possible transitions from the current state of the wire ( $A_t$ ) to the next state ( $A_{t+1}$ ). It is assumed that all these transitions are equally likely, which means that the probability of occurrence for all of them is:  $\frac{1}{4}$ .

**Table 1.** Transition table for a single wire

$A_t$	$A_{t+1}$	Prob.	Charge	Energy
0	0	1/4	0	0
0	1	1/4	$C_b V$	$C_b V^2$
1	0	1/4	0	0
1	1	1/4	0	0

As indicated by table 1, there is only one transition out of all four possible transitions where the power supply has to deliver energy to wire  $A$ . The energy from the power supply is partly stored in the capacitance ( $\frac{1}{2}C_b V^2$ ), and partly dissipated in the resistors (i.e. p-transistors) in the current path when charging the capacitance (also  $\frac{1}{2}C_b V^2$ ). The energy stored in the capacitance will be dissipated when discharging (i.e. in the n-transistors).

Expanding this transition table to multiple wires and using these tables to find which capacitances have to be charged is rather cumbersome. By describing the energy consumption of individual wires in a statistical way, we easily obtain the average charge and energy required for a transition. This allows for the extension to larger buswidths and for dealing with un-equal distributions of switching activity over the wires. In this paper, the transitions for a wire ( $A$ ) are indicated by:

1. ' $A = 0$ ':  $A$  was '0' and remains '0'
2. ' $A \uparrow 1$ ':  $A$  was '0' switches to '1'
3. ' $A = 1$ ':  $A$  was '1' and remains '1'
4. ' $A \downarrow 0$ ':  $A$  was '1' and switches to '0'

Since all transitions are equally likely, the average charge per transition delivered by the power supply is given by formula (1), and the average energy per transition by formula (2).

$$Q_{ave_A} = P(A \uparrow 1) \times C_b V = \frac{1}{4} C_b V \quad (1)$$

$$P_{ave_A} = P(A \uparrow 1) \times C_b V^2 = \frac{1}{4} C_b V^2 \quad (2)$$

### 3 Statistical Model for Multiple Wires

The case with two wires (see Fig. 2) is a little more complex because the switching wires influence each other via the mutual capacitance. When focussing on the charge delivered by the power supply to wire  $A$ , the following observations can be made:

- Wire  $A$  consumes energy when it is charged to '1' because the bottom capacitance  $C_b$  has to be charged. The probability of occurrence is denoted by  $P(A \uparrow 1)$ . The cross coupling capacitance  $C_m$  has to be charged if wire  $A$

switches to ‘1’ and wire *B* remains ‘0’. The probability for this to happen is denoted by  $P(A \uparrow 1) \times P(B = 0)$ .

- If wire *A* switches to ‘1’ and wire *B* switches to ‘0’, wire *A* has to be charged such that the polarity of the voltage across  $C_m$  is reversed, from  $-V_{dd}$  to  $+V_{dd}$ . This costs twice the charge ( $Q = C_m 2V$ ). Probability:  $P(A \uparrow 1) \times P(B \downarrow 0)$ .
- If wire *A* remains at ‘1’, the power supply will have to charge  $C_m$  via wire *A* if wire *B* switches from ‘1’ to ‘0’. Hence, charge is flowing through wire *A*. Probability:  $P(A = 1) \times P(B \downarrow 0)$ .

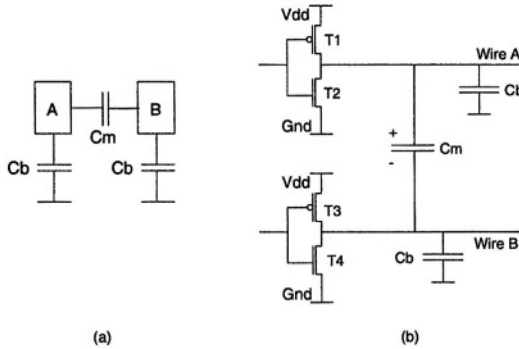


Fig. 2. Capacitances in the two wire case

Something special happens if wire *A* remains at ‘1’ when wire *B* switches to ‘1’. The capacitance  $C_m$  is short-circuited via the power supply network (see Fig. 2). The positive charge stored on  $C_m$  is injected into wire *A*, and forms a *negative* current for *A*. This current flows through transistor *T1* of the driver of wire *A*, through the power supply rail, and through transistor *T3* of the driver of wire *B*, to neutralize the negative charge on the *B*-side of capacitance  $C_m$ . Since the current in wire *B* flows in the *positive* direction (like with charging a capacitance), it is defined that wire *B* is *actively discharging*  $C_m$  (probability of  $P(A = 1) \times P(B \uparrow 1)$ ). Note that in this case, the power supply is not delivering any current and that according to our definition no energy is consumed for discharging  $C_m$ . However, since current is flowing, there will be heat generated by the resistivity (transistors) in the path from the positive side of  $C_m$  to the negative side. Likewise, if wire *B* was already at ‘1’, and wire *A* switches to ‘1’, wire *A* actively discharges  $C_m$  without drawing energy from the power supply.

The average charge per (possible) transition delivered by the power supply to wire *A* is described by Equation 3, and the average energy by Equation 4. Due to symmetry, the average charge and energy consumption per transition for wire *B* is identical.

$$\begin{aligned}
Q_{ave_A} &= P(A \uparrow 1)C_bV + P(A \uparrow 1)P(B = 0)C_mV \\
&\quad + P(A \uparrow 1)P(B \downarrow 0)C_m2V + P(A = 1)P(B \downarrow 0)C_mV \\
&= \frac{1}{4}C_bV + \frac{1}{16}C_mV + \frac{2}{16}C_mV + \frac{1}{16}C_mV = \frac{1}{4}(C_b + C_m)V \quad (3)
\end{aligned}$$

$$P_{ave_A} = \frac{1}{4}(C_b + C_m)V^2 \quad (4)$$

This model can be easily extended to multiple wires. Wire *B* in Fig. 3 is representative for a wire in a bus. For the two outer wires, *A* and *D*, formula (3) and (4) can be used. The average charge and energy consumption for wire *B* per transition are given by equations (5) and (6) respectively.

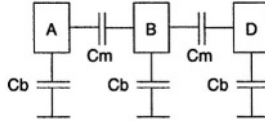


Fig. 3. Capacitances in the three wire case

$$\begin{aligned}
Q_{ave_B} &= P(B \uparrow 1)C_bV + P(B \uparrow 1)P(A = 0)C_mV + P(B \uparrow 1)P(D = 0)C_mV \\
&\quad + P(B \uparrow 1)\{P(A \downarrow 0) + P(D \downarrow 0)\}C_m2V \\
&\quad + P(B = 1)\{P(A \downarrow 0) + P(D \downarrow 0)\}C_mV \\
&= \frac{1}{4}C_bV + \frac{2}{16}C_mV + \frac{2}{16}C_m2V + \frac{2}{16}C_mV = \frac{1}{4}(C_b + 2C_m)V \quad (5)
\end{aligned}$$

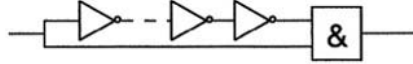
$$P_{ave_B} = \frac{1}{4}(C_b + 2C_m)V^2 \quad (6)$$

## 4 Preventing Opposite Transitions

As indicated in the previous section, the opposite transitions on adjacent wires in a bus cause a charge and energy consumption related to  $C_m2V$ . For the two outer wires, the average ‘wasted’ charge per wire per transition due to opposite transitions is  $\frac{1}{16}(C_m2V)$ . For the inner bus wires, the average charge due to opposite switching is  $\frac{2}{16}(C_m2V)$  per wire per transition. As indicated in section 2, short-circuiting wires does not draw charge from the power supply and therefore does not consume energy. When the capacitance  $C_m$  is short-circuited before being charged with the reverse polarity, charge and energy can be saved. We therefore propose to split opposite transitions into two transitions which are spaced in time, e.g. by delaying only the one-zero transition. This creates an intermediate state in which the coupling capacitance is short-circuited. This intermediate state be either ‘00’ or ‘11’, and can be obtained by delaying either the ‘0’  $\uparrow$  ‘1’ transition, or the ‘1’  $\downarrow$  ‘0’ transition. A circuit delaying the ‘0’  $\uparrow$  ‘1’ transition specifically is depicted in Fig. 4. Any ‘1’  $\downarrow$  ‘0’ immediately changes the



output of the and-gate, whereas any '0'  $\uparrow$  '1' is delayed by the inverter chain. This provides an intermediate '00' state in which  $C_m$  is short-circuited before being charged to reverse polarity. Consequently, due to the delay of a certain transition, the duty cycle of the signal is slightly changed. The duty cycle can be restored by using a similar circuit as presented in Fig. 4 at the end of the bus. This can be done by using an 'OR' instead of an AND', which delays the other transition.



**Fig. 4.** A transition dependent delay generator

Considering the charge balance for a bus wire, the charge consumed by reversing the polarity on  $C_m$  with this intermediate state is now only half the normal dissipation. The short circuiting does not cause the power supply to deliver any charge (and energy). The only charge needed is the charge required to charge  $C_m$  and  $C_b$  from '0' to  $V_{dd}$ .

The average charge required per transition for bus wire  $B$  is then given by equation (7) and the average energy consumption per transition by equation(8).

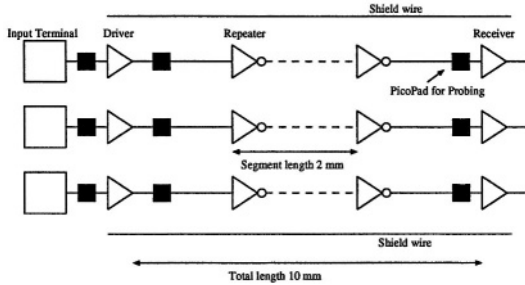
$$\begin{aligned}
 Q_{ave_B} &= P(B \uparrow 1)C_bV + P(B \uparrow 1)P(A = 0)C_mV + P(B \uparrow 1)P(D = 0)C_mV \\
 &+ P(B \uparrow 1)\{P(A \downarrow 0) + P(D \downarrow 0)\}C_mV + P(B = 1)\{P(A \downarrow 0) \\
 &+ P(D \downarrow 0)\}C_mV \\
 &= \frac{1}{4}C_bV + \frac{2}{16}C_mV + \frac{2}{16}C_mV + \frac{2}{16}C_mV = \frac{1}{4}(C_b + \frac{3}{2}C_m)V \quad (7)
 \end{aligned}$$

$$P_{ave_B} = \frac{1}{4}(C_b + \frac{3}{2}C_m)V^2 \quad (8)$$

Likewise, for the outer wires the average charge per transition per wire is:  $Q_{ave} = \frac{1}{4}(C_b + \frac{3}{4}C_m)V$ , and the average energy per transition per wire:  $P_{ave} = \frac{1}{4}(C_b + \frac{3}{4}C_m)V^2$ . Comparing these results with the initial power dissipation reveals that the power associated with the mutual coupling is reduced by 25%!

## 5 Simulation and Results

A layout of a bus has been made in a 130 nm CMOS technology to verify the results of the analytical approach. The layout consists of 3 parallel wires of 10 mm long at minimum spacing, with non-minimum sized inverting repeaters inserted every 2 mm. Two ground wires were implemented on both sides as shield wires of the 3 wire bus (see Fig. 5). Pico pads, mend for measurements on the test silicon, were included as well. This layout was extracted including parasitics and simulated to observe the cross-talk dependent time delay. The worst case delay



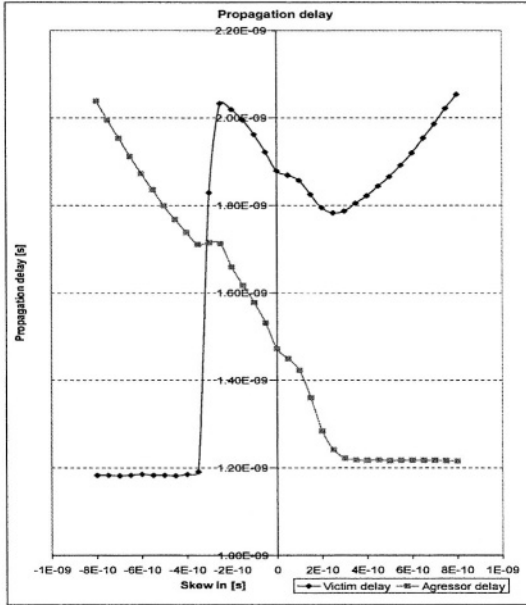
**Fig. 5.** Schematic representation of the simulated extracted circuit

should decrease due to the decrease in *effective* coupling capacitance when we impose a relative skew to prevent simultaneous opposite transitions on adjacent wires.

## 5.1 Analysis of Timing Behaviour

The two outer wires (aggressors) were switched oppositely with respect to the middle wire (victim) and the skew between the transitions of victim and aggressors was varied over a range from  $-800$  ps to  $+800$  ps. A positive skew implies a delay of the victim, whereas a negative skew implies a delay of the aggressors. The impact on the total propagation delay (including skew) was observed and plotted in Fig. 6. It can be noticed that the delay of the aggressors decreases linearly with the increase in skew of the victim. When the skew is larger than the propagation time needed for 2 mm of wire (one segment), no further decrease in propagation delay of the aggressors is observed, and the minimum victim delay is obtained. This is because the transitions of victim and aggressors are in different segments, and no *simultaneous* opposite transitions occur. Any further increase of skew leads to an equivalent increase in victim's propagation time.

Skewing the aggressors (left side of Fig. 6) gives a slightly different result. The propagation delay of the victim first increases due to the large impact of 2 aggressors switching oppositely. Only when the skew of the aggressors is that large that the transition of the victim has reached the second segment before the aggressor wires switch, the cross-talk induced delay on the victim disappears. Increasing the skew of the aggressors further, only increases the delay of the aggressor's signals. The different minimum propagation times of the victim and aggressor wires is due to the lower RC constants of the shield wires compared to signal wires. Hence the shield wires induce more delay than a constant signal wire. Therefore, a transition on an aggressor wire propagates less fast as a transition on a victim wire. Due to the faster propagation on the victim wire, there will be simultaneous opposite transitions on adjacent wires in later segments, despite the one-segment spacing between transitions initially. This effect is reflected in the smoother roll-off in aggressor delay compared with victim delay. Transitions on aggressors which are initially one-segment delayed compared to the transition



**Fig. 6.** Propagation delay results of simulated extracted layout

of the victim (negative skew) will not overtake the victim's transition, hence will not influence propagation times on later segments of the victim wire. The optimum skew is where victim and aggressor are equally fast, which provides an overall speed improvement of 8% compared to non-skewed transitions.

## 5.2 Relation with Prior Art

The reduction in delay is not in a straightforward way comparable with the results mentioned in [3] due to the different technology used. In [3], a 10 mm bus in a  $0.5 \mu\text{m}$  CMOS process was used. Splitting a 10 mm bus in  $0.5 \mu\text{m}$  technology in 5 segments of 2 mm implies a much more aggressive repeater insertion scheme than in our case, where a 10 mm bus in a 130 nm CMOS technology was split in 5 segments. Therefore, a comparison between the results for a 10 mm bus with 5 segments (4 repeaters) in the 130 nm technology should be made with approximately 2 segments (1 repeater). For even smaller (newer) technologies, more repeaters need to be used on the same physical distance to maintain the same relative speed advantage.

## 5.3 Energy Savings

With respect to energy savings, the gain is larger. Fig. 7 shows the energy required for an opposite transition on the three wires as a function of skew. Due to the fact that simultaneous opposite switching is prevented and charge is partly

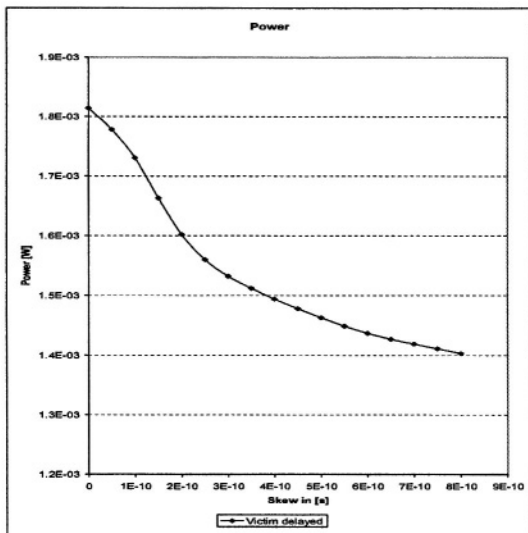


Fig. 7. Simulated power consumption of extracted layout

recycled, less energy is needed. From Fig. 7 it can be seen that as soon as the transitions of aggressor and victim are approximately one segment delay apart, the total energy consumption is reduced by approximately 15%. The maximum savings in energy (23%) is obtained at the longest inserted delay. Note that the maximum savings in power are not obtained after one-segment skew as with the minimum propagation delay. This is because the repeater already starts to drive the next segment, whereas the transition on the input segment is not yet at full swing. Therefore, there is still a transition on-going when the aggressor wires switched. Furthermore, the delayed transition of the victim is travelling faster than the transitions on the aggressors due to the higher RC-delay of the (stable) aggressor wires (driven by repeaters which have a certain output resistance). The propagation of the transitions on the aggressor wires suffers more from the (stable) shield wires, since they have a much lower RC-delay due to all the well and power-grid contacts.

## 6 Conclusions

A new technique has been presented to save on cross coupling power. By inserting a transition dependant delay, opposite switches on adjacent wires can be prevented, while transitions on adjacent wires in the same direction remain simultaneous. This technique saves almost 25% of the power associated with the coupling capacitance. For future IC technologies, where coupling capacitance can be up to 3 times the vertical capacitance, this implies a saving of the total wire power of about 20%! Furthermore, due to avoidance of opposite switches on adjacent wires, the worst case bus propagation delay is avoided as well. When a

sufficiently aggressive repeater insertion scheme is applied, the bus could operate upto 20% faster. Since the knowledge of signals of adjacent wires is not required, nor any knowledge of previous and next states of the wire, only limited circuitry is needed. This keeps area cost and self dissipation of this codec low.

**Acknowledgements.** The authors would like to thank Claudia Kretzschmar, Steven van Dijk and Roelof Salters for the many interesting and helpful discussions regarding this topic.

## References

1. <http://public.itrs.net/Files/2003ITRS/Home2003.htm>
2. Yu Cao, Chenming Hu, Xuejue Huang, Andrew B. Kahng, Sudhakar Muddu, Dirk Stroobandt, Dennis Sylvester, "Effects of Global Interconnect Optimizations on Performance Estimation of Deep Submicron Design", ICCAD, pp. 56–61, 2000.
3. Kei Hirose, Hiroto Yasuura, "A Bus Delay reduction Technique Considering Crosstalk", *Design Automation and Test in Europe (DATE)*, Paris (Fr.), pp. 441–445, 2000.
4. Ki-Wook Kim, Kwam-Hyun Baek, Naresh Shanbhag, C.L. Liu, Sung-Mo Kang, "Coupling-driven Signal Encoding Scheme for Low-Power Interface Design", *ICCAD*, pp. 318–321, 2000.
5. Paul P. Sotiriadis, Alice Wang, Anantha Chandrakasan, "Transition Pattern Coding: An approach to reduce Energy in Interconnect", *26<sup>th</sup> European Solid-State Circuit Conference (ESSCIRC)*, Stockholm (S), pp. 320–323, 2000.
6. Claudia Kretzschmar, André K. Nieuwland, Dietmar Müller, "Why Transition Coding for Power Minimization of On-Chip busses does not work", *DATE*, pp. 512-517, February 2004.
7. Paul P. Sotiriadis, Anantha P. Chandrakasan, "A Bus Energy Model for Deep SubmicronTechnology", *IEEE Transactions on VLSI systems*, vol. 10, No. 3, pp. 341–350, June 2002.
8. L. Di Silvio, D. Rossi, C. Metra, "Crosstalk Effect Minimization for Encoded Busses", *International On-Line Test Symposium (IOLTS)*, pp. 214–218, July 2003.

# Investigation of Low-Power Low-Voltage Circuit Techniques for a Hybrid Full-Adder Cell

Ilham Hassoune<sup>1</sup>, Amaury Neve<sup>2</sup>, Jean-Didier Legat<sup>1</sup>, and Denis Flandre<sup>1</sup>

<sup>1</sup> Dice, UCL, Batiment Maxwell, place de Levant 3,  
1348 Louvain-la-Neuve, Belgium  
hassoune@dice.ucl.ac.be

<sup>2</sup> IBM Entwicklung, Schönaicher Strasse 220,  
D-71032 Böblingen, Germany

**Abstract.** A full-adder implemented by combining branch-based logic and pass-gate logic is presented in this contribution. A comparison between this proposed full-adder (named BBL\_PT) and its counterpart in conventional CMOS logic, was carried out in a  $0.13\mu\text{m}$  PD (partially depleted) SOI CMOS for a supply voltage of 1.2V and a threshold voltage of 0.28V. Moreover, MTCMOS (multi-threshold) circuit technique was applied on the proposed full-adder to achieve a trade-off between Ultra-Low power and high performance design. Design with DTMOS (dynamic threshold) devices was also investigated with two threshold voltage values (0.28V and 0.4V) and  $V_{dd} = 0.6\text{V}$ .

## 1 Introduction

The reduction of the power dissipation at all design levels is one of the major challenges for the coming years in the semiconductor industry [1]. In particular for portable applications, designs have to minimize active and standby power in order to extend battery lifetime. Future applications such as distributed sensor networks have to operate reliably for long periods of time without the need to change the battery. The wide adoption of contactless smart cards also requires chips with minimal power consumption since they have to pull their energy from the RF link with the driver.

In this work we address the reduction of the power dissipation at the cell and transistor levels. Thus, two logic families have been investigated in order to finally create an optimal hybrid branch-based/pass-gate full-adder as proposed in section 2.

Logic CMOS families using the pass-transistor circuit technique to improve power consumption have long been proposed [2,3]. This logic style has the advantage to use only NMOS transistors, thus eliminating the large PMOS transistors used in conventional CMOS style. However, pass-transistor logic suffers from the reduced swing at the output ( $(V_{dd} - V_t)$  instead of  $V_{dd}$ ) and a full swing restoration is needed to avoid a bad high logic level.

Branch-based design can be considered as a restricted version of static CMOS where a logic level is only made of branches containing a few transistors in

series [5]. The branches are connected in parallel between the power supply lines and the common output node. By using the branch-based concept, it is possible to minimize the number of internal connections, and thus the parasitic capacitances associated to the diffusions, interconnections and contacts. This has been demonstrated in [12] for deep submicron technologies and complex 64b high-speed low-power adders.

On the other hand, as voltage supply reduction remains the main approach to reduce the power dissipation, a trade-off is needed to achieve this goal without a significant loss of speed. Indeed, maintaining a high performance when  $V_{dd}$  is reduced, requires an aggressive  $V_t$  scaling. However, this increases leakage currents. Thus, circuit techniques like MTCMOS [4,6] have been proposed to allow a leakage control when lowering voltage supply and when  $V_t$  is reduced.

In this circuit technique, high-threshold switch transistors are used in series with the low-threshold voltage logic block. In standby mode, high- $V_t$  transistors are turned off, thus suppressing leakage. While in the active mode, the high- $V_t$  transistors are turned on and act as virtual  $V_{dd}$  and ground.

The DTMOS technique was finally proposed [10] for ultra-low supply voltage operation ( $\leq 0.6V$ ) to improve circuit performance. Because of the substrate effect,  $V_t$  can be changed dynamically by using the DTMOS configuration where the transistor gate is tied to the body. By choosing this device configuration,  $V_t$  is lower during the on-state of the DTMOS device, thereby increasing the transistor drive current; while  $V_t$  is higher during its off-state. thus, suppressing leakage current.

In order to assess the potential of these design techniques for high-performance deep submicron CMOS applications, we have chosen the full-adder as benchmark cell.

## 2 The Proposed Full-Adder Gate

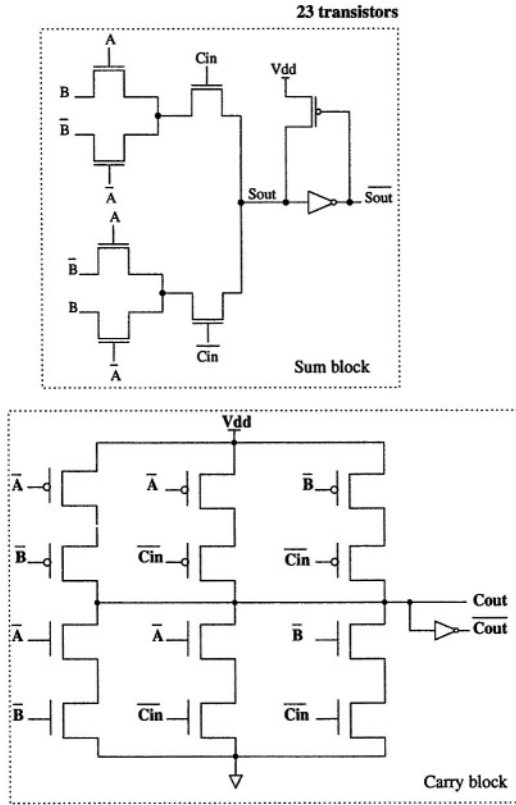
A CMOS gate is made of two dual N-ch and P-ch Networks [8]. In branch-based design, some constraints are applied on the N-ch and P-ch Networks: they are only composed of branches, this is a series connection of transistors between the output node and the supply rail. N-ch and P-ch networks are sums of products and are obtained from Karnaugh maps [7]. Simple gates are constructed of branches and more complex gates are composed of simple cells. In [9], the author presents a full-adder schematic using the branch based logic, with 7 simple BBL (Branch-Based Logic) gates and 28 transistors. BBL design does not implement gates with pass-transistors.

In this work, we use the simplification method given in [7] to implement the carry-block of a full-adder with a branch structure. The obtained equations of N-ch and P-ch networks are given by:

$$C_{N-ch} = \overline{A}.\overline{Cin} + \overline{B}.\overline{Cin} + \overline{A}.\overline{B}$$

$$C_{P-ch} = \overline{A}.\overline{Cin} + \overline{B}.\overline{Cin} + \overline{A}.\overline{B}$$

The resulting carry block is shown in figure 1.



**Fig. 1.** The proposed BBL\_PT full-adder

However, to implement the sum-block with only branches is not advantageous. Indeed, with this method, the sum-block needs 24 transistors for 1bit sum-generation and stacks of three devices. Therefore, a simple implementation with pass-transistors was used for the sum-block (figure 1).

The disadvantage lies in the resulting reduced swing due to reduced high output level in pass-transistors used in the sum-block of the proposed full-adder. However, the feedback realized by the pull-up PMOS transistor is sufficient to restore the high output level.

By choosing this implementation, we break some rules specified in the branch-based logic. Nevertheless, Branch-Based logic in combination with pass-gate logic, allows a simple implementation of full-adder gate, namely the BBL\_PT (Branch-Based Logic and Pass-Transistor) full-adder, with only 23 transistors versus the 28 transistors-conventional CMOS full-adder.

Comparison between the BBL\_PT full-adder and the conventional CMOS full-adder [11] is discussed in the next section.



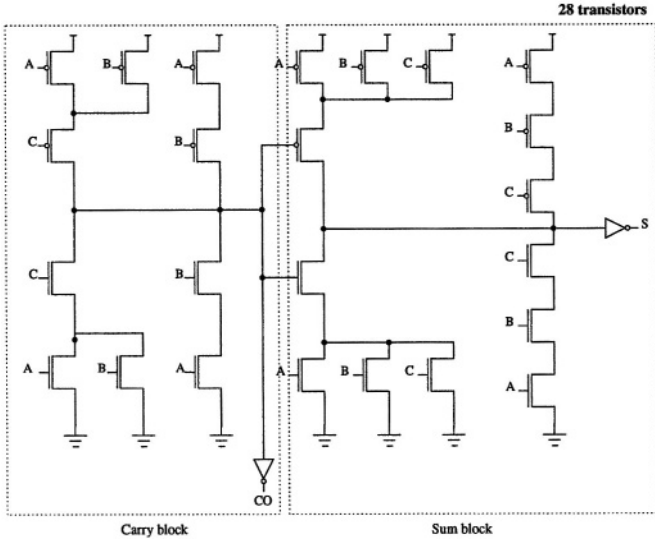


Fig. 2. The conventional CMOS full-adder

### 3 Comparison Between the Two Full-Adders

#### 3.1 Simulations Conditions

Simulations were carried out at schematic level for the proposed BBL\_PT full-adder shown in figure 1, and its counterpart in conventional CMOS shown in figure 2, in a  $0.13\mu\text{m}$  PD (partially depleted) SOI CMOS and with a supply voltage  $V_{dd} = 1.2\text{V}$ , floating body devices and  $V_t = 0.28\text{V}$ , at room temperature. The same  $\frac{W}{L}$  ratio was considered for the two full-adder gates, except for the pull-up transistor which must have a high ON resistance in order to restore the high logic level without affecting the low logic level at the output node. Thus, a lower  $\frac{W}{L}$  ratio was used for the pull-up transistor. Each full-adder is loaded by an inverter. This inverter has a separate supply voltage in order to be able to extract the power consumption of the full-adder cell. An input clock frequency of 100 MHz was considered.

#### 3.2 Results and Discussion

Simulation results, given in table 1, show that the delay of the BBL\_PT full-adder is lower than in the conventional CMOS one. The BBL\_PT sum-block needs the true carry signal and its complement. Therefore, it should be useful to evaluate also the  $\overline{CO}_{out}$  delay.

With regard to the total power consumption, the BBL\_PT full-adder consumes only 9% more than its counterpart in conventional CMOS.

**Table 1.** Simulations results with  $V_{dd} = 1.2V$ ,  $f = 100MHz$ ,  $V_t = 0.28V$ ,  $t = 27^\circ C$  and floating body devices

	Delay (ps)			Total power ( $\mu W$ )	Static power (nW)
	Cout	Sout	$\overline{Cout}$		
BBL_PT full-adder	51	34	98	4.80	150
Conventional CMOS full-adder	118	157		4.40	170
BBL_PT full-adder with MTCMOS circuit technique	59	34	130	4.23	0.0185

However, it must be pointed out that the main advantage of the BBL design is emphasized in the layout as the parasitic capacitances due to interconnections on internal nodes, are minimized because of the branch structure of the design.

With regard to the static power, it is shown in table 1 that static power is less in the BBL\_PT full-adder gate, thanks to the structure in branch and the lower number of transistors. Indeed, simulations have shown that the structure in branch helps to reduce the static power.

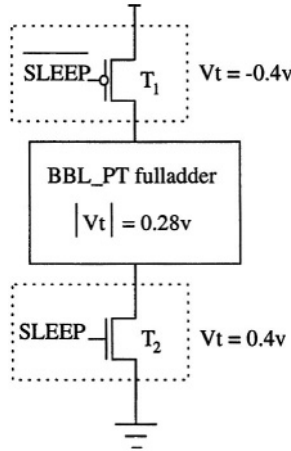
In order to highlight the better performance obtained with the BBL\_PT fulladder, comparison was performed between the critical paths in the BBL\_PT and the conventional CMOS gates. In the BBL\_PT full-adder, the worst delay is observed when  $A = '0'$ ,  $B = '1'$  and  $Cin$  goes from  $'1' \rightarrow '0'$ . In the sum-block, two devices in series are on the critical path to  $Sout$ . Concerning the carry-block, stacks of two devices are on the critical path to  $\overline{Cout}$ , while stacks of two devices and an inverter gate are on the critical path to  $\overline{Cout}$ .

In the conventional CMOS full-adder, the worst delay is observed when  $A$  goes from  $'1' \rightarrow '0'$ ,  $B = '0'$  and  $Cin = '1'$ . In the sum-block, the critical path to  $Sout$  goes through stacks of two devices before going through other stacks of two devices and an inverter gate. While, in the carry-block, stacks of two devices and an inverter gate are on the critical path to  $Cout$ .

Besides this, one can notice that the parasitic capacitance due to diffusions and gates should be lower in the BBL\_PT due to the lower number of internal connexions. These reasons make the BBL\_PT full-adder gate performing better than the conventional CMOS full-adder. Nevertheless, because of the presence of pass-transistors, the BBL\_PT full-adder must be used with high  $\frac{V_{dd}}{V_t}$  ratio ( $> 3$ ) due to the fact that the noise margins decrease in pass-transistor logic for reduced  $\frac{V_{dd}}{V_t}$  ratio.

#### 4 The MTCMOS Circuit Technique Applied to the BBL\_PT Full-Adder

The MTCMOS circuit technique was applied to the BBL\_PT full-adder as shown in figure 3. A high threshold voltage ( $|V_t| = 0.4V$ ) was used for the switch



**Fig. 3.** The MTCMOS circuit technique applied to the BBL\_PT full-adder

transistors ( $T_1$  and  $T_2$ ), while a low threshold voltage ( $|V_t| = 0.28V$ ) was used for the full-adder logic block.

Simulations were carried out with  $V_{dd} = 1.2V$  with floating body devices at room temperature. The same  $\frac{W}{L}$  ratio, the same fanout and input clock frequency as those used for the BBL\_PT full-adder using one threshold voltage value  $V_t = 0.28V$ , were considered.

Results are given in the third row of table 1. With regard to the delay, there is an increase of around 16% on  $C_{out}$  and about 33% on  $\overline{C_{out}}$  in the BBL\_PT with the MTCMOS technique in comparison with its counterpart using only low- $V_t$  ( $V_t = 0.28V$ ) devices. The delay on  $S_{out}$  is still unchanged due to the fact that sleep transistors are in series with the inverter and the pull-up PMOS transistor used to restore the high logic level, in the sum block (illustration is shown in figure 4). On the other hand, since the sleep transistors have a  $|V_t| = 0.4V$ , then  $\frac{V_{dd}}{V_t} = 3$  for those transistors. Therefore, their drive capability is just enough and the critical path on the sum block is not affected when the MTCMOS technique is used. With regard to the total power consumption, the full-adder gate consumes about 12% less when the MTCMOS circuit technique is used. And finally, the static power dissipation is reduced to a negligible value thanks to leakage current suppression in the standby mode.

## 5 The BBL\_PT Full-Adder with DTMOS Devices

As the use of DTMOS device is limited to voltage supply up to 0.6V [10], simulations were carried out on the BBL\_PT full-adder with  $V_t = 0.28V$  and  $V_{dd} = 0.6V$  in the same conditions than those given in section 3.1. The full-adder gate with DTMOS devices was compared with its counterparts using transistors in floating body configuration and with a third version using the MTCMOS technique

**Table 2.** Simulation results with  $V_{dd} = 0.6V$ ,  $f = 100MHz$ ,  $t = 27^\circ C$ 

	Delay (ps)			Total power ( $\mu W$ )	Static power (nW)
	$C_{out}$	$S_{out}$	$C_{out}$		
BBL_PT FA with DT-MOS devices, $V_t = 0.28V$	170	215	300	1.3	2
BBL_PT FA with floating body devices, $V_t = 0.28V$	120	160	273	1.03	1.56
BBL_PT FA with MTC-MOS technique and floating body devices	163	185	430	0.95	0.008

(shown in figure 3). A floating body devices configuration was also considered in the BBL\_PT full-adder with the MTCMOS technique.

From simulations results given in table 2, it appears that there is no gain in performance or power consumption when the DTMOS device is used with  $V_t = 0.28V$ . Results show that it is just the opposite and the BBL\_PT full-adder using the floating body devices is more advantageous. The increase of the switching load capacitance due to the depletion capacitance, degrades the performance when DTMOS devices and a low  $V_t = 0.28V$  are used. Thus, the expected gain in performance is not obtained when the DTMOS device is used.

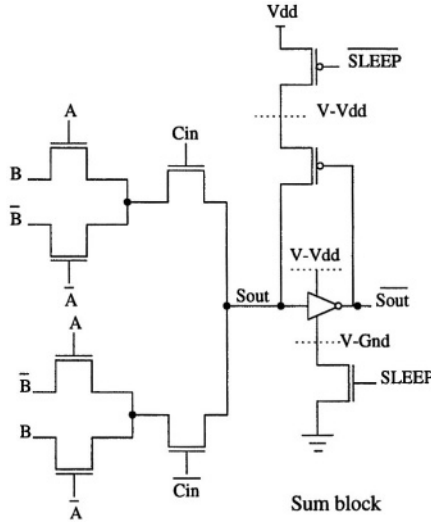
However, simulations results given in the next section, show that a gain in performance can be obtained when the DTMOS device is used in combination with a higher  $V_t$ .

With regard to the BBL\_PT with the MTCMOS technique and floating body devices, it appears that a degradation up to 58% is observed on delay, while a gain of 5% on total power consumption is obtained in relation to its counterpart using floating body devices and one  $V_t = 0.28V$ . And as observed in section 4, the static power dissipation becomes negligible.

It should be reminded that in section 4, the delay on  $S_{out}$  was the same in the BBL\_PT full-adder using MTCMOS technique and its counterpart using one  $V_t = 0.28V$ , with  $V_{dd} = 1.2V$ . This time, because of the high output level degradation in the sum block due to the lower  $\frac{V_{dd}}{V_t}$  ratio, and on the other hand, the sleep-transistors which work this time with a  $\frac{V_{dd}}{V_t} < 3$ , the high logic level and the delay on the critical path suffer much from this  $\frac{V_{dd}}{V_t}$  scaling. As a consequence, the delay on  $S_{out}$  is as well degraded when the MTCMOS technique is used.

## 6 The BBL\_PT Full-Adder with DTMOS Devices and High $V_t$

This section discusses a comparison between the BBL\_PT full-adder using DTMOS devices and its counterpart using floating body devices, when a high  $V_t$



**Fig. 4.** The MTCMOS circuit technique applied to the sum-block of the BBL\_PT full-adder

( $V_t = 0.4V$ ) and a supply voltage of  $0.6V$  are used. Simulations were carried out in the same conditions as before.

From simulations results given in table 3, it appears that a performance gain up to 36% is obtained when using the DTMOS technique. The most significant gain is observed on the critical path in the sum block. Therefore, DTMOS device might be a solution for the performance degradation of pass-transistors when an aggressive  $\frac{V_{dd}}{V_t}$  scaling is needed. However, it must be pointed out that the high logic level in the sum-block is significantly degraded with a high  $V_t = 0.4V$  under  $V_{dd} = 0.6V$ .

With regard to the total power dissipation, as it can be observed in table 3, the BBL\_PT full-adder consumes much more when the DTMOS device is used because of the increase in the parasitic load capacitance. The static power dissipation is as well much higher in this last case.

**Table 3.** Simulations results with  $V_{dd} = 0.6V$ ,  $f = 100MHz$ ,  $V_t = 0.4V$ ,  $t = 27^\circ C$

	Delay (ps)			Total power ( $\mu W$ )	Static power (nW)
	Cout	Sout	Cout		
BBL_PT FA with DT-MOS devices	280	1740	496	0.63	0.6
BBL_PT FA with floating body devices	327	2700	670	0.43	0.03

## 7 Conclusion

In this work, a hybrid architecture of a full-adder with only 23 transistors was presented. In comparison with its counterpart in conventional CMOS, it has shown an advantage in delay and static power consumption in a  $0.13\mu\text{m}$  PD SOI CMOS under  $V_{dd} = 1.2\text{V}$ . Moreover, a study of the MTCMOS circuit technique was carried out for a trade-off design to achieve both low-power and high performance at the cell level. An investigation at the transistor level by using DTMOS devices was also presented. Simulations done on the BBL\_PT full-adder under  $V_{dd} = 0.6\text{V}$ , showed that the DTMOS device offers a little gain in speed and only when high- $V_t$  transistors are used.

## References

1. International technology roadmap for semiconductors, 2003 edition.
2. Chandrakasan and al.: Low-Power CMOS Digital Design. IEEE Journal of Solid-State Circuits, vol. 27, no 4, April (1992), 473–484.
3. Abu-Khater I., Bellaouar A., Elmasry M.I.: Circuit Techniques for CMOS Low-Power High-Performance Multipliers. IEEE Journal of Solid-State Circuits, vol. 31, no 10, Oct. (1996), 1535–1546.
4. J.B.Kuo, J.-H.Lou: Low-voltage CMOS VLSI circuits. Wiley, (1999).
5. C.Piguet and al.: Basic Design Techniques for both low-power and high-speed ASIC's. EURO ASIC (1992), 220–225.
6. M. Anis, M. Elmasry: Multi-Threshold CMOS digital circuits, Managing leakage power. Kluwer Academic Publishers, (2003).
7. C.Piguet, A. Stauffer, J. Zahnd: Conception des circuits ASIC numériques CMOS. Dunod, (1990).
8. J.-M.Masgonty and al.: Branch-based digital cell libraries. EURO ASIC (1991), 27–31.
9. J.-M.Masgonty and al.: Technology and Power Supply Independent Cell Library Custom Integrated Circuits Conference (1991), 25.5.1–25.5.4.
10. F.Assaderaghi and al.: A dynamic threshold voltage Mosfet (DTMOS) for Ultra-Low voltage operation. International Electron device Meeting technical Digest, San Francisco, Decembre (1994), 809–812.
11. Ken Martin: Digital Integrated Circuit Design. Oxford University Press (2000).
12. A. Neve and al.: Power-delay product minimization in high-performance 64-bit carry-select adders. IEEE Transactions on VLSI Systems, vol.12, No. 3, March (2004).

# Leakage Power Analysis and Comparison of Deep Submicron Logic Gates\*

Geoff Merrett and Bashir M. Al-Hashimi

ESD Group, School of Electronics and Computer Science, University of Southampton, UK  
bmah@ecs.soton.ac.uk

**Abstract.** Basic combinational gates, including NAND, NOR and XOR, are fundamental building blocks in CMOS digital circuits. This paper analyses and compares the power consumption due to transistor leakage of low-order and high-order basic logic gates. The NAND and NOR gates have been designed using different design styles and circuit topologies, including complementary CMOS, partitioned logic and complementary pass-transistor logic. The XOR gate has been designed using a variety of additional circuit topologies, including double pass-transistor logic, differential cascade voltage switch logic and a gate designed specifically for low power. The investigation has been carried out with HSPICE using the Berkeley Predictive Technology Models (BTPM) for three deep submicron technologies (0.07 $\mu\text{m}$ , 0.1 $\mu\text{m}$  and 0.13 $\mu\text{m}$ ).

## 1 Introduction

The rising demand for portable systems is increasing the importance of low power as a design consideration. Considerable research is being performed into various techniques for lowering the power consumption of digital circuits [1]. As MOS transistors enter deep submicron sizes, undesirable consequences regarding power consumption arise. Decreasing the dimensions of the transistor requires a reduction in the supply voltage to keep the dynamic power consumption reasonable [2]. In turn, this demands a reduction of the threshold voltage to maintain performance, which causes an exponential increase in the subthreshold leakage current [3]. Recent research has shown that, with ever shrinking dimensions, the leakage current will become even greater than the dynamic current in the overall power dissipation [4].

Recently, numerous techniques have been proposed which aim to reduce leakage power. These include supply voltage reduction [5], [6], supply voltage gating [2], multiple or increased threshold voltages [2], [6], [7], [8], [9] and minimising leakage power consumption in sleep states [10], [11]. However, from a leakage power perspective, little work has been reported comparing different design styles and circuit topologies of the basic gates. Investigating this will allow designers to choose the correct design style for a gate to achieve low leakage power consumption. This paper presents a systematic comparison between NAND, NOR and XOR gates, at three DSM process technologies (0.07 $\mu\text{m}$ , 0.1 $\mu\text{m}$  and 0.13 $\mu\text{m}$ ), implemented using different design styles and circuit topologies. Furthermore, this paper demonstrates how the

---

\* This research is supported in part by EPSRC(UK) grant GR/595770.

stacking effect [12] can play an important role at the design stage in reducing leakage by considering transistor ordering.

## 2 Preliminaries

This section briefly analyses the leakage power of a 2-input NAND gate and reviews a number of design styles and circuit topologies that can be used to implement basic gates. This information will subsequently be used to explain the investigations and simulation results outlined in this paper.

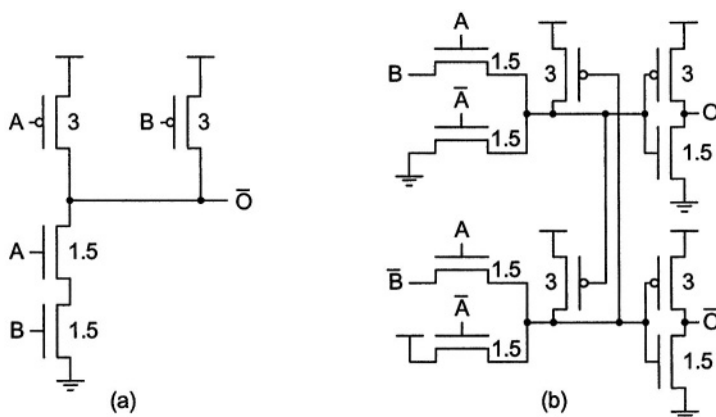


Fig. 1. 2-input NAND, (a) COMP [13], (b) CPL [15]

There are numerous design styles that can be used to realise digital CMOS gates. The three most commonly employed are examined in this paper: complementary CMOS [13], partitioned logic [13] and various pass-transistor designs [13], [14], [15] – focusing primarily on complementary pass logic (CPL). Complementary CMOS (denoted from here on as ‘COMP’) consists of a PMOS pull-up network and NMOS pull-down network. Fig. 1(a) shows a 2-input COMP NAND gate, and Fig. 1(b) shows the same gate designed using CPL. The advantages and disadvantages of CPL are well documented [13], [15], [16], [17]. The numbers next to the transistors in the figures represent their widths in relation to their lengths; the lengths are set to the smallest dimensions of the technology used. In designing high-order gates (>2 inputs), COMP and partitioning design styles can be employed. Fig. 2 shows an 8-input NAND gate implemented using the partitioned designed style.

Different design styles and circuit topologies have been proposed to implement a 2-input XOR gate. These include COMP [15], CPL [15], transmission gates [13], differential cascade voltage switch logic (DCVSL) [13], double pass logic (DPL) [15] and a gate specifically designed for low power (LP) [14]. Fig. 3 shows the two circuits for a 2-input XOR gate produced using COMP and LP design styles.



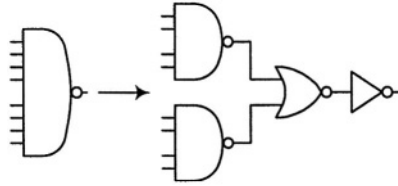


Fig. 2. 8-input partitioned NAND [13]

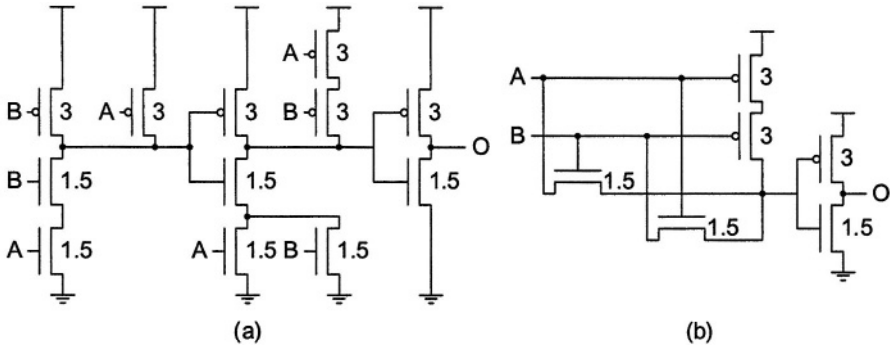


Fig. 3. 2-input XOR, (a) COMP [15], (b) LP [14]

In CMOS digital circuits, power dissipation consists of two main components: dynamic and static. As opposed to dynamic power (which is a result of switching activity), leakage power occurs as a result of the subthreshold leakage current present in deep submicron MOS transistors acting in the weak inversion region [5]. The leakage current is exponentially dependant on the gate-source voltage of the transistor, causing a drain current to flow even when the gate-source voltage is zero [3]. The leakage current for an NMOS transistor operating with  $V_{gs} \approx 0$  is given in equation (1), where the parameters have their usual meanings [7].

$$I_s = I_0 \exp\left(\frac{V_{gs} - V_T}{nV_{th}}\right) \left(1 - e^{\frac{-V_{ds}}{V_{th}}}\right) \tag{1}$$

In [3], analytical equations that model the leakage current for a series of stacked transistors were given. These equations give the subthreshold current for a stack of one transistor ( $IS1$ ), two transistors ( $IS2$ ) and three transistors ( $IS3$ ). It was also shown in [3] that the more transistors there are in a stack, the smaller is the leakage current (i.e.  $IS1 > IS2 > IS3$ ). The leakage current for transistors in parallel is given by the sum of the individual currents through each transistor. The leakage current in a CMOS gate depends on the input pattern applied; for example, in the case of the 2-input COMP NAND gate (shown in Fig. 1(a)), the leakage current is highest when  $A=1$  and  $B=1$ , since both PMOS transistors are off. In this case, the total leakage current is given by the sum of the individual currents through each of the parallel transistors ( $IS1+IS1=2*IS1$ ). The leakage current is smallest when  $A=0$  and  $B=0$ , since both NMOS transistors are off, and the total leakage current is given by  $IS2$ .

The leakage power results presented throughout this paper were obtained using HSPICE [18] with the Berkeley Predictive Technology Models (BPTM) [19] and minimum-size transistors [13]. The results were obtained from a transient analysis over an exhaustive set of input patterns. The average leakage power was calculated by averaging the power over all possible input combinations (i.e. all inputs have an equal probability of occurring). The leakage power consumption was calculated by multiplying the simulated leakage current by the supply voltage.

### 3 Design of Low Order Gates

Section 2 has shown that there are many design styles and circuit topologies to realise the functions of basic gates. Whilst comparisons exist between the different designs in terms of speed, area and dynamic power [15], there is little reported work giving a systematic comparison between different gate designs from a leakage power perspective – the main aim of this paper. Fig. 4(a) shows the simulated leakage power performance of 2-input NAND and NOR gates using the COMP and CPL design styles (Fig. 1) at  $0.07\mu\text{m}$ . As can be seen, the leakage power of a CPL based gate is nearly four times that of the equivalent COMP gate. This is because, as outlined in Section 2, the highest leakage current for the COMP NAND gate is  $2*IS1$  and occurs when the input is “1,1”. The lowest leakage current is  $IS2$ , and occurs when the input is “0,0”. In the case of the CPL NAND gate, the highest leakage current is  $5*IS1$  and occurs when the input is “0,1” or “1,0”. The lowest leakage current is  $3*IS1$  and occurs when the input is “0,0” or “1,1”. The leakage current is higher for the CPL design style, as there are no stacks (meaning all of the leakage currents are  $IS1$ s), and extra leakage current is drawn through level-restoring and output-driving circuitry.

Fig. 4(b) shows the leakage power performance of the 2-input NAND gate at different DSM technologies. It should be noted that, as the 2-input NOR gate has almost the same leakage as the NAND, the results are not shown in this paper. Fig. 4(b) reinforces the prediction that the leakage power in a CMOS circuit increases as the technology shrinks. It can be observed from Fig. 4 that leakage power is an issue in DSM gates and, to minimise this, designers should select the COMP design style in preference to CPL when implementing 2-input NAND and NOR gates.

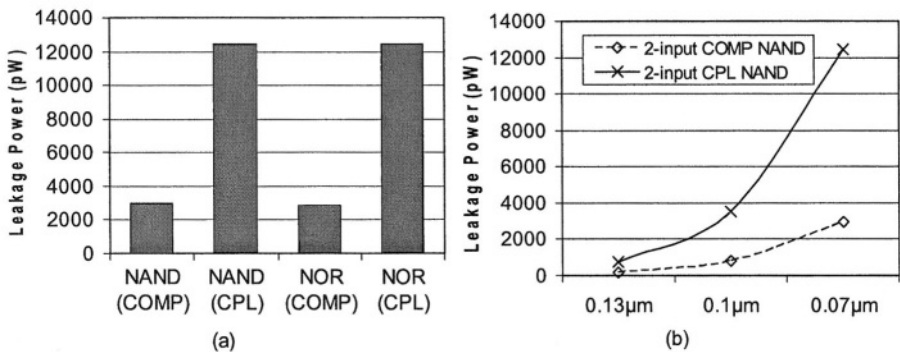


Fig. 4. Leakage power of COMP and CPL 2-input, (a) NAND and NOR at  $0.07\mu\text{m}$ , (b) NAND across DSM technologies

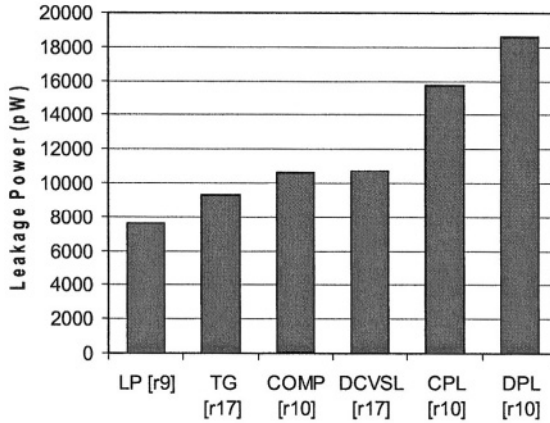


Fig. 5. Leakage power of 2-input XOR at  $0.07\mu\text{m}$

Up to this point, only NAND and NOR gates have been considered. Numerous design styles for 2-input XOR gates (a selection is shown in Fig. 3) have been reported in the literature, each with merits and shortcomings in terms of area, speed and dynamic power [15]. Fig. 5 shows the leakage power comparison of six different design styles using a  $0.07\mu\text{m}$  technology. As can be seen, the LP design (Fig. 3(b)) consumes the least leakage power, the COMP design consumes less leakage power than CPL, and the DPL design [15] has the highest leakage power. The LP gate performs best through the use of only six transistors, compared to 12 for COMP, 10 for CPL and 12 for DPL. The LP gate has a leakage current given by  $3 \cdot \text{IS1}$  for inputs “0,0”, “0,1” and “1,0”, and  $\text{IS1}$  for input “1,1”. This is in contrast to COMP having a leakage current given by  $4 \cdot \text{IS1}$ , CPL given by  $5 \cdot \text{IS1}$ , and DPL given by  $6 \cdot \text{IS1}$  (for all inputs).

It is informative to note at this stage that the subthreshold current through an NMOS transistor is not equal to that of a PMOS transistor. It is for this reason that, in Fig. 5, the difference between the leakage power of CMOS and CPL ( $5 \cdot \text{IS1} - 4 \cdot \text{IS1}$ ) is not equal to the difference between DPL and CPL ( $6 \cdot \text{IS1} - 5 \cdot \text{IS1}$ ).

Table 1 examines the leakage power performance of the 2-input XOR gate, implemented at different DSM technologies. As expected, the leakage power increases as the process is scaled down; for example, the leakage power of the COMP XOR at  $0.1\mu\text{m}$  is  $3050\text{pW}$  but increases to  $10600\text{pW}$  at  $0.07\mu\text{m}$ . It is known that DPL is worse than the other design styles for area, dynamic power and speed, and this has been reinforced by these results for leakage power. From Fig. 5 and Table 1, it can be observed that the LP design reported in [14] has the least leakage power. This design also has the fewest transistors compared to the other topologies outlined in Table 1, and has low delays and dynamic power [14]. This indicates that the LP XOR gate provides an attractive choice in terms of area, speed and power (dynamic and leakage).

We believe that, in applications where a unified design style is an important issue for basic gates, the COMP design style should be given serious consideration. This is because the COMP design style produces NAND and NOR gates with low leakage power across DSM technologies (NAND shown in Fig. 4(b)), while the XOR gate also performs comparatively well from a leakage power perspective – CPL is around 50% worse than COMP, while LP is less than 30% better than COMP.

**Table 1.** Leakage power of six designs for 2-input XOR gates at three DSM technologies

Gate	Design Style	No. of Trans.	Average Leakage Power (pW)		
			0.07 $\mu$ m	0.1 $\mu$ m	0.13 $\mu$ m
2-Input XOR	COMP [15]	12	10600	3050	709
	DCVSL [13]	8	10700	2620	659
	CPL [15]	10	15700	4300	922
	DPL [15]	12	18600	5340	1150
	TG [13]	6	9320	2670	577
	LP [14]	6	7570	2070	448

#### 4 Design of Higher Order Gates

To design higher-order basic gates ( $> 2$  inputs), the COMP (Fig. 1(a)) or partitioned (Fig. 2) design style can be chosen. As partitioning simply breaks down a high order gate into lower order gates, it could be implemented using CPL gates (Fig. 1(b)). However, as it was shown in Fig. 4 that the CPL leakage power is greater than COMP for individual gates, it is expected that the partitioned CPL gates would also have a higher leakage power consumption. For this reason, in this paper, only complementary CMOS and partitioned CMOS (denoted from hereon as ‘partitioned’) design styles are analysed. Higher-order XOR gates were not investigated as they have limited practical use.

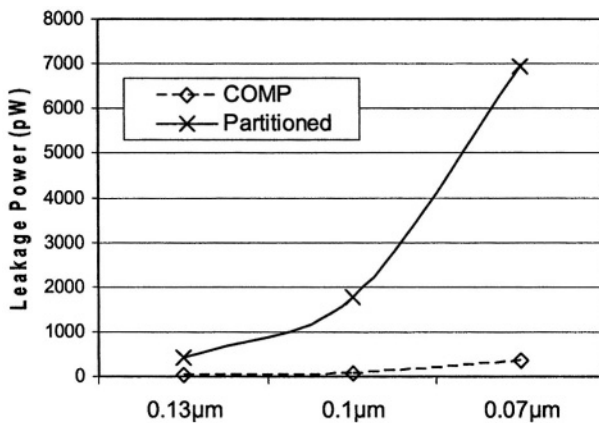
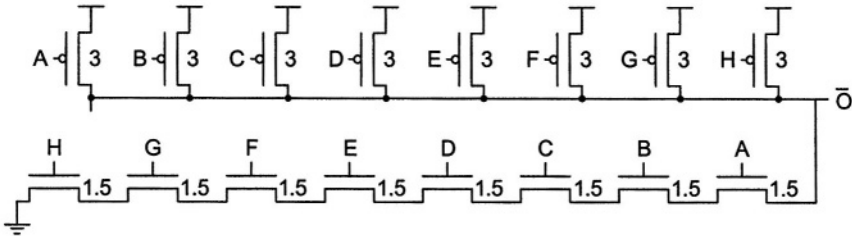
**Fig. 6.** Leakage power of COMP and partitioned 8-input NAND at three DSM technologies

Fig. 6 shows the leakage power of an 8-input NAND gate using COMP (Fig. 7) and partitioning (Fig. 2) at three DSM technologies. For a given technology, it can be seen that the COMP gate consumes less power through leakage than the equivalent partitioned gate. This is because, as outlined in Section 2, the more transistors there are in a stack, the lower the leakage current is through the stack. A COMP NAND or NOR gate has one deep stack – eight in Fig. 7 (leading to one leakage current). Partitioning, however, introduces several smaller stacks (giving rise to more, larger leak-

**Table 2.** Leakage power of two designs for high order gates at three DSM technologies

Design Style	Gate	Average Leakage Power (pW)		
		0.07 $\mu$ m	0.1 $\mu$ m	0.13 $\mu$ m
COMP	4ip NAND	1590	416	116
	6ip NAND	723	173	58
	8ip NAND	350	73	32
	4ip NOR	1390	417	91
	6ip NOR	557	181	40
	8ip NOR	221	83	20
Partitioned	4ip NAND	10600	2880	656
	6ip NAND	8580	2260	531
	8ip NAND	6940	1770	424
	4ip NOR	10300	3040	686
	6ip NOR	8130	2450	562
	8ip NOR	6440	2000	464



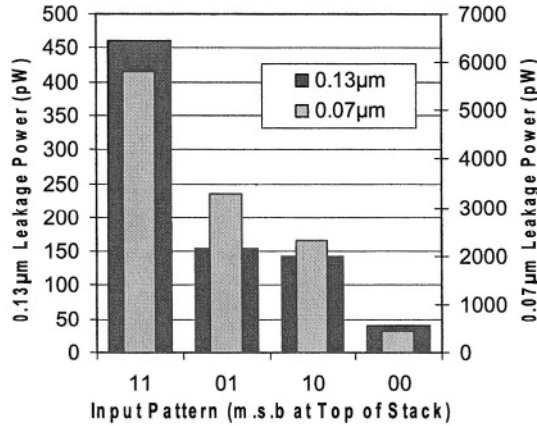
**Fig. 7.** 8-input COMP NAND gate

age currents). This means that the average leakage power in the partitioned gate is higher than that of the COMP gate. The leakage power performance of 4, 6 and 8 input COMP and partitioned NAND designs are given in Table 2.

We have analysed the leakage power performance of an 8-input NOR gate for a number of DSM technologies, and the results are given in Table 2. Again, this shows that the COMP design is more efficient than partitioning from a leakage power perspective. Fig. 6 and Table 2 indicate that designers should choose the COMP design style to obtain leakage power efficient high order NAND and NOR gates. However, this choice is achieved with an inferior speed performance [13], particularly when the number of gate inputs is greater than six. This trade-off needs to be given careful consideration. It can also be seen from Table 2 that, as the number of inputs to a gate increases, the average leakage power decreases. This is because a higher number of inputs causes a longer stack, and hence a lower average leakage power.

### 5 Input Pattern Ordering

As outlined in Section 2, the leakage power consumed in a CMOS circuit is dependant on the inputs to the gate. Previous research demonstrated that the leakage power is



**Fig. 8.** Leakage power of  $0.07\mu\text{m}$  and  $0.13\mu\text{m}$  2-input COMP NAND gate for all input patterns

the same for inputs “0,1” and “1,0” at  $0.35\mu\text{m}$  [7], and that leakage current in a stack is ‘almost’ independent of ordering for a constant number of ‘off transistors’ [12].

The 2-input COMP NAND gate (Fig. 1(a)) was simulated for a number of DSM processes. Fig. 8 shows that the leakage current for the inputs “0,1” and “1,0” are no longer equal, and the difference increases as the DSM process shrinks. It can be seen that the input combination “0,1” produces a larger leakage current than “1,0”.

Exhaustive simulations were carried out for high order NAND gates, and it was observed that the leakage power varied considerably for patterns containing only one ‘zero’ – shown in Fig. 9 for an 8-input NAND. It can be seen that the input pattern “01111111” produces the largest current whilst “11111110” produces the least. We believe this observation should be exploited. Similarly, the equivalent observation for NOR gates was found; for input combinations containing a single ‘one’, “10000000” gives rise to the smallest leakage current, while “00000001” gives the largest.

This observation can be explained by the fact that, for an 8-input COMP NAND, the input “01111111” gives rise to a greater body effect in the transistor acting in the weak inversion region than the input “11111110”. This is investigated further without reference to ordering in [11].

Table 3 shows the saving that can be made for IS1 leakage currents by taking note of input ordering. For example, with an 8-input NAND gate at  $0.07\mu\text{m}$ , a saving of 1230pA can be obtained by using the input “01111111” rather than “11111110”, and this equates to a percentage saving of 34%. In order to exploit this observation, the following guidelines should be followed:

- For NMOS stacks (NAND gates), a ‘zero’ closest to the output node of the gate will give rise to the largest IS1 leakage current, while a ‘zero’ closest to ground will give the smallest IS1 leakage current. For PMOS stacks (NOR gates), a ‘one’ closest to the output node of the gate will give rise to the largest IS1 leakage current, while a ‘one’ closest to Vdd will give the smallest IS1 leakage current.

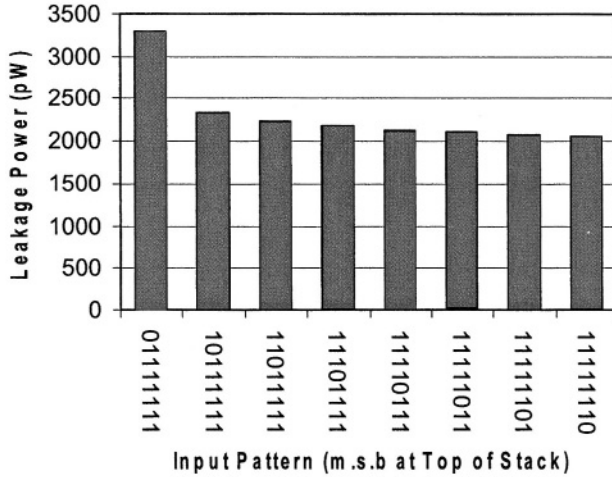


Fig. 9. Leakage power of an 0.07µm 8-input NAND gate for all inputs containing a single zero

Table 3. Leakage power savings through transistor ordering for three DSM technologies

Gate	Average Leakage Power (pW)		
	0.07µm	0.1µm	0.13µm
2ip NAND	969 (26%)	305 (41%)	11.1 (8%)
3ip NAND	1070 (29%)	330 (44%)	11.8 (8%)
4ip NAND	1130 (31%)	342 (46%)	11.8 (8%)
6ip NAND	1200 (33%)	354 (47%)	11.8 (8%)
8ip NAND	1230 (34%)	361 (48%)	11.8 (8%)
2ip NOR	1240 (39%)	373 (36%)	74 (36%)
3ip NOR	1370 (42%)	408 (40%)	81 (39%)
4ip NOR	1420 (44%)	426 (41%)	84 (40%)
6ip NOR	1490 (46%)	445 (43%)	85 (41%)
8ip NOR	1530 (48%)	456 (44%)	85 (41%)

## 6 Conclusions

We have presented a systematic comparison of the leakage power for basic gates, designed using different design styles and circuit topologies, and implemented at three DSM technologies. The results have shown that complementary CMOS is favoured over CPL and gate partitioning for implementing both low order and high order NAND and NOR gates. The low power XOR gate design developed in [14] provides a leakage power saving of 50% over CPL. These findings need to be considered carefully when choosing a particular design style due to the trade-offs that exist. Whilst complementary CMOS has the least leakage power for high-order gates, this is achieved at the expense of speed that the partitioning design style permits. We have shown how input pattern ordering may be exploited to reduce leakage power.

## References

1. Piguët, et al.: "Extremely Low-Power Logic," in Proc. Design Automation and Test in Europe Conf., Feb 2004, pp. 656-661
2. Y-F. Tsai, D. Duarte, N. Vijaykrishnan and M. J. Irwin: "Implications of Technology Scaling on Leakage Reduction Techniques" in Proc. Conf. Des. Automation, 2003, pp. 187-190
3. R. X. Gu and M. I. Elmasry: "Power Dissipation Analysis and Optimisation of Deep Submicron CMOS Digital Circuits," IEEE J. Solid-State Circuits, May 1996, vol. 31, pp. 707-713
4. N. S. Kim, et al.: "Leakage Current: Moore's Law Meets Static Power," IEEE J. Computer, Dec. 2003, vol. 36, pp. 68-75
5. L. Wei, K. Roy and V. K. De: "Low Voltage Low Power CMOS Design Techniques for Deep Submicron ICs," Int. Conf. VLSI Design, Jan. 2000, pp. 24-29
6. D. Liu and C. Svensson: "Trading Speed for Low Power by Choice of Supply and Threshold Voltages," IEEE J. Solid-State Circuits, Jan 1993, vol. 28, pp. 10-17
7. K. S. Khouri and N. K. Jha: "Leakage Power Analysis and Reduction During Behavioural Synthesis," IEEE Trans. VLSI Systems, Dec. 2002, vol. 10, pp. 876-885
8. K. Roy: "Leakage Power Reduction in Low-Voltage CMOS Designs," in Proc. Int. Conf. Electron., Circuits Syst., Sept 1998, pp. 167-173
9. N. Sirisantana and K. Roy: "Low-Power Design Using Multiple Channel Lengths and Oxide Thicknesses," in IEEE Design & Test of Computers, Jan. 2004, vol. 21, pp. 56-63
10. J. P. Halter and Farid N. Najm: "A Gate-Level Leakage Power Reduction Method for Ultra-Low-Power CMOS Circuits," in IEEE Custom Integrated Circuits Conf., 1997, pp. 475-478
11. Z. Chen, L. Wei, M. Johnson and K. Roy: "Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modelling of Transistor Stacks," in Proc. Symp. Low Power Design Electron., 1998, pp. 239-244
12. W. Jiang, V. Tiwari, E. de la Iglesia and A. Sinha: "Topological Analysis for Leakage Power Prediction of Digital Circuits," in Proc. Int. Conf. VLSI Design, Jan. 2002, pp. 39-44
13. J. M. Rabaey: "Digital Integrated Circuits (A Design Perspective)." New Jersey, Prentice Hall, 1996
14. J. M. Wang, S. C. Fang and W. S. Feng: "New Efficient Designs for XOR and XNOR Functions on the Transistor Level," IEEE Solid-State Circuits, July 1994, vol. 29, pp. 780-786
15. R. Zimmermann and W. Fichtner: "Low-Power Logic Styles: CMOS Versus Pass-Transistor Logic," IEEE J. Solid-State Circuits, July 1997, vol. 32, pp. 1079-1090
16. M. Alioto and G. Palumbo: "Analysis and Comparison on Full Adder Block in Submicron Technology," IEEE Trans. VLSI Systems, Dec. 2002, vol. 10, pp. 806-823
17. A. P. Chandrakasan, S. Sheng and R. W. Bordersen: "Low-Power CMOS Digital Design," in IEEE J. Solid-State Circuits, Apr. 1992, vol. 27, pp. 473-484
18. Avant!: "Star-Hspice Manual (release 1999.2)." June 1999
19. Berkeley Predictive Technology Model (BPTM): Device Group, Univ. California at Berkeley [online] Available: <http://www-device.eecs.berkeley.edu/~ptm/>.



# Threshold Mean Larger Ratio Motion Estimation in MPEG Encoding Using LNS

Jie Ruan and Mark G. Arnold

Lehigh University, Bethlehem PA 18015, USA,  
marnold@eecs.lehigh.edu

**Abstract.** The Logarithmic Number System (LNS) offers good performance for some multimedia applications at low precisions. Motion estimation is a key part of the MPEG encoding system. Usually the motion estimation is performed by using a fixed-point Mean Absolute Difference (MAD) cost function for block matching. LNS Mean Larger Ratio (MLR) is another cost function which can produce comparable video encoding results as fixed-point MAD. This paper proposes an optimized Threshold MLR (TMLR) cost function, which uses a threshold value to keep only the larger LNS ratio of the pixel values with shorter word lengths. The encoding results show that the optimization makes significant area savings with little quality degradation. This paper also shows that the threshold approach can be applied to the fixed-point MAD. The Threshold MAD (TMAD) can achieve similar area savings as the TMLR does.

## 1 Introduction

The Motion Picture Experts Group (MPEG) system is the most widely used video compression/decompression system. The MPEG needs large amounts of hardware/software resources, especially in its encoding process. The Logarithmic Number System (LNS) can offer a better performance than fixed-point for some applications at low precisions. Previous research in [1], [2], [10] and [11] has shown that LNS provided a better result than fixed-point in the DCT/IDCT part of the MPEG encoding/decoding process. Motion Estimation (ME) is another computationally intensive part in the MPEG encoding system. This paper introduces LNS into the motion-estimation process. Section 2 briefly introduces the Logarithmic Number System. Section 3 introduces Motion Estimation (ME) and its most widely used algorithm and cost function. Section 4 proposes Threshold Mean Larger Ratio (TMLR) by using thresholds to decrease the input word lengths of the accumulator. Section 5 shows the area analysis for the designs with and without threshold values. Section 6 shows the gate-level switch activities when TMLR and TMAD are used on block-matching motion estimation. Section 7 shows the synthesis results. Section 8 draws conclusions.

## 2 Logarithmic Number System (LNS)

The Logarithmic Number System (LNS) [8] represents a number ( $X$ ) with its exponent ( $x$ ) to a certain base  $b$  (such as  $b = 2$ ) and with the sign of the represented number ( $x_s$ ). Thus  $X \approx (-1)^{x_s} \cdot b^x$ .

Multiplication of two numbers simply requires the sum of the two numbers' parts. It only requires a fixed-point adder and an XOR gate, compared to the higher cost of a fixed-point multiplier.

However, the addition of two numbers,  $T = X + Y$  is not a linear operation in LNS. The basic method for the addition calculation [1] is:

1.  $z = y - x$  and  $z_s = x_s \oplus y_s$ ;
2. If  $z_s = 0$ ,  $w = s_b(z)$ ; otherwise  $w = d_b(z)$ ;
3.  $t = x + w$ .

Here,  $x$  and  $y$  denote approximations to  $\log_b(|X|)$  and  $\log_b(|Y|)$ ;  $x_s$  and  $y_s$  denote the corresponding sign bits of the two operands. In Step 1, the calculation of  $z = y - x$  is equivalent to calculating  $z = \log_b(Y/X)$ . Step 2 is a table look-up. Tables  $s_b(z)$  and  $d_b(z)$  correspond to addition or subtraction according to  $z_s$ :

$$s_b(z) = \log_b(1 + b^z), \tag{1}$$

$$d_b(z) = \log_b|1 - b^z|. \tag{2}$$

Thus, the final result is  $t = \log_b |X(1 \pm Y/X)| = \log_b |X \pm Y|$ .

## 3 Motion Estimation

Motion Estimation (ME) is a key part for moving-picture compression standards, such as MPEG and H.261/H.263. It exploits temporal redundancy in a series of pictures to achieve compression. It is the main difference between moving-picture compression and static-picture compression.

There are two main categories in motion-estimation algorithms: the pixel-recursive algorithm and the block-matching algorithm. Block matching has a more regular structure for the computation process so that it is suitable for hardware and software implementation. The block-matching process finds a matching macroblock ( $16 \times 16$  pixel area) within a certain search area in the reference frame for each macroblock in the frame to be encoded. The matching macroblock has the lowest cost value according to certain criteria, called cost functions. The displacement from the matching macroblock in the reference frame to the encoded macro block is recorded as the Motion Vector (MV).

There are different kinds of cost functions [6] for block matching that vary in cost and performance. The most commonly used is Mean Absolute Difference (MAD), as shown in Equation 3:

$$MAD = \sum_{i=1}^{16} \sum_{j=1}^{16} |X_{ij} - Y_{ij}|. \tag{3}$$

$X_{ij}$  and  $Y_{ij}$  are the pixel values of the macroblocks in the encoded and reference frames. MAD adds up all the absolute differences of the two macroblocks. Usually the pixel values are in fixed-point representation.

The reason that the MAD is the most commonly used cost function in motion estimation is because it can achieve high performance by using relatively inexpensive hardware/software compared to many other cost functions which require multiplications. In the MAD, only addition/subtraction operations are involved, which can be implemented in fixed-point-based hardware with low cost.

Because of the large number of macroblocks in the encoded frames, motion estimation requires a lot of computational resources. For a real-time MPEG encoding system, it is necessary to implement motion estimation in hardware. Considerable research has been done on the hardware implementation of block-matching motion estimation in [4], [5], [7] and [9].

## 4 Threshold MLR

In [12], a cost function, Mean Larger Ratio (MLR), is proposed for the LNS block-matching motion estimation which avoids addition, as shown in Equation 4:

$$MLR = \prod_{i=1}^{16} \prod_{j=1}^{16} \max\left(\frac{X_{ij}}{Y_{ij}}, \frac{Y_{ij}}{X_{ij}}\right) = \prod_{i=1}^{16} \prod_{j=1}^{16} \frac{\max(X_{ij}, Y_{ij})}{\min(X_{ij}, Y_{ij})}. \quad (4)$$

where  $X_{ij}$  and  $Y_{ij}$  are the pixel values in the macroblocks of the encoded and reference frames. In MLR, these pixel values are represented as LNS values. In this paper, the LNS pixel values are represented using 8-bit LNS numbers. In the 8-bit LNS representation, 3 bits are used as the integer part and 5 bits are used as the fractional part. Because the multiplication in LNS is equivalent to the addition in fixed-point numbers, the hardware cost of the LNS MLR is equal to the hardware cost of fixed-point MAD. [12] shows that the LNS MLR cost function can achieve a comparable video-compression ratio and video quality as the fixed-point MAD.

In most cases, the pixel values' larger ratios of  $\frac{X_{ij}}{Y_{ij}}$  and  $\frac{Y_{ij}}{X_{ij}}$  are small values, especially in the area close to the motion vector. Their several most significant bits have a high probability of being all zeros. It is possible to define a new cost function to make use of this property. In [3], a reduced-bit MAD is proposed using only the most significant bits from the pixel values' fixed-point representation when performing MAD block-matching calculations.

A new cost function, Threshold Mean Larger Ratio (TMLR), is proposed as shown in Equation 5:

$$V_{ij} = \begin{cases} \max\left(\frac{X_{ij}}{Y_{ij}}, \frac{Y_{ij}}{X_{ij}}\right), & \text{if } \max\left(\frac{X_{ij}}{Y_{ij}}, \frac{Y_{ij}}{X_{ij}}\right) \leq T \\ T, & \text{otherwise} \end{cases} \quad (5)$$

$$TMLR = \prod_{i=1}^{16} \prod_{j=1}^{16} V_{ij}.$$

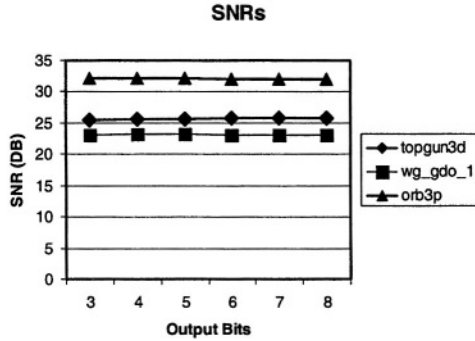


Fig. 1. SNRs for Different Thresholds.

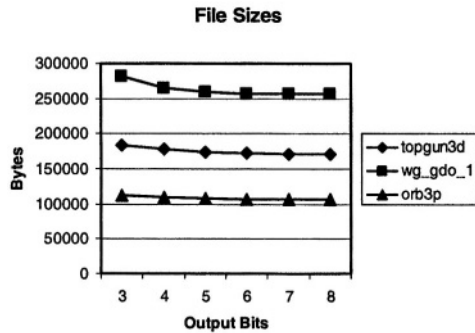


Fig. 2. Encoded File Sizes for Different Thresholds.

According to the TMLR, the LNS larger ratios keep their values if the ratios are less than the threshold  $T$ . Otherwise, the ratios are recorded as the threshold  $T$ . In this way, the larger ratios are kept in fewer bits than if they were kept in their full range. For example, if  $T = 2^{2^{-1}}$ , only 5 bits are needed to be kept instead of 8 bits.

To lower the hardware cost, the thresholds are chosen as powers of two. This allows the comparison of the LNS larger ratio value with the threshold to be implemented by ORING the ratio's several most significant bits.

Three video clips of different types were encoded using LNS MLR with different thresholds. Figure 1 shows the SNRs of the encoded videos with different thresholds (topgun3d, wg\_gdo\_1 and orb3p are the names of the encoded videos). Figure 2 shows the file sizes of the three videos with different thresholds.

The results show that at  $T = 2^{2^{-1}}$  (5-bit output after being thresholded), both the file sizes and the SNRs are quite close to the LNS MLR cost function without threshold.

The threshold value can be chosen to be a diminished power of two. The videos encoded using the diminished power-of-two thresholds have almost the same SNRs and file sizes with one output bit saved.

The threshold approach can also be applied on the fixed-point MAD cost function. When the Threshold MAD (TMAD) is used on the same videos, it produces similar SNR results and the same area saving result at the same output word lengths.

### 5 Area Analysis

The hardware cost of TLMR includes an extra threshold circuit for each larger ratio of  $X_{i,j}$  and  $Y_{i,j}$ . In this way, the larger ratios can be recorded in fewer bits, instead of the full range of 8 bits. There are 255 LNS multipliers in total for the product of the 256 ratios of each macroblock.

Figure 3 shows the hardware with the threshold  $T = 2^{2-1}$ . The input 8 bits,  $R_0$  to  $R_7$ , are used for the larger ratio of the two pixels' LNS value,  $\max(\frac{X_{i,j}}{Y_{i,j}}, \frac{Y_{i,j}}{X_{i,j}})$ . The outputs  $V_0$  to  $V_4$  are the 5-bit output. The three OR gates provide the control signal that indicates that the larger ratio exceeds the threshold. The four AND gates choose whether to pass the ratio value or the threshold value.

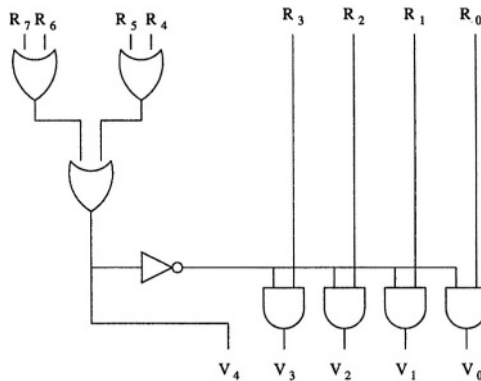
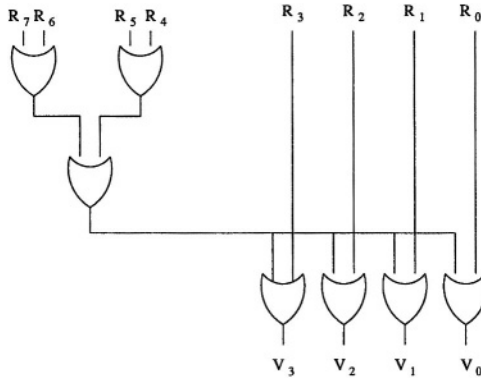


Fig. 3. Circuit for Power-of-two Thresholding of the Larger LNS Ratio.

Figure 4 shows the hardware on one pair of pixels' larger ratio with the threshold  $T = 2^{2-1-2-5}$ . The left three OR gates provide the control signal indicating that the larger ratio exceeds the threshold. The right four OR gates choose whether to pass the ratio value or the threshold value.

Tables 1 and 2 show, for different thresholds, the extra hardware TMLR adds and removes as well as the net hardware savings on the later calculation of the product of these ratios. Because one Full Adder (FA) consists of six 2-input gates, the smaller the threshold values, the larger the area savings. Savings occur except for  $T = 4$  in Table 1.

A main computation part of the block-matching motion estimation in MPEG is the accumulator circuits to compute the sum of the 256 absolute differences



**Fig. 4.** Circuit for Diminished Power-of-two Thresholding of the Larger LNS Ratio.

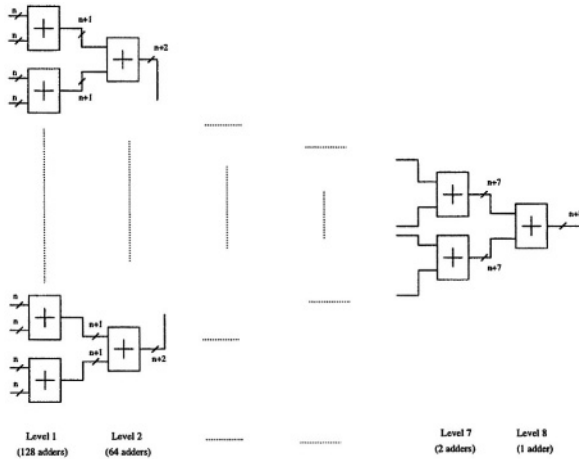
**Table 1.** Hardware Cost and Savings for Power-of-two Thresholds.

Threshold (T)	Added Hardware				Removed Hardware		Gates Saved
	OR gates	AND gates	Inverters	Total	Full Adders	Gates	
$2^{2^{-3}}$	5	2	1	8	5	30	22
$2^{2^{-2}}$	4	3	1	8	4	24	16
$2^{2^{-1}}$	3	4	1	8	3	18	10
$2^1$	2	5	1	8	2	12	4
$2^2$	1	6	1	8	1	6	-2

**Table 2.** Hardware Cost and Savings for Diminished Power-of-two Thresholds.

Threshold (T)	Added Hardware	Removed Hardware		Gates Saved
	OR gates	Full Adders	Gates	
$2^{2^{-3}-2^{-5}}$	7	6	36	29
$2^{2^{-2}-2^{-5}}$	7	5	30	23
$2^{2^{-1}-2^{-5}}$	7	4	24	17
$2^{1-2^{-5}}$	7	3	18	11
$2^{2-2^{-5}}$	7	2	12	5

(or larger ratios for LNS). The directly inferred structure is the 8-level adder-tree structure shown in Figure 5. The numbers of adders in each level are 128, 64, 32, 16, 8, 4, 2 and 1. Let the input word length at the first level be  $n$  bits. The word lengths of the adders in each level are  $n, n + 1, n + 2, \dots, n + 7$ . The threshold approach has a shorter word length in the input of the first level adder; thus it has shorter word lengths in each level of adders. The extra hardware for the threshold approach is an extra circuit to determine if the larger LNS ratio is greater than the threshold. The numbers of the FAs and the HAs in the method without the threshold are 2032 and 255. The number of Full Adders (FAs) for the threshold approach is  $255n - 8$ ; the number of Half Adders (HAs) is 255; the number of OR



**Fig. 5.** 8-level Adder-Tree Structure to Calculate the Product of the 256 Larger Ratios of the Pixel Values in One Macroblock.

**Table 3.** Area Analysis for Original Method and Power-of-two Thresholds.

	Without Threshold	With Threshold (T)				
	8-bit output	3-bit	4-bit	5-bit	6-bit	7-bit
FAs	2032	757	1012	1267	1522	1777
HAs	255	255	255	255	255	255
ANDs		512	768	1024	1280	1536
ORs		1280	1024	768	512	256
INVs		256	256	256	256	256
Total gates	13212	7610	9140	10670	12200	13730
Net savings		5602	4072	2542	1012	-518

**Table 4.** Area Analysis for Original Method and Diminished Power-of-two Thresholds.

	Without Threshold	With Threshold (T)				
	8-bit output	2-bit	3-bit	4-bit	5-bit	6-bit
FAs	2032	502	757	1012	1267	1522
HAs	255	255	255	255	255	255
ORs		1792	1792	1792	1792	1792
Total gates	13212	5824	7354	8884	10414	11944
Net savings		7388	5858	4328	2798	1268

gates is  $256(8 - n)$ . For power-of-two thresholds, where  $T = 2^{2^{n-F}-1}$ , the number of additional AND gates and inverters are  $256(n - 1)$  and 256, respectively. For diminished power-of-two thresholds, where  $T = 2^{2^{n-F}-2^{-F}}$ , the number of additional OR gates is  $256(n - 1)$ , bringing the total number of OR gates to 1792. The larger the number of output bits, the larger the number of gates needed for

the threshold method. Tables 3 and 4 show the area comparison of the number of gates needed for the 8-level adder-tree hardware implementation, assuming a half adder takes four gates.

The conversion between the fixed-point number representation and the LNS representation requires table-lookup hardware. If the whole MPEG encoding system is implemented in LNS, the extra hardware for conversion is added at the beginning and at the end of the system. Thus the conversion hardware will be relatively small compared to the whole system. The ability to perform competitive motion estimation in the LNS domain allows such an implementation to be cost effective.

## 6 Switching Activities

Simulation was performed for the accumulator circuits with  $T = 2^{2^{-1}}$  for LNS TMLR and  $T = 2^4$  for fixed-point TMAD. Figure 6 shows the comparison of the observed number of gate-level switches.

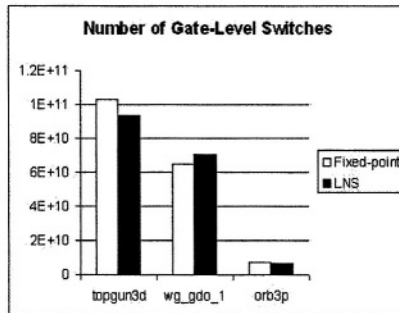


Fig. 6. Switching Activity Simulation Results.

For the three videos, the number of gate-level switches for the LNS TMLR in the videos *topgun3d* and *orb3p* are 9.4% and 4.3% fewer than the fixed-point TMAD. In the video *wg\_gdo\_1* LNS, TMLR is 8.6% more than the fixed-point MAD. Overall, the switching activities for LNS TMLR and fixed-point MAD are approximately equal.

## 7 Synthesis Results

Leonardo Spectrum is used as the synthesis tool. ASIC library *SCL05U* is selected as the target library. In this standard ASIC library, one inverter is counted



**Table 5.** Synthesis Result for Original Method and Power-of-two Thresholds.

Designs	Word Lengths to the Accumulator	Number of Gates	
Without Threshold	8	38620	
$T = 2^{2^{-3}}$	3	20569	47% less
$T = 2^{2^{-2}}$	4	26184	33% less
$T = 2^{2^{-1}}$	5	31837	18% less
$T = 2$	6	38168	1% less

**Table 6.** Synthesis Result for Original Method and Diminished Power-of-two Thresholds.

Designs	Word Lengths to the Accumulator	Number of Gates	
Without Threshold	8	38620	
$T = 2^{2^{-3}-2^{-5}}$	2	15832	59% less
$T = 2^{2^{-2}-2^{-5}}$	3	21069	45% less
$T = 2^{2^{-1}-2^{-5}}$	4	27106	30% less
$T = 2^{1-2^{-5}}$	5	33744	13% less

as 3 gates. Table 5 lists the synthesis results for the designs without a threshold and with different power-of-two thresholds.

Table 6 shows the synthesis results for the designs without a threshold and with different diminished power-of-two thresholds. The synthesis results show that the shorter the input word lengths are, the larger the area savings can be. Considering the quality degradation from the reduced word lengths,  $T = 2^{2^{-1}-2^{-5}}$  (4-bit output) may be the best choice. For instance, the synthesis results show that the hardware for the threshold  $T = 2^{2^{-1}-2^{-5}}$  takes 30% less area than the hardware without a threshold.

## 8 Conclusion

This paper proposes an optimized threshold cost function, Threshold Mean Larger Ratio (TMLR), for the block-matching motion estimation. The encoding and synthesis results show that the TMLR cost function can achieve area savings with little quality degradation and minor increased file sizes on the encoded videos. Because of the low-power property of LNS, this paper provides a solution to a major problem in MPEG encoding, thus making it possible to build a portable MPEG encoding system based on LNS.

## References

1. Mark G. Arnold, "LNS for Low Power MPEG Decoding," *SPIE Adv. Signal Proc. Algorithms, Arch. & Implementations XII*, Seattle, vol. 4791, pp. 369-380, July, 2002.
2. Mark G. Arnold, "Reduced Power Consumption for MPEG Decoding with LNS," *Appl. Specific Arch. & Proc.*, San Jose, pp. 65-75, July, 2002.
3. Yunju Baek, Hwang-Seok Oh and Heung-Kyu Lee, "An Efficient Block-Matching Criterion for Motion Estimation and its VLSI Implementation," *IEEE Trans. on Consumer Electronics*, vol. 42, pp. 885-892, November, 1996.
4. Sheu-Chih Cheng and Hsueh-Min Hang, "Comparison of Block-Matching Algorithms Mapped to Systolic-Array Implementation," *IEEE Trans. on Circuits & Systems for Video Technology*, vol. 7, issue 5, pp. 741-757, October, 1997.
5. Yuk Ying Chung, Man To Wong and N.W. Bergmann, "Custom Computing Implementation of Two-Step Block Matching Search Algorithm," *Proc. of IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing, ICASSP '00*, vol. 6, pp. 3231-3234, June, 2000.
6. Borko Furht, Joshua Greenberg and Raymond Westwater, *Motion Estimation Algorithms for Video Compression*, Kluwer Academic Publishers, 1997.
7. Young-Ki Ko, Hyun-Gyu Kim, Jong-Wook Lee, Yong-Ro Kim, Hyeong-Cheol Oh and Sung-Jea Ko, "New Motion Estimation Algorithm Based on Bit-Plane Matching and its VLSI Implementation," *IEEE TENCON*, pp. 848-851, 1999.
8. Israel Koren, *Computer Arithmetic Algorithms*, Prentice Hall, Inc, 1993.
9. Jeng-Huang Luo, Chung-Neng and Tihao Chiang, "A Novel All-Binary Motion Estimation (ABME) with Optimized Hardware Architectures," *IEEE Trans. on Circuits & Systems for Video Technology*, vol. 12, pp. 700-712, August, 2002.
10. Jie Ruan and Mark G. Arnold, "LNS Arithmetic for MPEG Encoding Using a Fast DCT," *Proc. of the Work-in-Progress Session of 29th EuroMicro Conf.*, Belek (Antalya), Turkey, pp. 47-48, September, 2003.
11. Jie Ruan and Mark G. Arnold, "Combined LNS Adder/Subtractors for DCT Hardware," *Proc. of First Workshop on Embedded Systems for Real-Time Multimedia*, Newport Beach CA, pp. 118-123, October, 2003.
12. Jie Ruan and Mark G. Arnold, "New Cost Function for Motion Estimation in MPEG Encoding Using LNS," to be presented at *SPIE Adv. Signal Proc. Algorithms, Arch. & Implementations XIV*, Denver, CO, August, 2004.

# Energy- and Area-Efficient Deinterleaving Architecture for High-Throughput Wireless Applications

Armin Wellig<sup>1</sup>, Julien Zory<sup>1</sup>, and Norbert Wehn<sup>2</sup>

<sup>1</sup> STMicroelectronics – Advanced System Technology,  
Chemin du Champ-des-Filles 39, 1228 Geneva, Switzerland  
{armin.wellig,julien.zory}@st.com

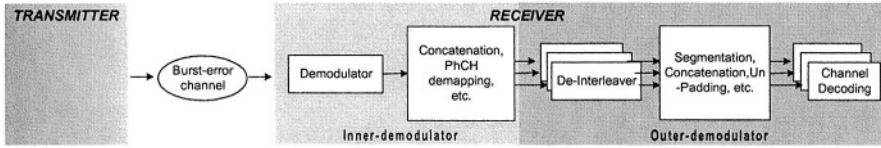
<sup>2</sup> University of Kaiserslautern, Postfach 3049,  
67653 Kaiserslautern, Germany  
norbert.wehn@eit.uni-kl.de

**Abstract.** Most of today's wireless systems implement some sort of channel interleaving to minimize burst errors. The interleaver breaks the temporal correlation of successive samples by reshuffling them prior to transmission. In state-of-the-art solutions, this memory intensive operation is both power consuming due to frequent accesses to large memories and area inefficient due to poor memory-reuse. In this paper, we present a cluster-based deinterleaving method which makes a two-level custom memory hierarchy design possible. The access rate to large memories is reduced by performing sub-array deinterleaving inside much smaller memories thereby lowering the energy dissipation significantly. Moreover, this novel approach solves the memory-reuse bottleneck. Costly adaptation buffers become obsolete yielding a scalable, energy- and area-efficient architecture. The high-speed data packet access (HSDPA) defined in 3GPP serves as target application, where energy reductions of up to 60% with negligible logic overhead is observed for STMicroelectronics' SRAM library in 0.13  $\mu\text{m}$  CMOS technology.

## 1 Introduction

System-on-Chip (SoC) are complex heterogeneous integrated circuits made of various components. Memories play an important role in such SoCs as they tend to dominate the die area. According to the ITRS roadmap [1], the fraction of die area occupied by memory will go up to 94% in 2014. In addition to the area issue, memories are also critical with respect to yield and throughput. Moreover, frequent memory accesses translate into dynamic power consumption which significantly contributes to the overall energy dissipated in SoCs for currently used process technologies [2][3].

A typical application domain for SoCs are handheld wireless devices. In such SoCs, the modem functionality is a basic building block whose main design challenge is to implement efficient receiver structures. As illustrated in Fig. 1, the outer demodulator consist of tasks such as data stream (de-)multiplexing, channel deinterleaving and decoding [4], which are used (among other things) to improve the system's communication performance. Interleaving breaks the temporal correlation among consecutive data samples and thus minimizes burst errors. Channel (de-)coding guar-



**Fig. 1.** The basic building blocks of a wireless demodulation system

antees system reliability by adding (structured) redundancy to the transmitted payload information.

Channel deinterleaving and decoding are both memory intensive operations which are among the top “power-drainers” in digital high-throughput receivers. Memory optimizations in channel decoders are beyond the scope of this paper and have already been discussed in e.g. [5][6]. There are two standard channel interleaving techniques commonly referred to as *block* and *convolutional* (de-)interleavers [7][8], where the first scheme operates on data blocks and the latter one on a continuous data stream. In this paper, we will focus on the design of block deinterleavers which is the method of choice in various cellular systems (e.g., 3GPP, IS-95, etc.). State-of-the-art implementations suffer high energy dissipation due to numerous sample-wide accesses to large memory components with limited temporal and spatial locality. Moreover, block deinterleaving results in memory elements with long “storage-lifetimes” which drastically limits continuous memory-reuse. As a result, large adaptation buffers are required that further increase die area and energy dissipation. Consequently, in this paper, we will focus on deinterleaving techniques that take advantage of spatial-correlation properties among block interleaved samples to implement an energy-efficient memory hierarchy and to get rid of additional adaptation buffers.

The paper is structured as follows: In Section 2, we will introduce the state-of-the-art solution highlighting its major limitations. A novel cluster-based deinterleaving concept is described in Section 3 by starting off with an analysis of the spatial correlation properties; techniques to reduce both energy dissipation and die area are subsequently described and the template architecture is introduced. In Section 4, both deinterleaving methods are compared with respect to die area and energy in a 0.13  $\mu\text{m}$  STMicroelectronics CMOS process technology. Section 5 finally concludes the paper.

## 2 State-of-the-Art Approach

In its simplest setup, the block interleaver writes the input data along the rows of a memory configured as a matrix, and then reads it out along the columns. A generalization of a (simple) block interleaver is a *pseudorandom block interleaver*, in which data is written in memory row by row in sequential order and read column by column in a pseudorandom order (defined by a permutation table). Note that this will be the default interleaving method thereafter. The deinterleaving simply performs the inverse operation to retain the original sequence. As illustrated below, in state-of-the-art implementations [7][9] a pseudorandom block deinterleaving unit writes the data samples one by one into the memory columns; inter-column permutation is performed either explicitly or implicitly via proper address computations; samples are finally read out row by row (and transferred to the next processing block).

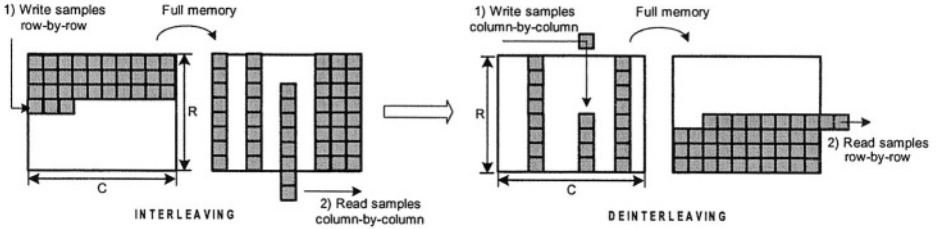


Fig. 2. Snapshot of the pseudorandom block interleaving (left) and deinterleaving (right)

As illustrated in Fig. 2, the classical deinterleaver offers poor data locality. Indeed, there is no temporal locality at all (each sample is only written and read once) and quite limited spatial locality; the write access is restricted to the sample width (thereafter referred to as *sample-wide* access), whereas the read access can be done per cluster (thereafter referred to as *cluster-wide* access) where one *cluster* is composed of several data samples. Banked memory designs are conceivable to allow both sample- and cluster-wide accesses. However, the additional control logic as well as the increased data and address bus bandwidth are not in favor of an energy-efficient implementation. Another important flaw in this explicit deinterleaving approach is its tight system dependency. That is, sample  $N$  of frame  $F$  has to be written to the same memory location as sample  $N$  of frame  $F+1$ ,  $F+2$  and so on. For a generic block deinterleaver arranged as a matrix with  $R$  rows and  $C$  columns, the first row of the memory can only be read after at least  $(C-1) \cdot R + 1$  samples have been written into it, yielding memory elements with long “storage-lifetimes”. Those constraints result in a poor memory-reuse; that is, to prevent storage overwrite in the presence of a continuous input flow (e.g. the output of the inner demodulator) adaptation buffers are required. To avoid this buffer penalty a fully-associative scheduling is feasible at the expense of increased data and control processing. For instance, the address look-up table (LUT) necessary to keep track of the “sample to memory address location” mapping would double the storage requirements.

Some previous work on block deinterleaver architecture was presented in [10][11], where the focus is on the design of (low-level) flexibility of IP-cores. For instance, the proposed solutions are flexible with respect to selecting the interleaving method, adjusting the permutation tables or configuring the size of the interleaver matrix. In [12], a “per group” parallel deinterleaving scheme is proposed. This concept is similar to our approach of embedded sub-array reordering presented next. Nonetheless, its application completely differs as authors of [12] are motivated by low-latency aspects while we exploit it to implement a custom memory hierarchy to decrease energy dissipation. The authors are not aware of any publication addressing energy efficiency and/or memory-reuse bottleneck of block deinterleaving architectures.

### 3 Embedded Reordering

This section introduces a novel deinterleaving approach targeting energy efficiency and memory-reuse. Due to the limited space of this paper, further details about the formalism and the micro-architecture will be the subject of a subsequent publication.

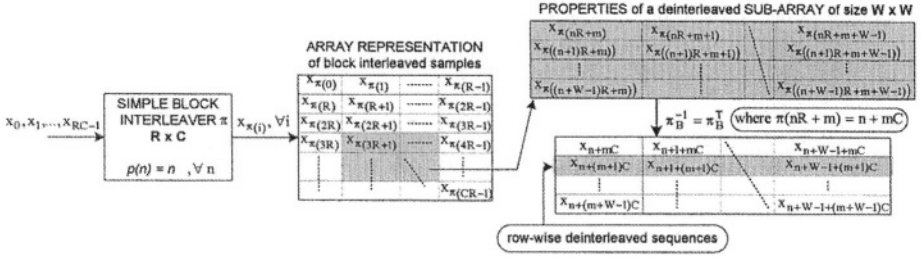


Fig. 3. Correlation properties of a (simple) block deinterleaved sub-array of size  $W \times W$

### 3.1 Spatial Correlation Properties Among Block Interleaved Samples

An interleaver can be formalized as a device  $\pi$  that rearranges the ordering of a sequence  $X$  of  $S$  samples  $x_0, x_1, \dots, x_{S-1}$  in some one-to-one deterministic manner; the output (interleaved) sequence  $Y$  is denoted as  $y_0, y_1, \dots, y_{S-1}$ . It can be shown that for any pseudorandom block interleaver with  $R$  rows and  $C$  columns, i.e.,  $S = R \times C$  which is referred to as deinterleaving data set (DDS) thereafter,  $y_i = x_{\pi(i)}$  with  $\pi(i)$  defined by

$$\pi(i) = p(\lfloor i/R \rfloor) + (i \bmod R) \cdot C \quad (1)$$

where  $\lfloor \cdot \rfloor$  denotes the floor function,  $\bmod$  the modulo operator and  $p(j)$  specifies the original column position of the  $j$ -th permuted column of the interleaver matrix of size  $R \times C$ . Equation (1) simply describes that for any sample at position  $i \in \{0, RC-1\}$  in the interleaved sequence, one can retrieve (compute) the corresponding position  $\pi(i)$  in the original input sequence. To better illustrate how this relationship translates into interesting spatial correlation properties among interleaved samples, let us represent the interleaved sequence in a matrix  $A$  of size  $(R_A=C) \times (C_A=R)$  with  $n \in \{0, \dots, C-1\}$  the row index and  $m \in \{0, \dots, R-1\}$  the column index. If, for simplification, we limit the illustration to a simple block interleaver (the permutation table follows  $p(n)=n$ ) one interleaved sample  $A_{n,m}$  in the matrix  $A$  is defined by  $A_{n,m} = X_{\pi(n.R+m)} = X_{n+m.C}$ .

Two interesting properties are illustrated in Fig. 3 above:

- (p1) Following the matrix representation the deinterleaving can be seen as a matrix transpose. It can be shown that deinterleaving (hence transposing) a sub-array of arbitrary size  $W \times W$  yields  $W$  row-wise *reordered* sequences (each of size  $W$ ).
- (p2) Thanks to equation (1) the addresses of all samples contained in a cluster of size  $W$  can be retrieved (computed) from the address of the first sample (for  $W \leq R$ ).

**From sub-array deinterleaving to energy-efficient memory hierarchy:** the possibility to perform partial sub-array deinterleaving can be exploited to implement a custom energy-efficient memory hierarchy. It is well known that for on-chip memories (which are not intensively partitioned) the access energy increases *more than linearly* with its size. Hence, power savings can be achieved by moving data accesses with locality to smaller memories [13][14], yielding memory hierarchies such as the traditional caches. As illustrated in Table 1, one can also exploit the property that the access energy increases *less than linearly* with the access bit-width for a fixed memory size. Let us define word-accessible (Wa-)SRAM a memory device that is word-

**Table 1.** Average energy dissipation in  $pJ$  per SRAM access (in  $0.13 \mu m$  CMOS process)

SRAM size	Word format	8-bit	16-bit	32-bit	64-bit
960 bytes (1 DDS per SRAM)		22	28	38	64
1920 bytes (2 DDS per SRAM)		26	32	43	67
3840 bytes (4 DDS per SRAM)		36	39	51	73
7680 bytes (8 DDS per SRAM)		51	52	65	89

addressable and offers word-wide data accesses (e.g., a 13-bits wide Wa-SRAM of size 1664 bits results in a 7-bit wide address bus and a 13-bit wide data bus). Based on Table 1, for a fixed SRAM size of 7680 bytes, replacing an 8-bit word Wa-SRAM cut (i.e. 7680 words of 8 bits) with a 32-bit word Wa-SRAM cut (i.e. 960 words of 32 bits) results only in an energy increase of only  $1.7x$  per access for STMicroelectronics' SRAM design library in  $0.13 \mu m$  CMOS technology; hence, the "average energy dissipation per 8-bit sample access" (EDPSA) can be significantly reduced. The concept of embedded reordering is to fetch the data from the large deinterleaving SRAM cluster by cluster instead of sample by sample and to perform the sub-array deinterleaving task in an energy-efficient smaller memory component (called reordering buffer thereafter). A sample-accessible (Sa-)SRAM is then replaced by a cluster-accessible (Ca-)SRAM so as to reduce the EDPSA.

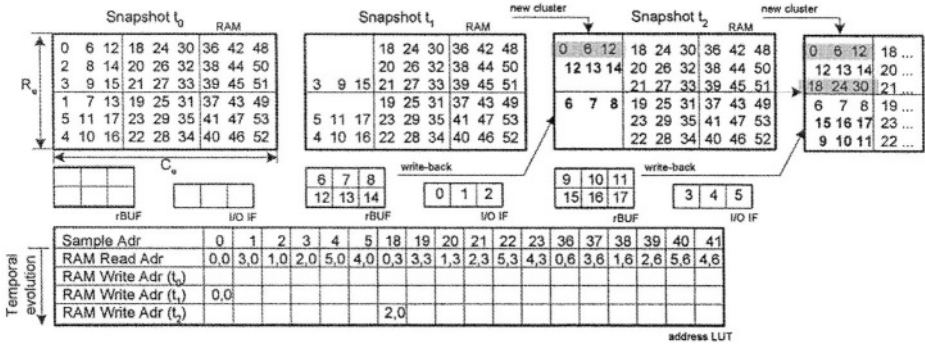
**A memory-reuse deinterleaving approach:** thanks to the afore-mentioned property ( $p2$ ) efficient memory-reuse is made possible. Empty memory locations resulting from partial deinterleaving can be updated continuously with new clusters of the next DDS. The underlying address LUT needed to support fully-associative access scheduling requires only one entry per cluster to retrieve (or compute from equation (1)) all memory mappings. Thus, the LUT size is reduced by approximately  $W$ .

### 3.2 Embedded Reordering – An Example

The two key ideas of *embedded reordering* are (1) to deinterleave partial sub-arrays rather than the entire deinterleaver matrix at once and (2) to update empty memory locations in a continuous way with clusters of the next DDS. Thus, the *large* deinterleaving memory is "downgraded" to a simple *cluster-accessible* storage unit. The burden of deinterleaving the sequence is pushed onto a *much smaller* reordering buffer (rBUF) with much lower power consumption per access. An address LUT is necessary to guarantee consistency of a fully-associative scheduling.

The embedded reordering method is best explained by going through an example step by step. Let us assume a block interleaver with  $R = 9$ ,  $C = 6$  and inter-column permutations defined as follows:  $(C_0, C_1, C_2, C_3, C_4, C_5) \rightarrow (C_0, C_2, C_3, C_1, C_5, C_4)$ . Moreover, the cluster size  $W$  of the deinterleaving RAM, which is arranged as a matrix of size  $R_c \times C_c$  with  $R_c = C$  and  $C_c = R$ , is chosen to be a multiple of  $R_c$  and  $C_c$ ; that is  $W = 3$  in our example. Assuming the original sample address sequence contained in one DDS being  $0, 1, \dots, 53$  the chronological order of the samples after pseudorandom block interleaving becomes:

$0, 6, 12, 18, 24, 30, 36, 42, 48, 2, 8, 14, 20, 26, 32, 38, 44, 50, 3, 9, 15, 21, 27, 33, 39, 45, 51,$   
 $1, 7, 13, 19, 25, 31, 37, 43, 49, 5, 11, 17, 23, 29, 35, 41, 47, 53, 4, 10, 16, 22, 28, 34, 40, 46, 52$



**Fig. 4.** Snapshot of the *access with reordering (A&R)* mode; at snapshot  $t_0$ , the RAM (initially empty) is updated with new clusters; at snapshot  $t_1$ , the content of the reordering buffer (rBUF), output FIFO (I/O IF), address LUT and RAM after the first A&R phase is illustrated; at snapshot  $t_2$ , the memory content after the second A&R phase is shown

Note that the interleaved sequence satisfies the properties discussed in Section 3.1. Initially, data is written cluster-wise to the RAM row by row as illustrated in Fig. 4 (snapshot  $t_0$ ). As discussed in Section 2, all samples of one DDS (e.g., 3GPP sub-frame) are to be received first before the deinterleaving can be initiated.

To deinterleave the current DDS a finite state machine (FSM) keeps track of the reordering states. Initially, the sample with sample address 0 needs to be accessed. The corresponding address location is stored in the address LUT (here,  $adr = 0,0$ ). The first sample ( $= 0$ ) of the cluster is extracted and transferred to the output FIFO. The remaining  $W-1$  samples (here, 6 and 12) are stored in the 1<sup>st</sup> column of rBUF. Then, the cluster containing the next sample address ( $= 1$ ) is read and is processed in the same way as before. This time, the remaining  $W-1$  samples (here, 7 and 13) are stored in the 2<sup>nd</sup> column of rBUF. The same processing is repeated  $W$  times until the reordering buffer is full. This access mode is referred to as *access with reordering (A&R)*. At the end of each A&R phase (snapshots  $t_1$  and  $t_2$ ), the RAM is updated with a new cluster of the next DDS (shaded cluster in Fig. 4). To simplify the FSM and LUT handling, the new clusters are systematically written to the first memory location accessed within one A&R phase. Moreover, in order to reuse the rBUF in the next A&R phase, its content is written back to the RAM at the  $W-1$  last accessed locations. This A&R access mode is repeated  $C/W (= 2)$  times. Snapshots of the RAM, rBUF and address LUT content is shown in Fig. 4. As described by property ( $p1$ ) sub-array deinterleaving yields row-wise reordered sequences.

Next, there are  $(W-1) \cdot C/W (= 4)$  *accesses without reordering (Aonly)*. The FSM needs to retrieve the memory location which points to sample address 6. To do so, note that the cluster composed of sample address 6 is *always* stored in the memory location previously occupied by the cluster composed of sample address 1 (here,  $adr = 3,0$ ). This property is achieved by the systematic “write-back” of the rBUF content to the  $W-1$  last accessed RAM addresses at the end of each A&R phase. This time, all samples of the cluster have already been reordered during the A&R phase and can thus be directly transferred to the output FIFO. Similarly, the cluster composed of sample address 9 is read from the memory and so on. Note that the RAM can be updated with a new cluster after each *Aonly* phase. To extract  $W$  ordered samples during the *Aonly* phase only *one* RAM access is required (space locality). Finally, to deinter-



the *Aonly* phase only *one* RAM access is required (space locality). Finally, to deinterleave the entire DDS,  $C/W$  accesses *with* sub-array reordering are followed by  $(W-1) \cdot C/W$  accesses *without* reordering in a repetitive fashion.

### 3.3 Template Architecture

High-throughput applications often results in several (independent) deinterleaving data sets to be supported in parallel (e.g., 3GPP). The underlying cluster-based deinterleaving architecture is illustrated in Fig. 5. It consists of  $M$  kernel units, where each kernel is composed of a *two-level* custom memory hierarchy (i.e., a (large) deinterleaving Ca-SRAM and a (small) Sa-SRAM reordering buffer), an address LUT, finite state machine (FSM) and cluster-based processing (CBP) logic. The trade-off between mapping  $N$  DDSs onto the same Ca-SRAM, which also impacts the number of kernel units ( $M$ ) required to support a certain system throughput, is the duplication of memory control, interconnection circuitry, architecture throughput and power consumption.

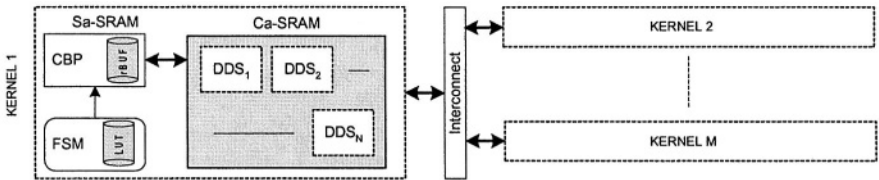


Fig. 5. Template architecture implementing embedded reordering

Note that this template architecture hides micro-architecture details such as single- vs. multi-port or single- vs. multi-bank SRAM, hardware (HW) versus software (SW) or details about the interfaces; those aspects will be addressed in a subsequent publication. Nonetheless, synthesis results of the CBP logic are presented in Section 4 to put the logic overhead into perspective. Moreover, this template is a means to perform analytical and quantitative design space exploration as discussed in the next section. To reiterate, the important design parameters impacting computational processing, die area and energy dissipated per sample access are 1) cluster size  $W$ , 2) “DDS to RAM” mapping ( $N$ ) and 3) number of kernels ( $M$ ).

## 4 Design Space Exploration

In this section, we will compare the state-of-the-art and embedded reordering solutions. By formalizing the complexity (i.e., the memory size and access rate as shown in Table 2), we can demonstrate the impact of high-level design parameters (i.e.,  $M$ ,  $N$ ,  $W$ ) on low-level design metrics such as die area ( $mm^2$ ) and energy dissipation ( $\mu J$ ). The analysis results are based on STMicroelectronics’ single-port SRAM design library in 0.13  $\mu m$  CMOS process technology.

To assess memory size and access rate statistics the storage requirement of the embedded reordering includes 3 units, i.e. a large deinterleaving Ca-SRAM, a small Sa-SRAM reordering buffer (rBUF) and the address LUT. Note that the greatest part of

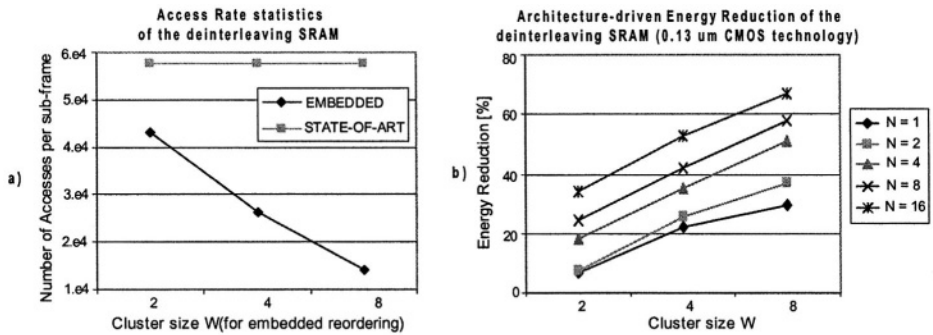
**Table 2.** Memory size and Access rate statistics per kernel unit;  $N$  accounts for the number of DDSs (each of size  $C \cdot R$ ) mapped to the same SRAM;  $W_s$  specifies the sample width (for typical wireless systems  $W_s \leq 8$  bits) and  $W_A$  the number of bits required to represent one LUT entry;  $\tilde{C} = \lceil C/C_w \rceil$  and  $\tilde{R} = \lceil R/C_w \rceil$ , where  $\lceil \bullet \rceil$  denotes the ceiling function

	SRAM (soa)	SRAM (emb)	rBUF	LUT
<b>Size</b>	$C \cdot R \cdot N \cdot W_s$	$C \cdot R \cdot N \cdot W_s$	$W(W-1) \cdot W_s$	$2 \cdot \tilde{C} \cdot R \cdot W_A$
<b>Write</b>	$C \cdot R \cdot N$	$N \lceil C \cdot R / W \rceil + N \cdot \tilde{R} \cdot \tilde{C} \cdot (W-1)$	$N \cdot \tilde{R} \cdot \tilde{C} \cdot W(W-1)$	$N \cdot \tilde{C} \cdot R$
<b>Read</b>	$C \cdot R \cdot N$	$N \cdot \tilde{R} \cdot \tilde{C} \cdot (2 \cdot W - 1)$	$N \cdot \tilde{R} \cdot \tilde{C} \cdot W(W-1)$	$N \cdot \tilde{C} \cdot R$

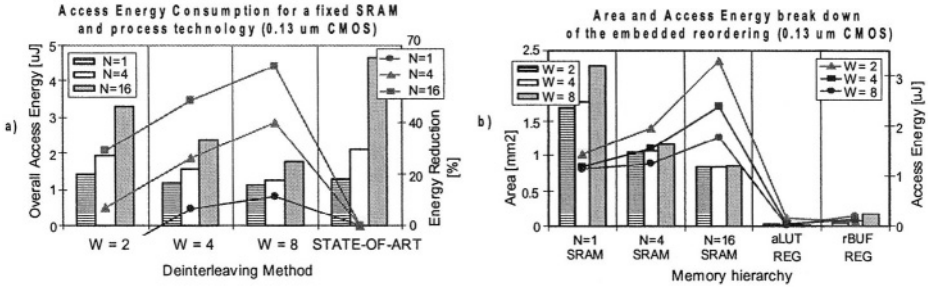
the accesses is moved to the much smaller reordering buffer. Similarly, for state-of-the-art solutions, the storage units to be considered are the deinterleaving Sa-SRAM and an adaptation buffer. The buffer size depends on system parameters (e.g., input rate, DDS size, etc.), the micro-architecture (e.g., single- vs. multi-port SRAM) and technological limits (e.g., read access delay). Buffer size estimates are given below.

The high-speed data packet access (HSDPA) defined in 3GPP [4] serves as target application. To support data rates of up to  $10$  Mbps the standard implies a maximum of  $30$  DDSs of size  $(R=30) \times (C=32)$  to be supported. Moreover, since all data channels are time aligned and of the same frame format one address LUT suffices to monitor all DDSs. The architectural design parameters are set to  $W = \{2, 4, 8\}$  (i.e.  $16$ -,  $32$ - and  $64$ -bit wide Ca-SRAMs), DDS to SRAM mapping  $N = \{1, 2, 4, 8, 16\}$  which in turn results in  $M = \{30, 15, 8, 4, 2\}$  kernel units to implement the  $10$  Mbps HSDPA class.

The access reductions to the deinterleaving SRAM achieved with embedded reordering are depicted in Fig. 6a. The resulting decrease in (access) energy dissipation is plotted in Fig. 6b for different architectural design parameters (i.e.,  $N$  and  $W$ ). To ensure consistency, only results achieved with single-port SRAM cuts of STMicroelectronics' high-density design library are illustrated. Taking now into account the additional energy dissipated in the LUT and rBUF, for appropriate selection of the cluster size  $W (=8)$  and "DDS to RAM" mapping  $N (=16)$ , our novel embedded deinterleaving achieves an energy dissipation reduction of up to  $60\%$  (Fig. 7a).



**Fig. 6.** Access rate statistics to the deinterleaving SRAM for the state-of-the-art and embedded reordering solutions (a); architecture-driven energy reduction of the deinterleaving SRAM achieved with the embedded reordering in  $0.13 \mu\text{m}$  CMOS process technology (b)



**Fig. 7.** Overall access energy in  $\mu\text{J}$  (left ordinate) and energy reduction in percentage (right ordinate) shown with bars and lines, respectively (a); Area ( $\text{mm}^2$ ) and energy ( $\mu\text{J}$ ) consumption breakdown of the memory hierarchy represented by bars and lines, respectively (b)

The LUT and reordering buffer area is at least one order of magnitude smaller than the SRAM one (Fig. 7b). Finally, note that the logic overhead of the embedded reordering (CBP) is in the order of  $0.007\text{-}0.08 \text{ mm}^2$  depending on  $W$ ,  $N$  and  $M$ ; hence, much smaller than the SRAM cuts. The CPB logic energy dissipation is negligible with respect to the energy dissipated in the memories.

Thanks to its memory-reuse capability the embedded reordering does not require any adaptation buffer. As discussed above, the actual buffer size for state-of-the-art techniques depends on several system and design parameters. For instance, with a 10 Mbps HSDPA configuration, assuming a 100 MHz clock and a soft-combining task in the following “Hybrid ARQ unit” which takes 5 cycles, the buffer would have to store a minimum of  $1k$  samples for  $N = 4$  and  $5k$  samples for  $N = 16$ . This leads to Sa-SRAM cuts of  $0.05 \text{ mm}^2$  and  $0.16 \text{ mm}^2$ , respectively and further increases the energy dissipation due to the required transfer of the samples from the buffer to the deinterleaving memory.

## 5 Conclusion

In this paper we presented a 3GPP-HSDPA compliant block deinterleaving architecture which addresses the following important design issues: *continuous memory-reuse* and *low energy consumption*. High-level design parameters can be tuned to trade-off low-level design metrics such as die area and energy dissipation. The results show, that for an appropriate selection of those parameters this novel *cluster-based* design implementing a two-level custom memory hierarchy can save up to 60% in energy with a negligible logic overhead. Moreover, the memory hierarchy promotes a system-independent design; that is, the fully-associative memory access pattern eliminates the need for large adaptation buffers and hence saves area. This work completes the memory optimizations and architecture-driven voltage-scaling approach described in [6] to propose a scalable, energy- and area-efficient outer demodulator solution for high-throughput wireless systems.

## References

1. International Technology Roadmap for Semiconductors (ITRS), 2001 edition, available from <http://public.itrs.net>
2. F. Cathoor, E. de Greef, S. Suytack.: Custom Memory Management Methodology: Exploration of Memory Organization for Embedded Multimedia System Design. In: Kluwer Academic Publishers, Norwell, MA (1998)
3. N. Kim *et al.*: Leakage Current – Moore’s Law Meets Static Power. In: IEEE Computer Society, Vol. 36, No. 12, (2003) 68-75
4. Third Generation Partnership Project: TS 25.212 Release 5, available from [www.3gpp.org](http://www.3gpp.org)
5. E. Yeo, P. Pakzad, B. Nikolic.: VLSI Architectures for Iterative Decoders in Magnetic Recording Channels. In: IEEE Transactions on Magnetics, Vol. 37, No. 2, (2001) 748-755
6. F. Gilbert, N. When.: Architecture-Driven Voltage Scaling for High-Throughput Turbo-Decoders. In: International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS ’03), Torino, Italy (2003) 379-388
7. J. Heiskala, J. Terry.: OFDM Wireless LANs – A theoretical and Practical Guide. SAMS publisher, 1<sup>st</sup> edition, 2001
8. J. L. Ramsey.: Realization of optimum interleavers. In: IEEE Transactions on Information Theory, Vol. IT-16, (1970) 338-345
9. Y. Uchida, M. Ise, T. Onoye, I. Shirakawa, I. Arungsirsanhai.: VLSI Architecture of Digital Matched Filter and Prime Interleaver for W-CDMA. In: IEEE International Symposium on Circuits and Systems (ISCAS ’03), Vol. 3, Arizona, USA (2002) 269 - 272
10. Symbol Interleaver/Deinterleaver MegeCore Function User Guide, available from [www.altera.com](http://www.altera.com)
11. Xilinx Interleaver/Deinterleaver Features & Data Sheets, available from [www.xilinx.com](http://www.xilinx.com)
12. T. Richter, G. Fettweis.: Parallel Interleaving on parallel DSP architectures. In: IEEE Workshop on signal processing systems (SIPS ’02), San Diego, USA (2002)
13. S. Wuytack, J.P. Diguët, F. Cathoor, H. De Man.: Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Mappings. In: Special issue of IEEE Transactions on VLSI Systems on low power electronics and design, Vol. 6, No. 4, (1998) 529-537
14. L. Benini, A. Macii, E. Macii, M. Poncino.: Synthesis of application-specific memories for power optimization in embedded systems. In: Design Automation Conference (2000) 300-303

# Register Isolation for Synthesizable Register Files

Matthias Müller, Andreas Wortmann, Dominik Mader, and Sven Simon

Hochschule Bremen, Flughafenallee 10, D-28199 Bremen, Germany  
Matthias.Mueller@hs-bremen.de

**Abstract.** In this paper, a low power synthesizable register file architecture is presented. The register file is modeled on register-transfer level (RTL). It is suitable for technology independent IP-core and processor models. The data inputs of all registers of the register file are connected to a data bus. Similar to operand isolation for combinational logic blocks we use blocking logic in order to isolate registers from transitions on the data bus. The registers are divided into groups and blocking logic is inserted for each group. Even if no spurious transitions occur on the data bus, the power dissipation of the register file can be reduced applying this register file architecture.

## 1 Introduction

In the last decade, the design methodology for complex CMOS circuits has moved from the traditional full custom design style to an automated semi-custom design flow. The main reason for these structural changes is the design productivity which must cope with the exponential increase in design complexity according to Moore's Law. Even with the introduction of an automated semi-custom design flow, the ever-increasing design complexity has led to the necessity of further productivity improvements. In order to achieve this, design reuse with synthesizable processors and IP cores has become the focus of attention in the last years [1]. These cores are implemented as soft macros applying a hardware description language (HDL) like Verilog or VHDL.

At the same time the design methodology changed, power dissipation has become a severe constraint on the physical level of VLSI implementations in the last years [2,3,4]. There are basically three reasons for this evolution. Firstly, the limited energy capacitance of batteries used for mobile systems and devices like laptops, mobile phones, digital cameras or personal digital assistance (PDA) constrains the operating time without battery recharging. Secondly, the capacitance of batteries could not keep up with the increasing energy demand of portable devices in the last years [5,6]. Thirdly, with increasing clock frequencies and chip complexity the costs for heat removal and packaging due to a significant power dissipation has become a major concern.

In this paper, a low power register file architecture being part of synthesizable processors or IP cores is presented. Register files are widely used, e.g. for

FIFOs, data transfer buffers, elastic buffers, memories and for the storage of state values of signal processing applications like digital filters. Another classical application of register files are processors [7]. The register file architecture presented here is synthesized as part of the RTL core in a semi-custom design flow. Alternatively, a RAM could be used instead. RAMs are designed on the physical level and are very technology dependent. Considering the design flow (RTL description, logic and timing simulation, static timing analysis, testability, formal verification, place & route, etc.), RAMs have to be treated like hard macros. Therefore, synthesizable register files which are part of the design on RTL are often preferred to RAM if the area penalty is not too high. Other reasons to prefer synthesizable register files to RAMs is the maximum achievable clock rate.

The dynamic power dissipation of CMOS circuits is caused by the charging and discharging of capacitances in the circuit. It mainly depends on the value of the supply voltage, the clock frequency of the circuit, the value of the capacitance recharged and the switching activity of the nodes in the circuit. Low power design methodologies consider all these parameters. Modifications applicable at the architectural level of the design are well suited for the reduction of the power dissipation of technology independent designs. For instance, the switching activity can be reduced by architectural modifications on RTL. A widely used technique for power reduction based on the reduction of the switching activity of combinational logic is operand isolation, see [8,9,10]. Additional blocking logic is connected to the primary inputs of combinational blocks. If the data output of these blocks are not needed in the next clock cycle, the data inputs are set to a constant value. Due to this, the power dissipation caused by glitch propagation in the blocks is reduced. For low power register files, the power dissipation due to changes of the clock signal is reduced by the application of clock gating. Clock gating cells are integrated into the clock signal of each register. If a register is not addressed, the value of its clock signal is kept constant, hence the switching activity is reduced [8,9]. In this paper, however, the principle of the reduction of the switching activity is applied to the data inputs of registers. In register files, the data inputs of all registers are connected to a data bus. Transitions on the data bus due to a write access to one register generate power dissipation in the data path of all registers connected to the data bus. In the same way as at operand isolation, blocking logic is inserted at the data inputs of groups of registers of the register file. Using this blocking logic, registers that are not addressed can be isolated from transitions on the data bus. Due to the analogy to operand isolation, we use the phrase 'register isolation' in the rest of the paper.

The paper is organized as follows. In Section 2, a model for the power dissipation of library flip-flops and conventional synthesizable register files is derived. The architectural modifications of the register file with register isolation and its power dissipation are discussed in Section 3. In Section 4, experimental results of the evaluation of the proposed architecture are presented. The main aspects of this paper are summarized in the conclusion in Section 5.

## 2 Power Dissipation in Synthesizable Register Files

In this section, the power dissipation of synthesizable register files is discussed. The register file architectures considered are very similar in terms of layout properties. Especially the capacitive output load of the registers and the power dissipation of the multiplexers at the register file output can be assumed to be identical for the different architectures. A power model is derived on the lowest level of abstraction required to compute the difference in power dissipation between the different architectures. Due to the comparison of different implementation alternatives, there is no need to know the power dissipation in absolute values. The abstraction level of a standard cell is sufficient for our purpose. We do not require a more detailed model on transistor level.

### Power Dissipation of Library Flip-Flops

The storage elements of synthesizable register files are flip-flops available in the standard cell library. These flip-flops are mostly edge-triggered master-slave D flip-flops. Unless otherwise expressly mentioned, we consider D flip-flops exclusively in the following. The power dissipation of a flip-flop can be split up into two parts. Firstly, the power dissipation of the data path  $P_{FD}$ . Secondly, the power dissipation of the clock path  $P_{FC}$ . The switching activity  $\alpha_{FD}$  is the number of changes of the data signal per clock period. The switching activity of the clock signal is defined to be  $\alpha_{FC} = 1$ . Let  $V_{DD}$  be the supply voltage and  $f$  be the clock frequency of the flip-flop. The total dissipated power of a flip-flop  $P_F$  is given by the following equation:

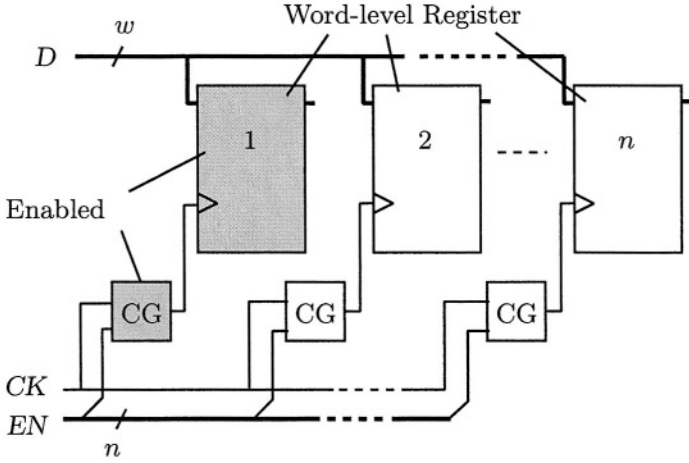
$$P_F = P_{FD} + P_{FC} \quad (1)$$

$$= \alpha_{FD} C_{FD} V_{DD}^2 f + \alpha_{FC} C_{FC} V_{DD}^2 f. \quad (2)$$

The parameter  $C_x$  with  $x \in \{FD, FC\}$  is the equivalent capacitance which models the power dissipation on the abstraction level of standard cells. This equivalent capacitance is defined as the measured power dissipation of the cell divided by switching activity, square of the supply voltage and frequency:  $C_x = P_x / (\alpha_x V_{DD}^2 f)$ . Thus, the power dissipation of all transistors of a standard cell and the short circuit currents across n- and p-transistors are included in this equivalent capacitance. Wire capacitances of the inputs and outputs are also included.

### Power Dissipation of Register Files

In register file structures, clock gating allows power efficient addressing, see [9]. In most applications the value of only one register changes at each clock cycle while the data values of the remaining registers are not altered. Modern EDA-tools used in semi-custom design flows are capable of integrating special clock gating cells into the design [8]. Fig. 1 depicts a register file of  $n$  registers with  $w$  bits connected to the data bus  $D$ . The multiplexer at the output of the register file is



**Fig. 1.** Power efficient addressing of the registers in the register file using clock gating.

not depicted. The address logic is realized by one clock gating cell for each word-level register. The addressed register and the corresponding clock gating cell are indicated in Fig. 1 by the shaded elements. Using this clock gating technique, the clock signal propagates only to the flip-flops of the addressed word-level register whereas the clock signal input of all other flip-flops is constant. Therefore, the power dissipation due to transitions of the clock signal of the register file is reduced by clock gating.

The total power dissipation  $P_{\text{RFCG}}$  of the register file with clock gating consists of the power dissipation of the clock gating cells  $P_{\text{CG}}$ , the flip-flops  $P_{\text{F}}$  and the multiplexer  $P_{\text{M}}$  which is discussed in Appendix A. The power dissipation of the register file without clock gating is denoted as  $P_{\text{RF}}$  and consists of the power dissipation of the flip-flops  $P_{\text{F}}$  and the multiplexer  $P_{\text{M}}$ .

$$P_{\text{RF}} = \sum_{i=1}^{nw} P_{\text{F}}(i) + P_{\text{M}} \quad (3)$$

$$P_{\text{RFCG}} = \sum_{i=1}^n P_{\text{CG}}(i) + \sum_{i=1}^{nw} P_{\text{F}}(i) + P_{\text{M}}. \quad (4)$$

As only one word-level register is addressed for the register file with clock gating and all registers are assumed to be identical, we obtain:

$$P_{\text{RF}} = nwP_{\text{FC}} + nwP_{\text{FD}} + P_{\text{M}} \quad (5)$$

$$P_{\text{RFCG}} = nP_{\text{CG}} + wP_{\text{FC}} + nwP_{\text{FD}} + P_{\text{M}}. \quad (6)$$

Obviously, the product  $nwP_{\text{FC}}$  of  $P_{\text{RF}}$  is larger than  $nP_{\text{CG}} + wP_{\text{FC}}$  of  $P_{\text{RFCG}}$  especially for large  $n$  and  $w$  such that  $P_{\text{RFCG}} \ll P_{\text{RF}}$ . Therefore, clock gating should be applied for low power register files in any case. In commercial semi-custom design flows, automated clock gating has been established as a standard



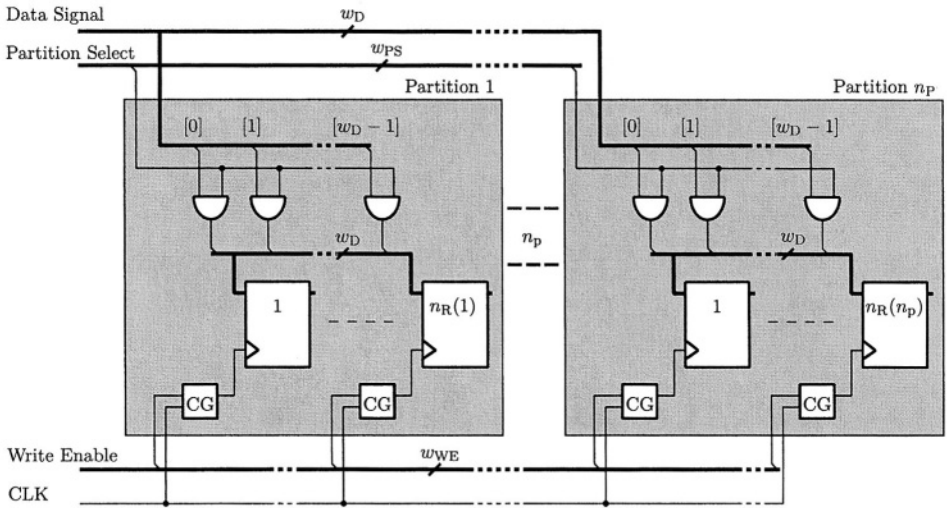


Fig. 2. Register file architecture with blocking logic for register isolation.

methodology [8,11]. Although the methodology presented in the next section can also be successfully applied to register files without clock gating, we consider register file architectures with clock gating in the following.

### 3 Register Isolation for Synthesizable Register Files

A widely used technique for power reduction of combinational logic is operand isolation, see [8,9,10]. Additional blocking logic to avoid glitch propagation is connected to the primary input of combinational blocks. A control signal disables the inputs of these blocks from data propagation if the output values of the combinational blocks are not used in the following stages during the next clock cycle. This methodology is especially effective if the blocking logic disables the combinational block for many clock cycles. Obviously, the principle of operand isolation is applicable to the data input of register files because transitions due to a write access to one register consisting of  $w$  flip-flops generate power dissipation in the data path of all  $wn$  flip-flops of the register file. In the same way, blocking logic is inserted at the input of groups of registers of the register file, in the following called as partitions.

In Fig. 2, a register file with AND gates as blocking logic is shown. OR gates or latches might also be chosen although latches may be unsuitable from a semi-custom design flow point of view. It is important to note that the power consumption of the register file can be reduced even if no glitches occur on the data input because each write access to any register of the register file will cause transitions at the data input of all flip-flops.

The number of registers which belong to partition  $i$  is referred to  $n_R(i)$ . For each partition,  $w$  AND gates are used as blocking logic for the data input

signal. The value  $w$  denotes the wordlength of the data signal. As shown in Fig. 2, one input of all AND gates of one partition is connected to one bit of the partition select signal. If the partition select signal is '1', the value on the data bus is connected to the data input of all flip-flops of the corresponding partition. Otherwise, the data input of the flip-flops are constant zero.

Let  $P_{\text{RFCGR}}$  be the power dissipation of the register file with clock gating and register isolation and let  $P_{\text{AR}}$  be the power dissipation of the AND gates. Due to the AND gates as transition barriers, the power dissipation of the flip-flop data paths of the register file with register isolation  $P_{\text{FDR}}$  differ from the power dissipation of the register file without register isolation  $P_{\text{FD}}$ . The power consumption of the clock gating cells  $P_{\text{CG}}$ , of the multiplexers  $P_{\text{M}}$  and the power consumption of the internal clock generation  $P_{\text{FC}}$  of the flip-flops is not affected.

The power dissipation of the register file with register isolation  $P_{\text{RFCGR}}$  and the power savings due to the application using partitioning  $P_{\Delta} = P_{\text{RFCG}} - P_{\text{RFCGR}}$  with  $P_{\text{RFCG}}$  computed in Eq. (4) is given as follows:

$$P_{\text{RFCGR}} = P_{\text{AR}} + P_{\text{FDR}} + P_{\text{FC}} + P_{\text{CG}} + P_{\text{M}} \quad (7)$$

$$P_{\Delta} = P_{\text{FD}} - P_{\text{FDR}} - P_{\text{AR}}. \quad (8)$$

The equivalent capacitance of all AND gates for one partition is  $wC_{\text{A}}$ . Usually, the partition select signal has a significant lower switch activity than the data signal  $\alpha_{\text{AD}}$ . The power consumption of the AND gates of all partitions  $n_{\text{p}}$  is approximately:

$$P_{\text{AR}} \approx n_{\text{p}}\alpha_{\text{AD}}wC_{\text{A}}V_{\text{DD}}^2f. \quad (9)$$

The power dissipation  $P_{\text{FDR}}$  of the flip-flops of all partitions  $n_{\text{p}}$  due to transitions on the data signal is:

$$P_{\text{FDR}} = \left( \sum_{i=1}^{n_{\text{p}}} n_{\text{R}}(i)\alpha_{\text{FD}}(i)C_{\text{FD}} \right) wV_{\text{DD}}^2f. \quad (10)$$

From the equation above, the sum  $P_{\Delta}$  can be computed to:

$$P_{\Delta} = \left( n\alpha_{\text{FD}}C_{\text{FD}} - \sum_{i=1}^{n_{\text{p}}} n_{\text{R}}(i)\alpha_{\text{FD}}(i)C_{\text{FD}} - n_{\text{p}}\alpha_{\text{AD}}C_{\text{A}} \right) wV_{\text{DD}}^2f. \quad (11)$$

If data are written to the same partition  $i$  during several clock cycles, the power savings  $P_{\Delta}$  can be rewritten. Both switching activities  $\alpha_{\text{AD}}$ ,  $\alpha_{\text{FD}}(i)$  are equal in this case, whereas the switching activities of the other partitions  $\alpha_{\text{FD}}(k)$ ,  $k \neq j$  are equal to 0:

$$P_{\Delta} = (n\alpha_{\text{FD}}C_{\text{FD}} - \alpha_{\text{FD}}(i)n_{\text{R}}(i)C_{\text{FD}} - n_{\text{p}}\alpha_{\text{FD}}(i)C_{\text{A}})wV_{\text{DD}}^2f. \quad (12)$$

From the equation above, the power savings can be affected profitably if the number of registers  $n_{\text{R}}(i)$  of the activated partition and the total number of

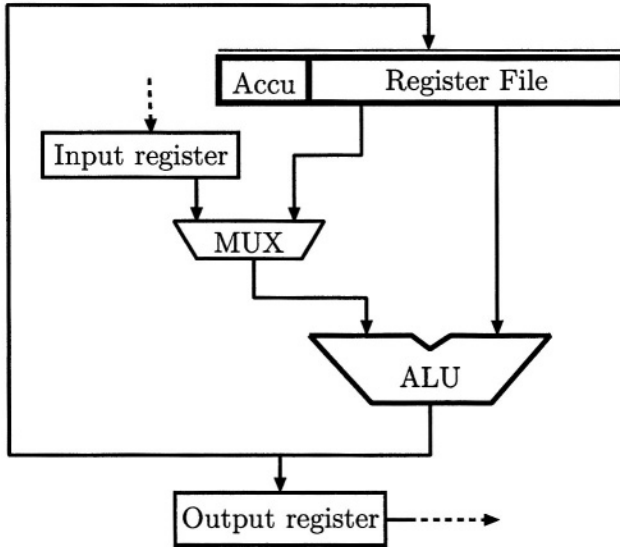


Fig. 3. Datapath of the processor core.

partitions  $n_P$  is chosen small. Therefore, partitions with high subsequent write access rates should have a small register size  $n_R(i)$ , e.g.  $n_R(i) = 1$ . This can be applied in the typical case when intermediate results are written relatively often to an accumulator register of a processor core. All other registers which are used rarely belong to other partitions.

If write access is changed from partition  $i$  to partition  $j$ , the power savings  $P_\Delta$  of Eq. (12) is reduced by the factor  $\alpha_{FD}(j) n_R(j) C_{FD} w V_{DD}^2 f$  contributed by the activation of partition  $j$ . Thus, the change of partitions should be minimized if possible. Obviously, the power dissipation also depends on the program and the register allocation. From this point of view, partitioning is well suited for multichannel applications. For this case, registers assigned to one channel can be grouped into one partition. Since there is no dependency of the data of different channels, the channels can be processed subsequently without interference and hence, the partitions are activated subsequently. Further, if there are only a few registers, such as temporary storages, which are accessed often and subsequently, these can be grouped into one partition. In this case, power dissipation can be reduced applying register isolation.

## 4 Experimental Results

In order to evaluate the register file architectures with and without partitioning, we have implemented a processor core in VHDL. The datapath of the processor core consists of an ALU, the register file and I/O registers, see Fig. 3. We synthesized the VHDL model to gate level using a standard cell library of a  $0.18\mu$

**Table 1.** Comparison concerning the power dissipation in  $\mu W$  of a register file with and without register isolation.

			Without register isolation	#Partitions $n_P$		
$w_D$	$n_R$			2	4	8
<b>8</b>	<b>16</b>	Flip-flops + CG	654	322	297	255
		Blocking Logic	-	64	144	196
		<b>Register file (<math>\Sigma</math>)</b>	<b>654</b>	<b>386</b>	<b>441</b>	<b>451</b>
	<b>32</b>	Flip-flops + CG	988	543	409	383
		Blocking Logic	-	115	259	353
		<b>Register file (<math>\Sigma</math>)</b>	<b>988</b>	<b>658</b>	<b>668</b>	<b>736</b>
<b>16</b>	<b>16</b>	Flip-flops + CG	1572	475	391	369
		Blocking Logic	-	124	208	442
		<b>Register file (<math>\Sigma</math>)</b>	<b>1572</b>	<b>599</b>	<b>599</b>	<b>811</b>
	<b>32</b>	Flip-flops + CG	2886	734	525	484
		Blocking Logic	-	151	338	547
		<b>Register file (<math>\Sigma</math>)</b>	<b>2886</b>	<b>885</b>	<b>863</b>	<b>1031</b>

**Table 2.** Relative power savings  $P_\Delta$  due to register isolation.

			#Partitions $n_P$		
$w_D$	$n_R$		2	4	8
<b>8</b>	<b>16</b>	Flip-flops + CG	51%	55%	61%
		<b>Register file</b>	<b>41%</b>	<b>33%</b>	<b>31%</b>
		<b>32</b>	Flip-flops + CG	45%	59%
	<b>Register file</b>		<b>33%</b>	<b>32%</b>	<b>26%</b>
	<b>16</b>		Flip-flops + CG	70%	75%
		<b>Register file</b>	<b>62%</b>	<b>62%</b>	<b>48%</b>
<b>32</b>		Flip-flops + CG	75%	82%	83%
	<b>Register file</b>	<b>69%</b>	<b>70%</b>	<b>64%</b>	

process technology. The power consumption of the circuit was determined by circuit level simulation of the transistor level netlist with SPICE accuracy.

Different combinations of wordlength, number of registers and number of partitions have been chosen. A FIR filter was implemented in software on the processor core. Due to the algorithm, a write access to the accumulator occurred subsequently for 50% of the clock cycles. According to the conclusions of the previous section, the accumulator was treated as one partition for all simulations. The remaining registers of the register file are uniformly distributed over all partitions. In addition, we minimized the number of switches between partitions by appropriate register allocation and scheduling as mentioned in the previous section. As a reference, the power dissipation of the registers without register isolation is given.

The simulation results in Table 1 show that the power dissipation of the register file can be significantly reduced by register isolation. The optimal number of partitions varies for different cases but are typically small numbers e.g. 2 or

4. This is due to a trade-off between the power savings of the registers and the additional power dissipation for the blocking logic. As shown in the previous section, the power dissipation for the blocking logic increases with the number of partitions. The values given for the blocking logic in Table 1 include the power dissipation of the partition enable signals. The increase of the power dissipation of the blocking logic is compensated by the power savings due to the reduction of the transition activity at the register file input.

Table 2 shows the relative reduction of the power consumption related to the register file without register isolation. In the case of a wordlength  $w = 8$  bit and the number of  $n = 16$  registers, two partitions are the best choice and 41% of the power consumption is saved. In the case of a wordlength of 16 bit and a number of 32 registers, 70% of the power consumption is saved if the register file is divided into 4 partitions.

## 5 Conclusion

In this paper, a method is presented to reduce the power dissipation of synthesizable register files. The basic idea is to reduce the switching activity of the data inputs of the registers. In order to achieve this, the registers of the register file are divided into groups. By the insertion of additional blocking logic for each group of registers, it is possible to isolate the data inputs of the registers from transitions on the data bus. This is similar to operand isolation, where additional blocking logic is inserted in order to avoid glitch propagation into combinational blocks. It is shown that the method presented is especially effective if many subsequent write accesses are performed to registers grouped in a small partition. For instance, an accumulator register of a processor core that is accessed relatively often in subsequent clock cycles can be assigned to its own partition. Another typical application is the implementation of multichannel algorithms. The registers allocated to one channel can be grouped in an appropriate partition. Experimental results show that the power dissipation of the register file can be reduced by more than 30% for typical cases.

## References

1. ([www.design-reuse.com](http://www.design-reuse.com))
2. Chandrakasan, A.P., Brodersen, R.W.: Minimizing power consumption in digital CMOS circuits. *Proceedings of the IEEE* **83** (1995) 498–523
3. Yeap, G.K.: *Practical Low Power VLSI Design*. Kluwer Academic Publishers (1998.)
4. Chandrakasan, A.P., Sheng, S., Brodersen, R.W.: Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits* **27** (1992) 473–484
5. Borkar, S.: Low power design challenges for the decade. In: *Proceedings of the ASP DAC 2001*. Asia and South Pacific Design Automation Conference 2001. (2001) 293–296
6. Pedram, M., Wu, Q.: Battery-powered digital CMOS design. *IEEE Transactions on Very Large Scale Integration VLSI Systems* **10** (2002) 601–607

7. Patterson, D.A., Hennessy, J.L.: Computer Organization & Design. Morgan Kaufmann Publishers (1998)
8. Renu Mehra, Synopsys, I.: Power aware tools and methodologies for the asic industry. Low Power Tutorial, Design Automation Conference 2003 (2003)
9. Raghunatan, A., Dey, S., Jha, N.K.: Register transfer level power optimization with emphasis on glitch analysis and reduction. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **18** (1999) 1114–1131
10. Munch, M., Wurth, B., Mehra, R., Sproch, J., Wehn, N.: Automating rt-level operand isolation to minimize power consumption in datapaths. In: Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000. (2000) 624–631
11. Synopsys Inc.: Power Compiler Reference Manual. (2000)

## A Power Dissipation of the Multiplexer

The power dissipation of the multiplexer  $P_M$  is part of the total power dissipation of the register file. This multiplexer can either be implemented using combinatorial logic or a tri-state bus. A tri-state bus requires certain timing constraints which have to be set and controlled for the synthesis tool and the back-end design flow. From an IP reuse point of view, combinatorial logic is much easier to handle because the IP customer does not have to control timing constraints in the IP macro. For this reason we focus on multiplexer using combinatorial logic.

Usually, the combinatorial logic is synthesized. During the technology mapping phase a multitude of cells of the semi-custom library can be selected such that a power computation based on the standard cell approach with a small set of cells as used throughout in this paper is difficult. In the previous sections it was shown that the power dissipation of the multiplexer will not influence the decisions concerning the best choice of a low power register file architecture. Therefore, it is sufficient to have a rough estimate for  $P_M$ . The exact values  $P_M$  are included in the power figures of the sections concerning experimental results.

A rough estimate can be computed assuming a 2:1 Multiplexer with the power dissipation  $P_{M21}$  as basic building block.

$$P_M = w \sum_{j=0}^{\text{ld}(n)-1} \sum_{i=0}^{2^j-1} P_{M21}(i, j). \quad (13)$$

If we assume that the average power consumption of each multiplexer is the identical ( $P_{M21}(i, j) = P_{M21}$ ) to simplify the computation, we obtain a rough estimate for  $P_M$  with  $n = 2^m$ ,  $m$  integer:

$$P_M = P_{M21} w \sum_{j=0}^{\text{ld}(n)-1} 2^j = P_{M21} w (n - 1) \approx P_{M21} w n. \quad (14)$$

# Discrete-Event Modeling and Simulation of Superscalar Microprocessor Architectures

C. Brandolese, W. Fornaciari, and F. Salice

Politecnico di Milano, Piazza L. da Vinci, 32 - 20133 Milano, Italy

**Abstract.** This paper presents a methodology and framework to model the behavior of superscalar microprocessors. The simulation is focused on timing analysis and ignores all functional aspects. The methodology also provides a framework for building new simulators for generic architectures. The results obtained show a good accuracy and a satisfactory computational efficiency. Furthermore, the C++ SDK allows rapid development of new processor models making the methodology suitable for design space exploration over new processor architectures.

## 1 Introduction

Nowadays, at least two out of three processors are employed to realize embedded systems, since they help the designers to trade-off flexibility, chip complexity and productivity. However, with respect to conventional general purpose applications, harder requirements to the design tool-chain are emerging, due to the need of taking into account, at a finer granularity, not purely functional aspects, driving the design of the overall system, such as execution timing. Instruction Set Simulators (ISS) provide a sufficient level of abstraction to capture the details necessary to analyze the performance and the software functionality, while avoiding to move down into the architectural aspects. As far the simulation technology is concerned, accurate methodologies and tools have been developed performing dynamic and/or compilation-based analyses strongly related to proprietary ISSs [8,3,5,7,6]. A comprehensive reviews of both ISA description languages and simulation technologies can be found in [10,6]. This approaches exhibit good accuracy and efficiency but suffer from some problems in terms of generality and model reusability across similar processors. The proposed approach shows some advantages over the existing simulation and modeling environments. Firstly, the model is built using C++ only, thus enforcing generality and code reusability. Nevertheless, the object-oriented SDK has been designed to hide much of the complexity of the simulation algorithms and provides a wide set of extensible classes for common pipeline stages modeling. In addition, the tool-chain allows modeling and simulation a number of components external to the processor such as memory hierarchies, busses, co-processors etc. This makes the environment much more general and flexible than usual ISSs or performance estimation toolsets such as SimpleScalar [11]. As an example, this paper presents the model developed for the microSPARC-II architecture.

## 2 Simulation Framework

The simulation framework described in the following focuses on architectural aspects such as execution time and energy consumption while neglecting functional issues. This means that the input of the simulation cannot be object code but rather an *execution trace*, i.e. a dump of all the assembly instructions executed by the processor during a run of the program under analysis. Such a trace, in fact, carries the same information that an ISSs[3] usually generates.

### 2.1 Behavioral Simulator

The ideas behind the proposed simulation approach is based on 5 keypoints:

**Discrete time.** The clock cycle is the time unit.

**Functional units.** Are active and autonomous elements and model the timing and energy behavior of the processing elements of the architecture.

**Resources.** Are passive elements queried by the functional units to determine some aspects of their behavior.

**Tokens.** Are used to model assembly instructions flowing through the processors' pipelines and to implement functional units synchronization.

**Messages.** Implements communication between functional units and resources.

The behavior of the architecture is thus distributed among interacting components that manipulate tokens according to resource availability in the domain of discrete time. This implies that the status of the architecture is also distributed among resources and functional units. The temporal evolution of the status of the architecture is controlled both by the flow of tokens through the different functional units and by the global clock signal. The arrival of a token into a functional unit represents an *event*.

Figure 1 shows the basic structure of a simulation model. In further detail, tokens are not passed directly from a functional unit to another but are rather stored into variable-depth buffers that actually implement the synchronization

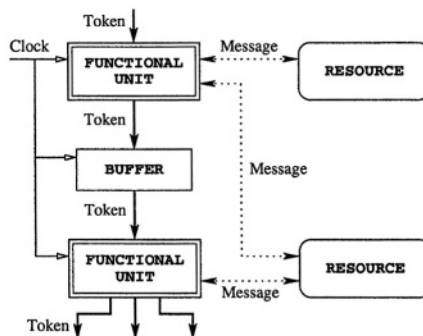


Fig. 1. Behavioral simulator structure



mechanism. At each clock cycle, a functional unit checks its input buffer for available tokens and whenever at least one is present, the functional unit processes it, removes it from the input buffer and forwards one or more possibly modified tokens to the output buffers. Each token models an assembly instruction by means of a set of simple operations referred to in the following as *micro-instructions*. A micro-instruction specifies an elementary action that functional units have to perform when traversed by the token the micro-instruction belongs to. As previously mentioned, tokens may be modified when processed by functional units. In particular, the micro-instructions are deleted from the token as they are executed, so that a token will always leave the pipeline as an empty set.

## 2.2 Micro-Compilation

Assembly instructions forming the execution trace are translated into an ordered sequence of micro-instructions by the process of micro-compilation. It is worth noting that while the complete functionality of an assembly instruction would require a wide set of micro-instructions, the timing behavior only can be captured by a few elementary operations such as load/store, read/write, delay and resource requests. The choice of micro-compiling the assembly code before actual simulation is motivated by the following reasons:

- Generality.** A loosely encoded, internal micro-code, allows the simulator to be more general and to abstract from the details of the specific instruction set.
- Code reuse.** The code developed for a particular target architecture simulator may be reused for other processors with similar characteristics. Furthermore, a platform-independent micro-code enforces the development of configurable modules that can be integrated into different architectures.
- Performance.** The proposed micro-code describes a very small set of operations with extremely simple behavior leading to a faster execution.

As a consequence, each instruction set architecture requires a specific micro-compiler. The development of such micro-compiler implies some knowledge of the target architecture and instruction set. In particular, the elements of the data-path that are used and their corresponding latencies must be known for each instruction. It is worth noting that while micro-code syntax is general, compilation is tightly bound to the structure of the simulated processor. This means that the micro-compiler can only be built once the simulator structure has been clearly defined. It is worth noting that defining a specific simulator architecture still leaves a number of degrees of freedom (e.g. number of GPRs, branch prediction schemes, cache size and policies, etc.) that can be configured at run-time. This makes the simulation environment suitable for ISA-level design space exploration. Though not trivial, micro-compiler development requires much more limited effort with respect to the development of a new simulator from scratch. This activity is further simplified by a SDK providing core support libraries and macros. In the following, this two-phase approach will be shown to be sufficiently general for most of today's pipelined and superscalar CPUs.

### 3 Processor Features Modeling

Despite the variety and high complexity of today's processors, the simulator architecture proposed in this work must only focus on a few aspects that turn out to be relevant for performance analysis, i.e.: (i) pipelining, superscalarity and ILP, (ii) branch prediction techniques, and (iii) memory hierarchy access.

#### 3.1 Pipelining, Superscalarity, and ILP

With pipelining, a number of functional elements are employed in sequence to perform a single computation. These elements form an assembly line or pipeline: each unit performs a certain stage of the computation and each computation goes through the pipeline. Pipelining fits perfectly in the model, as can be seen directly from Figure 1. To best exploit potential instruction parallelism superscalar processors have been introduced. Such processors can execute more than one instruction per clock cycle by using fairly independent pipelines. Concerning superscalarity, it is important to ensure that the proposed simulation model can deal with: (i) superscalar instruction issue (alignment, shelving and dispatching, register renaming or forwarding), (ii) parallel execution, and (iii) preservation of sequential integrity. Superscalar issue relies on two types of instruction alignment techniques: aligned issue and unaligned issue. In the former case, instructions are issued in fixed-size groups only when such groups are completely full; in the latter case, instructions are issued independently as soon as they are available. Going back to the simulator model, in both cases, the functional unit implementing the issue stage of the pipeline models this behavior: for aligned issue, it waits until the tokens arriving at its input queue fill the token group (modeling the instruction group), while for unaligned issue it fires tokens as they arrive (Figure 2). This behavior requires that the functional unit responsible of issuing instructions has an *internal queue* with variable width used to temporarily hold tokens until their processing is finished. In physical devices issue alignment must be combined with dispatching hardware whose job is to forward instructions to subsequent execution elements. In the simulator, this is modeled using multiple functional units connected to a single token source.

Considering commercial architectures, the most common way of detecting instruction dependencies in superscalar execution is the use of *shelving buffers*, i.e. buffers that hold instructions before they are passed to the execution stage

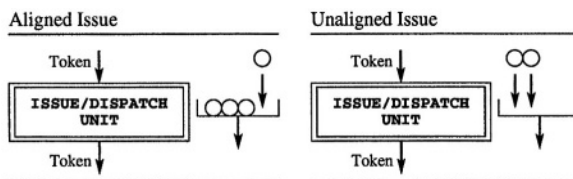


Fig. 2. Modeling of superscalar issue alignment

until all dependencies are satisfied. In the simulator, this structure can be directly mapped to a simulation model where a functional unit can selectively fire tokens to multiple buffers. To allow this behavior, each functional unit, and its input buffer as a consequence, is associated to a *unique identifier*. In modern processors, register renaming and forwarding are two common means of avoiding data dependencies between instructions. These approaches can be modeled within the proposed framework with resources: a resource behaving as a register file can include all the logic pertaining register renaming while a resource flag can be set whenever forwarding is not possible, thus leading to stall conditions. Concerning parallel execution, in the proposed framework it is a direct consequence of the simulator architecture and does not require to be explicitly modeled. To represent *out-of-order execution*, where eventually instructions are guaranteed to be retired in sequential order, reorder buffers can be modeled with a functional unit interacting with a buffer resource returning the oldest token it contains. To model the behavior of processors having units dedicated to the decoding of CISC instructions into simpler RISC ones two solutions are viable. The first closely matches its physical counterpart and is thus based on a functional unit that outputs as many tokens as the number of the corresponding RISC instructions. The second approach considers the RISC core as a black-box and thus directly models the behavior of CISC instructions. As an example, an architecture where RISC instructions have data-independent latencies could be modeled at CISC level with tokens having variable latency equal to the overall execution time of the corresponding RISC instructions.

### 3.2 Branch Prediction Techniques

Branch prediction is modeled by means of a dedicated resource that is accessed by the functional unit implementing the instruction prefetch/fetch stage. All the logic pertaining to branch prediction schemes is embedded into the resource allowing thus functional units to be unaware of such details. For every branch, a given functional unit asks the branch prediction resource whether the branch was correctly predicted or not, and stalls the simulated architecture in case of misprediction. It is worth noting that the instruction following a branch in the execution trace is always the one actually executed, whether the branch was taken or not. Hence, the pipeline does not need to be flushed but rather bubbles, i.e. empty instructions, are generated to emulate the stall instead.

### 3.3 Memory Hierarchy Access

Memory access is implemented by functional units requesting data to memory resources. If the requested data is not immediately available, i.e. there are some wait states, the requesting functional unit is stalled until data is available. This leads to an implementation of memory hierarchy similar to the modeling of branch prediction techniques: all the logic representing memory access is hidden

into a dedicated resource. This way, functional units can abstract from this details and use a uniform interface to access different memory structures. The same resource can be used thus to represent arbitrarily complex cache hierarchies.

### 3.4 Simulation Algorithm

The simulation algorithm is structured into the following steps, repeated for the number of clock cycles needed to execute the chosen program.

1. To avoid access conflicts on resources, concurrent access is granted only to the oldest token: functional units are sorted according to the *age* of the instructions they contain, from oldest to youngest. This behavior is not mandatory, and could be modified to best fit the target architecture.
2. Each functional unit executes a cycle in the given order.
3. Each buffer is updated: tokens are moved from input to output, so that can be accessed from the following functional units in the next cycle.
4. Eventually, resources are updated. It is worth noting that this step is optional, since resources may be clock-independent.

## 4 Case Study: microSPARC-II

The microSPARC™-II is a RISC processor featuring a relatively simple instruction set on a complex data path [1,9]. Much of the simulator complexity resides then in the simulated architecture, while the micro-compiler much simpler. The next sections describe the internal structure of both. The simulated architecture is modeled over the physical one, splitting the processor into nine functional units. Each of these functional units represent a particular pipeline stage of the microSPARC™-II data-path. The connection between functional units is sufficient to model the processor pipeline while all other features are either embedded into specific functional units or modeled by resource access. Adjacent functional units are connected by means of an *instruction buffer* acting as a master-slave register between pipeline stages. Figure 3 shows the simulator structure: functional units are displayed as double-line squares, resources as rounded boxes and instruction queues as small rectangles. Dashed lines connecting the functional units with resources indicate that the functional unit has access to the resource, other connections indicate the possible flows of instructions.

### 4.1 General Structure

As shown by the block diagram of figure 3, instructions flowing into the simulator can follow two paths: the integer data-path and the floating point data-path. The integer ALU unit (ALU<sub>INT</sub>) issues instructions in the correct data-path, where they are then executed. Another interesting aspect shown in the diagram is that the microSPARC™-II can fetch up to two instructions per clock cycle: this may happen if and only if one of the instructions is a branch and the

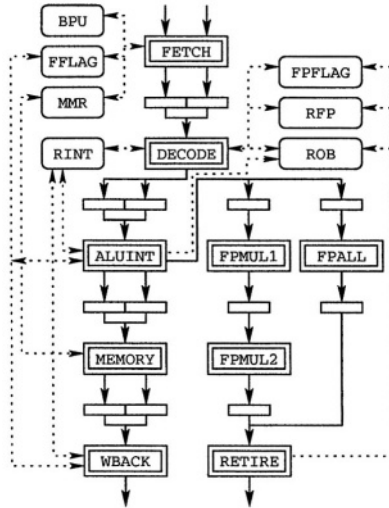


Fig. 3. The microSPARC™-II pipeline

other is an integer ALU operation. For this reason, the integer data-path has a width of two instructions for all its length. Moreover, the floating point data-path can concurrently execute up to three instructions since it integrates a 2-stage pipelined multiplier and a generic floating point ALU working in parallel. Each functional unit has access to a set of resources, such as flags, register files, memory and others. The input queue (IQ) reads instructions from an input file as it receives requests, ignoring the system clock. In this way an arbitrary number of instructions can be read in a clock cycle. This models the data stream between processor cache and the prefetch buffer. The output queue works similarly: every instruction received is simply written to an output file, annotated with execution time figures. The functional units and resources used to model the microSPARC processor are described in detail in the following paragraphs.

**Fetch Unit.** Instructions are read from the input queue (IQ), loaded into a 3-slot prefetch buffer and finally passed to the fetch unit (FETCH). Different delays of memory read operations are modeled accessing the memory management resource (MMR) which returns the number of clock cycles that the unit has to stall before continuing. The fetch unit handles branches also and implements *branch folding* by using the folding flag resource (FFLAG). Whenever the flag is set, folding cannot take place. The flag is set by some instructions to model the architecture limitations concerning this technique. Another peculiarity related to branches is the *annulling state*: when a branch is annulling, its delay slot must traverse the pipeline without any effect. To do this, the instruction in the delay slot is stripped off all its micro-instructions, and traverses the data-path as an empty instruction.

**Decode Unit.** The decode unit (DECODE) loads operands by checking and setting locks on registers modeled by the integer register file resource (RINT).

This unit can process up to two instructions per clock cycle (a branch and an ALU instruction). The decode unit accesses two flags: `FFLAG` to manage branch folding and `FPFLAG` to detect some peculiar floating point hazards.

**Integer ALU Unit.** The integer ALU unit (`ALUINT`) performs two different tasks: executing integer ALU operations and dispatching instructions to subsequent units. Proper micro-instructions tell the unit which queue the instruction has to be sent to among the memory unit (`MEMORY`), the floating point multiplier unit `FPMUL1` and the generic floating point ALU (`FPALL`). The integer ALU unit also uses the folding flag to determine whether register forwarding is possible or not. Forwarding is enabled by unlocking registers in the integer register file resource (`RINT`). Instructions sent to the floating point data-path, are also registered into the reorder buffer resource (`ROB`) which emulates a 4-slot queue that grants in-order-completion of floating point instructions. Trying to insert an instruction in a full reorder buffer, causes this instruction, and consequently the entire integer pipeline, to be stalled until one of the slots is freed.

**Floating Point Units.** These units model the floating point data-path and simply forward instructions from input to output with a given latency. A generic ALU unit (`FPALL`) implements most of the operations while multiplication is done by means of a two-stage pipeline (`FPMUL1` and `FPMUL2`).

**Memory and Write-back Units.** The memory (`MEMORY`) and write-back unit (`WRITEBACK`) do little more than passing instructions from input to output. The former accounts for load/store latencies by accessing the memory resource (`MMR`) and, when forwarding has been disabled, unlocks integer registers in order to model a single-cycle stall in case of data dependency between instructions. The latter detects hazards related to integer register file port usage and produces a one-cycle stall whenever all ports are in use.

**Retire unit.** The retire unit (`RETIRE`) spills instructions from the floating point data-path, unlocks registers and clears flags whenever necessary, and, finally, sends instructions to the output queue. It is worth noting that at this point all instructions are empty. In-order-retire is implemented by a 3-slot internal queue holding completed instructions and by querying the reorder buffer resource (`ROB`) to determine the next instructions to be retired.

## 4.2 The SparcV8 Micro-Compiler

The micro-compiler reads an assembly execution trace and translates it into a simulator readable format. Tracing is performed by the `shade` toolset [2], a performance analysis package distributed by Sun. The trace format is an ASCII file listing the mnemonics of the instructions in the order in which they have actually been executed. The micro-compiler is structured in a ISA-specific front-end performing trace parsing and a general back-end for micro-code synthesis. As an example, Figure 4 shows one of the translation rules composing the front-end. Each instruction has an identification number (`ID()`) and a nominal latency in clock cycles (`LATENCY()`). What follows is a list of micro-instructions that describes the activities in term of resource usage and possibly additional

```

T_PREFIX T_ADD src1, src2, dst {
  ID(49); LATENCY(5);
  READ(RINT,src1); READ(RINT,src2);
  WRITE(RINT,dst); WRITE(RINT,ALUINT,dst); WRITE(RINT,WRITEBACK,dst);
  END;
}

```

**Fig. 4.** Translation rule sample

delays in the different functional units. As an example, the micro-instruction `WRITE( RINT, ALUINT, dst );` is used to unlock the destination register `dst` by letting the `ALUINT` unit access the `RINT` resource. It is worth noting that this micro-instruction will be executed by the `ALUINT` unit as, similarly, the last `WRITE` instruction will be executed by the `WRITEBACK` unit. The behavior of some instructions may change depending on the data or the registers the instruction uses. To model this behavior, conditional statements can be inserted in the translation rules, As an example, the `move` instruction disables forwarding whenever a special registers is used. In particular, when the register is above `R31`, i.e. it is a special register, the `move` unlocks the register during `WRITEBACK` rather than `ALUINT` thus modeling the forwarding disabling mechanism.

## 5 Experimental Results

A complete tool-chain has been built for the microSPARC-II and the Intel i486 cores. A set of 10 benchmarks [4], has been used for validation, leading to the results summarized in Table 1.

**Table 1.** Simulator accuracy results

Code	Error (%)		Code	Error (%)	
	i486	microSPARC		i486	microSPARC
<b>adpcm</b>	-14.88	-9.23	<b>crc16</b>	+2.74	+2.37
<b>gsm</b>	-7.65	-7.48	<b>md5</b>	-5.90	+5.15
<b>lagrange</b>	-6.41	-3.01	<b>rle</b>	-13.31	-1.59
<b>g723</b>	-3.84	-3.00	<b>bsort</b>	-15.72	-0.11
<b>fdct</b>	-13.72	-9.48	<b>matrix</b>	-27.88	-3.38

The actual execution time has been measured by running the benchmarks on the target platform configured to enable microstate accounting (microSPARC) and to use high-resolution timers (i486). Each benchmark has been run with different sets of input data. The delays caused by cache misses, write-buffer overflows and memory refresh have not yet been modeled for the i486 architecture, leading to significantly higher estimation errors. The experiments have been performed on a Pentium III 1.0GHz machine running Linux RedHat 7.2. The performance of all the processing phases have been measured leading to the

following results: instruction tracing runs at 2.4 Minst/s, micro-compilation at 210 Kinst/s and behavioral simulation at 370 Kinst/s. Since the simulation flow can be run in a single pipeline, the resulting simulation throughput is around 200 Kinst/s. It is worth noting that, though the focus of the toolset has been posed on flexibility and ease of use, the simulation speed is still acceptable. An analysis of the bottlenecks of the simulation kernel has shown that microinstruction decoding based on plain text microcode accounts for roughly 80% of the execution time. For this reason, a more efficient *binary mode* is currently being implemented and preliminary test indicate a 5-times speed-up ( $\approx 1$  Minst/s).

## 6 Conclusions

The paper has presented a flexible framework for execution time estimation of assembly code for superscalar architectures. The proposed framework provides a complete SDK that greatly simplifies the development of a simulator for a new architecture. The simulators developed for the two cores considered have shown good accuracy and satisfactory performances. Furthermore, it is worth noting that the developed simulators can be made parametric with respect to different aspects of the architecture such as cache size, associativity and policies, branch prediction algorithms, prefetch buffer size etc. This feature of the tool-chain is especially interesting to perform design exploration of custom architectures.

## References

1. The sparc architecture manual, version 8, 1990.
2. R. F. Cmelik and D. Keppel. Shade: A fast instruction-set simulator for execution profiling. Technical Report SMLI 93-12, 93-06-06, Sun Microsystems, 1993.
3. T. Conte and C. Gimarc. *Fast Simulation of Computer Architectures*. Kluwer Academic Publishers, 1995.
4. R. Desikan, D. Burger, and S. W. Keckler. Measuring experimental error in micro-processor simulation. In *Proceeding of the 28th Annual International Symposium on Computer Architecture*, pages 49–58, 2001.
5. J. Emer. Asim: A performance model framework. *Computer*, 35(2):68–76, 2002.
6. A. Hoffmann, H. Meyr, and R. Leupers. *Architecture Exploration for Embedded Processors with LISA*. Kluwer Academic Publishers, Boston, MA, USA, 2002.
7. M. Lazarescu, J. Bammi, E. Harcourt, L. Lavagno, and M. Lajolo. Compilation-based software performance estimation for system level design. In *Proc. of IEEE International High-Level Validation and Test Workshop*, pages 167–172, 2000.
8. J. Liu, M. Lajolo, and A. Sangiovanni-Vincentelli. Software timing analysis using hw/sw cosimulation and instruction set simulator. In *Proceedings of the Sixth International Workshop on Hardware/Software Codesign*, pages 65–69, 1998.
9. S. microsystems. *microSPARC™-Ilep User's Manual*. Sun microsystems, 1997.
10. S. S. Mukherjee, V. S. Adve, T. Austin, J. Emer, and S. P. Magnusson. Performance simulation tools. *VLSI Design Journal*, 35(2), 2002.
11. SimpleScalar LCC. <http://www.simplescalar.com>.



# Design of High-Speed Low-Power Parallel-Prefix VLSI Adders

G. Dimitrakopoulos<sup>1\*</sup>, P. Kolovos<sup>1</sup>, P. Kalogerakis<sup>1</sup>, and D. Nikolos<sup>2</sup>

<sup>1</sup> Technology and Computer Architecture Laboratory  
Computer Engineering and Informatics Dept., University of Patras, Greece  
{dimitrak,kolobos,kalogera}@ceid.upatras.gr

<sup>2</sup> Computer Technology Institute, 61 Riga Feraiou Str., 26221 Patras, Greece  
nikolosd@cti.gr

**Abstract.** Parallel-prefix adders offer a highly-efficient solution to the binary addition problem. Several parallel-prefix adder topologies have been presented that exhibit various area and delay characteristics. However, no methodology has been reported so far that directly aims to the reduction of switching activity of the carry-computation unit. In this paper by reformulating the carry equations, we introduce a novel bit-level algorithm that allows the design of power-efficient parallel-prefix adders. Experimental results, based on static-CMOS implementations, reveal that the proposed adders achieve significant power reductions when compared to traditional parallel-prefix adders, while maintaining equal operation speed.

## 1 Introduction

Binary addition is one of the primitive operations in computer arithmetic. VLSI integer adders are critical elements in general-purpose and DSP processors since they are employed in the design of arithmetic-logic units, in floating-point arithmetic datapaths, and in address generation units. When high operation speed is required, tree-like structures such as parallel-prefix adders are used. Parallel-prefix adders are suitable for VLSI implementation since they rely on the use of simple cells and maintain regular connections between them. The prefix structures allow several tradeoffs among the number of cells used, the number of required logic levels and the cells' fanout [1].

The high-operation speed and the excessive activity of the adder circuits in modern microprocessors, not only lead to high power consumption, but also create thermal hotspots that can severely affect circuit reliability and increase cooling costs. The presence of multiple execution engines in current processors further aggravates the problem [2]. Therefore, there is a strong need for designing power-efficient adders that would also satisfy the tight constraints of high-speed and single-cycle operation.

Although adder design is a well-researched area, only a few research efforts have been presented concerning adders' performance in the energy-delay

\* This work has been supported by D. Maritsas Graduate Scholarship.

space [3]–[5]. On the other hand, several optimization approaches have been proposed that try to reduce the power consumption of the circuit, either by trading gate sizing with an increase in the maximum delay of the circuit [6], or by using lower supply voltages in non-critical paths [7]. The novelty of the proposed approach is that it directly reduces the switching activity of the carry-computation unit via a mathematical reformulation of the problem. Therefore its application is orthogonal to all other power-optimization methodologies. The proposed parallel-prefix adders are compared to the fastest prefix structures proposed for the traditional definition of carry equations using static CMOS implementations. In all cases the proposed adders are equally fast and can achieve power reductions of up to 11%, although they require around 2.5% larger implementation area. Additionally, for an 1.5% increase in the maximum delay of the circuit, power reductions of up to 17% are reported.

The remainder of the paper is organized as follows. Section 2 describes the intuition behind the proposed approach. In Section 3 some background information on parallel-prefix addition are given, while Section 4 introduces the low-power adder architecture. Experimental results are presented in Section 5 and finally conclusions are drawn in Section 6.

## 2 Main Idea

Assume that  $A = a_{n-1}a_{n-2} \dots a_0$  and  $B = b_{n-1}b_{n-2} \dots b_0$  represent the two numbers to be added, and  $S = s_{n-1}s_{n-2} \dots s_0$  denotes their sum. An adder can be considered as a three-stage circuit. The preprocessing stage computes the carry-generate bits  $g_i$ , the carry-propagate bits  $p_i$ , and the half-sum bits  $h_i$ , for every  $i$ ,  $0 \leq i \leq n-1$ , according to:

$$g_i = a_i \cdot b_i \quad p_i = a_i + b_i \quad h_i = a_i \oplus b_i, \quad (1)$$

where  $\cdot$ ,  $+$ , and  $\oplus$  denote the logical AND, OR and exclusive-OR operations respectively. The second stage of the adder, hereafter called the carry-computation unit, computes the carry signals  $c_i$ , using the carry generate and propagate bits  $g_i$  and  $p_i$ , whereas the last stage computes the sum bits according to:

$$s_i = h_i \oplus c_{i-1}. \quad (2)$$

The computation of the carries  $c_i$  can be performed in parallel for each bit position  $0 \leq i \leq n-1$  using the formula

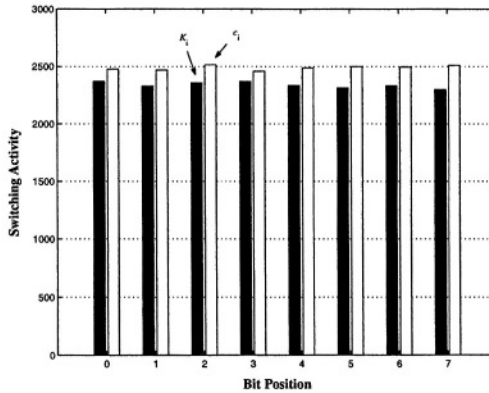
$$c_i = g_i + \sum_{j=-1}^{i-1} \left( \prod_{k=j+1}^i p_k \right) g_j, \quad (3)$$

where the bit  $g_{-1}$  represents the carry-in signal  $c_{in}$ . Based on (3), each carry  $c_i$  can be written as

$$c_i = g_i + \mathcal{K}_i, \quad (4)$$

where  $\mathcal{K}_i = \sum_{j=-1}^{i-1} \left( \prod_{k=j+1}^i p_k \right) g_j$ . It can be easily observed from (4) that the switching activity of the bits  $c_i$  equally depends on the values assumed by the carry-generate bits  $g_i$  and the term  $\mathcal{K}_i$ .

In order to quantify the difference between the switching activity of the bits  $\mathcal{K}_i$  and the carries  $c_i$ , an experiment was performed. In particular, a set of 5000 random vectors were generated using Matlab and the switching activity of the bits  $c_i$  and  $\mathcal{K}_i$  was measured for the case of an 8-bit adder. Figure 1 shows the measured switching activity for each bit position. It is evident that the bits  $\mathcal{K}_i$  have reduced switching activity that range from 5% to 7.5%.



**Fig. 1.** Switching activity of the bits  $\mathcal{K}_i$  and the traditional carries  $c_i$ .

The direct computation of the bits  $\mathcal{K}_i$  using existing parallel-prefix methods is possible. However, it imposes the insertion of at least one extra logic level that would increase the delay of the circuit. Therefore our objective is to design a carry-computation unit that will compute the bits  $\mathcal{K}_i$  instead of the bits  $c_i$ , without any delay penalty, and will benefit from the inherent reduced switching activity of the bits  $\mathcal{K}_i$ .

### 3 Background on Parallel-Prefix Addition

Several solutions have been presented for the carry-computation problem. Carry computation is transformed to a prefix problem by using the associative operator  $\circ$ , which associates pairs of generate and propagate bits and is defined, according to [8], as follows,

$$(g, p) \circ (g', p') = (g + p \cdot g', p \cdot p'). \quad (5)$$

Using consecutive associations of the generate and propagate pairs  $(g, p)$ , each carry is computed according to

$$c_i = (g_i, p_i) \circ (g_{i-1}, p_{i-1}) \circ \dots \circ (g_1, p_1) \circ (g_0, p_0). \quad (6)$$

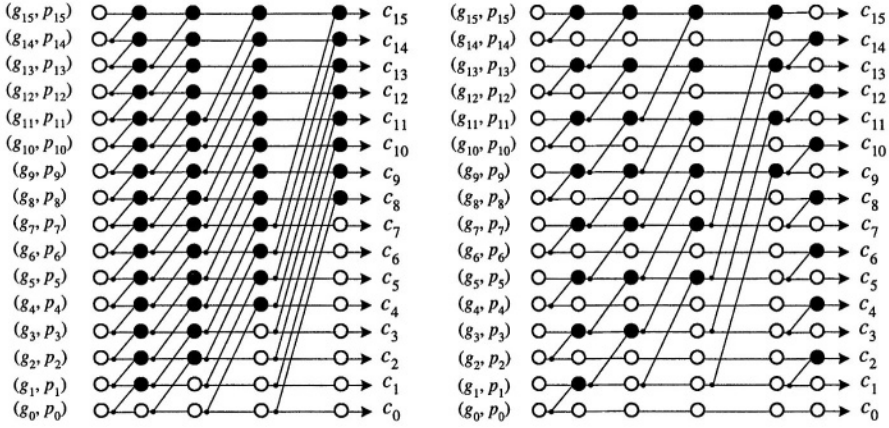


Fig. 2. The Kogge-Stone and Han-Carlson parallel-prefix structures.

Representing the operator  $\circ$  as a node  $\bullet$  and the signal pairs  $(g, p)$  as the edges of a graph, parallel-prefix carry-computation units can be represented as directed acyclic graphs. Figure 2 presents the 16-bit parallel-prefix adders, proposed by Kogge-Stone [9] and Han-Carlson [10], where white nodes  $\circ$  are buffering nodes. A complete discussion on parallel-prefix adders can be found in [1] and [11].

## 4 The Proposed Design Methodology

In the following we will present via an example the parallel-prefix formulation of the computation of the bits  $\mathcal{K}_i$ . Assume for example the case of  $\mathcal{K}_5$ . Then according to (4) it holds that

$$\begin{aligned} \mathcal{K}_5 = & p_5 \cdot g_4 + p_5 \cdot p_4 \cdot g_3 + p_5 \cdot p_4 \cdot p_3 \cdot g_2 + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot g_1 \\ & + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot g_0 \\ & + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_{in}. \end{aligned} \quad (7)$$

Based on the definition (1) it holds that  $p_i \cdot g_i = g_i$ . Then equation (7) can be written as

$$\begin{aligned} \mathcal{K}_5 = & p_5 \cdot p_4 \cdot (g_4 + g_3) + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot (g_2 + g_1) \\ & + p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot (g_0 + c_{in}). \end{aligned} \quad (8)$$

Assuming that

$$G_i = g_i + g_{i-1} \quad (9)$$

and

$$P_i = p_i \cdot p_{i-1}, \quad (10)$$

with  $G_0 = g_0 + c_{in}$  and  $P_0 = p_0$ , then equation (8) is equivalent to

$$\mathcal{K}_5 = P_5 \cdot G_4 + P_5 \cdot P_3 \cdot G_2 + P_5 \cdot P_3 \cdot P_1 \cdot G_0. \quad (11)$$

The computation of  $\mathcal{K}_5$  can be transformed to a parallel-prefix problem by introducing the variable  $G_i^*$ , which is defined as follows:

$$G_i^* = P_i \cdot G_{i-1} \quad (12)$$

and  $G_0^* = p_0 \cdot c_{in}$ . Substituting (12) to (11) we get

$$\mathcal{K}_5 = G_5^* + P_5 \cdot G_3^* + P_5 \cdot P_3 \cdot G_1^*, \quad (13)$$

which can be equivalently expressed, using the  $\circ$  operator as

$$\mathcal{K}_5 = (G_5^*, P_5) \circ (G_3^*, P_3) \circ (G_1^*, P_1). \quad (14)$$

In case of an 8-bit adder the bits  $\mathcal{K}_i$  can be computed by means of the prefix operator  $\circ$  according to the following equations:

$$\mathcal{K}_7 = (G_7^*, P_7) \circ (G_5^*, P_5) \circ (G_3^*, P_3) \circ (G_1^*, P_1)$$

$$\mathcal{K}_6 = (G_6^*, P_6) \circ (G_4^*, P_4) \circ (G_2^*, P_2) \circ (G_0^*, P_0)$$

$$\mathcal{K}_5 = (G_5^*, P_5) \circ (G_3^*, P_3) \circ (G_1^*, P_1)$$

$$\mathcal{K}_4 = (G_4^*, P_4) \circ (G_2^*, P_2) \circ (G_0^*, P_0)$$

$$\mathcal{K}_3 = (G_3^*, P_3) \circ (G_1^*, P_1)$$

$$\mathcal{K}_2 = (G_2^*, P_2) \circ (G_0^*, P_0)$$

$$\mathcal{K}_1 = (G_1^*, P_1)$$

$$\mathcal{K}_0 = (G_0^*, P_0)$$

It can be easily derived by induction that the bits  $\mathcal{K}_i$  of the odd and the even-indexed bit positions can be expressed as

$$\mathcal{K}_{2k} = (G_{2k}^*, P_{2k}) \circ (G_{2k-2}^*, P_{2k-2}) \circ \dots \circ (G_0^*, P_0) \quad (15)$$

$$\mathcal{K}_{2k+1} = (G_{2k+1}^*, P_{2k+1}) \circ (G_{2k-1}^*, P_{2k-1}) \circ \dots \circ (G_1^*, P_1) \quad (16)$$

Based on the form of the equations that compute the terms  $\mathcal{K}_i$ , in case of the 8-bit adder, the following observations can be made. At first the number of terms  $(G_i^*, P_i)$  that need to be associated is reduced to half compared to the traditional approach, where the pairs  $(g, p)$  are used (Eq. (6)). In this way one less prefix level is required for the computation of the bits  $\mathcal{K}_i$ , which directly leads to a reduction of 2 logic levels in the final implementation. Taking into account that the new preprocessing stage that computes the pairs  $(G_i^*, P_i)$  requires two additional logic levels compared to the logic-levels needed to derive the bits  $(g, p)$  in the traditional case, we conclude that no extra delay is imposed by the proposed adders. Additionally, the terms  $\mathcal{K}_i$  of the odd and the even-indexed bit positions can be computed independently, thus directly reducing the fanout requirements of the parallel-prefix structures.

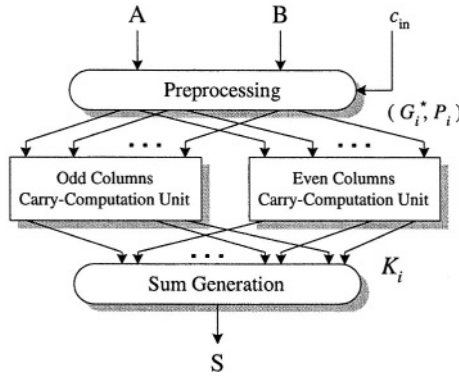


Fig. 3. The architecture of the proposed parallel-prefix adders.

The computation of the bits  $\mathcal{K}_i$ , instead of the normal carries  $c_i$ , complicates the derivation of the final sum bits  $s_i$  since in this case

$$s_i = h_i \oplus c_{i-1} = h_i \oplus (g_{i-1} + \mathcal{K}_{i-1}). \tag{17}$$

However, using the Shannon expansion theorem on  $\mathcal{K}_{i-1}$ , the computation of the bits  $s_i$  can be transformed as follows

$$s_i = \bar{\mathcal{K}}_{i-1} \cdot (h_i \oplus g_{i-1}) + \mathcal{K}_{i-1} \cdot h_i. \tag{18}$$

Equation (18) can be implemented using a multiplexer that selects either  $h_i$  or  $(g_{i-1} \oplus h_i)$  according to the value of  $\mathcal{K}_{i-1}$ . The notation  $\bar{x}$  denotes the complement of bit  $x$ . Taking into account that in general a XOR gate is of almost equal delay to a multiplexer, and that both  $h_i$  and  $(g_{i-1} \oplus h_i)$  are computed in less logic levels than  $\mathcal{K}_{i-1}$ , then no extra delay is imposed by the use of the bits  $\mathcal{K}_i$  for the computation of the sum bits  $s_i$ . The carry-out bit  $c_{n-1}$  is produced almost simultaneously with the sum bits using the relation  $c_{n-1} = g_{n-1} + \mathcal{K}_{n-1}$ .

The architecture of the proposed adders is shown in Figure 3, and their design is summarized in the following steps.

- Calculate the carry generate/propagate pairs  $(g_i, p_i)$  according to (1).
- Combine the bits  $g_i, p_i, g_{i-1}$ , and  $p_{i-1}$  in order to produce the intermediate pairs  $(G_i^*, P_i)$ , based on the definitions (9), (10), and (12).
- Produce two separate prefix-trees, one for the even and one for the odd-indexed bit positions that compute the terms  $\mathcal{K}_{2k}$  and  $\mathcal{K}_{2k+1}$  of equations (15) and (16), using the pairs  $(G_i^*, P_i)$ . Any of the already known parallel-prefix structures can be employed for the generation of the bits  $\mathcal{K}_i$ , in  $\log_2 n - 1$  prefix levels.

In Figure 4 several architectures for the case of a 16-bit adder are presented, each one having different area, delay and fanout requirements. It can be easily

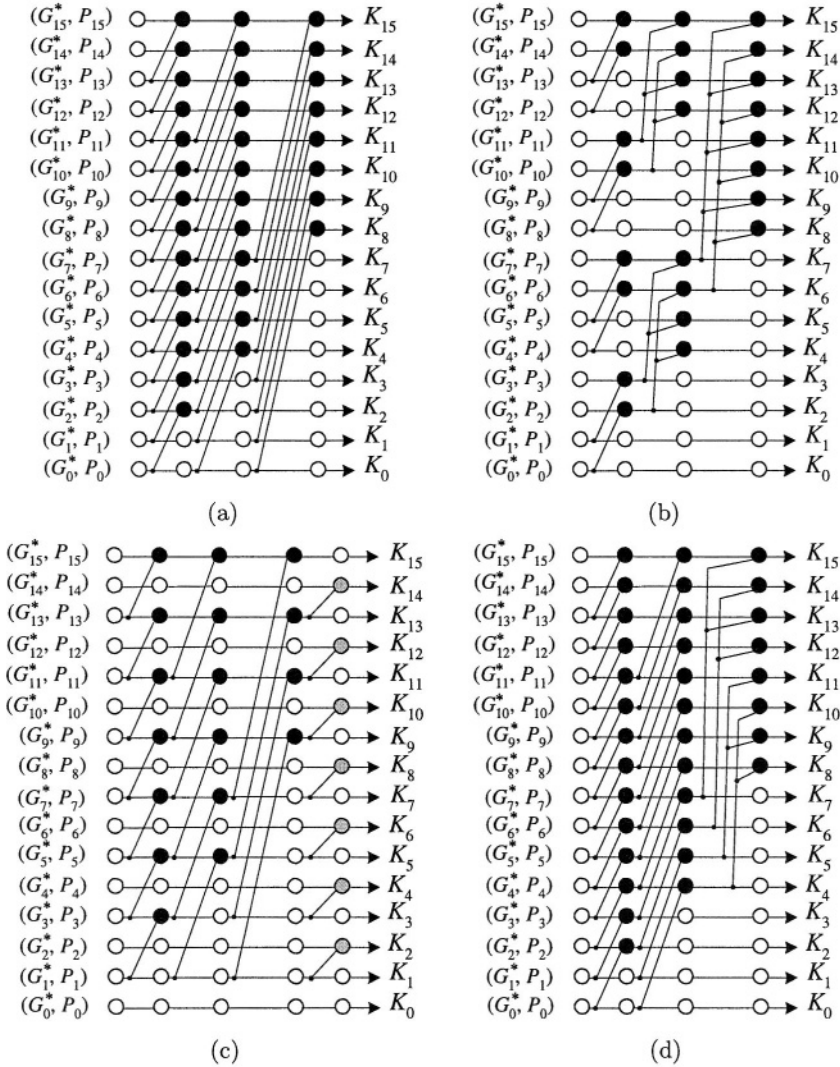
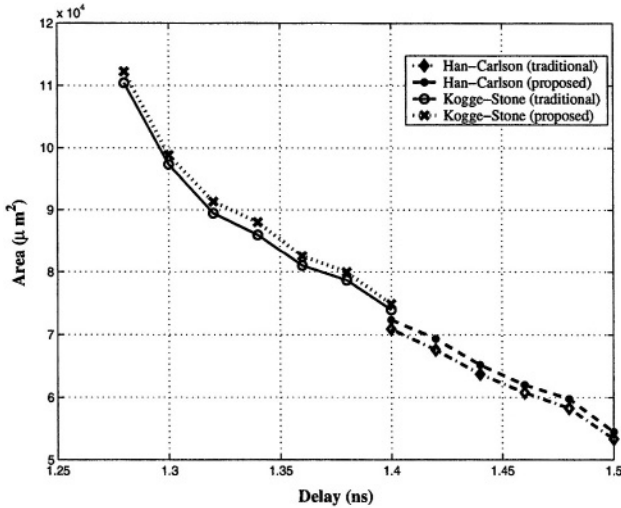


Fig. 4. Novel 16-bit parallel-prefix carry computation units.

verified that the proposed adders maintain all the benefits of the parallel-prefix structures, while at the same time offer reduced fanout requirements. Figure 4(a) presents a Kogge-Stone-like parallel-prefix structure, while in Figure 4(c) a Han-Carlson-like scheme is shown. It should be noted that in case of the proposed Han-Carlson-like prefix tree only the terms  $\mathcal{K}_i$  of the odd-indexed bit positions are computed. The remaining terms of the even-indexed bit positions are computed using the bits  $\mathcal{K}_i$  according to

$$\mathcal{K}_{i+1} = p_{i+1} \cdot (g_i + \mathcal{K}_i), \tag{19}$$

which are implemented by the grey cells of the last prefix level.



**Fig. 5.** The area and delay estimates for the traditional and the proposed 64-bit adders using a Kogge-Stone [9] and Han-Carlson [10] parallel-prefix structures.

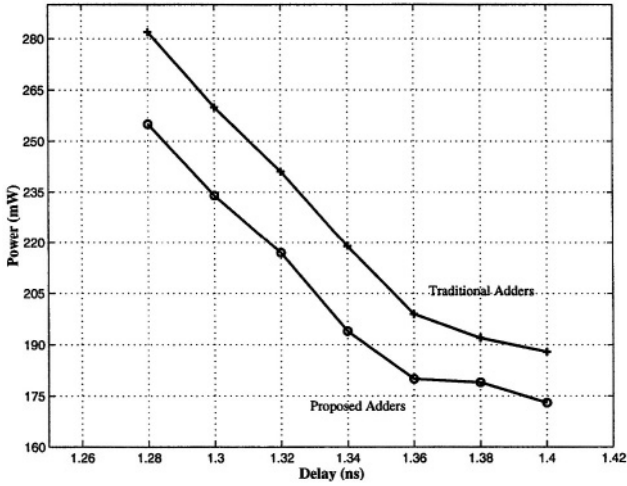
## 5 Experimental Results

The proposed adders are compared against the parallel-prefix structures proposed by Kogge-Stone [9] and Han-Carlson [10] for the traditional definition of carry equations (Eq. (6)). Each 64-bit adder was at first described and simulated in Verilog HDL. In the following, all adders were mapped on the  $0.25\mu\text{m}$  VST-25 technology library under typical conditions (2.5V,  $25^\circ\text{C}$ ), using the Synopsys® Design Compiler. Each design was recursively optimized for speed targeting the minimum possible delay. Also, several other designs were obtained targeting less strict delay constraints.

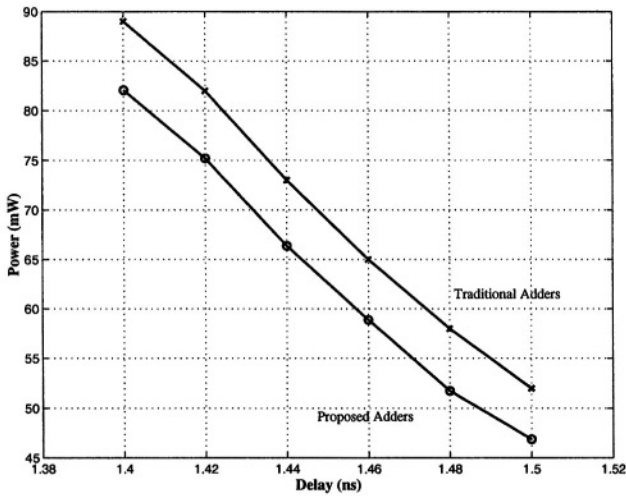
Figure 5 presents the area and delay estimates of the proposed and the traditional parallel-prefix adder architectures. It is noted that the proposed adders in all cases achieve equal delay compared to the already known structures with only a 2.5% in average increase in the implementation area. Based on the data provided by our technology library, it is derived that the delay of 1 fanout-4 (FO4) inverter equals to 120 – 130ps, under typical conditions. Thus the obtained results fully agree with the results given in [12] and [5], where the delay of different adder topologies is calculated using the method of Logical Effort.

In Figure 6 the power consumption of the proposed and the traditional Kogge-Stone 64-bit adders are presented versus the different delays of the circuit. It can be easily observed that the proposed architecture achieves power reductions that range between 9% and 11%. All measurements were taken using PrimePower of Synopsys toolset after the application of 5000 random vectors. It should be noted that although the proposed adders require slightly larger implementation area, due to the addition of the extra multiplexers in the sum





**Fig. 6.** Power and delay estimates for the traditional and the proposed 64-bit adders using a Kogge-Stone [9] parallel-prefix structure.



**Fig. 7.** Power and delay estimates for the traditional and the proposed 64-bit adders using a Han-Carlson [10] parallel-prefix structure.

generation unit, they still offer power efficient designs as a result of the reduced switching activity of the the novel carry-computation units. Similar results are obtained for the case of the 64-bit Han-Carlson adders, as shown in Figure 7.

Finally, since the application of the proposed technique is orthogonal to all other power optimization methods, such as gate sizing and the use of multiple supply voltages, their combined use can lead to further reduction in power dissipation.

pation. Specifically, the adoption of the proposed technique offers on average an extra 10% gain over the reductions obtained by the use of any other circuit-level optimization.

## 6 Conclusions

A systematic methodology for the design of power-efficient parallel-prefix adders has been introduced in this paper. The proposed adders preserve all the benefits of the traditional parallel-prefix carry-computation units, while at the same time offer reduced switching activity and fanout requirements. Hence, high-speed datapaths of modern microprocessors can truly benefit from the adoption of the proposed adder architecture.

## References

1. Simon Knowles, "A Family of Adders", in Proceedings of the 14th IEEE Symposium on Computer Arithmetic, April 1999, pp. 30–34.
2. S. Mathew, R. Krishnamurthy, M. Anders, R. Rios, K. Mistry and K. Soumyanath, "Sub-500-ps 64-b ALUs in 0.18- $\mu\text{m}$  SOI/Bulk CMOS: Design and Scaling Trends" IEEE Journal of Solid-State Circuits, vol. 36, no. 11, Nov. 2001.
3. T. K. Callaway and E. E. Swartzlander, "Low-Power arithmetic components in Low-Power Design Methodologies", J. M. Rabaey and M. Pedram, Eds. Norwell, MA: Kluwer, 1996, pp. 161–200.
4. C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-Time-Power tradeoffs in parallel adders", IEEE Trans. Circuits Syst. II, vol.43, pp. 689–702, Oct.1996.
5. V. G. Oklobdzija, B. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-Delay Estimation Technique for High-Performance Microprocessor VLSI Adders", 16th IEEE Symposium on Computer Arithmetic, June 2003, pp. 15–22.
6. R. Brodersen, M. Horowitz, D. Markovic, B. Nikolic and V. Stojanovic, "Methods for True Power Minimization, International Conference on Computer-Aided Design, Nov. 2002, pp. 35–42.
7. Y. Shimazaki, R. Zlatanovici, and B. Nikolic, "A Shared-Well Dual-Supply-Voltage 64-bit ALU", IEEE Journal of Solid-State Circuits, vol. 39, no. 3, March 2004.
8. R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders", IEEE Trans. on Computers, vol. 31, no. 3, pp. 260–264, Mar. 1982.
9. P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations", IEEE Trans. on Computers, vol. C-22, pp. 786–792, Aug. 1973.
10. T. Han and D. Carlson, "Fast Area-Efficient VLSI Adders", in Proc. 8th IEEE Symposium on Computer Arithmetic, May 1987, pp. 49–56.
11. A. Beaumont-Smith and C. C. Lim, "Parallel-Prefix Adder Design", 14th IEEE Symposium on Computer Arithmetic, Apr. 2001, pp. 218–225.
12. D. Harris and I. Sutherland, "Logical Effort of Carry Propagate Adders", 37th Asilomar Conference, Nov. 2003, pp. 873–878.

# GALSification of IEEE 802.11a Baseband Processor

Miloš Krstić and Eckhard Grass

IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany  
{krstic, grass}@ihp-microelectronics.com

**Abstract.** In this paper a Globally Asynchronous Locally Synchronous (GALS) implementation of a complex digital system is discussed. The deployed asynchronous wrappers are based on a novel request-driven technique with embedded time-out function. Each GALS block is fitted with a local clock generator for flushing the internal pipeline stages when there are no incoming requests. This request-driven technique is applied for the ‘GALSification’ of an IEEE 802.11a compliant baseband processor. Details of the GALS partitioning and some additionally developed blocks will be discussed. Furthermore, the design-flow, implementation results and power estimation numbers are reported.

## 1 Introduction

Current trends in System on Chip (SoC) integration impose a number of technical challenges on designers and tools. In the area of wireless communication systems, the number of difficulties for the SoC implementation additionally grows. There is a constant demand for elegant solutions for decreasing power consumption, reducing electromagnetic interference (EMI) between analog and digital circuit blocks, and effective clocking schemes for complex digital systems. Many proposals have been published, but the question which of them will be applied in industrial practice is without answer, yet.

GALS techniques attempt to solve some problems of SoC integration. There are many different proposals how to create a stable and efficient GALS system. In papers [1, 2] some of the most popular ways of synchronous system GALSification can be found. Usually, a GALS system consists of a number of locally synchronous blocks surrounded with asynchronous wrappers. Communication between synchronous blocks is performed indirectly via asynchronous wrappers. The current level of GALS development offers a relatively stable framework for the implementation of complex digital systems. However, known GALS techniques may introduce some drawbacks in performance, additional constraints in the system and hardware overhead. On the other hand, all proposed GALS techniques are oriented towards some general architecture with sporadic and not too intensive data transfer on its interfaces.

In paper [3] we have introduced a new request-driven GALS technique, optimized for a datapath architecture with bursty, but very intensive data transfer. In this paper the structure of the asynchronous wrapper was described in detail. We decided that after GALSification of relatively small experimental designs, the complete spectrum of advantages and disadvantages of our proposed request-driven GALS technique could be investigated best in a complex datapath application. Since we are working on

a system-on-chip implementation of an IEEE 802.11a modem, the baseband processor for this standard appeared a good candidate for introducing GALS. In particular the reduction of EMI for this mixed-signal design is in the focus of this work. The baseband processor is a relatively complex design (around 700k gates) with an internal datapath structure, which includes sub-blocks like Viterbi decoder, FFT/IFFT processor, and different CORDIC processors. Introducing the GALS mechanisms in this baseband processor was a challenge and some results of this work will be reported here.

In the following chapter we will first briefly describe the concept of our request driven GALS technique and possible advantages of using that approach. In the third chapter some basic information about the IEEE 802.11a baseband processor will be given and our proposed way of GALS partitioning will be shown. Subsequently, there are the chapters describing our design flow and showing implementation details and results. Finally, some conclusions are drawn.

## 2 Concept of Request Driven GALS Technique

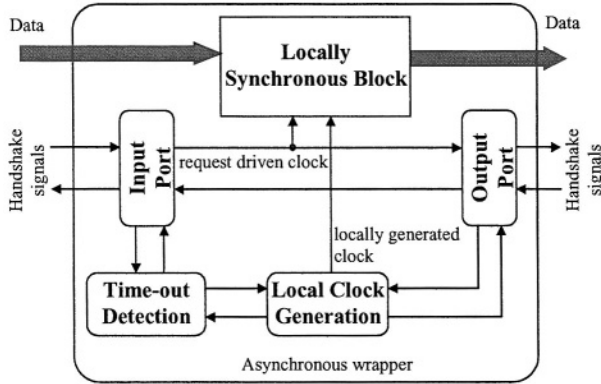
During development of our GALS system, a main goal was avoiding the unnecessary transitions and the attendant waste of energy during data transfer between GALS blocks, as reported in [3]. A secondary issue is to achieve high data throughput with low latency.

For a constant input data stream, GALS blocks could operate in a request-driven mode, i.e. it is possible to synchronize the local clock generators with the request input signal. This way no unnecessary transitions are generated. However, for bursty signals, when there is no input activity, the data stored inside the locally synchronous pipeline has to be flushed out. This can be achieved by switching to a mode of operations in which a local clock generator drives the GALS block independently. To control the transition from request driven operation to flushing out data, a time-out function is proposed. The time-out function is triggered when no request has appeared at the input of a GALS block, but data is still stored in internal pipeline stages.

Our proposed concept is intended for systems, mainly consisting of point-to-point connections between complex synchronous blocks. The principle architecture of our modified asynchronous wrapper around a locally synchronous module is shown in Fig. 1. The locally synchronous circuit can be driven both by the incoming request as well as the local clock signal. The driver of the request input signal is in the asynchronous wrapper of the previous block. It is synchronized with the transferred data, and can be considered as a token carrier.

When there is no incoming request signal for a certain period of time (equivalent to  $T_{\text{time-out}}$ ), the circuit enters a new state where it can internally generate clocks using a local ring oscillator. The number of internally generated clocks is set to match the number of cycles needed to empty the locally synchronous pipeline. When there is no valid token in the synchronous block, the local clock will stall and the circuit remains inactive until the next request transition, indicating that a fresh data token has appeared at the input. This way we avoid possible energy waste.

More complex and demanding is the scenario when after time-out and starting of the local-clock generation, a new request appears before the synchronous pipeline is emptied. In this case it is necessary to complete the present local clock cycle to pre-



**Fig. 1.** New asynchronous wrapper around locally synchronous block

vent metastability at data inputs. Subsequently it is possible to safely hand over clock generation from the local ring oscillator to the input request line. To deal with this situation it is necessary to implement additional circuitry to prevent metastability and hazards in the system.

The proposed architecture has several potential advantages. First, and the most obvious is the decreased latency, when the circuit is operating in request-driven mode. In that mode arbitration is not needed and incoming requests are automatically considered as clock signals. Therefore, the performance drop, always present in other GALS solutions, is considerably decreased.

This GALS technique offers an efficient power-saving mechanism, similar to clock gating. A particular synchronous block is clocked only when there is new data at the input, or there is a need to flush data from internal pipeline stages. In all other cases there is no activity within this block.

In a complex system, designed with the proposed GALS technique, different locally synchronous modules are driven by independent clocks. These clocks will have arbitrary phases due to varying delays in different asynchronous wrappers. During local clock generation they have even different frequencies. In this way EMI and the attendant generation of substrate and supply noise can be reduced.

The target applications of the proposed GALS technique are datapath architectures, often deployed for baseband processing in communication systems. For designing ‘joins’ and ‘forks’ between asynchronous wrappers, standard asynchronous methods can be employed.

Based on the proposed concept asynchronous wrappers have been developed and were described in [3]. In this paper, the reader can find detailed information on the construction of specific individual blocks of the request-driven GALS system.

### 3 GALS Baseband Processor

Our work on GALS implementations is linked to a project that aims to develop a single-chip wireless broadband communication system in the 5 GHz band, compliant with the IEEE802.11a [4, 5] standard. This standard specifies a broadband communi-

cation system using Orthogonal Frequency Division Multiplexing (OFDM) with data rates ranging from 6 to 54 Mbit/s. The design of the required mixed-signal baseband processor for this standard involves a number of challenges. In particular the global clock tree generation, power consumption, testability, EMI and crosstalk are very demanding issues for the designer. Our request-driven GALS architecture was developed as a possible solution for those problems.

Initially, the baseband processor was implemented as an ASIC (Application Specific Integrated Circuit) working synchronously. The natural style of the baseband processor implementation is a datapath architecture. In order to achieve lower power dissipation, the baseband architecture is partitioned in to two blocks: Transmitter and Receiver as shown in Figure 2.

The standard [4] defines the procedures for receiver and transmitter datapath processing. Our transmitter comprises an input buffer, scrambler, signal field generator, encoder, interleaver, mapper, 64-point IFFT (Inverse Fast Fourier Transform) and circuitry for pilot insertion, guard interval insertion, and preamble insertion. Fundamental issues of the receiver are the synchronization mechanism (including 64-point FFT - Fast Fourier Transform) and the channel estimation. Other blocks of the receiver are demapper, deinterleaver, descrambler, Viterbi decoder, and additional buffers.

The transmitter GALS partitioning is relatively easy to be performed. It is a straightforward datapath architecture, based on a token-flow design technique. The complete baseband transmitter is divided into three modules. This block division takes into account the functionality and complexity of the existing blocks.

Block Tx\_1 comprises 80 MHz components such as signal field generator, scrambler, encoder, interleaver and mapper. The complexity of the block is about 17k gates (inverter equivalent). Block Tx\_1 is not GALSified due to the complex off-chip interface protocol. The intermediate block Tx\_2 (20k gates) has to perform pilot insertion and serial to parallel conversion. Additionally, this block performs token rate adaptation because it does receive tokens at a frequency of 80 Msps and should generate tokens at 20 Msps for the subsequent block. This is performed in the following way: when block Tx\_2 is in request-driven mode, it will collect data at 80 Msps, but it will not send any token to the neighboring block Tx\_3. When a complete data burst for one symbol is collected (48 data), the local clock generator, set to 20 Msps, will initiate data transfer to block Tx\_3. Block Tx\_3 contains very complex (153k gates) processing stages, as IFFT, guard interval insertion and preamble insertion. There is an additional problem in the communication between blocks Tx\_2 and Tx\_3. Generally, communication between those two blocks is performed only in bursts of 8 data tokens and than block Tx\_3 has to process this data for the next 72 cycles without any new incoming data. In order to enable that data transfer free period of 72 cycles, a special signal is generated in Tx\_3 that grants a transfer of tokens from the Tx\_2 side.

It is significantly more difficult to introduce GALS in the receiver. There are several reasons for that: The gate count is more than double in comparison with the transmitter. Additionally, the data-flow is much more complicated. The channel estimator is based on a feedback loop. Accordingly, inside the receiver there is a token ring between some blocks (Fig. 2).

The activation scheme of the baseband processor shows that the receiver stage is processing data all the time, except for the periods when data is transmitted. Most of its own time the receiver will just search for the synchronization sequence (preamble). It appears plausible to implement the tracking synchronizer as a separate GALS

block. This block (Rx\_1) is comparatively small (80k gates), and if we can limit the switching activities inside the baseband to this block for most of the time, we can keep the power consumption low. Block Rx\_2 will be activated when coarse synchronization is reached, and it will perform fine synchronization, frequency error correction, FFT and channel estimation. This block is very complex (235k gates) and all processing is performed at 20 Msps. Block Rx\_3 (168k) will perform demapping, deinterleaving, Viterbi decoding, descrambling and loop backward processing needed for channel estimation. All these blocks naturally operate at 80 Msps.

One of the main challenges in the receiver is how to connect blocks Rx\_2 and Rx\_3, and how to arrange token rate adaptation **20 Msps**  $\rightarrow$  **80 Msps** (in forward data transfer Rx\_2  $\rightarrow$  Rx\_3), and **80 Msps**  $\rightarrow$  **20 Msps** (in backward data transfer Rx\_3  $\rightarrow$  Rx\_2). An additional problem is how to synchronize backward data transfer with forward data transfer. A possible solution can be achieved by adding two types of interface blocks: GALS block Rx\_TRA and the asynchronous FIFO\_TA, as shown in Figure 2. This way the backward dataflow is treated as an independent dataflow that creates asynchronous tokens. One task of those blocks is token rate adaptation. Rx\_TRA will perform datarate transformation from 20 Msps to 80 Msps and FIFO\_TA, datarate transformation from 80 Msps to 20 Msps. A critical point in this solution is the control of the asynchronous token ring. Since the latency in this feedback loop can vary, joins and forks that are created in the system should be designed in such way that operation of the system is not disrupted. Having this in mind, an elegant implementation of the join circuit for the alignment of tokens entering block Rx\_2 is crucial for making this system functional.

Using the proposed partitioning inside the baseband processor it is possible to efficiently implement power saving mechanisms. After reset the baseband processor will produce no switching activity until the first data from outside arrives. After that, in receive mode, most of the time only block Rx\_1 will be active, trying to find the synchronization pattern. This block is relatively small and the power consumption level will be acceptable. When synchronization is reached, block Rx\_1 becomes inactive and block Rx\_2 will start its activity. This is achieved by switching the token multiplexer that is termed “activation interface” in Fig. 2 from tracking to the datapath synchronizer. During operation of the receiver, the transmitter is inactive, and vice versa. The amount of power saving achieved by this technique is comparable to clock gating with a similar strategy.

The baseband processor will normally be used in a standard synchronous environment. In the transmit mode it has to send data tokens every clock cycle (20 Msps) to DACs (Digital to Analog Converters). In receive mode it should absorb data tokens from the ADCs (Analog to Digital Converter) in every clock cycle (20 Msps).

Connection between a synchronous producer and an asynchronous consumer is achieved in a very simple manner. Due to the request driven property of the asynchronous wrapper, we have just connected the request line of the wrapper with the clock source that creates incoming synchronous data. The only condition that must be fulfilled is the ability of asynchronous wrappers to absorb incoming tokens. Fulfillment of this condition is safely achieved with the insertion of decoupling circuitry in blocks RX\_TRA and Tx\_2.

To connect the asynchronous producer with a synchronous consumer we have used pipeline synchronization [6] (blocks Rx\_int and Tx\_int in Fig. 2). This way we can safely connect an asynchronous producer with a synchronous consumer. By addition of more pipeline stages we can easily increase the robustness of our system.

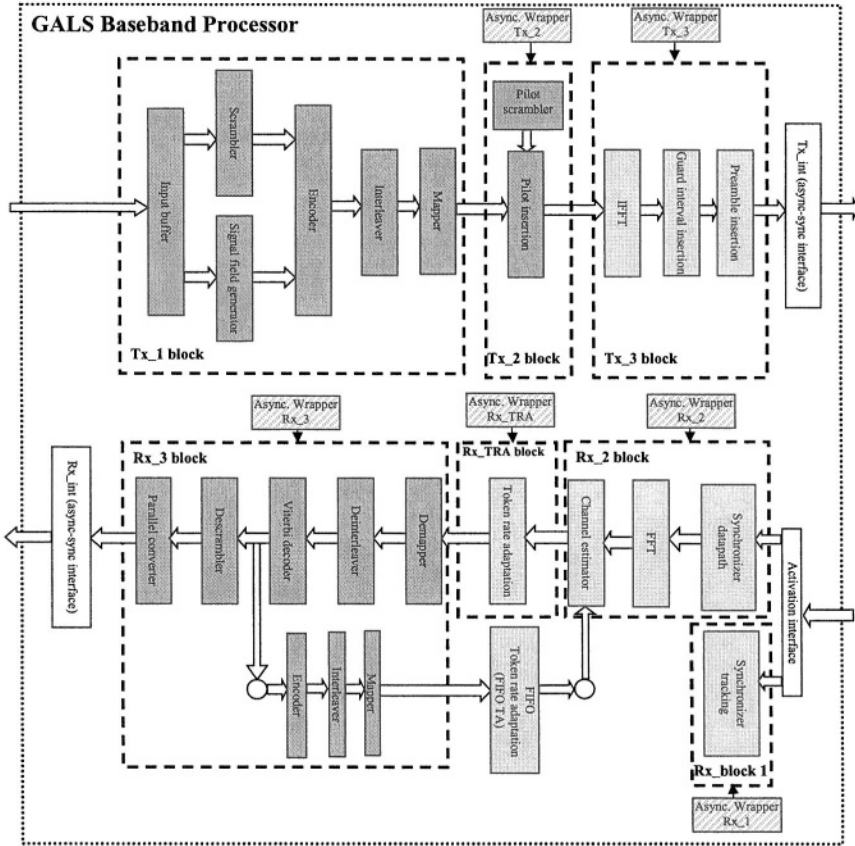


Fig. 2. GALS partitioning in baseband processor

In order to efficiently test the designed baseband processor, dedicated units based on Built-in Self-Test (BIST) were developed. The implemented BIST circuitry allows us to test both the global system and local block operation. However, the test approach is not in the focus of this paper and it will not be described in detail.

### 4 Design Flow and Implementation

The design flow for developing our GALS system is a combination of the standard synchronous design-flow with specific asynchronous synthesis tools. A graph of our design flow is shown in Fig. 3. All synchronous circuits are designed in VHDL and synthesized for our in-house 0.25 μm standard cell library using Synopsys Design Compiler. All asynchronous controllers are modeled as asynchronous finite-state machines and subsequently synthesized using the 3D tool [7]. 3D guarantees hazard-free synthesis of extended and normal burst-mode specifications. The logic equations generated by 3D are automatically translated using an own developed converter tool, called 3DC, into structural VHDL and subsequently synthesized. Two types of behav-



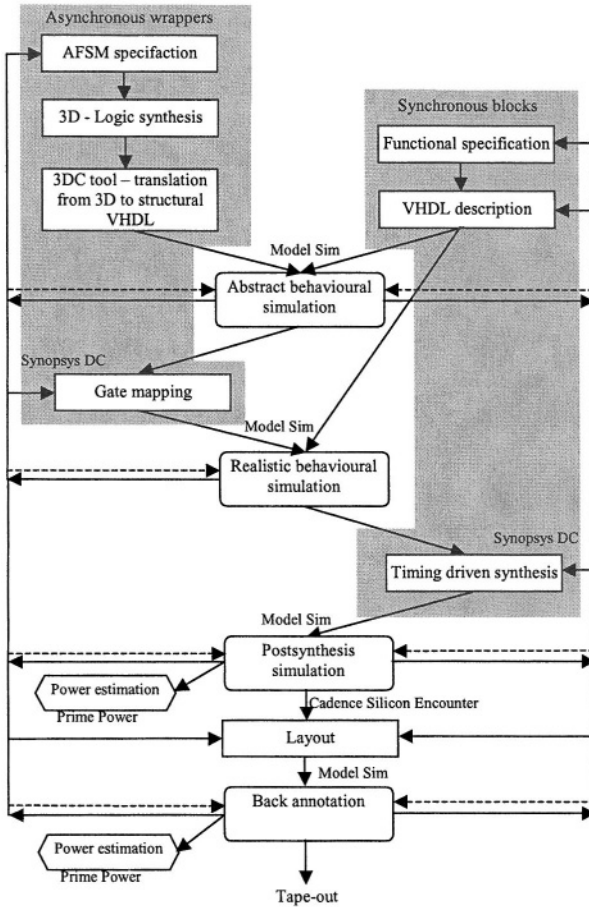


Fig. 3. Design flow for GALS circuits

ioral simulations are carried out. Initially functional verification is performed at the level of VHDL descriptions. This behavioral simulation is to confirm conceptual correctness of the system. Gate-level timing verification is performed after the abstract simulation, on the basis of the synthesized asynchronous parts and VHDL descriptions of the locally synchronous (LS) blocks. This simulation will with high probability prove the correctness of system operation. Nevertheless, it is executed very fast because all LS blocks are still modeled in behavioral VHDL. Subsequently a complete gate-level simulation is performed using both asynchronous and synchronous parts as synthesized netlists. After layout, back-annotation is performed. Behavioral, post-synthesis simulation and back-annotation are performed using a standard VHDL-Verilog simulator. Power estimation based on realistic switching activities is done using the Prime Power tool.

All asynchronous wrappers are fitted with a reset logic, necessary for initialization. The local clock generator for each GALS block is designed as a tunable ring oscillator, similar as described in [1]. The circuit area for the complete asynchronous wrap-

**Table 1.** Area and power distribution in GALS baseband processor

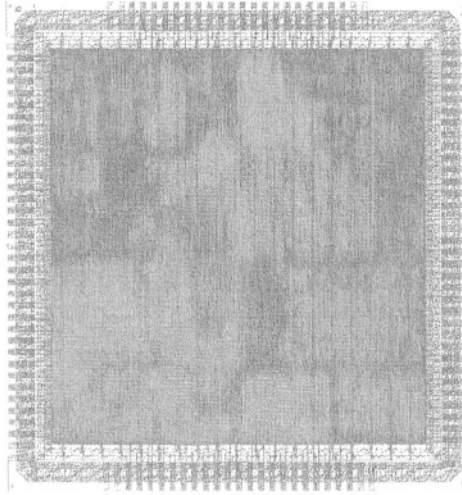
Component	Cell area [mm <sup>2</sup> , %]		Power [mW, %]	
Baseband chip	45,68	100%	151	100%
Sync. blocks	41,62	91,1%	125	82,7%
Rx_1 block	4,85	10,6%	17,7	11,7%
Rx_2 block	14,11	30,9%	40,4	26,7%
Rx_3 block	10,12	22,6%	17,7	11,7%
Rx_TRA	1,10	2,4%	0,9	0,6%
Tx_1 block	1,06	2,3%	23,1	15,3%
Tx_2 block	1,17	2,6%	1,6	1%
Tx_3 block	9,19	20,1%	23,6	15,6%
Asyn. blocks	0,93	2%	4,6	3 %
AW_Rx1	0,11	0,2%	1,2	0,8%
AW_RxTRA	0,13	0,3%	0,3	0,2%
AW_Rx2	0,13	0,3%	1,2	0,8%
AW_Rx3	0,13	0,3%	0,6	0,4%
AW_Tx2	0,13	0,3%	0,3	0,2%
AW_Tx3	0,28	0,6%	0,7	0,4%
Asyn. interface	0,73	1,6%	0,2	0,1%
Join	0,03	0,1%	0,2	0,1%
FIFO_TA	0,70	1,5%	0,0	0,0%
As-Sy interface	0,60	1,3%	13,5	8,9%
Rx_int	0,24	0,5%	6,4	4,2%
Tx_int	0,36	0,8%	7,1	4,7%
BIST	1,63	3,6%	6,6	4,3%
CBC	0,11	0,2%	1,5	1 %
TDE	1,23	2,7%	2,1	1,3%
TPG	0,29	0,6%	3	2%

per is equivalent to about 1.6 k inverter gates. From the complete wrapper, the largest area (1166 inverter gates) is needed by the tunable clock generator. Designing the clock generators as tunable circuits causes significant hardware overhead. The throughput of the GALS system after synthesis was determined by simulation. In our technology an asynchronous wrapper is operational up to about 110 Msp. This is well sufficient for our application and fast enough for moderate speed circuitry.

The complete system was verified using two different sets of test vectors. One is based on embedded BIST testing and the other one on a real transceiver application in the synchronous environment of a IEEE 802.11a modem. Results of the synthesis and power estimation are shown in Table 1.

It can be observed that synchronous functional blocks occupy more than 90% of the total area. The BIST circuitry requires around 3.6%, interface blocks (asynchronous interfaces and asynchronous to synchronous interfaces) 2.9% and asynchronous wrappers only 2% of the area.

Based on the switching activities, in the real transceiver scenario, power estimation with Prime Power has been performed. In the simulation example the baseband processor receives a frame of 43 bytes and then transmits a standard acknowledge frame of 14 bytes. Synchronous blocks are using most of the power (around 83%). Other important power consumers are asynchronous-to-synchronous interfaces with 13.5%, BIST 6.6% and asynchronous wrappers with 3%.



**Fig. 4.** Layout of the GALS baseband processor

Our GALS baseband processor is taped-out and it is currently under fabrication. The layout of the GALS processor is given in Fig. 4. The total number of pins is 120 and the silicon area including pads is  $45.1 \text{ mm}^2$ . After layout, the estimated power consumption is 324.6 mW. This represents a significant increase compared with the after-synthesis power estimation due to the clock-tree and pad insertions. We have also generated a pure synchronous version of the baseband processor that uses clock-gating as power-saving method. According to PrimePower, this synchronous baseband is consuming around 393mW in the same transceiver application. That means the GALSification leads to an additional power reduction compared to a synchronous implementation of around 17,4%. This result gives us hope that our GALS technique can be used as an effective power saving technique in complex digital systems.

## 5 Conclusions

The design process for GALSification of a complex IEEE 802.11a compliant WLAN baseband processor was discussed in this paper. The GALS technique used here is based on a request-driven operation with additional local clock generation. The challenging process of power-optimal GALS partitioning is described. The structure of our GALS baseband processor is complemented with several additional blocks necessary for efficient operation of the system. Finally, the design flow used for this system is described and implementation results are presented.

Even though the driving force for this work is the reduction of EMI in the context of a single-chip implementation, it is shown that the GALS implementation consumes less power than a functionally equivalent synchronous version.

## References

1. J. Muttersbach, *Globally-Asynchronous Locally-Synchronous Architectures for VLSI Systems*, Doctor of Technical Sciences Dissertation, ETH Zurich, Switzerland, 2001.
2. S. Moore, G. Taylor, R. Mullins, P. Cunningham, P. Robinson, Self Calibrating Clocks for Globally Asynchronous Locally Synchronous System, *In Proc. International Conference on Computer Design (ICCD)*, September 2000.
3. M. Krstić, E. Grass, New GALS Technique for Datapath Architectures, *In Proc. International Workshop PATMOS*, pp. 161-170, September 2003.
4. IEEE P802.11a/D7.0, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High Speed Physical Layer in the 5 GHz Band*, July 1999.
5. E. Grass, K. Tittelbach-Helmrich, U. Jagdhold, A. Troya, G. Lippert, O. Krüger, J. Lehmann, K. Maharatna, K. Dombrowski, N. Fiebig, R. Kraemer, P. Mähönen, On the Single-Chip Implementation of a Hiperlan/2 and IEEE 802.11a Capable Modem, *IEEE Personal Communications*, Vol. 8, No. 6, pp. 48 – 57, 2001.
6. J. Seizovic, Pipeline Synchronization, *In Proc. International Symposium of Advanced Research in Asynchronous Circuits and Systems*, pp 87-96, Nov 1994.
7. K. Yun, D. Dill, Automatic synthesis of extended burst-mode circuits: Part I and II, *IEEE Transactions on Computer-Aided Design*, 18(2), pp. 101-132, February 1999.

# TAST Profiler and Low Energy Asynchronous Design Methodology

Kamel Slimani, Yann Rémond, Gilles Sicard, and Marc Renaudin

TIMA Laboratory  
46, Av. Felix Viallet  
38031 – Grenoble CEDEX  
Kamel.Slimani@imag.fr

**Abstract.** In this paper, we present a tool for estimating the activity of asynchronous circuit specifications. This tool is well suited for monitoring how and where the activity is spread in a circuit. The quality and the precision of the results allow the designer to perform optimizations early in the design flow in order to improve the energy consumption and the speed of asynchronous circuits. Based on the TAST profiler, an energy optimization methodology is defined, focusing on micro-architecture decomposition, choice structures unbalancing and channels splitting. All these techniques are illustrated using an example.

## 1 Introduction

The steady evolution of integrated circuits in terms of surface, speed, energy consumption, electromagnetic emission is the result of a real investigation by research groups to develop tools to get information on the circuit behavior at different stages of the design flow. A better information on the circuit behavior results in a better implementation of it. An activity estimator of a circuit for a specific simulation is an excellent tool to observe early in the design flow the repartition of the switching activity in a circuit. So, optimizations can be brought as soon as possible in the design of a circuit to improve different parameters. Getting an activity estimation of a circuit is a good way: (1) to shrink the surface of a circuit by suppressing parts of a circuit which are never used (2) to increase the speed by doing judicious scheduling on the critical path (3) to reduce the energy consumption by optimizing the dynamic energy of parts which are most often used and (4) to decrease electromagnetic emission by spreading the activity of the circuit in the time.

Some commercial tools exist and are well used by designers to get these estimations on the activity of a circuit. These tools are commonly designed as code coverage tools. Mentor Graphics proposes in Modelsim an option –coverage [1] which allows the designer to get code coverage during a specific simulation. Actually, this tool is just a line counter of a specification described in VHDL or Verilog high level languages. A counter is associated to each line and is incremented when the line is executed. Synopsys proposes a similar tool for code coverage estimation, the name of this tool is CoverMeter [2]. Both these tools are based on a counter of lines, however other tools exist which proceed differently. Indeed, tools based on computing metrics instead of counter of lines are developed [3]. These commercial tools used for design-

ing synchronous circuits are very precise and interesting for functional high level description but does not reflect the activity of the final circuit since it gives information on effective work of a simulation. But, we all know that in the reality, the activity of a synchronous circuit is not only located on part of the circuit which does the effective treatment, other parts of the circuit which are not involved in the treatment are also active due to the use of a global synchronization by the mean of a unique signal: the clock.

In the asynchronous world, as far as the authors are aware, no activity estimator as it will be presented in this paper exists. Some estimators of power consumption exist; the pioneers in this research are the University of Utah [4] which proposes a technique for estimating power consumption of the Petri Net representation of asynchronous circuits. Some works have been done on energy estimation of speed independent control circuits [5] which take as intermediate representation the STG format. Other methods based on Production Rules were proposed to estimate the energy consumption of QDI asynchronous circuits [6].

These estimators are limited since on the one hand the estimation can be only applied on one particular kind of asynchronous circuit (SI, QDI etc.) and on the other hand the estimation is weighted by the executed instructions, thus we can not say that the part which consumes the most is systematically the part which represents the highest activity. So, these tools are really restricted to energy estimation and only energy optimizations can be achieved according to the results obtained by the estimator. Our estimator is more general, that is to say it can be used for a larger number of goals even if it has been specified for energy optimization.

## 2 Motivations

Basically, our motivation to set up an activity estimator tool is to design low power asynchronous circuits. We want to get information on the switching activity of a specification during simulations. We know that reducing the switching activity of a circuit reduces the dynamic energy of it. Indeed, the dynamic energy is directly related to the commutation of CMOS gates during an execution (charge and discharge of capacitors). This kind of information is useful to precisely identify parts of the circuit which are the most often solicited. Armed with this estimation, the designer precisely knows parts of the circuit that must be optimized to reduce the energy consumption of the circuit.

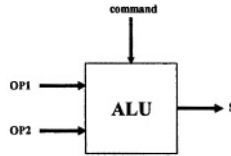
The estimator tool we have developed, namely the TAST Profiler tool, is integrated in the TAST design flow [7]. TAST stands for “Tima Asynchronous Synthesis Tool”, it is a complete asynchronous design flow developed by the CIS group at TIMA laboratory. The richness of this design flow claims to propose to the designer a variety of tools to facilitate the realization of complex and efficient asynchronous circuits.

## 3 TAST Profiler Tool Features

The TAST Profiler tool is an asynchronous estimator tool. It allows the designer to monitor the switching activity of a high level asynchronous circuit specification dur-

ing a particular simulation. The high level language used in TAST is a modified version of the CHP language. This language is based on VHDL and the original CHP language from Caltech. The CHP specification is compiled in an intermediate Petri Net format. The simulation of a CHP specification corresponds finally to the simulation of the intermediate Petri-Net format. From this simulation, the estimation of the activity is obtained. The results obtained are rich and precise, they are presented in different forms according to the option chosen by the user. All the commands and options proposed by the TAST Profiler tool have been set up to fulfill the need of the designer to get the useful information that can help him to achieve efficient circuits. It is effectively possible to observe the code coverage, to trace particular variable and channel, or to establish statistics on the execution.

We choose an ALU to describe the options of our TAST profiler (Figure 1) and to introduce the language.



**Fig. 1.** ALU component

The ALU is a component which takes two operands *op1* and *op2*, manipulates these operands with an arithmetic or logical operator depending on the *command*, and generates a result *S*. The corresponding CHP code is written in Figure 2.

A module in CHP is described as a top component with its associated input (*command*, *OP1*, *OP2*) and output (*S*) ports. The ports use channel to communicate with other components, these channels are of multi rail type (MR[B][L]) where *B* is the radix and *L* is the length. The component is composed of processes; the ALU component contains only one process named *P\_ALU*. A process has almost the same meaning as process usually used in VHDL language. Nevertheless in CHP, parallelism can be used inside a process whereas in VHDL the process contains only sequential instructions. The sequence operator is a semi-colon and the parallel operator is a comma. The CHP language constructs are listed below:

“[]” represents a choice with only one guard which is always true. Generally, this instruction represents the enclosure of a whole process.

“@ []” is the deterministic choice structure. This choice structure must have more than one guard, and only one guard can be valid at a time (property of mutual exclusion).

“@ @ []” is the non deterministic choice. It is in fact a structure including several guards (equal or superior than 2) and many guards can be true. The structure has to take a decision to choose one guard among the true guards. There is no property of mutual exclusion to respect for this instruction.

“!” is the send instruction. It emits a data on a channel.

“?” is the dual of the previous instruction, it represents the receive instruction, it allows the reception from a channel.

“:=” is the assignment of a variable inside a process.

And finally “skip” is the instruction that does nothing, it just passes.

```

Component ALU
Port ( command : in DI MR[2][3];
      OP1      : in DI MR[2][16];
      OP2      : in DI MR[2][16];
      S        : out DI MR[2][16]
    );
const cont_or : MR[2][3] := "0.0.0"[2];
const cont_and : MR[2][3] := "0.0.1"[2];
const cont_plus : MR[2][3] := "0.1.0"[2];
const cont_minus : MR[2][3] := "0.1.1"[2];
const cont_sll : MR[2][3] := "1.0.0"[2];
const cont_srl : MR[2][3] := "1.0.1"[2];

begin
process P_ALU
Port ( command : in DI MR[2][3];
      OP1      : in DI MR[2][16];
      OP2      : in DI MR[2][16];
      S        : out DI MR[2][16]
    );

begin
variable cmd : MR[2][3];
variable op1_var, op2_var : MR[2][16];

[ command ? cmd;
  @[ cmd = const_or =>
    op1 ? op1_var, op2 ? op2_var;
    S ! (op1_var or op2_var)
    cmd = const_and =>
    op1 ? op1_var, op2 ? op2_var;
    S ! (op1_var and op2_var)
    cmd = const_plus =>
    op1 ? op1_var, op2 ? op2_var;
    S ! (op1_var + op2_var)
    cmd = const_minus =>
    op1 ? op1_var, op2 ? op2_var;
    S ! (op1_var - op2_var)
    cmd = const_sll =>
    op1 ? op1_var, op2 ? op2_var;
    S ! sll(op1_var, op2_var)
    cmd = const_srl =>
    op1 ? op1_var, op2 ? op2_var;
    S ! srl(op1_var, op2_var)
  ]
loop;
];
end; -- process
end; -- component

```

Fig. 2. CHP code of the ALU

So, our ALU component is a choice structure with 6 guards corresponding to 6 arithmetic and logical operators (or, and, plus, minus, shift left logical, shift right logical). The command selects the right operation to perform, OP1 and OP2 are manipulated depending on the operator and finally a result is generated and sent through the port S.

### 3.1 Code Coverage

The activity estimation tool allows the designer to achieve a profiling of the code to get code coverage information of a CHP description. The aim of this code coverage command is to identify parts of a description which are never used during a particular simulation. The information of unexecuted code is written in a file to be directly visible by the designer. If a line of code is never executed, the resulting file is as follow:

path.process_name	instr	ref
-------------------	-------	-----

Where path.process\_name is the exact path to identify the process inside which the instruction is never executed. "Instr" identifies the instruction never used, and "ref" gives column and line information of the instruction never used.

In the ALU, for example if the guard "cmd=const\_srl" is never used, the code coverage command generates a file which contains the following information:

alu.p_alu.G1.G6	op1?op1_var[15..0][0]	(L=68, C=37)
alu.p_alu.G1.G6	op2?op2_var[15..0][0]	(L=68, C=52)
alu.p_alu.G1.G6	s!(op1_var[15..0][0] srl op2_var[15..0][0])	(L=68, C=66)



The guards inside a structure of choices are by default labeled with the number of the guard. The first guard is labeled G1, the second G2 etc. If the instruction never used is inside a guard of a choice structure, the identification of this instruction includes the guard labeled to precisely identify the instruction. In our example, three instructions are written in the code coverage file because the guard G6 of the structure of choice is never used and this guard contains the three instructions written in the file.

### 3.2 Trace

The trace command of the TAST profiler tool aims at observing the behavior of a (several) channel(s) or a (several) variable(s).

- Channels

The first option of this trace command allows the designer to precisely analyze the behavior of one or several channel(s) in a circuit. To see the number of time it is activated and the values of data it receives or it emits. This information is written in a Trace file which can be analyzed by the designer to check the behavior of a particular channel.

This trace command goes ahead by offering to the designer the possibility to see the statistics of the behavior inside a channel, that is to say to examine the behavior of the separated digits inside one channel. The designer can observe which wires of a channel are the most used or are never used. The results of these statistics are presented in Figure 3 for the command channel in the ALU component, that is to say a channel composed of three digit dual rail vector. The statistics computed by this tool show that the wire 0 of the three digits is more often used than the wire 1. That can be a good information of the activation of wire in a channel for future optimizations.

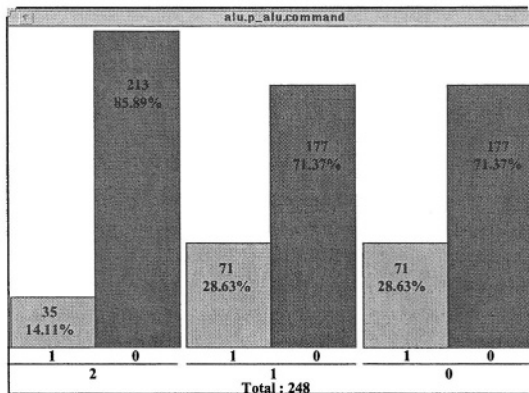
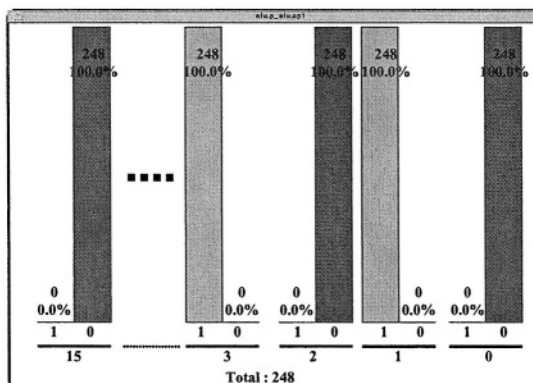


Fig. 3. Statistics on digits inside a channel

- Variables

The option of variables follows exactly the same reasoning as what we explained previously on channels. Here, the treatment is done on variable instead of channel, but



**Fig. 4.** Statistics on digits inside a variable

the results obtained by the tool are of same type. We get a file which establishes the whole behavior of one or several variable(s) and statistics are produced on the behavior of the digits inside a variable as shown in Figure 4. This figure shows the behavior of the variable OPI\_var. This variable receives the channel OPI, we see that the digits 1 and 3 receive the value 1 all the time, and the other digits receive the value 0. We will see later that this kind of results can be used to optimize a circuit.

### 3.3 Activity

The activity estimation of a circuit is a relevant information to observe the activity repartition in the circuit. The aim of this information is to see how the activity of a circuit is spread for a given simulation (representing the application of the circuit) and to identify parts of a circuit which are the most frequently used. The results are given in form of statistics. To make relevant comparison, the statistics are computed in different natures explained hereafter.

- **Instruction Types**

The first kind of statistics the designer can get is statistics on instructions. Inside a process, statistics are made to see how the instructions are solicited. The results are represented by a histogram which establishes the number of time the instructions are executed and gives statistics on the execution of these instructions among all existing instructions. Figure 5 shows the results of statistics given by the simulation of the process P\_alu of our ALU component.

We observe that the process P\_alu contains neither non deterministic choice nor assignment. The reception instruction “?” is the most often used.

- **Choice Structures**

In a choice structure including many guards, it is interesting to know the statistics of the execution of every guard. The unique process in the ALU is a structure with 6 guards, we can write it in a pseudo CHP code as shown in Figure 6.

G1, G2, G3, G4, G5 and G6 represent the guards of the choice structure; I1, I2, I3, I4, I5 and I6 the respective arithmetic and logical instructions. For a given simulation, the results of the statistic tool have the shape shown in Figure 6. On this histogram,

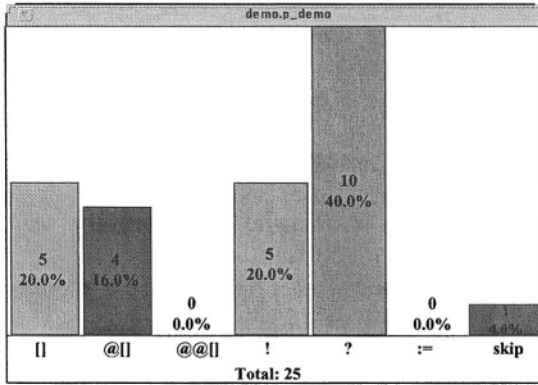


Fig. 5. Statistics on instructions

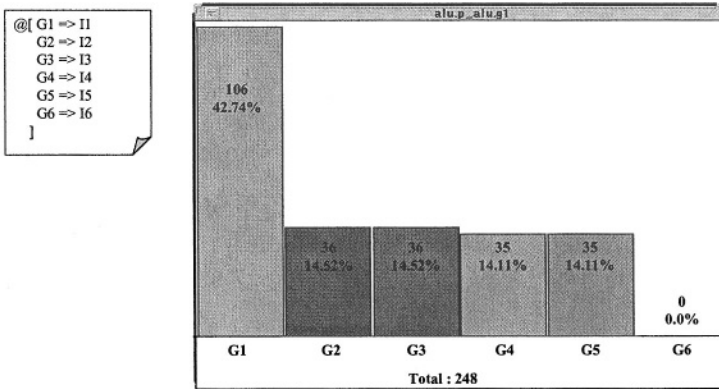


Fig. 6. Statistics on choice structures

useful information is visible. One can see that the guard G6 is never used and the guard G1 is the most often used. The other guards have almost the same probability of execution. We will see later that the designer can exploit this information to optimize a circuit.

The general approach of the TAST Profiler tool has been presented in this section. It is possible to refine the information depending on what useful information the designer wants to get. The CHP developed at TIMA allows the designer to insert label in the CHP programs, hence he can observe particular parts of the code.

## 4 Energy Optimization Methodology

The TAST profiler presented previously is added to the TAST design flow in order to precisely monitor the switching activity in a circuit during a given simulation. The information produced by the TAST profiler tool allows the designer to bring optimizations in several ways. We present 3 examples of optimization: the first one is on a

general micro-architecture optimization, then we show how to optimize a channel and finally an optimization on choice structure is presented.

#### 4.1 Micro-Architecture Optimization

When a component or a process is often used during a simulation, it is interesting to bring optimizations on this component or process to reduce its switching activity, consequently reducing its energy consumption. A way to optimize a process is to decompose it into a set of sub-processes, thus the complexity of a given computation is reduced. Only the sub-processes which are involved in the treatment are activated, the sub-processes which do nothing are not consuming. For the ALU component, the decomposition result is shown in Figure 7.

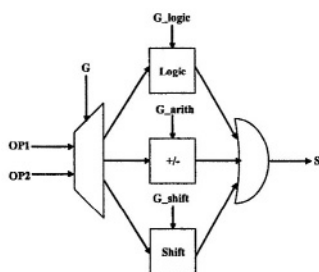


Fig. 7. Process decomposition

Each block on the figure represents a process. Each process is reduced at the lowest complexity. If a logical operation is performed, only the path involved in the treatment of the corresponding logical operation has an activity. Moreover, the first multiplexer is now composed of three guards instead of the original 6 guards multiplexer. This multiplexer is used to orientate the operands OP1 and OP2 to the right arithmetic and logical block (The corresponding command has consequently a smaller channel length). The multiplexer can be optimized to privilege the case where operands are sent to the logical block which represents the highest probability (see Sect. 4.3). New commands to the arithmetic and logical block have been added, these commands are sent only if the corresponding block is involved in the treatment. Moreover, these commands are just dual rail commands since there are 2 operators per block: two logical operators (or, and), two arithmetic operators (plus, minus) and two shift operators (sll, srl). For this example, the activity is reduced at its minimum. It has been achieved by the knowledge of the repartition of the activity thanks to the TAST Profiler tool.

#### 4.2 Channel Optimization

Information on channels is useful to choose the best encoding. The activity of digits in a channel is interesting to see which digits of a channel are rarely or never used. According to this analysis, the channels can be sliced into a set of sub-channels that are conditionally active (channel width adaptation). Therefore, instead of sending all the

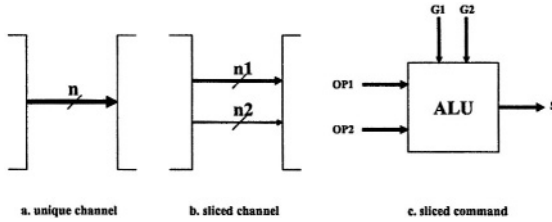


Fig. 8. Channel optimization

time  $n$  digits through a channel, it is better to slice the channel in many sub-channels with lower lengths ( $n_1$  and  $n_2$  for example) as shown in Figure 8-a and Figure 8-b.

If we consider Figure 4 which shows the statistics of the variable  $OP1\_var$  which receives the channel  $OP1$  (NB: the statistics of the channel  $OP1$  give exactly the same results since the  $OP1\_var$  receives only  $OP1$  channel). According to this information, a better implementation of the channel  $OP1$  is to slice it into two sub-channels  $n_1$  (four low significant digits) and  $n_2$  (the other digits). Consequently, only the four low significant digits can be sent because only these digits are useful. So, we cancelled the activity of the other wires.

This kind of statistics is also useful for restructuring the command channel (only the needed digits of a command are sent to a module) (Figure 8-c).

### 4.3 Choice Structure Optimization

Statistics on choice structures are essential to privilege cases with the highest probabilities. The statistics obtained can guide the designer to unbalance the structure of choice in favor of the cases with the highest probabilities [8]. For example, consider Figure 6 which shows the statistics of the execution of the 6 guards  $G_1, G_2, G_3, G_4, G_5$  and  $G_6$  of the ALU component. We can see that  $G_6$  is never executed and  $G_1$  has the highest probability, so the CHP code can be modified as shown in Figure 9-a. The initial choice structure is now replaced by a hierarchical structure. Guard  $G_6$  has been suppressed because it is never used. In addition, the structure has been unbalanced to privilege case  $G_1$ . The effect on the final circuit by decomposing the structure in that way is to reduce the whole energy consumption by suppressing guard  $G_6$  (choice with 5 guards instead of 6) and by privileging the execution of the most probable guard  $G_1$ . Indeed, every time guard  $G_1$  is executed, it costs the energy of a two-input multiplexer instead of a structure with 5 inputs. Figure 9-b shows the initial circuit. The resulting circuit after optimization is shown in Figure 9-c.

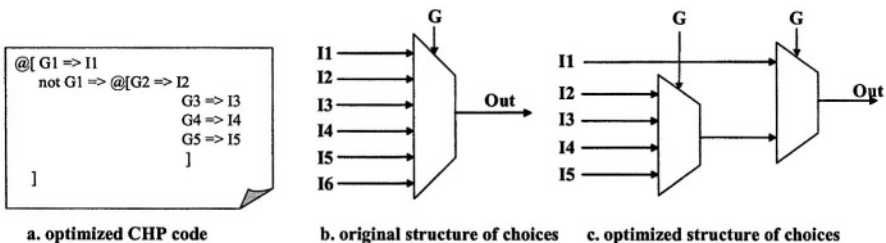


Fig. 9. Choice structure optimization

A complex circuit including complex choice structures having unequal probabilities results in large energy savings. Note that because there is no global clock, the speed is increased at the same time.

## 5 Conclusion

The TAST profiler provides the designer a tool to perform activity analysis of high level programs. It is shown how the TAST profiler features enable to apply an energy optimization methodology early in the design cycle. It is noticeable that optimizing the energy of an asynchronous specification is also improving its speed. Apart from energy optimization, the tool we designed can also be used to estimate other criterion than energy such as speed, area and electromagnetic emission.

## References

1. [http://www.model.com/products/tours/58/code\\_coverage\\_web/code\\_coverage\\_web.htm](http://www.model.com/products/tours/58/code_coverage_web/code_coverage_web.htm)
2. <http://www.covermeter.com>
3. Fallah-F; Devadas-S; Keutzer-K. "OCCOM-efficient computation of observability-based code coverage metrics for functional verification". *IEEE-Transactions-on-Computer-Aided-Design-of-Integrated-Circuits-and-Systems*. Vol.20, no.8; Aug. 2001; p.1003-15.
4. Prabhakar Kudva and Venkatesh Akella. A technique for estimating power in self-timed asynchronous circuits. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 166-175, November 1994.
5. P. A. Beerel, C.-T. Hsieh, and S. Wadekar. Estimation of energy consumption in speed-independent control circuits. In *International Symposium on Low-Power Design*, pages 39-44, 1995.
6. Paul I. Péntzes and Alain J. Martin. An energy estimation method for asynchronous circuits with application to an asynchronous microprocessor. In *Proc. Design, Automation and Test in Europe (DATE)*, pages 640-647, March 2002.
7. <http://tima.imag.fr/cis/> TAST Project
8. The Energy and Entropy of VLSI Computations. Jose A. Tierno, Rajit Manohar, and Alain J. Martin. *Async96 Second International Symposium on Advanced Research in Asynchronous Circuits and Systems*, March 1996.

# Low Latency Synchronization Through Speculation

D.J. Kinniment and A.V. Yakovlev

School of Electrical and Electronic and Computer Engineering, University of Newcastle,  
NE1 7RU, UK

{David.Kinniment,Alex.Yakovlev}@ncl.ac.uk

**Abstract.** Synchronization between independently clocked regions in a high performance system is often subject to latencies of more than one clock cycle. We show how the latency can be reduced significantly, typically to half the number of clock cycles required for high reliability, by speculating that a long synchronization time is not required. The small number of synchronization metastability times longer than normal are detected, and subsequent computations re-evaluated, thus maintaining the reliability of the system.

## 1 Introduction

Systems on silicon are increasingly designed using IP blocks from different sources. These blocks are optimised to operate at a particular clock rate and there may therefore need to be several different clocks in the system. If the clocks are all phase locked, there is no need for re-synchronisation of data passing between from block to another, since data originating in one clock domain and passing to the next will always arrive at the same point in the receiving clock cycle. Even if the phase difference is not the same, it may not be known until the clock has run for some time, but it is then predictable, and a suitable data exchange time can be found, [1].

In practice it is difficult to achieve accurate and reliable locking between all the clock domains for a number of reasons.

- Jitter and noise in the clock generation circuit, particularly if two different frequencies must be locked.
- Crosstalk between the data and the clock tree connections introducing noise into both.
- Dynamic power supply variations between different parts of a clock tree and the data distribution, affecting the path delay difference.
- Temperature changes may alter the relative phase delay of two clock trees designed in different ways.
- Input output interfaces between the system and the outside world are not controllable and phase relationships cannot be predicted.

These effects cause unpredictable variation in the time of arrival of a data item relative to the receiving clock, which is particularly noticeable in high performance systems using IP blocks with large clock trees [2]. Figures of 150ps noise [4] and 110ps clock skew [3], have been reported in 0.18 $\mu$  systems, and this is likely to increase with smaller geometries. Design of interfaces in high performance systems with fast

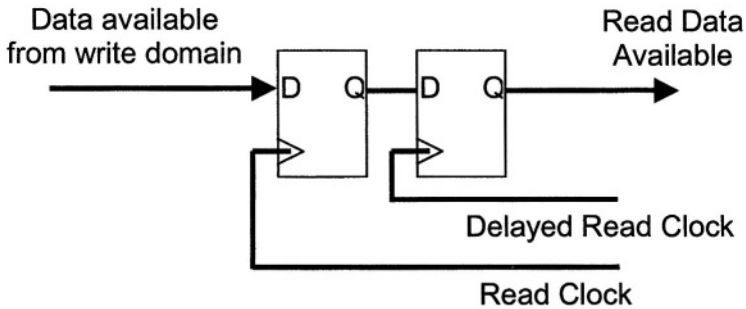


Fig. 1. Synchronizer

clocks and large timing uncertainties becomes more difficult as these uncertainties increase as a proportion of the receiving clock cycle and it may be simpler to assume that the two domains are independent.

Exchange of data between two independently clocked domains requires the use of a synchronizer. This ensures that data originally synchronized to the writer's clock is synchronized to the reader's clock so that it arrives at the correct time in relation to the reader's computation cycle. Conventionally this is done with the circuit of Fig. 1, [5].

Here metastability may occur in the first flip-flop, caused by the asynchronous nature of the write data available signal and the read clock. This is allowed to settle for a fixed amount of time before being clocked into the second flip flop by the later delayed read clock with the aim of obtaining a clean read data available signal. The reliability of the synchronizer depends critically on the time allowed between the read clock and the delayed read clock, which determines the time allowed for metastability to settle. In terms of the metastability time constant,  $\tau$ , of the flip-flop, increasing the

delay by a time  $t$  increases the MTBF by a factor of  $e^{\frac{t}{\tau}}$ , and in order to keep system failures to less than 1 in  $10^8$  s (2 years) the total settling time may need to be over  $30\tau$ . Since  $\tau$  is comparable to a gate delay,  $30\tau$  may be equivalent to several clock cycles, if  $\tau = 50\text{ps}$  then a single clock cycle at 1 GHz is insufficient. Using several clock cycles for synchronization can lead to unacceptably large latencies because the data cannot be read until the read data available is asserted. While throughput can be increased by using an intermediate FIFO to separate the read and write operations, the total time between generation of the data and its availability in the receiver cannot. As a result, considerable latency is added to the data transmission between separately clocked zones, and this latency can be a problem, particularly in long paths where a computation cannot proceed until a requested result is received.

By allowing the read data to be used as soon as metastability is complete, fully asynchronous systems, or those with stoppable clocks allow the data transmission latency to be closer to the average metastability time rather than always using that required to accommodate the worst metastability time [6]. We will show that it is also possible to reduce the synchronization latency in a fully synchronous system, by assuming that the synchronization time does not normally need to be long, but allowing



for recovery from any errors if an event with a long metastability time is encountered. In section 2 we describe how a FIFO is normally used to achieve high throughput, in section 3 how possible errors can be detected, and in section 4 we show the operation of a synchronizer circuit with error detection. Finally a simple system is described in section 5 where error detection and recovery is combined with a FIFO to give high throughput as well as lower than normal latency.

## 2 High Throughput Synchronization

Synchronization of the data available signal and any acknowledgement of the reading of the data may require a number of clock cycles, and therefore there can be a delay between the write data becoming available and the actual reading of the data, and also between the acknowledgement of the reading and the writing of the next data item. This leads to low throughput, but inserting a FIFO between the writer and the reader as shown in Fig. 2, [7], allows both reading and writing to take place immediately if the FIFO is always partially filled, and always has some space available for a new write. These conditions are indicated by the two Flags Full, which is true if there is only sufficient space to accommodate a certain number of data items and Not Empty, which is true if there are at least a certain number of unread items in the FIFO. Provided there are always at least  $n$  data items in the FIFO, the delay of  $n$  cycles required to synchronize the Not Empty signal can be overlapped with the reading of these  $n$  items. Similarly with  $n$  spaces the writes can also be overlapped. Unfortunately this does not improve the latency, because the time between writing an item, and reading it is at least  $n$  cycles of the read clock plus between 0 and 1 cycle for the Not empty signal to appear as a Request. Similarly between  $n$  and  $n + 1$  cycles of the write clock are needed to recognise that the FIFO needs another item. Forward latency can only be reduced by reducing the time needed to synchronize the Not Empty signals, and reverse latency by reducing the Full synchronization time.

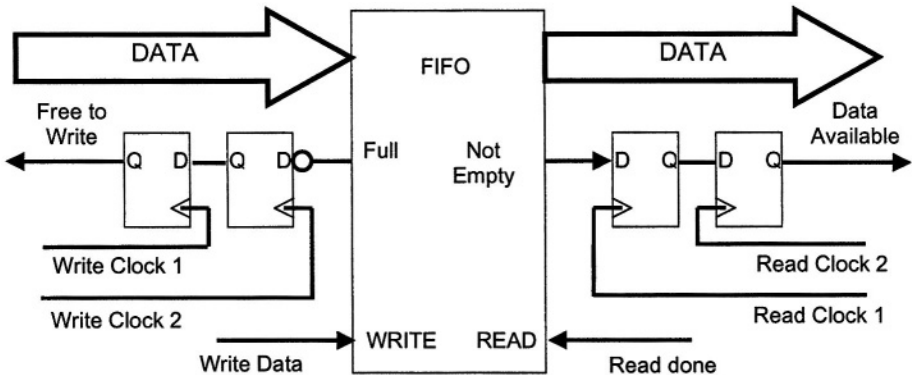


Fig. 2. Synchronizer with FIFO

### 3 Synchronization Error Detection

If the synchronization delay between the read clock and the delayed read clock is reduced the system performance is improved, but the probability of failure for each data transmission will be much higher, for example for a  $10\tau$  delay it would be 1 in 20,000. However if it were possible to later detect the cases where synchronisation failed, and recover by returning to the system state before failure, the performance of the system could be improved with little penalty in reliability. Note that this proposal does not eliminate the metastability problem, because there will still be cases where metastability lasts longer than the failure detector allows, and the system state will not be recoverable.

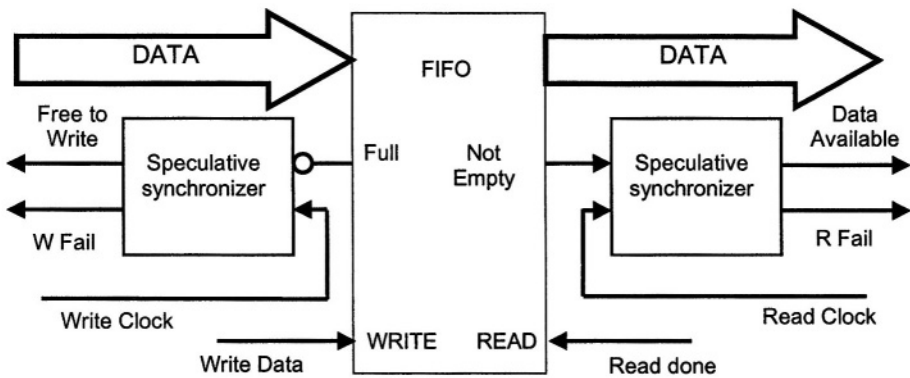


Fig. 3. Low latency synchronization

Fig. 3 shows how this can be done by replacing both conventional n-cycle synchronizers with speculative versions in which the data available, or Free to write signals are produced early, and allowed to proceed if they subsequently prove to be correct. However, if there is any doubt about their validity at a later time, the R Fail or W Fail flag is raised so that computation can be repeated.

In the speculative synchronizer the first flip-flop must be tested to detect whether it has resolved in a short time or whether it takes longer than average. In order to achieve this we must use a metastability filter on its output as shown in Fig. 4. This circuit, sometimes known as a Q-Flop [8], ensures that half levels will not appear at the output, and the uncertainty due to metastability appears (in the case of the read synchronizer) as a variable delay time from the synchronized write data available to the read data available output.

In the Q-Flop only clean level transitions occur, because the output is low during metastability, and only goes high when there is a significant voltage difference between Q and Q'. It can be shown that the transition between a voltage difference of  $V_1$

and  $V_2$  takes a time  $\tau \cdot \ln \frac{V_2}{V_1}$ , so when the voltage between Q and Q' is sufficient to produce an output (more than the p-type threshold,  $V_p$ ) the transition to a full output of  $V_{dd}$  will take less than  $2\tau$ . This voltage difference is also much greater than the flip

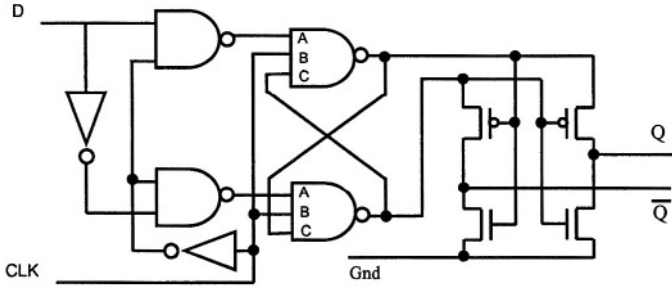


Fig. 4. Q-Flop

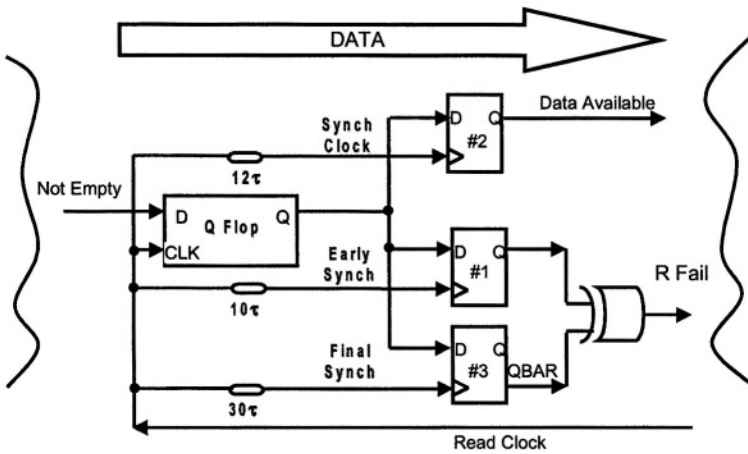


Fig. 5. Speculative synchronizer

flop internal noise level, so when the read data available signal leaves the metastable region to make a high transition, it happens in a fixed time and cannot return low. On the other hand, the time at which this transition takes place cannot be determined, and may be unbounded.

If we use further timing signals derived from the read clock to sample the read data available at  $10\tau$ ,  $12\tau$ , and  $30\tau$ , as shown in Fig. 5, it is possible to determine whether the partially synchronized read signal changed between  $10\tau$  and  $30\tau$ .

We will assume that at  $30\tau$  all the samples are stable, an assumption that is true to a high degree of probability. There are then only four combinations of the three samples to consider since if the  $10\tau$  sample has reached a high level, the  $12\tau$  and  $30\tau$  samples must also be high. Similarly if the  $12\tau$  sample has reached a high level, the  $30\tau$  sample must also be high. These four combinations, together with the possibility of metastability at  $30\tau$  in Table 1.

The probability of the  $30\tau$  sample being metastable, is  $e^{-30}$  and will be discounted. If the  $10\tau$  and  $30\tau$  samples are both 0, we can infer that there was no data available signal, and the output at  $12\tau$  was correctly flagged as 0. If the  $10\tau$  and  $30\tau$  samples

**Table 1.** Synchronizer possibilities

$10\tau$	$12\tau$	$30\tau$	Fail	Comment
0	0	Metastable	?	Unrecoverable error
0	0	0	0	No data available
0	0	1	1	Change between $10\tau$ and $30\tau$ , return to original state
0	1	1	1	Change between $10\tau$ and $30\tau$ , return to original state
1	1	1	0	Normal data Transfer

are both 1, we can infer that there was a data available signal, and the output at  $12\tau$  was correctly flagged as 1. The only other possible case is that the  $10\tau$ , and  $30\tau$  samples are 0 and 1 respectively. In this case the output of the Q Flop changed some time between  $10\tau$ , and  $30\tau$  and the  $12\tau$  value is unreliable. The fail signal is then set and the data transfer must be repeated. The MTBF for this arrangement is the same as that of a  $30\tau$  synchronizer, because even if the  $10\tau$  sample is metastable, the chances of this are 1 in  $e^{10}$ , and further  $20\tau$  is allowed for the metastability to settle. Given that it is metastable, chance of it remaining metastable at  $30\tau$  is 1 in  $e^{20}$ , and the overall probability is 1 in  $e^{10} \cdot e^{20} = e^{30}$ .

## 4 Circuit Operation

To verify the operation of the circuit of Fig. 5 we simulated it using SPICE to model the TSMC 0.18 $\mu$  process. We used inverter chains to implement the delays, because both the value of  $\tau$  and the delay of an inverter are related to the same circuit parameters, and are likely to track approximately with power supply variations and temperature. Trial and error simulations of the Q-flop constructed of standard gates as shown in Fig. 4 gave a  $\tau$  of about 53ps, and typical delay of 280ps in the circuit. With the D input 1.97ps ahead of the Read Clock input the metastability time became longer than  $12\tau$ , and with the clock data overlap increased to 2.15 ps, the delay was 600ps (280ps delay +  $6\tau$ ). With a clock period of 1ns, the probability of getting an overlap within 0.18 ps of the metastable point would be less than once in 5000 events.

Fig 6 shows the Read Clock to the Not Empty Q-flop is on the bottom plot at 100ps, and two delayed version of this clock., one 540ps after the clock (Early Synch, derived from a chain of 14 inverters) and one at 630ps, (Synch Clock from 16 inverters). The output from the Q-flop goes high at around 600ps (middle trace), so that it is low for the first Early Synch signal, but is high for the Synch Clock. The final Data Available output is shown on the top trace and goes high at around 1ns. This is the case represented by the penultimate line in Table 1, where a Fail signal would appear at  $30\tau$  after the Final Synch signal, a delay of about 1.9ns, since the 540ps sample is low, and the 1.5ns sample is high. Latency is therefore about 900ps for the Data Available signal. For a standard synchronizer with  $30\tau$  reliability the latency would be  $280 + 30 \times 53 = 1870$ ps.

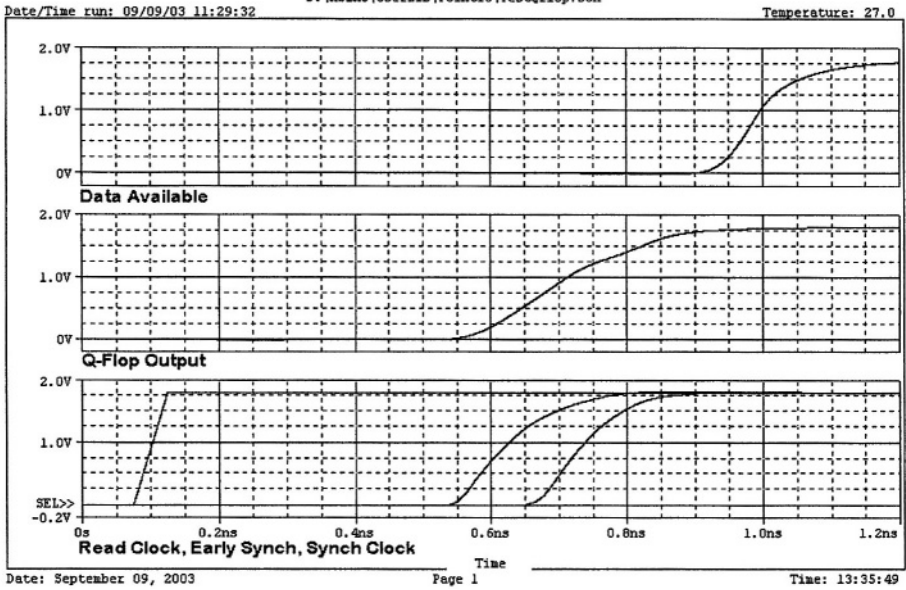


Fig. 6. Standard gate implementation

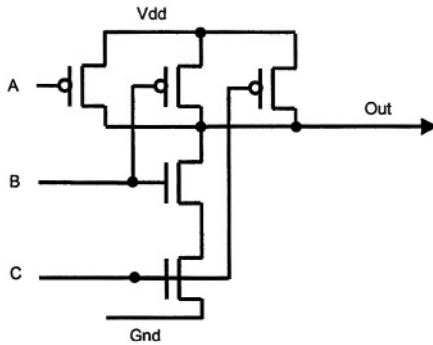


Fig. 7. Modified 3 input NAND gate

A value for  $\tau$  of 53ps is significantly greater than that achievable with a  $0.18\mu$ , but it is possible to produce a Q-flop with a  $\tau$  comparable with state of the art designs. We redesigned the Qflop by reducing the input coupling to the flip flop formed by the three input NAND gates of Fig 4. In the circuit of Fig. 7 just two small p types transistors are used to provide the A input, and some of the other transistors are resized to optimise the feedback from the C input. If B and C were both high at the same time that A was low, there could be a short circuit, but that cannot happen, except transiently, in the Q-flop because both A's must be high when the B inputs are high. This modification gives a  $\tau$  of about 26ps (half the previous value). The times for each of the delayed clocks can now be reduced, and in this case we use 10 inverters for Early Synch (379ps from the Read Clock) and 12 inverters for the Synch Clock (491ps from the Read Clock).

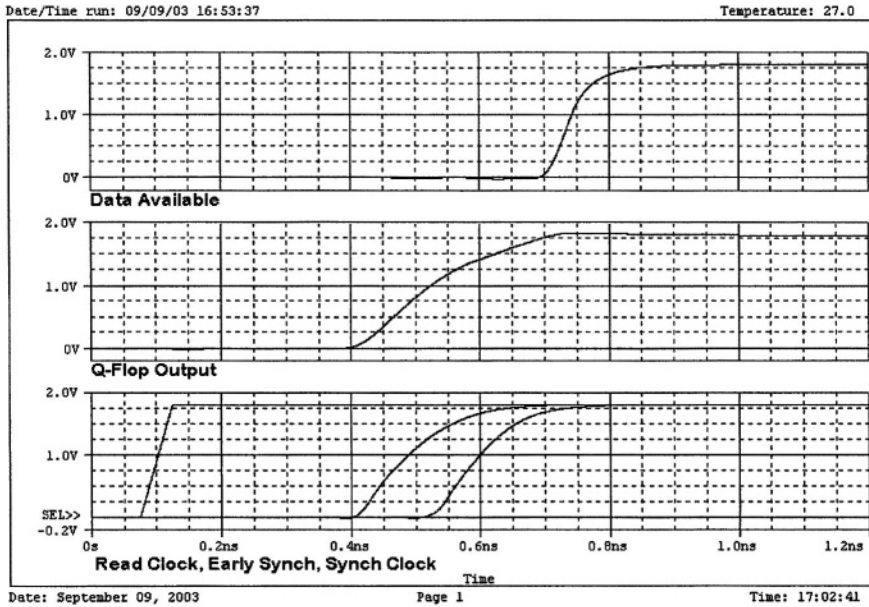


Fig. 8. Synchronizer with modified gate

This is shown in the SPICE plot of Fig. 8. A Q-flop output response at 511ps (411ps from the Read Clock) is again chosen to illustrate the Fail situation. Responses longer than 379ps may fail, and this occurs in about 1 in 5000 cases, because the Early Synch transition is set at 379ps, and the delay in the Q-flop is about 200ps leaving 179ps, or  $7\tau$  approx for metastability. Total latency is about 650ps for the Data Available signal, compared with  $200 + 30 \times 26 = 980$ ps in a conventional synchronizer.

## 5 Recovery

A clock period of  $20\tau$  would allow synchronization latency of between 1 and 2 clock cycles, but the fail signal is not generated until midway in the second cycle, so it is necessary to duplicate the Q-flop so that metastable events created in one cycle are preserved until the second. This is done with the arrangement of Fig. 9, where a data available is generated within one cycle, but the fail signal corresponding to that Data Available appears in the following cycle.

To show how recovery can be achieved after a synchronization failure, we give a simple example in Fig. 10. Here input data from the writer is normally clocked into the input register, and then added to the R register which has a slave refreshed whenever the input register is overwritten. Read done is signalled back to the synchronizer during the next cycle by the flip-flop B which is clocked by a delayed read clock sufficiently far before the following cycle to enable the read pointer to be updated.

We will assume that the receive clock period is  $20\tau$ , that the data available signal appears at  $12\tau$  in the first cycle, and that a fail signal does not arrive until middle of the second cycle at  $30\tau$ .

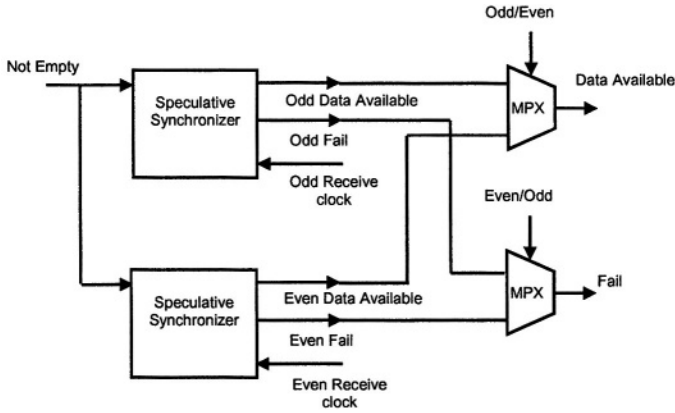


Fig. 9. Pipelined speculative synchronizer

If there is a failure of synchronisation such that the data available signal is still metastable at the receive clock time, the input register, the slave, and the flip-flop A may all be corrupted. At  $30\tau$  the fail signal will force the flip flop B to be set to 0, and the R register master will not be altered. Since the read done is not generated, the read pointer is not updated, and the data available is effectively allowed another cycle to settle before both the input and R register slave are refreshed.

In this scheme only one extra cycle is required every time a synchronization failure occurs, and provided the rate of writing is slower than the rate of reading this extra delay will rapidly be made up. Average latency is now  $0 - 1$  read clock cycle for the time between movement of the write pointer which may alter the state of the Not Empty signal, and the next read clock cycle, plus 1 cycle for synchronisation plus 1 in  $e^{10}$  cycles for possible failures.

This is a relatively simple example, but in general, our strategy is to stop a pipeline when the failure is detected, and recompute or refresh corrupted values.

It is important here to ensure that corrupted data, or control states are always recoverable, but this does not usually present a problem. This is similar to schemes in which a fast, but unreliable computation path may need to be backed up by a slower, but reliable one [9]. In more complex systems, or those in which failure cannot be detected in one cycle more than the read, every register that may be altered during a calculation dependent on correct input data is replaced by a stack. This stack needs to be of sufficient depth to cover the number of clock cycles between the initial read clock, and the final sample ( $30\tau$ ), then the following strategy will ensure recovery:

- a. Push each new value on to the head of each stack.
- b. If failure occurs pop the stacks by a number sufficient to recover the old values, and repeat the transfer.

This procedure may take several clock cycles, but only need occur very infrequently.

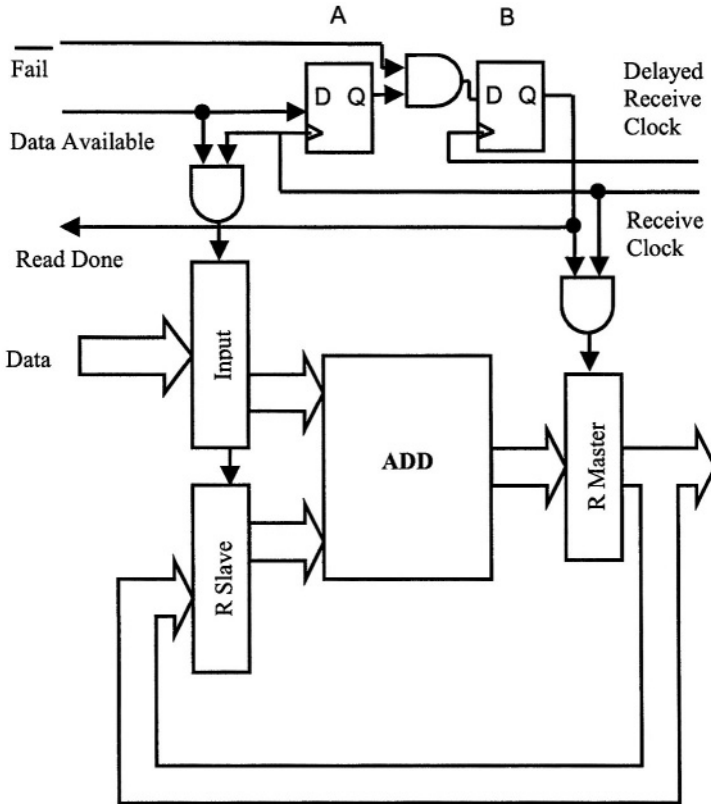


Fig. 10. Example of failure recovery

## 6 Conclusion

Synchronization in systems where the relationship between write and read clock domains is unpredictable may require more than one clock cycle. This is particularly true of high performance system where the clock period is only a small number of gate delays. Normally the latency of a synchronizer is fixed at the time required to give an acceptable reliability. Here, if a synchronization time of  $n$  cycles is needed to achieve adequate reliability, say  $e^{-30}$ , for every event, then it will usually be possible to approximately halve the synchronization time while accepting a probability of  $e^{-10}$  (1 in 20,000) that a computation may have to be carried out again. In this case the latency is variable, but the worst case is no greater than that of the standard solution.

This recomputation procedure may take more than one clock cycle, but only need occur approximately once in several thousand data transmissions, so does not significantly impact on the system performance. Because the synchronization delay is reduced, data transmission latency is also reduced by a factor of approximately two when compared to conventional; synchronizers. The additional hardware cost for the simple example described here is minimal, but latency is reduced from 2.5 cycles to



approximately 1.5 cycles. In more complex examples, as the number of cycles required in a conventional synchronizer increases, so does both the hardware and the recovery time needed for a speculative synchronizer. Typically every additional cycle would require an additional layer of registers in the affected pipeline stages, increasing both hardware and power dissipation. This is to be expected with the speculative approach, must be traded off against the latency improvements of around 0.5 cycle.

## References

1. Chakraborty, and M. Greenstreet, "Efficient Self-Timed Interfaces for crossing Clock Domains". Proceedings ASYNC2003, Vancouver, 12 – 16 May 2003, pp78-88.
2. N.A Kurd, J.S. Barkatullah, R.O.Dizon, T.D.Fletcher, and P.D.Madland "Multi-GHz Clocking Schemes for Intel Pentium 4 Microprocessors" Proc. ISSCC 2001 Feb 2001 pp404-405
3. S Tam, S Rusu, U. N. Desai, R Kim, J. Zhang, and I Young., "Clock Generation and Distribution for the first IA-64 Microprocessor". IEEE JSSC Vol. 35 No.11, Nov. 2000, pp1545-1552.
4. Y. Semiat, and R. Ginosar "Timing Measurements of Synchronization Circuits" Proceedings ASYNC2003, Vancouver, 12 – 16 May 2003, pp68-77.
5. C.Dike and E.Burton. "Miller and Noise Effects in a Synchronizing Flip-Flop". IEEE Journal of Solid State Circuits Vol. 34 No. 6, pp.849-855, June 1999
6. S.W.Moore, G.S.Taylor, P.A.Cunningham, R.D.Mullins and P.Robinson. "Using Stoppable Clocks to Safely Interface Asynchronous and Synchronous sub-systems". Proceedings AINT2000, Delft, 19-20 July 2000 pp.129-132.
7. W.J.Dally, and J.W.Poulton "Digital Systems Engineering" Cambridge University Press, 1998.
8. F.U.Rosenberger, C.E.Molnar, T.J.Chaney and T-P. Fang, "Q-Modules: Internally Clocked Delay-Insensitive Modules". IEEE Transactions on Computers, Vol. 37, No. 9, Sept 1988, pp 1005-1018
9. Bystrov, D. Sokolov, and A. Yakovlev "Low Latency Control Structures with Slack" Proceedings ASYNC2003, Vancouver, 12 – 16 May 2003, pp164-173.

# Minimizing the Power Consumption of an Asynchronous Multiplier

Yijun Liu and Steve Furber

Advanced Processor Technologies Group  
Department of Computer Science  
University of Manchester  
Manchester M13 9PL, UK  
{yijun.liu, sfurber}@cs.man.ac.uk

**Abstract.** As the demand for low power electronic products continues to increase there is a need for the designers of CMOS circuits to find ways to reduce the power consumption of their circuits. This paper introduces a practical approach to improve power-efficiency based upon the analysis of a breakdown of the power consumption of an existing design. The breakdown is used to identify the most promising subcircuits for improvement. A  $32 \times 32$  asynchronous pipelined integer multiplier is used as a case-study. Following the proposed methodology, the redesigned multiplier uses less than 40% of the energy per instruction of its predecessor. An asynchronous latch controller is also proposed which is smaller and faster than previous 4-phase fully-decoupled latch controllers.

## 1 Background

The semiconductor industry has witnessed an explosive growth in very large-scale integrated circuits over several decades. Only recently has the rapid progress of CMOS technology made integrating a complete system-on-a-chip possible. The rapid increase in the number of transistors on a chip dramatically improves system performance. However, the improvements in transistor density and performance also result in a rapid increase in power dissipation, which has become a major challenge in the use of modern CMOS technology. In order to prolong battery life and reduce the need for expensive cooling systems, significant effort has been focused on the design of low-power systems for portable applications, digital signal processors and ASIC implementations. Power consumption has emerged as an important parameter in VLSI design and is given comparable weight to or, for some applications, more weight than performance and silicon area considerations.

Dynamic power dissipation is the dominant factor in the total power consumption of a CMOS circuit and typically contributes over 80% of the total system power [1], although leakage is also becoming a major problem in deep sub-micron CMOS. Three factors determine the dynamic power dissipation of a CMOS circuit: the supply voltage, capacitance and switching activity. The supply voltage of a CMOS circuit is decided by the characteristics of the CMOS

technology used to fabricate the circuit, so we will ignore techniques based on reducing  $V_{dd}$ .

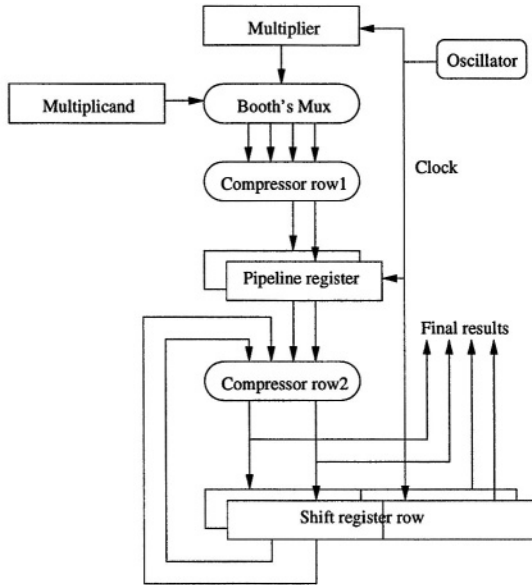
Multiplication is an essential function in microprocessors and digital signal processing systems and contributes significantly to the overall system power consumption, thus low-power design is important here. A number of techniques have been proposed to minimize the power dissipation in multipliers [2][3][4]. These approaches can be divided into two categories. The first category focuses on high-level algorithms to decrease the switching activity. The modified Booth's algorithm [5] is the most popular algorithm for low-power multipliers, because it scans multipliers two bits at a time, thus decreasing the number of partial products by a factor of two compared to a one-bit-at-a-time algorithm and reducing the number of compressor cells. The modified sign-generate algorithm (MSG) [4] is another power-efficient multiplication algorithm; instead of extending the sign bit to the most significant bit of the array, the MSG algorithm needs only two sign-extension bits, thus saving power and silicon area.

The second category of techniques used in power-efficient multipliers put their emphasis specifically on details of the CMOS implementation. Different Booth's encoders and partial product generators have been proposed to minimize the number of glitches [4][6]. The structure of compressors is another area of interest. 3-2 compressors (CSAs), 4-2 compressors, and even 5-2 compressors are claimed to have their advantages in low-power design [7][8][9]. Pass-transistor logic [10] is often used because it is very efficient for building small compressors.

The low-power methodology used in this paper is presented in the context of a pipelined multiplier — the Amulet3 multiplier [11], but can be used to reduce the power dissipation of other pipelined systems. The remainder of the paper is organized as follows: Section 2 presents the context of the work by introducing the architecture and power analyses of the Amulet3 multiplier. Section 3 describes the low-power strategies derived from the analyses of Section 2. Section 4 gives the detailed circuit implementation of the new multiplier and the experimental results. Section 5 summarizes the low-power techniques used in the new multiplier.

## 2 The Amulet3 Multiplier

Amulet3 [12] is a 32-bit asynchronous processor core that is fully instruction set compatible with clocked ARM cores. The Amulet3 multiplier uses a radix-4 modified Booth's algorithm [5] to decrease the number of partial products by a factor of two. An early termination scheme is also used to improve performance and save power. The early termination algorithm is based on the idea that if the multiplier is very small so that several of the most significant bits are all 1s or 0s, the multiplication can be sped up by ignoring these bits. Using a radix-4 modified Booth's algorithm and 4-2 compressors, 8 bits of multiplier are processed in each cycle and it takes 4 cycles to complete a multiplication. So for a 32-bit multiplier, if  $s_{31}..s_{24}$  are all 0s or 1s, one cycle of the multiplication can go faster. Similarly, two or three cycles can go faster if the next one or two



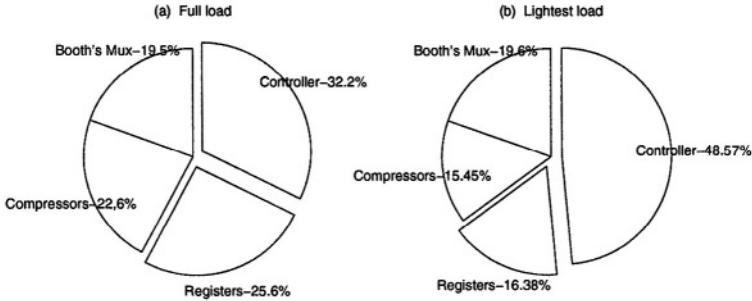
**Fig. 1.** The architecture of the Amulet3 multiplier

groups of eight bits are all the same as the top group. The multiplication cycles can be separated into two kinds — normal cycles and early termination cycles. In normal cycles, the multiplier operates as usual, but during early termination cycles the 4-2 compressors are idle and shift registers simply shift the output to the proper position. Consequently, power can be saved in early termination cycles.

The Amulet3 multiplier is an asynchronous multiplier, which cooperates with other parts of the Amulet3 microprocessor using an asynchronous handshake protocol. However, from the internal architecture point of view, the Amulet3 multiplier is a synchronous pipeline because its internal clock signal is generated by an inverter oscillator and this clock signal is distributed throughout the whole multiplier to provide the timing reference. The architecture of the Amulet3 multiplier is shown in Figure 1.

The Amulet3 multiplier does not use a tree or array architecture because of the tight silicon budget; instead an iterative architecture is used. Pipelining improves the speed of the multiplier by a factor of two.

In order to find the critical areas of power dissipation, we need to define some specific multiplication samples. The special features of the Amulet3 multiplier — a pipelined iterative multiplier with an early termination scheme — mean that its power dissipation is not constant. Multiplications without early termination consume the most power (the full load) and those with 3 cycles of early termination use the least power (the lightest load). Both situations must be analyzed. The power dissipation contributions of the different components are presented



**Fig. 2.** The power breakdown of the Amulet3 multiplier

in Figure 2(a) with a full load and Figure 2(b) shows the power breakdown with the lightest load.

As can be seen from Figure 2, the control circuit uses the largest share of the system power, followed by the registers. The registers in the Amulet3 multiplier are substantial — they include more than 200 edge-triggered flip-flops which put a heavy load on the clock signal. Therefore large buffers are needed in the output of the clock generator to drive the registers, and these buffers themselves consume a lot of power.

The interesting phenomenon we found when analyzing the lightest load situation is the fraction of the power used by the Booth's Mux. It is no different from that on full load. We might expect that during early termination cycles the Booth's Mux is idle and it should use less power. Moreover, the share of the power used by the shift registers is the same as that under full load, while the total power is decreased, so the power percentage of the shift registers should be bigger than that under full load. The most likely explanation is that during early termination cycles the datapath, including the Booth's Mux and compressor rows, still does some unnecessary work. In another words, the datapath is not fully idle during early termination cycles.

## 3 Low Power Strategies

### 3.1 Asynchronous Control

From the power analysis of the Amulet3 multiplier we can see that the power consumption of the edge-triggered registers and the control circuits is the crux of the problem. One efficient way to decrease the power consumption of the multiplier is to use smaller registers or latches and to simplify the control circuit. However, the TSPC register is very small compared to other edge-triggered registers. To get any smaller we must use level-sensitive transparent latches. With level-sensitive transparent latches, synchronous pipelines have only 50% occupancy. Fortunately, with fully-decoupled asynchronous pipeline latch controllers [13], we can use normal level-sensitive latches to construct an asynchronous pipeline

with 100% occupancy. As is well known, level-sensitive latches present half the capacitance of edge-triggered registers to their clock signals. Moreover, in asynchronous circuits several locally-generated signals are used instead of the global clock signal, which means that the load on the global signal is shared by several local signals and each local signal drives fewer gates and needs smaller driver buffers. As a result, changing the architecture of the Amulet3 multiplier from a synchronous pipeline to an asynchronous pipeline has the potential to reduce its power consumption.

The major cause of wasted power is unnecessary switching activity. It has been argued that asynchronous logic offers a more flexible power management strategy, which is that an asynchronous system only performs processing ‘on demand’ [14]. With asynchronous latch controllers we can easily isolate the datapath of the Amulet3 multiplier as soon it has finished performing normal cycles, so during early termination cycles only the shift registers are active to shift the output to the proper position, thus saving power by suppressing unnecessary activity.

The asynchronous latch controller used in the multiplier is shown in Figure 3(a). The latch controller is constructed from two C-elements [15]. The Signal Transition Graph (STG) description of the latch controller is illustrated in Figure 3(b). The dashed arrows indicate orderings which are maintained by the environment; the solid arrows represent orderings which the circuit ensures by itself. As we can see from Figure 3(b), the latch controller is normally closed, which means that the latch opens only briefly when a new stable data value is available on its inputs and it is closed in the idle state which eliminates unnecessary glitches. Normally-closed latch controllers are good for low power [16]. A normally-open latch controller, on the other hand, holds its latches in the transparent state when idle, so a glitch at the front of the datapath may propagate through the whole datapath. As a result, noise and glitches can easily cause unnecessary switching activity in the whole system and dissipate a lot of power. To achieve low power we should minimize the switching activity; the latches should therefore be closed as soon as possible after they finish passing data to avoid unnecessary activity throughout the system. The normally-closed latch controller can thus yield a significant power advantage in low-performance systems.

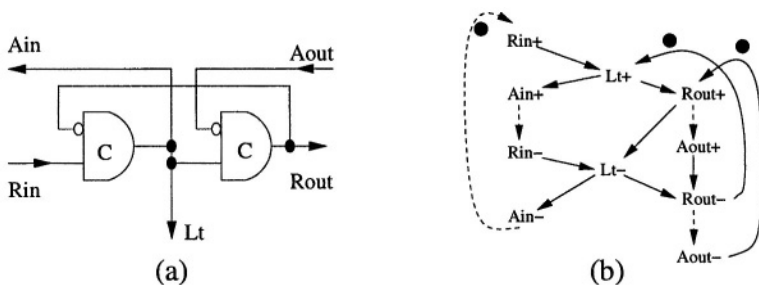


Fig. 3. A normally-closed latch controller

The latch controller is also fully-decoupled [13], and we can use normal level-sensitive latches to construct an asynchronous pipeline with 100% occupancy. Clocked logic requires edge-triggered registers to construct pipelines with 100% occupancy. Level-sensitive latches present half the capacitive load of edge-triggered registers to clock signals. As a result, level-sensitive latches controlled by the fully-decoupled latch controller shown in Figure 3 require about half as much area and power as edge-triggered designs. In the new multiplier, the pipeline registers are level-sensitive latches and the iterative registers are edge-triggered (because the iterative registers act both as pipeline registers and as shifters).

The multiplier is a 2-stage pipeline, so its control circuit includes three latch controllers, controlling the multipliers, the pipeline registers and the iterative registers respectively. The latch controller is not truly speed-independent, because we should not allow the input data to change until after  $Ain-$ , which indicates that the latch in the next stage has closed. Here,  $Rout-$  enables  $Lt+$ , violating the protocol at the next stage; a speed-independent latch controller should have  $Aout- \rightarrow Lt+$  instead of  $Rout- \rightarrow Lt+$ . However, since the delay of the logic block is much longer than the delay of a C-element, the new latch controller operates correctly despite this violation of speed-independence. As a result of this timing assumption, the proposed latch controller is smaller and faster than previous normally-closed and/or fully-decoupled latch controllers [13] [17].

The shifter shifts the result of multiplication right 8 bits per cycle. Initially the least-significant 32 bits of the shifter are empty, so we can put the multiplier in the least significant 32 bits of the shifter to save an additional 32 bits of register for storing the multiplier.

We can further increase the speed of the latch controller by eliminating two inverters' delay as shown in Figure 4. With this latch controller, we get an improved performance because the driver delay is no longer included in the critical path of the control circuit. Again, the timing assumptions upon which this optimisation is based must be checked carefully.

### 3.2 Non-Booth's Algorithm

We have argued [18] that the modified Booth's algorithm may have power-efficiency disadvantages as a result of its inverting operations: with the modified Booth's algorithm, partial products have a 50% probability of being  $-1\times$  or  $-2\times$  the multiplicand. These inverting operations introduce many transitions into the adder tree. This phenomenon is especially critical in the multiplication of two small integer numbers.

In order to overcome the disadvantage of the modified Booth's algorithm, we should avoid the inverting operations. We can use a much simpler algorithm by scanning the multiplier bit by bit. If the bit is 0, the respective partial product is 0; otherwise, the partial product is the value of multiplicand. This simple algorithm is not very efficient when the multiplier is negative; as a result of the two's-complement format, the multiplier sign bit will be propagated to the

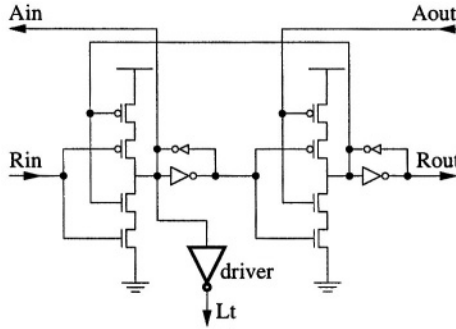


Fig. 4. A fast implementation of the latch controller

most significant bit. As a result, the multiplication has 32 partial products. Fortunately, we can modify this simple algorithm using the following equations:

$$a \times b = \bar{a} \times \bar{b} + \bar{a} + \bar{b} + 1 \tag{1}$$

$$a \times b = \overline{\bar{a} \times \bar{b} + b - 1} \tag{2}$$

$$a \times b = \overline{a \times \bar{b} + a - 1} \tag{3}$$

We first check the sign bits of the two operands. If both multiplicand and multiplier are negative (say  $a$  and  $b$ ), we invert both multiplicand and multiplier and send them to the non-Booth’s multiplier. Then we add the result to  $\bar{a} + \bar{b} + 1$  to get the final result. If only one of the operands is negative, we invert it. We send two positive numbers to the non-Booth’s multiplier and get the result. Then we invert the sum of the result, the positive operand and  $-1$  to get the final result.

### 3.3 Split Register

Multiplication operands have the characteristic of unbalanced distribution, which means that the least significant bits switch more frequently than the most significant bits. Some of the least significant bits toggle at approximately half the maximum frequency (uniform white noise). The most significant bits have a very low toggle rate. In our non-Booth’s algorithm, all operands sent into the multiplier are positive, so we can draw the effects of data correlation on bit switching frequency as shown in Figure 5. The  $n$  in Figure 5 is about 8 [19]

Because of the unbalanced distribution of input vectors, we can split the 32-bit registers into small groups. We only drive the ‘significant bits’ and ignore a series of 0s on the most significant bit side. In our design, we split the registers into 3 groups —  $Bit_{31}..Bit_{16}$ ,  $Bit_{15}..Bit_8$ ,  $Bit_7..Bit_0$ , as shown in Figure 6. We already have an 8-bit zero detector to decide the number of early termination cycles, so the only overhead incurred for splitting the registers is 2 clock gates. Testing the split registers by simulating the netlist we found a 12% energy saving.



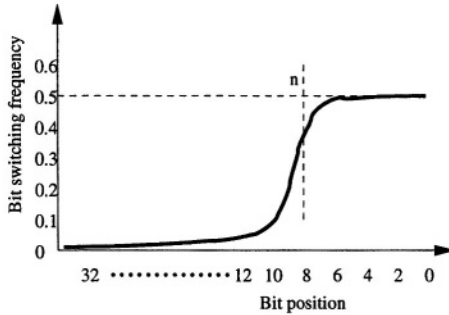


Fig. 5. The effects of data correlation on bit switching frequency

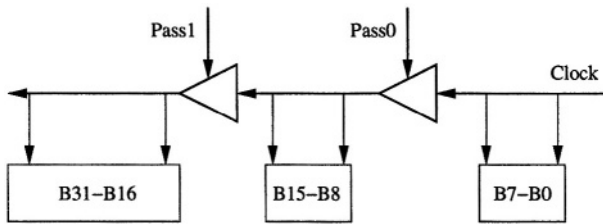


Fig. 6. 32-bit split register organization

Because the netlist-based simulation does not include wire capacitance we think it likely that in a real circuit splitting the registers will offer a greater power saving.

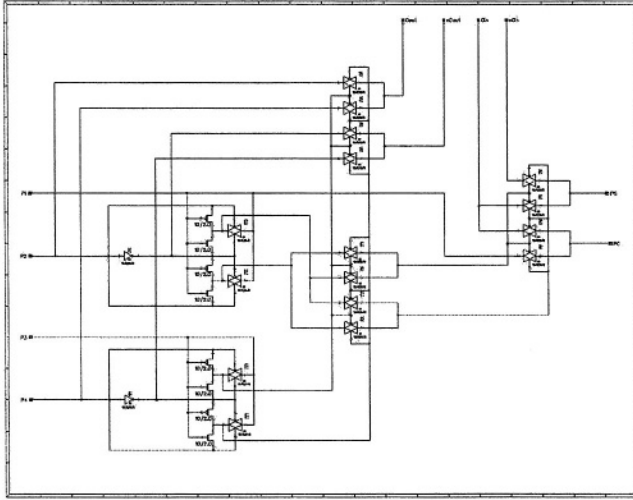
The split-register technique is inspired by a low-power register bank [20] and can be used in any register- or latch-based CMOS circuit with an unbalanced distribution of input operands.

## 4 Circuit Implementation and Experimental Results

### 4.1 DPL 4-2 Compressor

Using 4-2 compressors we can build a multiplier with a more regular structure than can be achieved with 3-2 compressors. A 4-2 compressor row reduces 4 partial products at once compared to only 3 partial products in a 3-2 compressor row. A 4-2 compressor can be constructed simply from two 3-2 compressors with four XOR-gates' delay. However, with careful design a 4-2 compressor can have only 3 XOR-gates' delay. So the multiplier using 4-2 compressors is faster than that using 3-2 compressors. Since the carry out signal is logically independent of the carry in, there is no carry propagation problem when several 4-2 compressors with the same weight are abutted into the same row; this is the key idea behind the use of the 4-2 compressor.

Pass-transistor logic is known to have an advantage over conventional static complementary CMOS logic in arithmetic circuits because it is well-suited to



**Fig. 7.** The schematic of a DPL 4-2 compressor [11]

the construction of XOR-gates and multiplexers. Complementary pass-transistor logic (CPL) [10] and differential pass-transistor logic (DPTL) [21] both consist of two nMOS logic networks to increase the noise immunity which is potentially compromised by low signal swings. CPL and DPTL make full use of the advantages of nMOS transistors, which are faster and smaller than pMOS transistors. However, both of them have disadvantages in high performance and low power areas:

- The degraded output signals increase leakage power dissipation.
- Low signal swings decrease the speed of the circuit.
- A swing restoration circuit is needed in the output, which causes additional power dissipation.

Double pass-transistor logic (DPL) [22] retains the advantages of pass-transistor logic but avoids the drawbacks mentioned above because it provides full signal swings. The schematic of the DPL 4-2 compressor used in the new multiplier is shown in Figure 7. From the schematic we can see that the paths of *Sum* and *Carry* are well balanced, which decreases glitches thus minimizing unnecessary switch activity. The balanced delay also results in a lower worst-case latency.

## 4.2 Pipeline Latches and Registers

In this multiplier, because of the employment of the hybrid pipeline latch controllers, normal transparent latches are used to act as pipeline latches rather than edge-triggered registers. An edge-triggered register is often constructed from two latches with a common clock input, so the latency of an edge-triggered register is double the latency of a latch, and the register's capacitance attached to the

clock signal is also doubled. As a result, this multiplier is lower power and faster than its synchronous counterparts which use edge-triggered registers.

Registers are used as a shifter to store the output of a multiplication. True single-phase clocking (TSPC) logic [23] is power-efficient when used to build the registers because a TSPC register is faster than a conventional static register and occupies less silicon area. Moreover, a TSPC register puts a smaller capacitance on the clock signal than a static register. PowerPC603 master-slave registers and latches [24] are well-known in the field of low-power design. In our experiment, we found that the PowerPC603 master-slave register is slightly better than the TSPC register when used in low-performance systems.

### 4.3 Experimental Results

The new multiplier was compared with the original Amulet3 multiplier using HSPICE under the conditions of a 1.8 volt supply and 27 °C temperature on a 0.18 micron CMOS technology. The average power consumption of the new multiplier is 6.47 mW at 100 million operations per second, whereas the Amulet3 multiplier burns 16.17 mW at the same speed. Power dissipation alone is not enough to evaluate a circuit because we can simply make it very low power by reducing its performance. In this paper, *power × delay* or *energy per operation* is also used as a metric to evaluate the multipliers. Compared to the Amulet3 multiplier, the new multiplier is 10% faster than the Amulet3 multiplier because the delay of the transparent latches is less than that of edge-triggered registers, and we gain from the average performance of the two well-balanced pipeline stages. High performance is a side-effect of choosing asynchronous logic. Under the same conditions and with the same operands, the new multiplier uses only 37% of the *energy per operation* of the Amulet3 multiplier.

## 5 Conclusion

In the work described in this paper we succeeded in decreasing the power dissipation of an asynchronous pipelined iterative multiplier — the Amulet3 multiplier — starting from an analysis of the power breakdown. The new multiplier uses less than 40% of the *energy per operation* of its predecessor. The improvement in the low-power characteristics of the new multiplier is due to redesigning the control circuit and discarding large edge-triggered registers which contribute the most power dissipation in the Amulet3 multiplier. With the new asynchronous latch control circuits, more compact control circuits and smaller transparent level-sensitive latches are used in the new multiplier to minimize the load capacitance and remove the large driver buffer. The asynchronous control scheme causes the proposed multiplier to perform processing ‘on demand’, thus minimizing unnecessary switching activity. A low-power non-Booth’s algorithm is also used in this multiplier to avoid the inverting operations incurred by the modified Booth’s algorithm. Finally, to exploit the unbalanced bit switching frequency, we use a split register scheme to decrease the power dissipation of the large registers.

## References

1. D. Soudris, C. Pigué and C. Goutis (eds). "Designing CMOS Circuits for Low Power". Kluwer academic publishers, 2002.
2. T. Callaway and E. Swartzlander. "The Power Consumption of CMOS Adders and Multipliers", in "Low power CMOS Design", A. Chandrakasan and R. Brodersen (eds), IEEE Press, 1998.
3. L. Bisdounis, D. Gouvetas and O. Koufopavlou. "Circuit Techniques for Reducing Power Consumption in Adders and Multipliers", in [1].
4. R. Fried. "Minimizing Energy Dissipation in High-Speed Multipliers", International Symposium on Low Power Electronics and Design, 1997.
5. A. D. Booth, "A Signed Binary Multiplication Technique", Quarterly J. Mech. Appl. Math., vol. 4, part 2, pp. 236-240, 1951.
6. N.-Y. Shen and O. T.-C. Chen, "Low power multipliers by minimizing switching activities of partial products", IEEE International Symposium on Circuits and Systems, 2002, Volume: 4, 26-29 May 2002.
7. J. Gu and C.-H. Chang, "Ultra low voltage, low power 4-2 compressor for high speed multiplications", International Symposium on Circuits and Systems, 2003, Volume: 5, 25-28 May 2003.
8. S.-F. Hsiao, M.-R. Jiang and J.-S. Yeh, "Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers", Electronics Letters, Volume: 34, Issue: 4, 19 Feb. 1998.
9. K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors", Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, 2001, Volume: 1, 4-7 Nov. 2001.
10. R. Zimmermann and W. Fichtner, "Low-Power Logic Styles: CMOS versus Pass-Transistor Logic", IEEE Journal Of Solid State Circuits, 32(7):1079-1090, July 1997.
11. J. Liu, "Arithmetic and Control Components for an Asynchronous System", PhD thesis, The University of Manchester, 1997.
12. S. B. Furber, D. A. Edwards and J. D. Garside, "AMULET3: a 100 MIPS Asynchronous Embedded Processor", IEEE International Conference on Computer Design, 2000, 17-20th September 2000.
13. S. B. Furber and P. Day, "Four-Phase Micropipeline Latch Control Circuits", IEEE Transactions on VLSI Systems, vol. 4 no. 2, June 1996, pp. 247-253.
14. S. B. Furber, A. Efthymiou, J. D. Garside, M. J. G. Lewis, D. W. Lloyd and S. Temple, "Power Management in the AMULET Microprocessors", IEEE Design and Test of Computers Journal special issue, pp. 42-52 (Ed. E. Macii), March-April 2001.
15. J. Sparsø, S. Furber (eds). "Principles of Asynchronous Circuit Design: A systems perspective". Kluwer Academic Publishers, 2001.
16. M. Lewis, J. D. Garside, L. E. M. Brackenbury. "Reconfigurable Latch Controllers for Low Power Asynchronous Circuits". International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1999, April 1999.
17. P. A. Riocreux, M. J. G. Lewis and L. E. M. Brackenbury, "Power reduction in self-timed circuits using early-open latch controllers", Electronics Letters, Volume: 36, Issue: 2, 20 Jan. 2000.
18. Y. Liu and S. B. Furber, "The Design of a Low Power Asynchronous Multiplier", International Symposium on Low Power Electronics and Design, 2004, Newport Beach, California, USA, August 9-11, 2004.

19. P. E. Landman and J. M. Rabaey, "Architectural power analysis: The dual bit type method", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Volume: 3, Issue: 2, June 1995.
20. V. Zyuban and P. Kogge, "Split Register File Architectures for Inherently Low Power Microprocessor", *Power Driven Microarchitecture Workshop at ISC98*, June 1998.
21. J. H. Pasternak and C. A. T. Salama, "Differential pass-transistor logic", *Circuits and Devices Magazine, IEEE* , Volume: 9, Issue: 4, July 1993.
22. M. Suzuki, N. Ohkubo, T. Yamanaka, A. Shimizu and K. Sasaki, "A 1.5 ns 32 b CMOS ALU in double pass-transistor logic", *IEEE International Conference on Solid-State Circuits* , 40th ISSCC., 24-26 Feb. 1993.
23. J. Yuan and C. Svensson, "New TSPC latches and flipflops minimizing delay and power", *IEEE International Symposium on VLSI Circuits, Digest of Technical Papers.*, 1996 , 13-15 June 1996.
24. G. Gerosa, S. Gary, C. Dietz, P. Dac, K. Hoover, J. Alvarez, H. Sanchez, P. Ippolito, N. Tai, S. Litch, J. Eno, J. Golab, N. Vanderschaaf, and J. Kahle, "A 2.2 W, 80 MHz superscalar RISC microprocessor", *IEEE Journal on Solid-State Circuits*, vol. 29, pp. 1440-1452, Dec. 1994.

# A Channel Library for Asynchronous Circuit Design Supporting Mixed-Mode Modeling

T. Bjerregaard, S. Mahadevan, and J. Sparsø

Informatics and Mathematical Modeling  
Technical University of Denmark (DTU), 2800 Lyngby, Denmark  
{tob, sm, jsp}@imm.dtu.dk

**Abstract.** This paper presents the use of SystemC to model communication channels for asynchronous circuits at various levels of abstraction. Our channel library supports transactions through a CSP-like interface (implementing *send()* and *receive()* commands) as well as through one of many specific handshake protocols e.g. 4-phase-bundled-data push etc. Our SystemC implementation enables a seamless design flow which makes possible: (i) modeling and simulation at different *and mixed* levels of abstraction, and (ii) easy use of different data types and handshake protocols on different channels in the circuit being designed. The paper also illustrates the use of this design flow for several asynchronous Networks-on-Chip all the way from system level to handshake components.

## 1 Introduction

Over the past years, research in design of asynchronous circuits and systems have lead to a better understanding of where and how to apply asynchronous techniques, and significant industrial-scale asynchronous IC's have been fabricated. The main factor hindering more widespread use of asynchronous design is the lack of suitable CAD tools. Almost all work on the high-level modeling and synthesis of asynchronous circuits is based on specialized asynchronous hardware description languages (HDLs) derived from CSP [1]. The basic CSP primitives for channel-based communication (*send*, *receive* and *probe*) abstracts away the low-level details of the handshaking that controls the flow of data in an asynchronous circuit. Examples of such asynchronous HDLs are: Tangram [2], Balsa [3], CHP [4].

These specialized HDLs and related synthesis tools for asynchronous design have a number of disadvantages: (i) They are not well integrated with industry standard CAD tool frameworks based on VHDL, Verilog and/or SystemC. (ii) Hybrid designs including both synchronous and asynchronous sub-systems are not supported. (iii) The use of different handshake protocols in different parts of the circuit is not supported. (iv) The designer is restricted to working at a high abstraction level, and the transformation to actual circuit netlist is hidden. Thus low level circuit optimization is not possible. In this regard, there is a *language*

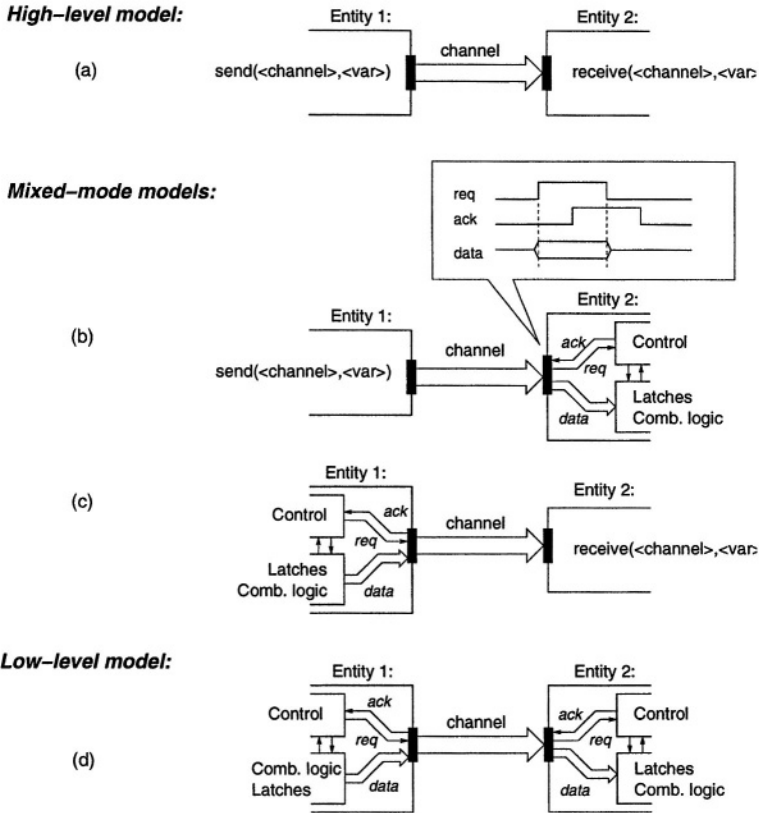


Fig. 1. Channel usage at different levels of abstraction.

*barrier* which must be crossed, making design iterations at the circuit level cumbersome and error prone.

These and related observations have motivated attempts to provide CSP-style communication within industry-standard HDLs such as VHDL and Verilog. In [5, pp. 133-152], [6] VHDL resolution functions were used to model channel communication at mixed levels of abstraction. VHDL however does not support overloading for such functions, obstructing the implementation of different channel protocols within the same circuit. In [7] similar ideas were explored using Verilog, but this work extends the source language and relies on PERL scripts invoked through Verilog's Programming Language Interface to deal with the extensions. The bottom line is that both VHDL and Verilog have fundamental limitations which prevent the implementation of sufficiently flexible abstract and mixed-mode communication.

During the past couple of years, SystemC [8] has gained popularity in industry and academia. It is a synthesizable library within C++ which enables communication between entities via abstract interfaces, while also supporting

event-based modeling primitives required for signal level design. Thus it can be used across a wider range of abstraction levels than traditional HDLs. This paper presents a SystemC based channel library which overcomes the limitations found in VHDL and Verilog based approaches.

Figure 1 illustrates the aim of our work: to enable, within a single language framework, a seamless design flow which supports (1) modeling and simulation of asynchronous circuits at different *and mixed* levels of abstraction as would be useful in for example a top-down stepwise refinement design methodology, and (2) easy use of different data types and handshake protocols on different channels in the circuit being designed. The contributions of the paper are the concepts used in the modeling of asynchronous channel communication between entities which may be implemented at different levels of abstraction and the implementation of these concepts using SystemC. To demonstrate the use of the presented channel library the paper briefly explains how the channels have been used in ongoing research, where we study and develop asynchronous implementations of Networks-on-Chip.

The rest of the paper is organized as follows: Section 2 details the concepts behind our channel-based communication and in particular mixed-mode modeling. Section 3 explains how these concepts are implemented in SystemC. Section 4 demonstrates the use of our channels in implementing asynchronous Networks-on-Chip. Finally conclusions are drawn in Sect. 5.

## 2 Channel Model

In this section, we introduce our channel concept which allows flexible mixed-mode simulation. First the workings of our abstract channel, used in high-level modeling, is explained. Then the transformation mechanisms used to switch between abstract and physical asynchronous protocols are detailed.

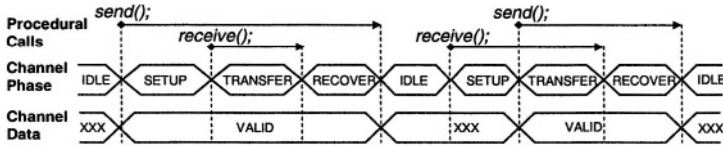
We have implemented 2- and 4-phase (2ph and 4ph), push and pull versions of both bundled data and dual rail protocols (bd and dr). In this paper we will use the 4ph-bd-push protocol as an example to illustrate the workings of the channel. It contains three signals: request (*req*), acknowledge (*ack*) and data wires, as illustrated in Fig. 1(b). Details of this and other well known protocols can be found in [5].

### 2.1 Abstract Channel Communication

During design space exploration at high levels of abstraction, details of the physical protocol are unwarranted; a purely abstract communication channel is called for. The important aspects of such a channel are (i) the availability of communication primitives, which take the form of CSP-like procedural calls such as *send(channel, data)*, *receive(channel, data)* and *probe(channel)*, and (ii) data type flexibility, i.e. support for a wide range of data types.

The abstract channel implements the *concepts* of channel-based communication. Such communication has a setup phase during which the entities at either





**Fig. 2.** Channel phase transitions in our abstract channel.

end of the channel get ready to communicate, a transfer phase during which the actual communication takes place, and a recover phase, during which the entities disengage. The phases of the abstract channel must represent a superset of the phases of the physical channels that it will be mapped to during mixed-mode simulations. We have implemented an abstract channel that meets these requirements.

Figure 2 shows its behaviour for two transactions. Initially the channel is in the IDLE phase. By invoking *send()* the sender initiates the communication. The channel thus enters the SETUP phase. At some point, the receiver invokes the corresponding *receive()* and the channel enters the TRANSFER phase, during which data is transferred. Upon completion of the transfer the receiver disengages, and the channel enters RECOVER phase. When the sender has also disengaged, the channel re-enters the IDLE phase.

The ordering of the *send()* and *receive()* calls is not important. During the next transaction shown in the figure, *receive()* is invoked first and *send()* later. When both sender and receiver have invoked their respective calls, the channel enters TRANSFER phase, thus synchronizing the connected modules.

A possible extension to this abstract communication channel is bi-put functionality, in which data is passed in both directions during each transaction.

## 2.2 Mixed Mode Communication

In order to enable the seamless transition from system specification to circuit implementation independently for different parts of the system, support for mixed-mode communication as illustrated in Fig. 1 (b) and (c) is needed. Figure 3 shows the constituent of our channel implementations. The channel implements both an abstract and a physical interface. A translator module handles the translation between the two. Thus the channel supports mixed-mode communication. It is illegal and unwarranted to simultaneously use both the abstract and the physical interface at the same end of the channel.

Communication via the abstract interfaces is already explained in Sect. 2.1. In the following we will discuss the implications of the translation between this and our example, the 4ph-bd-push protocol. Such mapping to a specific handshake protocol is a simple matter of relating handshake signal events with the phase transitions of the abstract channel, as shown in Figs. 4 and 5. The arrows in the figures indicate the driving of signals by the translator.

During translation from the abstract to the physical 4-ph-bd-push protocol shown in Fig. 4, corresponding to the situation of Fig. 1 (b), the translator

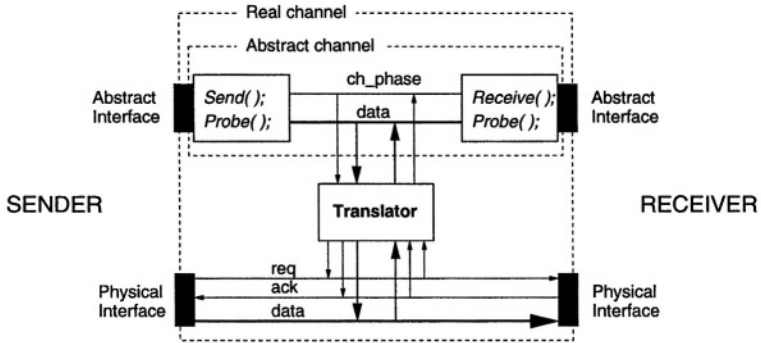


Fig. 3. Protocol-specific channel translator example.

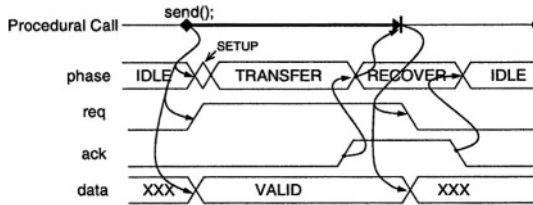


Fig. 4. Illustration of abstract to 4-ph-bd-push protocol translation.

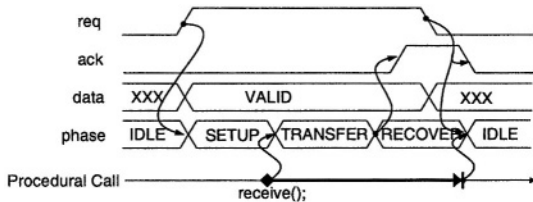


Fig. 5. Illustration of 4-ph-bd-push to abstract protocol translation.

drives *req* and *data* signals as would a physical sender. When the abstract sender invokes a *send()* command, the input data is placed on the *data* lines, and *req* is driven high. The channel enters the SETUP phase, but moves immediately and spontaneously to the TRANSFER phase. In a 4-ph-bd-push protocol, the receiving end of the channel is inherently ready for a transaction. The *ack* signal being low indicates this. This illustrates the fact that the abstract protocol is a superset of the given physical protocol. The SETUP phase has no meaning. When the physical receiver indicates that it has accepted the data, by driving *ack* high, the channel completes the TRANSFER phase, and moves into the RECOVER phase. The *send()* completes and *req* is lowered. In response, the receiver lowers *ack*, and the channel returns to the IDLE phase.

A similar channel phase transition is observed when translating from the physical to the abstract protocol, as shown in Fig. 5. This corresponds to Fig. 1

(c). The physical sender initiates the transaction by driving *req* high. The channel thus enters the SETUP phase. When the abstract receiver invokes a *receive()*, the channel enters the TRANSFER phase. The rest of the channel phase transition from TRANSFER to RECOVER to IDLE are self-explanatory. Note that in this case, all four phases of the abstract channel are used. The reason is that the abstract receiver is not passive, as is the physical 4-ph-bd-push receiver.

Finally a word on modeling accuracy: the phase transitions in the abstract channel model cannot be controlled or observed directly; *send()* and *receive()* are atomic primitives which “step” the channel through all its phases. It may prove useful to provide simpler primitives like *init\_send()* and *complete-send()* in order to allow a refined behavioural model to reflect the handshake reshuffling which may be implemented in the actual hardware. Our channel model can be easily extended with such intermediate primitives.

### 3 Channel Implementation

In this section we explain the implementation details of our channel library. First some of the key SystemC and C++ features that we have used are reviewed. Then the programming structure and the binding mechanism is described. We use SystemC v2.0 as it has introduced a set of features for generalized modeling of communication and synchronization: *channels*, *interfaces*, and *events* [8].

#### 3.1 Useful SystemC and C++ Features

In SystemC, a channel is an object that serves as a container for communication and synchronization. The channel implements one or more interfaces. An interface specifies a set of access methods (e.g. *send()* and *receive()*) to be implemented within the channel. The interface class is *virtual*, and as such does not implement these methods itself. Since modules bound to the channel via the interface are only allowed access to the methods specified within the interface class, code-integrity between various implementations is maintained. To perform a data transfer, the data source and sink bind themselves to the interfaces at either end of the channel, and then simply invoke the required method specified in the interface. A SystemC event is a low-level synchronization primitive, which can be used to synchronize execution of different methods within the channel.

Besides these special SystemC features, we also exploit the inheritance property of C++. Inheritance is a means of specifying hierarchical relationships between objects. C++ classes can inherit both data and function members from the parent classes. We use these properties of inheritance to relate the abstract channel with the physical channel. The object oriented hierarchy (OOH) makes the development of new protocol-specific channels easy and safe. Since the same abstract channel class is inherited by all physical channel classes, all channels share the same abstract interface and abstract channel implementation. Additionally, C++ allows flexible type-casting of the channel data known as data templating. Thus the abstract channel model can be instantiated for any type of data.

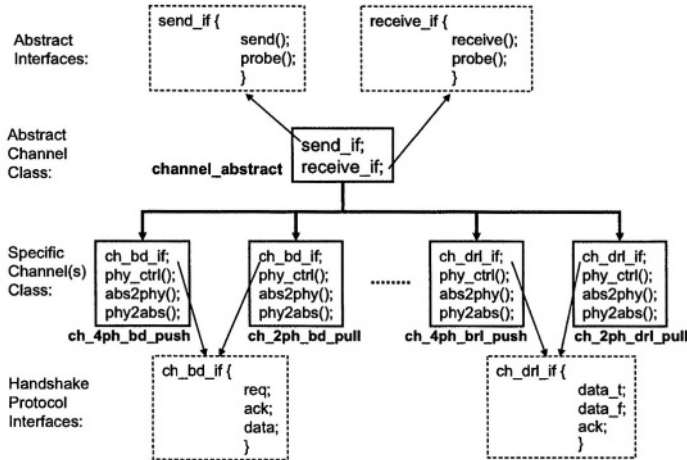


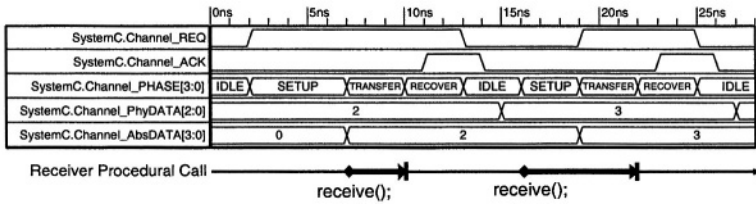
Fig. 6. The class-based Object Oriented Hierarchy (OOH) of our channel library.

### 3.2 Hierarchy-Based Channel Design

Figure 6 shows the class-based OOH of our design. The abstract channel called *channel\_abstract*, is protocol independent and inherits the abstract interface classes *send\_if* and *receive\_if*. This channel communicates via sequential phase transitions as described in Section 2.1 and is one in which the signals have no circuit-level meaning. The *send\_if* specifies the *send()* method for transmission of data and the *receive\_if* specifies the *receive()* method for reception of data. Each interface also specifies a *probe()* method that detects pending transactions.

The different protocol-specific channels inherit the abstract channel class. Thus, the abstract function calls are readily available also to the physically connected entities. On top of this, the protocol-specific channels implement the corresponding handshake protocol interface which defines the control and data signals. The *phy\_ctrl()*, *abs2phy()* and *phy2abs()* processes, shown in Figure 6 implement the translator module shown in Figure 3. The former implements the protocol translation (driving the protocol signals defined in handshake protocol interface), while the latter two can be user defined for data encoding/decoding between any user defined type and a physical bit vector signal. Thus by building on the proven protocol translation in the channel, any data type defined by the user at high abstraction level can be used in a mixed-mode simulation. Our library provides a number of standard data encoders/decoders such as *int*, *char*, *bool*, etc.

An alternate prospect is the implementation of our channel model in SystemVerilog. Similarly to SystemC this extension to Verilog provides *interfaces* which may encapsulate physical signal ports and/or procedural calls. Also, a limited form of OOH is supported, in that SystemVerilog allows single-level inheritance. Complex user defined types can be declared using the *struct* construction which is similar to the *record* of VHDL, and data encoding/decoding



**Fig. 7.** Simulation trace of physical to abstract mixed-mode communication, with two transactions. In the first, the sender is faster and in the second the receiver is faster.

as described above is straight forward since even the complex data types have well-defined translation to and from scalar bit vectors [9].

## 4 Demonstration

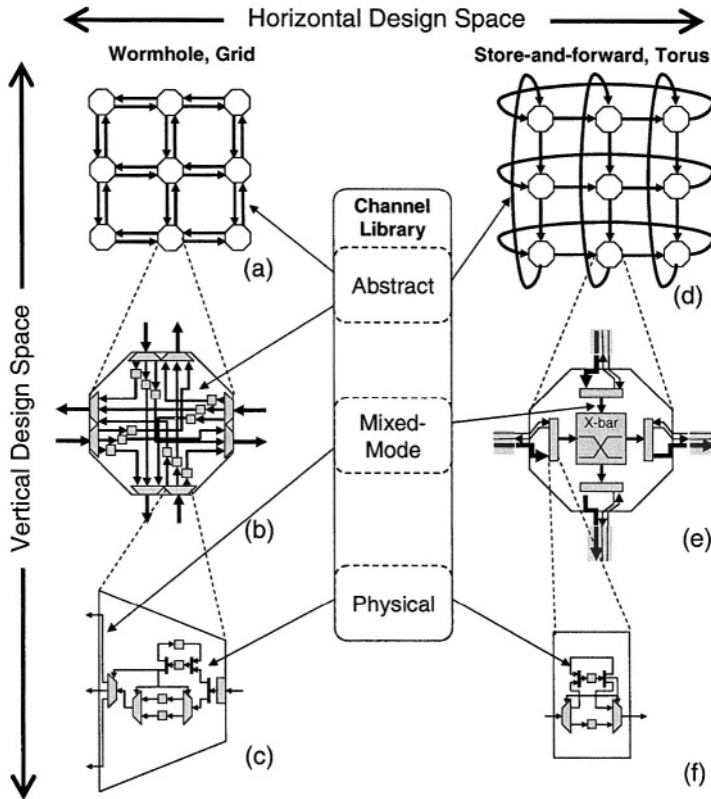
Our channel library [10] has been used in support of our ongoing research of networks-on-chip (NoC). NoC are segmented communication structures typically implemented by links and routing nodes across the chip [11,12]. In particular, we have used the channels for design space exploration of asynchronous NoC implementations.

Figure 8 shows two dimensions of the design space. The horizontal space indicates alternative implementations, while the vertical space relates to the level of abstraction. The flexibility of mixed-mode channel-based design proved valuable during design iterations as well as during architectural experiments. Our channels were used at all abstraction levels, providing a flexible tool for both horizontal and vertical design space exploration. Figure 7 shows a simulation trace of the 4-ph-bd-push channel in a mixed-mode situation. It is seen how a physical sender communicates with an abstract receiver. The trace is equivalent to the timing diagram shown in Fig. 5.

Among the NoCs that we have looked at are two variants of source routing packet-switched networks, as shown in Figs. 8(a) and 8(d): Example I, a grid with wormhole routing, and Example II, a torus with store-and-forward routing. The abstract channels were first used for exploring system-level issues. Hereafter the channels were used during refinement of the designs in several ways.

In Example I, abstract channels were used to model a structural implementation of the node (Fig. 8(b)), and following this each sub-module was refined into a structure of handshake components (Fig. 8(c)), this time replacing the abstract channels with physical channels. Finally the handshake components were implemented using standard cells, thus leading to the final netlist.

In Example II, a similar refinement was done. As shown in Fig. 8(e) the node uses physical channels on its interface and mixed mode channels inside – the node consists of physical implementations of input and output latches and a behavioural model of the routing switch.



**Fig. 8.** Demonstration of our channel model in two select NoC platforms. The arrows indicate instances of the channels i.e. either physical or abstract.

Another use of mixed mode channels would be a scenario where only a single node is refined into physical implementation, keeping other nodes of the network behavioral – a situation which will be useful when exploring alternative node implementations. This will keep the simulation time down, compared to a situation where all nodes are represented by their physical implementation.

In addition to the mixed mode modeling, our channel library also allows the designer to chose different protocols for different parts of the system if this is deemed necessary. In our NoC designs, we chose 4-ph-bd-push for the internal node components, while the power efficiency and delay insensitivity of pipelined 2-ph-dr-push circuits were used on the node-to-node links.

The demonstration has served as an example of the flexibility of our channel-based design approach. As illustrated in Fig. 7, the channels allowed functional verification at any intermediate design step, using the same top level testbench.

## 5 Conclusion

The paper presented a channel concept which enables the modeling and simulation of asynchronous handshake channels at mixed abstraction levels. A channel library, which allows simulation at any intermediate step in a top-down design refinement, was implemented in SystemC. Unlike the Verilog and VHDL languages, SystemC features the constructs necessary for this. Our channel library provide translation between abstract CSP-like procedural calls (*send*, *receive* and *probe*) and a given physical protocol, and allow flexible translation for any user defined data type. The library features a number of well known asynchronous handshake protocols such as 4-phase bundled data push, 2-phase dual rail pull, etc. Thus a channel-based top-down design methodology for asynchronous circuit design is contained within a single language. The channel library is currently being used in research on asynchronous implementations of Networks-on-Chip and it has proven stable and versatile.

## References

1. Hoare, C.A.R.: Communicating Sequential Processes. Communications of the ACM **21** (1978) 666–677
2. Berkel, K.v., Kessels, J., Roncken, M., Saeijs, R., Schalijs, F.: The VLSI-Programming Language Tangram and Its Translation into Handshake Circuits. In: Proceedings of the European Conference on Design Automation, EDAC. (1991) 384–389
3. Bardsley, A., Edwards, D.: Compiling the language Balsa to delay-insensitive hardware. In Kloos, C.D., Cerny, E., eds.: Hardware Description Languages and their Applications (CHDL). (1997) 89–91
4. Martin, A.J.: Formal program transformations for VLSI circuit synthesis. In Dijkstra, E.W., ed.: Formal Development of Programs and Proofs. UT Year of Programming Series, Addison-Wesley (1989) 59–80
5. Sparsø, J., Furber, S.: Principles of Asynchronous Circuit Design. Kluwer Academic Publishers, Boston (2001)
6. Pedersen, M.: Design of Asynchronous circuits using standard CAD tools. Technical Report IT-E 774, Technical University of Denmark, Dept. of Information Technology (1998) (In Danish).
7. Saifhashemi, A.: Verilog HDL: A Replacement for CSP. In: 3rd Asynchronous Circuit Design Workshop - ACiD-WG, Heraklion, Greece (2003) <http://www.scism.sbu.ac.uk/ccsv/ACiD-WG/Workshop3FP5/>.
8. : Systemc workgroup website. (<http://www.systemc.org>)
9. Fitzpatrick, T.: SystemVerilog for VHDL Users. In: Proceedings of the 2004 Design, Automation and Test in Europe Conference (DATE'04), IEEE (2004)
10. Mahadevan, S., Bjerregaard, T.: DTU Channel Package (2003) <http://www.imm.dtu.dk/~tob/SystemC/channels.html>.
11. Benini, L., Micheli, G.D.: Networks on Chips: A new SoC Paradigm. IEEE Computer **35** (2002) 70 – 78
12. Dally, W.J., Towles, B.: Route packets, not wires: On-chip interconnection networks. In: Proceedings of the 38th Design Automation Conference. (2001) 684–689

# L0 Cluster Synthesis and Operation Shuffling\*

Murali Jayapala<sup>1</sup>, Tom Vander Aa<sup>1</sup>, Francisco Barat<sup>1</sup>,  
Francky Catthoor<sup>2</sup>, Henk Corporaal<sup>3</sup>, and Geert Deconinck<sup>1</sup>

<sup>1</sup> ESAT/ELECTA, Kasteelpark Arenberg 10, K.U.Leuven, Heverlee, Belgium-3001  
{mjayapal, tvandera, fbaratqu, gdec}@esat.kuleuven.ac.be

<sup>2</sup> IMEC vzw, Kapeldreef 75, Heverlee, Belgium-3001  
catthoor@imec.be

<sup>3</sup> Department of Electrical Engineering, Eindhoven University of Technology (TUE),  
P.O. box 513, 5600 MB Eindhoven, The Netherlands  
h.corporaal@tue.nl

**Abstract.** Clustered L0 buffers are an interesting alternative for reducing energy consumption in the instruction memory hierarchy of embedded VLIW processors. Currently, the synthesis of L0 clusters is performed as a hardware optimization, where the compiler generates a schedule and based on the given schedule L0 clusters are synthesized. The result of clustering is schedule dependent, which offers a design space for exploring the effects on clustering by scheduling. This paper presents a study indicating the potentials offered by shuffling operations within a VLIW instruction on L0 cluster synthesis. The simulation results indicate that potentially up to 75% of L0 buffer energy can be reduced by shuffling operations.

## 1 Introduction

Current embedded systems for multimedia applications like mobile and handheld devices, are typically battery operated. Therefore, low energy is one of the key design goals of such systems. Many of such systems often rely on VLIW ASIPs [10]. However, power analysis of such processors indicate that a significant amount of power is consumed in the instruction caches [15,5]. For example in the TMS320C6000 (TI C6X), a VLIW processor from Texas Instruments, up to 30% of the total processor energy is consumed in the instruction caches alone [15]. Loop buffering or L0 buffering is an effective scheme to reduce energy consumption in the instruction memory hierarchy [4,14]. In any typical multimedia application, significant amount of execution time is spent in small program segments. Hence, by storing them in a small L0 buffer instead of the big instruction cache, up to 60% of energy consumption in instruction memories can be reduced [4,14].

While this reduction is substantial, further optimizations are still necessary to ensure such high energy efficiency in the future processors. Orthogonal optimizations applied on different aspects of the processor, like the datapath, register

\* This work is supported in part by MESA under the MEDEA+ program.



files and data memory hierarchy [8], reduce the overall processor energy. However, the instruction cache energy, including the L0 buffer, is bound to increase or remain at around 30%. Of the two main contributors of energy consumption in the instruction memory hierarchy, the L0 buffers are the main bottleneck; they consume up to 70% of the energy. To alleviate the L0 buffer energy bottleneck, a clustered L0 buffer organization has been proposed in the recent past [12]. Essentially, in a clustered L0 buffer organization the L0 buffers are partitioned and grouped with certain functional units in the datapath to form an instruction cluster or an L0 cluster. In each cluster the buffers store only the operations of a certain loop destined to the functional units in that cluster. Further architectural details are provided in [4,12].

## 1.1 Motivation

Currently, the generation of L0 clusters is at the micro-architectural level. For a given application and a corresponding schedule, the clustering tool analyzes the functional unit activation trace and generates a cluster configuration optimal with respect to energy[12]. The left-side in Figure 1, illustrates the flow for generating clusters. For a given application, a pre-compiler tool analyzes the profile and identifies certain interesting parts<sup>1</sup> of the code to be mapped on to the L0 buffers. In the example of Figure 1, a certain for-loop is mapped on to the L0 buffer. The compiler then generates a certain schedule for the selected parts of the code (for-loop in the example). Here, the schedule is generated for an 8-issue VLIW processor, which has 3 instructions with 4 operations in first instruction and 1 operation in second and third instructions. The clustering tool generates L0 clusters that minimizes the L0 buffer energy, and which is the optimum for this given schedule. No other clustering will have a lower energy consumption for this schedule. In the example, the optimum is 4 clusters with energy consumption of 1.0 units. The associated functional unit grouping and the L0 buffer partition sizes are as indicated in the Figure 1. For example, functional units (or issue slots) 3, 5 and 7 are grouped into a single cluster and it has an L0 buffer partition of width 3 operations and a depth of one word.

However, the resulting clusters from the clustering tool are sensitive to the schedule. In the above example, if the schedule is slightly modified as indicated in Figure 1, then the same clustering tool would now produce different clusters: 3 clusters, with different functional unit grouping and L0 buffer partitions. The L0 buffer energy is also reduced from 1.0 units to 0.91 units. The example shows just one possible way to modify the schedule. The schedule can be modified in several ways, some of which are capable of reducing L0 buffer energy. This is a good motivation to investigate the potentials offered by a modified scheduling algorithm to reduce L0 buffer energy.

Two levels of freedom can be explored in scheduling. In the example, operations within each instruction (or within an instruction cycle) were assigned to different functional units. For instance, operation ‘CMP’ was reassigned from

<sup>1</sup> Mainly loops, but a group of basic blocks within a loop can also be mapped.

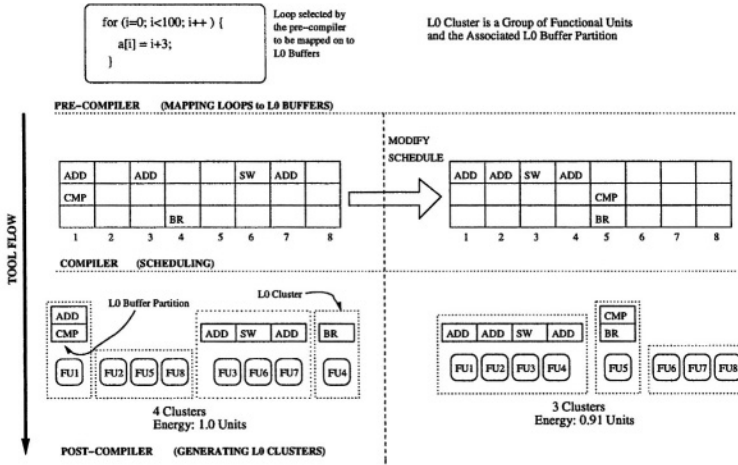


Fig. 1. Example Illustrating Scheduling Sensitivity on L0 Cluster Generation

FU1 to FU5. This is the first level of freedom: to shuffle operations within an instruction. Instead of restricting to reassigning an operation within an instruction, the operation can also be rescheduled to a different functional unit in a different instruction cycle. For instance operation ‘ADD’ scheduled to FU7 in the first instruction could be rescheduled to FU2 in the second instruction. This is the second level of freedom: to reschedule operations across instructions. Note that FUs do not have to be identical to allow such a move of certain operations. By adapting also the instruction selection, most of the operations in the source code can be moved to different (free) FU slots without a penalty in performance.

In this paper we focus on exploring the first level of freedom. In order to shuffle operations within an instruction, only a schedule with a certain operation to functional unit assignment is needed, the dependency between operations are not necessary. Hence, a post-compiler tool which shuffles the operations for a given schedule is sufficient to explore the first level of freedom. Section 3 describes the corresponding algorithm and section 4 describes the simulation results.

## 2 Related Work

Conventional clustered VLIW processors are primarily clustered in the datapath, where each datapath cluster derives data from a single register file. In contrast, the functional units in an L0 cluster derive instructions from a single L0 buffer partition. If register files can be viewed as the lowest level storage in the data memory hierarchy, then the L0 buffers can be viewed as the lowest level of storage in the instruction memory hierarchy. Some clustered VLIW architectures like the Lx [9] processor, have a notion of instruction cluster similar to the L0 clusters (except for the L0 buffers). However in their architecture, a datapath cluster and an instruction cluster are synonymous. In contrast, we distinguish between

the two explicitly. For instance, we could employ several L0 clusters within a datapath cluster (or vice versa). In a typical system design, the design criteria and optimizations for instruction and data memory hierarchies are different, but not necessary independent. An overview of several datapath cluster generation tools and techniques are provided in [13]. Most of the scheduling techniques (or operation assignment) for datapath clusters target the register file energy consumption, and they are primarily constrained by the inter-register communication bandwidth [13]. Since instructions need not be transferred from one cluster to another during execution, inter-L0 cluster communication is largely absent. Hence, most of the scheduling techniques for datapath clusters are not directly applicable in the context of scheduling for L0 clusters.

Shuffling of operations within a VLIW instruction has been studied in the recent past [6]. Similar to the work presented in this paper, the main objective is to reduce instruction memory energy consumption. However, they target the instruction cache, while we target L0 buffers and also L0 cluster generation.

### 3 Operation Shuffling

The objective is to evaluate the potentials of operation shuffling and L0 clustering to reduce L0 buffer energy. For this, the L0 cluster generation tool presented by the authors in [12] has been extended to search the combined design space of operation assignment and functional unit grouping. The freedom to shuffle the operations comes from the fact that, within a certain instruction the operations (all of which are scheduled in the same cycle) can be mapped to more than one functional unit, and hence potentially different L0 cluster can be generated. Since the schedule times of the operations are not modified, the process of shuffling does not alter performance.

The problem of assigning operations within an instruction to L0 clusters, and also to simultaneously generate the L0 clusters, is formulated as a 0-1 assignment problem.

$$\begin{aligned} & \min\{ L0Cost(L0clust, Sched, Weight_{BB}) \} \\ & \text{Subject To,} \\ & \sum_{i=1}^{N_{max\ CLUST}} L0clust_{ij} = 1; \forall j \\ & \sum_{i,j} Sched_{bcij} = BB_{cb}; \forall c, \forall b \\ & \sum_{i=1}^{N_{FU}} Sched_{bcij} * FUmap_{ji} \leq 1; \forall j, \forall c, \forall b \\ & \sum_{j=1}^{N_{FU}} Sched_{bcij} * FUmap_{ji} \leq 1; \forall i, \forall c, \forall b \end{aligned}$$

Where,

$$\begin{aligned} L0Clust_{ij} &= \begin{cases} 1 & \text{if } j^{th} \text{ FU is assigned to cluster } i \\ 0 & \text{otherwise} \end{cases} \\ Sched_{bcij} &= \begin{cases} 1 & \text{if } j^{th} \text{ Op is assigned to FU 'i'} \\ & \text{in cycle 'c' of BB 'b'} \\ 0 & \text{otherwise} \end{cases} \\ FUmap_{ji} &= \begin{cases} 1 & \text{if FUs } j, i \text{ can execute same set of ops} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

$Weight_{BB}$  = is an array containing the execution weight of each basic block (BB) mapped on to the L0 buffers

$BB_{cb}$  = is an array containing the number of operations in an instruction cycle 'c', for a certain basic block 'b' mapped on to L0 buffers

$N_{FU}$  = is the total number of functional units

$N_{maxCLUST}$  = is the maximum number of feasible clusters ( $N_{FU}$ , at most each functional unit can be an  $L0cluster$ )

Here, the objective function  $L0Cost(..)$  is the energy consumption of a particular clustering scheme given a schedule and a certain FU grouping. The result of the optimization is a certain FU grouping ( $L0Clust_{ij}$  matrix) and a certain schedule ( $Sched_{bcij}$  matrix) to minimize energy. The resulting schedule is that the operations within each instruction are reassigned to different functional units.

The first constraint is related to clustering. In essence, the constraint ensures that all the FUs are mapped to some cluster and only to one cluster. The second, third and fourth constraints are related to scheduling. The second constraint is to ensure that all the operations in a certain cycle are mapped to some FU. The third constraint ensures that each operation is mapped on to at most one FU. And the fourth constraint is to ensure that for each functional unit at most only one operation is mapped in any instruction cycle.

*Estimation of the Cost Function* : Given the triplet  $\{L0clust, Sched, Weight_{BB}\}$ , the L0 buffer energy consumption can be estimated as follows. In a certain  $Sched$  matrix the summations  $[\sum_j Sched_{bc1j}, \sum_j Sched_{bc2j}, \dots, \sum_j Sched_{bcN_{FU}j}]$  give the FU activation in a certain cycle 'c' and in a certain basic block 'b'. Repeating the same process  $\forall c$ , an FU activation for one iteration of a certain basic block 'b' is obtained. This activation is replicated  $Weight_{BB}(b)$  number of times to give the FU activation of that basic block. By repeating the same process  $\forall b$ , an FU activation trace for the whole program can be obtained. Given a certain  $L0Clust$  and the FU activation trace, the number of accesses to each of the L0 buffer partitions can be obtained by masking each row of the  $L0Clust$  with the FU activation trace. Then, the cost function is calculated using the following equation:  $Cycles * E_{LC} + \sum^{clusters} (access_i * E_{acc-i})$ , where,  $E_{LC}$  is the local controller energy per activation (power),  $access_i$  is the number of accesses to each L0 buffer partition in cluster 'i' and  $E_{acc-i}$  is the L0 buffer energy per access.

*Implementation* : The outline of an implementation of the above formulation is presented in Algorithm 1. Given certain inputs, the algorithm searches for a certain optimum clustering and scheduling over all combinations of clustering and scheduling.

*Initialize()* : All the functional units are assigned into a single cluster. And the  $Sched_{bcij}$  matrix is initialized with the default input schedule.  
*valid\_cluster()* : Over all the possible combinations of the  $L0clust_{ij}$  matrix valid clusters are the ones which satisfy the first constraint in the above formulation.  
*certain\_min\_schedule()* : For every valid clustering the number of combinations of all possible scheduling is very large and grows exponentially with

**Algorithm 1** Clustering and Operation Shuffling Optimization Implementation*Data Structures**Sched*<sub>bcij</sub> // The Schedule matrix*L0clust*<sub>ij</sub> // The Cluster matrix*Inputs**BBProfile* // Basic Block profile (weight and number of ops in each instruction)*LoopProfile* // Loop Profile (the corresponding basic blocks in one loop)*FUmap*<sub>ji</sub> // The matrix indicating functional units which execute same set of operations

```

optimal_schedule_and_clusters(){
  Initialize();
  for_every ( valid_cluster() ){
    certain_min_schedule();
    est_energy = estimate_energy();
    if (est_energy != min_energy) {
      update_clusters();
      update_schedule();
    }
  }
}

```

the total number of instructions. The time complexity of such a trivial search is large enough to make the implementation unrealistic for nominally sized benchmarks. Hence, a heuristic is employed to make a realistic implementation.

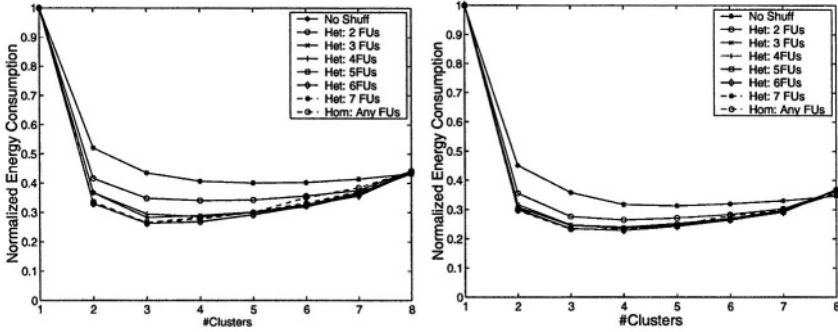
The heuristic is as follows. For every valid cluster, instead of searching among all possible assignments for all the instructions, each instruction is considered independent of others, and an operation assignment is chosen so that it minimizes energy due to that instruction. This sub-optimization essentially translates to mapping of operations on to functional units that are in smaller clusters. A smaller cluster is the one, with a smaller partition of the L0-Buffer<sup>2</sup>. If two or more clusters have similar sized L0-Buffer partitions, then the first cluster in the list is chosen first and then the rest.

Essentially, for every valid cluster, searching for a certain minimum schedule for each instruction independently of others instead of searching among all possible schedules for all the instructions, makes the complexity of the whole search linear in number of instructions. This linear complexity is more realistic than the exponential complexity of the trivial full search.

## 4 Experimental Results

This section outlines some of the results of the above exploration algorithm proposal to shuffle operations. The experimental setup is as follows. Each benchmark is passed through three phases of the tool chain. The first phase is the

<sup>2</sup> Here, we assume that the depths of the L0-Buffer in each partition is the same



**Fig. 2.** L0 Buffer Energy Reduction for Increasing Number of Clusters(a) For Mpeg2 Encoder (b) Averaged Over All Benchmarks

pre-compiler, which maps the interesting parts of that application on to the L0 buffers. The second phase is the compiler suite, which is an extended version of the Trimaran framework [2]. The third phase is the post-compiler, where the tool to shuffle operations and generating clusters is implemented. The underlying architecture under consideration is an unclustered (datapath) VLIW processor with 8 functional units. The energy models of the L0 buffers are derived from Watch [7] for a 0.18 $\mu$ m technology. For each L0 buffer partition in a cluster, the data-array, the address decoder and the local controller are modeled as register files. The applications for evaluation are chosen from a combination of Media-bench [1] and SpecCPU2000 [3] benchmark suites.

Our solution is compared with the original clustering for a given predefined schedule. Figure 2(a) shows the variation of energy with increasing number of clusters for Mpeg Encoder benchmark, and Figure 2(b) shows the variation of energy averaged over all the benchmarks under consideration. #clusters = 1 implies that a single L0 buffer feeds the instruction for all the functional units. #clusters = 8 implies that each functional unit has an L0 buffer partition of its own, whose activation can be controlled by local controllers. The curve corresponding to the legend ‘No Shuff’ represents the energy variation due to clustering for a given default schedule. The remaining curves represent the energy variation due to combined clustering and scheduling. In comparison with ‘No Shuff’ (only clustering), we can see that the results are significantly different. The optimal number of clusters is different and the corresponding L0 buffer energy is reduced significantly. This additional reduction is achieved by exploiting the freedom in assigning operations to functional units in each instruction, instead of assigning randomly.

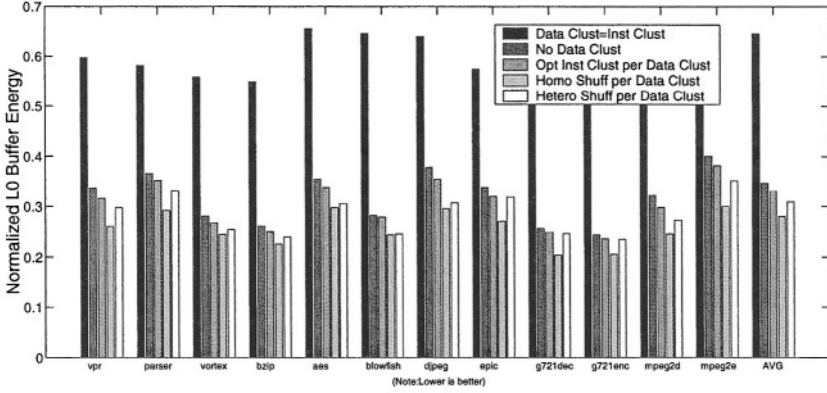
The curve corresponding to legend ‘Het: 2 FUs’ represents the result of clustering and scheduling with the restriction that any operation can be mapped on to at most ‘2’ functional units. Similarly, the curves corresponding legends ‘Het: 3 FUs’, ‘Het: 4 FUs’ and so on, represent the result where any given operation can be assigned to ‘3’ functional units, ‘4’ functional units and so on. As ex-

pected we can see that as we increase the scheduling freedom (#FUs to which an operation can be mapped), the energy reduction also increases. The largest reduction (about 35% additional reduction) is obtained when any operation can be mapped to any of the available functional units. Refer the curve corresponding to legend ‘Hom: Any FUs’ in Figure 2(a). However, the gains with heterogeneous assignment restrictions are still considerable. Thus the gains obtainable in the datapath itself by the heterogeneity can be combined with the gains achieved in the instruction memory hierarchy.

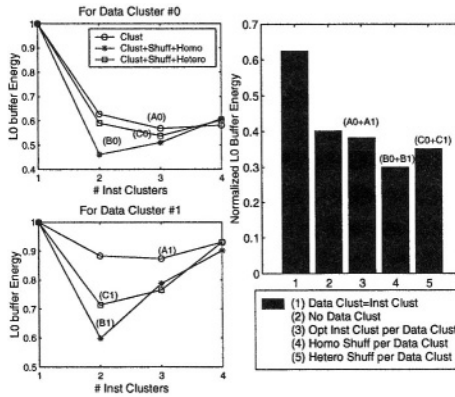
*L0 (Instruction) Clusters and Data Clusters* : In this section we describe some experimental results in the context of data clustered processors. As mentioned in section 2 we explicitly differentiate between L0 and data clusters. We observed that the number of L0 clusters are typically around four to six for a non-data clustered datapath with eight functional units. Here we apply the operation shuffling and clustering scheme to each data cluster. Two data clusters are modeled based on a model similar to commonly used VLIW processors, i.e., two register files and four functional units per register file.

Figure 3, shows the L0 buffer energy reductions for different configurations. The reductions for all the configurations are normalized against a centralized L0 buffer (a single L0 buffer which stores instructions for all the functional units). The various configurations are as follows. (1) ‘Data Clust=Inst Clust’: the L0 clusters (FU grouping) is identical to the two given data clusters. (2) ‘No Data Clust’: the L0 clusters as obtained by the original clustering tool [11] without operation shuffling and assuming no data clusters. (3) ‘Opt Inst Clust per Data Clust’: the L0 clusters as obtained by original clustering tool without operation shuffling, but applied per data cluster. (4) ‘Homo Shuff per Data Clust’: the L0 clusters as obtained by the clustering tool described in section 3, assuming homogeneous functional units and as applied per data cluster. (5) ‘Hetero Shuff per Data Clust’: similar to (4), but the functional units are heterogeneous with limited scheduling freedom based on TMS320C6000 processor from Texas Instruments [16]. The figures indicate that configuration (4) achieves the best possible reduction of about 75% on average. In spite of the limited scheduling freedom in configuration (5), the reductions are higher than in all other configurations except (4), namely about 70% on average. From these results, we see that the realistic constraint of data clustering, which is crucial to decrease the foreground data storage related energy, is fully compatible with the gains related to instruction memory hierarchy.

Figure 4 shows in detail the results of clustering and shuffling applied per data cluster for Mpeg2Encoder benchmark. The sub-figures on the left side show the variation of L0 buffer energy with number of clusters for each data cluster. The three curves in each sub-figure represent clustering only [11], clustering and shuffling (section 2) with homogeneous functional units and clustering and shuffling with TMS320C6000 like functional units, respectively. The reduction for homogeneous functional units is larger than the reduction for heterogeneous functional units. However, the latter is still better than clustering only. The sub-figure on the right side shows the energy reductions for various configurations as described



**Fig. 3.** Energy Reductions of Clustering and Clustering with Operation Shuffling for TMS320C6000 like datapath



**Fig. 4.** L0 Buffer Energy Vs Number of Clusters per Data Cluster (for Mpeg2 Encoder)

in the previous paragraph. Also, it shows how the values have been obtained for data clustered configurations. For instance, in configuration (5), the best possible reductions from clustering and scheduling are obtained by adding values corresponding to points marked C0 and C1 (refer sub-figures on the left side of Figure 4). Similarly, the values for other data clustered configurations are obtained.

The implementation of the optimization formulation as described in the previous section, is heuristic based and hence the presented results might be sub-optimal. For smaller toy-benchmarks it was observed that the full search results were much lower than the ones obtained for the heuristic based implementation. Due to the inherent complexity of the search, applying full search for the larger benchmarks used by us, was not feasible. Nevertheless, the substantial reductions indicate the clear value of shuffling operations along with clustering.



## 5 Summary and Future Work

In summary, this paper presents a study indicating the potentials offered by scheduling for L0 clusters in terms of energy reduction. Operations in a given instruction are reassigned to different clusters. The simulation results indicate that potentially up to 75% of L0 buffer energy can be reduced by shuffling operations, and also substantial reductions can be achieved for data clustered processors.

Current scheduler for VLIW processors employs modulo scheduling in one form or another. Extending a modulo scheduler for L0 clusters would be the next topic for our future work. Datapath clusters and L0 clusters can coexist. However, current schemes for generating datapath clusters and L0 clusters are mutually exclusive, and the resulting clusters might be in conflict. The synchronicity between datapath and L0 clusters needs to be investigated as well.

## References

1. *Mediabench*. <http://www.cs.ucla.edu/leec/mediabench>.
2. *Trimaran: An Infrastructure for Research in Instruction-Level Parallelism*. <http://www.trimaran.org>, 1999.
3. *Standard Performance Evaluation Corporation: SPEC CPU2000*. <http://www.spec.org>, 2000.
4. R. S. Bajwa, M. Hiraki, H. Kojima, D. J. Gorny, K. Nitta, A. Shridhar, K. Seki, and K. Sasaki. Instruction buffering to reduce power in processors for signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(4):417–424, December 1997.
5. L. Benini, D. Bruni, M. Chinosi, C. Silvano, and V. Zaccaria. A power modeling and estimation framework for vliw-based embedded system. *ST Journal of System Research*, 3(1):110–118, April 2002. (Also presented in PATMOS 2001).
6. A. Bona, M. Sami, D. Sciuto, V. Zaccaria, C. Silvano, and R. Zafalon. An instruction-level methodology for power estimation and optimization of embedded vliw cores. In *Proc of Design Automation and Test in Europe (DATE)*, March 2002.
7. David Brooks, Vivek Tiwari, and Margaret Martonosi. Watch: A framework for architectural-level power analysis and optimizations. In *Proc of the 27th International Symposium on Computer Architecture (ISCA)*, pages 83–94, June 2000.
8. T. Burd and R. W. Brodersen. *Energy Efficient Micorprocessor Design*. Kluwer Academic Publishers, January 1992 (1st Edition).
9. P. Faraboschi, G. Brown, J. Fischer, G. Desoli, and F. Homewood. Lx: A technology platform for customizable vliw embedded processing. In *Proc of 27th International Symposium on Computer Architecture (ISCA)*, June 2000.
10. Margarida F. Jacome and Gustavo de Veciana. Design challenges for new application-specific processors. *Special issue on Design of Embedded Systems in IEEE Design & Test of Computers*, April-June 2000.
11. Murali Jayapala, Francisco Barat, Tom Vander Aa, Francky Catthoor, Geert Deconinck, and Henk Corporaal. Clustered 10 buffer organization for low energy embedded processors. In *Proc of 1st Workshop on Application Specific Processors (WASP), held in conjunction with MICRO-35*, November 2002.

12. Murali Jayapala, Francisco Barat, Pieter OpDeBeeck, Francky Catthoor, Geert Deconinck, and Henk Corporaal. A low energy clustered instruction memory hierarchy for long instruction word processors. In *Proc of 12th International Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS)*, September 2002.
13. Viktor Lapinskii, Margarida F. Jacome, and Gustavo de Veciana. Application-specific clustered vliw datapaths: Early exploration on a parameterized design space. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 21(8):889–903, August 2002.
14. Lea Hwang Lee, William Moyer, and John Arends. Instruction fetch energy reduction using loop caches for embedded applications with small tight loops. In *Proc of International Symposium on Low Power Electronic Design (ISLPED)*, August 1999.
15. Texas Instruments Inc., <http://www.ti.com>. *TMS320C6000 Power Consumption Summary*, November 1999.
16. Texas Instruments Inc, <http://www.ti.com>. *TMS320C6000 CPU and Instruction Set Reference Guide*, October 2000.

# On Combined DVS and Processor Evaluation

Anders Brødløs Olsen<sup>1</sup>, Finn Büttner<sup>2</sup>, and Peter Koch<sup>1</sup>

<sup>1</sup> Center for Indlejrede Software Systemer (CISS)

Aalborg University, Denmark

{abo, pk}@kom.aau.dk

<sup>2</sup> Analog Devices

Aalborg, Denmark

finn.buettner@analog.com

**Abstract.** Numerous Dynamic Voltage Scaling (DVS) methods have been proposed over the years, and since several commercial programmable processors supporting DVS are now available, the demand for DVS evaluation tools is obvious. In this paper we propose a simulator based on soft- and hardware parameters that make it possible to get realistic performance evaluations from a high abstraction level. Traditionally, DVS methods have been tested using unrealistic assumptions, both from a software and hardware point of view, but this problem is now alleviated due to our simulator. We show that DVS methods are very depended on the application specification as well as the processor parameters.

## 1 Introduction

Over the last years an intense interest for portable devices has highlighted a need for power aware design. The increase in computational complexity and the ever increasing demand for longer battery life time have initiated an urge for energy reducing methods. This is also driven by the ever larger gap between battery capacity, and the energy consumption by high performance processors, displays, and other modern peripherals. One promising low power technique for programmable processors is Dynamic Voltage Scaling (DVS), which belong to the category of Dynamic Power Management (DPM). DPM represent a number of techniques which use dynamic properties for bringing different system components e.g., processor, and memories into various power modes. The reason for DVS being efficient is due to the fact that power consumption for CMOS technology is a quadratic function of the supply voltage ( $P \approx f_{clock} \cdot V_{dd}^2 \cdot C_L$ ). At the same time, however, the supply voltage and the clock frequency is related by a time delay restriction ( $t_{delay} = \frac{1}{V_{dd}}$ ), and thus we use the notation “speed” throughout the paper. The basic idea of DVS is to change the processor speed according to the utilization state, while still maintaining application timing constraints.

Research within DVS mainly explores the possible energy reduction benefit from a theoretical point of view. When calculating a new speed, almost all DVS

algorithms requires knowledge of the future, e.g., the remaining processing time for the current task, time to next event, and deadline time for the task. In practical software implementations, this required knowledge is a challenge to derive, mainly due to two reasons;

- “*Software knowledge*” both means static and dynamic knowledge, e.g., how many cycles are needed to conduct a certain task, and what is the size of the remaining part of a task already executing. This knowledge requires wide state knowledge, e.g., is the required memory available at the right time, or has a task to await other tasks to free memory first.
- “*Application knowledge*”. E.g., in a GSM cellular phone the monitoring of a paging channel is a well known cyclic task. However when a paging is actually received, or the user suddenly presses hook-off, waves of tasks will be launched, and the task pattern becomes unknown. The GSM application also includes some major timing shift, e.g., handover is violating the cyclic behavior.

Based on these considerations of practical implementations, we see that there are much to be said before the DVS methods are suitable for industrial utilization. First and foremost, how do we provide accurate estimates of task execution times for software implemented in dynamic environments, and what about the timing overheads caused by the DVS facilitating processors? In this paper we will give some of the answers.

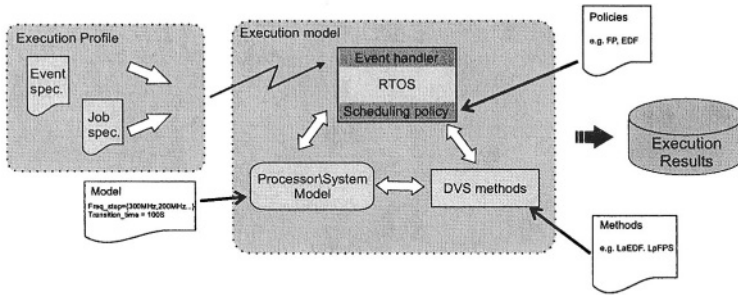
## 2 Simulating DVS Methods

When surveying the literature it becomes obvious that frameworks for testing DVS methods are lacking. To the best of our knowledge there has been only one attempt to make a uniform test environment, the SimDVS [1], [2]. Normally, researchers have tested their DVS algorithms in ideal environments where the typical assumptions are; 1) the processor “speed” can have any value within a given interval, and 2) speed transitions are free of cost. In most case we see no considerations of the overhead introduced by; 1) the DVS algorithm, 2) the underlying operating system, and 3) the DVS related hardware parameters.

The contribution of our work is to make a simulator where such soft- and hardware parameters can be evaluated. In Figure 1, the principle design of our simulator is illustrated.

### 2.1 Proposed DVS Methods

Over the years numerous DVS methods have been developed, and basically they can be categorized into two groups, the Intra- and Inter-task DVS methods [2]. The Intra-task DVS methods, requiring minimal runtime overhead, are operating within the individual tasks and are typically based on offline analysis of task graphs, providing scaling points and speed settings, [3], [4], the advantage being that no operating system is needed. The problem is that the methods



**Fig. 1.** The overall structure of our simulator. The *execution profiler* is a task-set or application specification, which either can be manually specified or randomly generated. The *execution model* is mimicking an actual processor with an RTOS, DVS method, and the important processor/system model.

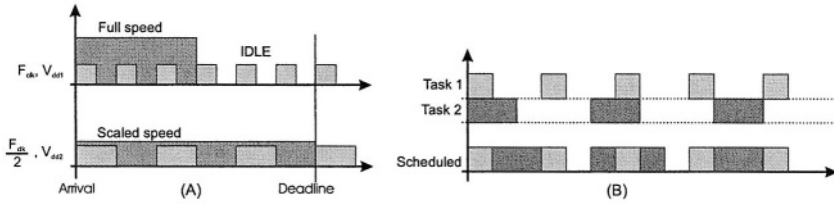
heavily depend on accurate timing analysis of the task execution times, which make them inflexible when software is changed. On the contrary, together with an operating system, Inter-task DVS methods are working in a “task by task” manner. Typical Inter-task DVS methods are based on traditional scheduling policies, such as Fixed Priority (FP) or Earliest Deadline First (EDF) where the speed scaling is conducted using e.g. task-set knowledge, [5], [6], [7], [8]. The overhead of these methods is therefore also larger as compared to Intra-task DVS methods, but are more flexible for software changes. When surveying the DVS literature one observe that the minority of the publications deal with Inter-task DVS algorithms.

In this work we have chosen to present experimental results from three different proposed DVS methods. The **l**ow-**p**rower **p**riority based real time **s**cheduling using EDF (lppsEDF) [5], the **c**ycle **c**onserving EDF (ccEDF) [6], and the **l**ook **a**head EDF (laEDF) [6]. The reason for choosing these methods is twofold; the experimental simplicity, and because most used approaches are represented; stretching to next task arrival, and slack time passing.

## 2.2 DVS Methods and Software Parameters

All DVS methods need information from the task-set. To get an understanding of the software parameters, a simple example is considered. In Figure 2(A) the basic idea of DVS is illustrated. The task is specified by; 1) arrival time, 2) deadline time, and 3) workload noted in Worst Case Execution Time (WCET) or Worst Case Execution Cycles (WCEC). In Figure 2(B) the scheduling of a task-set using an Earliest Deadline First (EDF) is illustrated. The idea of the Inter-task DVS methods is to convert the task-set slack time into energy savings by lowering the processor speed, while maintaining the task-set timing constraints.

The software parameters can be divided into two main groups, event and job parameters. Most software is event driven, basically being described by periodic,



**Fig. 2.** (A) Basic idea of speed scaling. (B) Task-set scheduling, using an EDF policy.

aperiodic, or sporadic natures. In this work we will look into periodic task-sets only, since the DVS methods chosen only consider this. Job parameters are related to task workload, basically being a cycle count. The workload can be specified using a constant or variable value, where the variable value is defined by a Best to Worst Case Ratio (BWCR).

### 2.3 DVS Methods and Hardware Parameters

When DVS is applied, additional hardware is needed in order to enable the speed transitions. This hardware is typically a programmable PLL and a programmable DC-DC converter. Important parameters are the number of available speed settings, and the time it takes to conduct a transition.

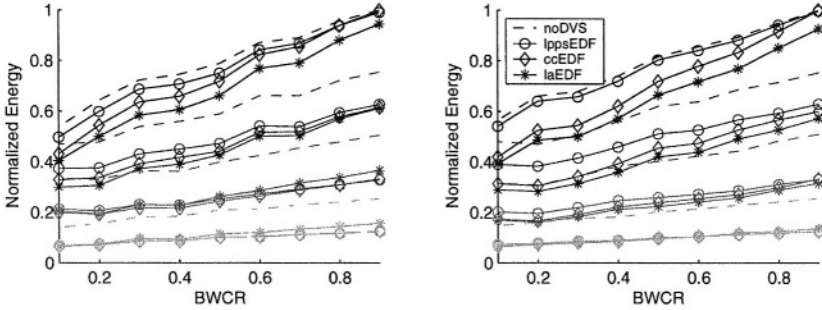
In our work the power consumption in the various speed steps is modelled using measurements from an actual processor platform (Analog Devices Blackfin BF535 EZ-KIT Lite platform ). The model is compatible with the traditional power function, and we assume equal instruction execution times for each speed step despite potential utilization of e.g., memory hierarchies.

## 3 Experimental Results, Software Issues

In our quest of evaluating DVS performance, a number of experiments related to software parameters (this section), and related to hardware parameters (next section) are conducted. Two parameters important for the workload are 1) task-set utilization, and 2) the BWCR.

### 3.1 Task-Set Utilization and BWCR

As the task-set utilization and BWCR determines the actual workload we will explore the effect of the two parameters on the three selected DVS methods. In our experiments we compare the DVS methods with an implementation not using DVS, but running at full processor speed and then IDLE'ing during the slack times. Results are presented in percentage energy saved as compared to the noDVS implementation. The experiments are made with four different levels of utilization {1, 0.75, 0.5, 0.25}, and a BWCR in the interval [0.1;0.9]. In Figure 3,



**Fig. 3.** On the left and right hand side, respectively, simulations of the different DVS algorithms are shown on a task-set with 5 and 20 tasks. Four different utilization factors are employed, conspicuous as the four groups on the y-axis.

simulations for tasks-sets with 5 and 20 tasks are illustrated, and task-set timing in the interval  $[2;40]$ ms.

It is observed that the number of tasks has an impact on the DVS performance, e.g., the lppsEDF are obtaining an energy saving of 7.5% for 5 task case and 4.2% for 20 task case, with utilization equal to 1 and BWRC 0.1. The utilization also influences the DVS performance, e.g., using a utilization of 0.5 and BWCR of 0.9 the results are 35%, 34%, and 27% respectively, for the three DVS methods and in the 5 task case. This indicates that for low utilization the choice of DVS method becomes less important.

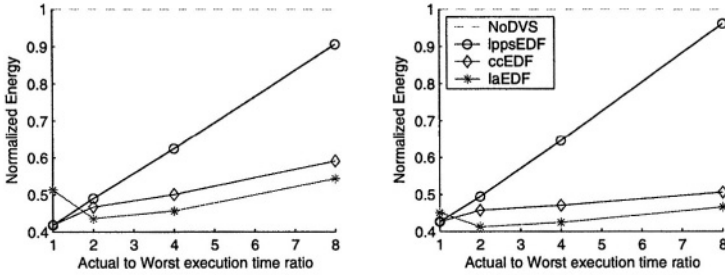
The BWCR also influences the individual DVS methods differently. The lppsEDF performs relatively poor for a high utilization. However, if the utilization is 1 and the BWCR is 0.9 and 0.1, respectively, the lppsEDF saves {1%, 7.4%} for the 5 task case and {0.3%, 4.2%} for the 20 task case. The laEDF saves {5.5%, 24.5%} for the 5 task case and {7.3%, 30.3%} for the 20 task case, showing that the optimistic nature of the laEDF benefits from a large BWCR.

The conclusion is that one must be careful choosing the DVS algorithm against the task-set specification; task-set utilization and BWCR will significantly influence the DVS performance.

### 3.2 WCET Estimation

WCET is of overall importance when the DVS methods compute the scaling factor. When the actual task execution cycles are larger than the WCET it is obvious that task will miss a deadline, which is not acceptable. The WCET must therefore be estimated either very precisely or overestimated.

From Figure 4 it is observed that the DVS algorithms perform differently when the WCET is estimated as 1 to 8 times the actual task execution cycles. In the 5 task case, the lppsEDF saves from 58.1% to 9.4% compared to the noDVS implementation, and the ccEDF saves from 58.1% to 40.9%. For the 20 task case, comparable results are 57.3% to 3.8% and 57.3% to 49.4%. These



**Fig. 4.** Experiments with a utilization of 0.125, and BWRC equal to 1. To the left 5 task, and to the right 20 tasks. Soft- or hardware overheads, are neglected.

results show that the lppsEDF depends very much on the WCET where as the ccEDF, though losing some performance, still maintain significant performance improvements.

The conclusion is that the accuracy of the WCET estimations have a serious impact on the choice of DVS methods.

## 4 Experimental Results, Hardware Issues

On DVS facilitated processors there are mainly two important hardware components that must be considered; 1) the supply voltage regulator, and 2) the processor clock generator, where both are limited by constraints such as number of speed steps and transition times.

### 4.1 Number of Speed Steps

In this experiment, the influence of having different speed settings is investigated using the following three settings;

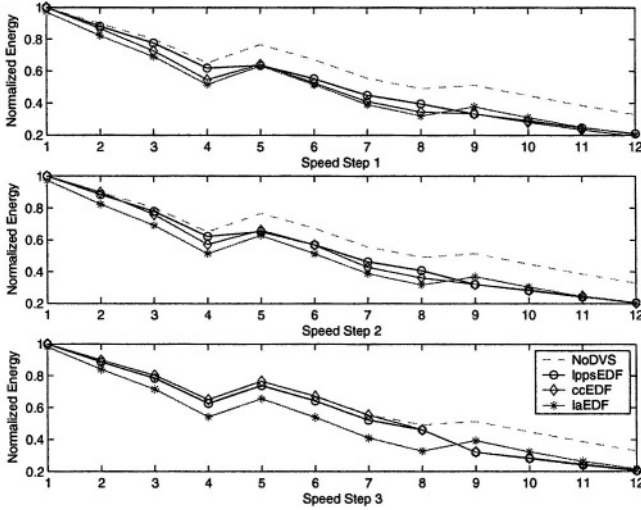
1. 20 to 300Mhz with 20MHz steps
2. 50 to 300MHz with 50MHz steps
3. 300MHz multiplied by  $[\frac{1}{1}, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}]$

From Figure 5 it is observed that the difference between speed step 1 and 2 are minor, e.g., at utilization 0.75, and BWCR equal to 1 (fifth point on the x-axis), the saving is 17.5% for speed setting 1, and 17.9% for speed setting 2 using laEDF. This indicates that a large number of speed steps are not necessarily more beneficial as compared to a lower number.

For speed step 3, the lppsEDF and ccEDF cannot gain significant saving as compared to the noDVS implementation. Again for utilization 0.75, and BWCR equal to 0.75, the savings are; 3.5% for lppsEDF, 0% for ccEDF, and 14.3% for laEDF.

Again, the conclusion is that a large number of speed steps is not necessarily better, but in addition we may say that the speed step values are impacting the DVS performance.

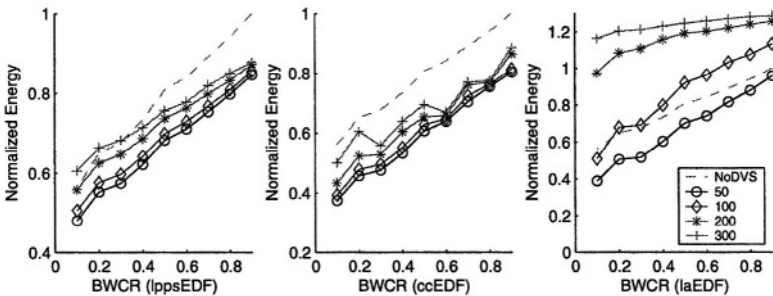




**Fig. 5.** The task-sets are random generated using 5 tasks, with a utilization of  $\{1, 0.75, 0.5\}$  and a BWCR of  $\{1, 0.75, 0.5, 0.25\}$ . The x-axis represent the task-set with the first point representing *utilization 1 and BWCR 1*, the second representing *utilization 1 and BWCR 0.75*, and so on.

### 4.2 Transition Overhead

In previous DVS publications, the transition overhead value has unrealistically been considered negligible, but in our experiments we will show that the overhead plays a significant role in relation to the task-set specification. Available DVS-facilitated processors have transition overhead from  $20\mu$  to  $300\mu s$ , [9], [6], and thus Figure 6 illustrates the results of our experiments with overhead settings of  $\{50, 100, 200, 300\}\mu s$ .



**Fig. 6.** Experiments with 20 task randomly generated task-sets restricted by the time period  $[2, 40]ms$ , utilization of 0.75, and BWCR in the interval  $[0.1;0.9]$ .

The lppsEDF and ccEDF methods provide energy saving as compared to the noDVS implementation. When the laEDF is used, more energy, as compared to execution without DVS, is consumed. One reason is that the laEDF is inserting significantly more scaling points than the other two methods. Comparing laEDF to lppsEDF, the increase in scaling points are by factor 9.4, 8.1, 5.2, and 5, respectively for the four overheads.

An important issue in relation to transition overhead is the task-set timing. The problem arises when the shortest event period becomes almost equal to the transition time. In today's DVS-facilitated processors, a speed transition causes an IDLE period, meaning that potential execution time is lost.

The conclusion is that the transition overhead combined with the number of scaling points and task-set timing can cause significant DVS performance degradation.

## 5 Selecting the Right DVS Method

A number of soft- and hardware parameters relevant when evaluating a DVS method have now been investigated. In order to narrow down the most efficient DVS method, 3 different applications, and 3 different hardware platforms are now specified for our final experiment, see table 1.

**Table 1.** The application specifications, and the platform specification all using the power model extracted from the Blackfin BF535.

	Application 1	Application 2	Application 3
Task number	10	15	20
Time distribution [ms]	[0.1 10]	[1 100]	[10 500]
Utilization, WBRC	0.75, 0.5	0.75, 0.5	0.75, 0.5

	Platform 1	Platform 2	Platform 3
Speed steps [MHz]	[20,40,...,300]	[50,100,...,300]	$300 \cdot \{\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, \frac{1}{1}\}$
Transition Overhead [ $\mu$ s]	50	200	0

In table 2, the simulation results are illustrated, and the results are normalized to a simulation without DVS for the individual applications. Application 1 on platforms 1 and 2 are exceeding deadlines, and we will discard them as valid solutions. When comparing the application timing and the transition overhead, we observe that the two are almost equal, simply not allowing any speed transitions. Platform 3 is able to handle all three DVS methods with laEDF as the most optimal. For application 2 the best implementation is laEDF on platform 3, but if the other platforms are considered, the ccEDF is obtaining the best performance. For application 3, the laEDF is still superior but only on platform 1, and we also observe that laEDF is the best choice on all platforms (the ccEDF on platform 2, though, has identical energy consumption).

**Table 2.** Application, platform simulation results. The Energies are normalized to a simulation without DVS. If deadlines are missed, the solution is not valid.

lppsEDF	Application 1			Application 2			Application 3		
	Plat. 1	Plat. 2	Plat. 3	Plat. 1	Plat. 2	Plat. 3	Plat. 1	Plat. 2	Plat. 3
Energy	1.001	1.097	0.963	0.842	0.891	0.981	0.827	0.858	0.969
Deadline Miss	yes	yes	no	no	no	no	no	no	no

ccEDF	Application 1			Application 2			Application 3		
	Plat. 1	Plat. 2	Plat. 3	Plat. 1	Plat. 2	Plat. 3	Plat. 1	Plat. 2	Plat. 3
Energy	1.084	1.189	0.999	0.756	0.777	0.998	0.735	0.760	0.999
Deadline Miss	yes	yes	no	no	no	no	no	no	no

laEDF	Application 1			Application 2			Application 3		
	Plat. 1	Plat. 2	Plat. 3	Plat. 1	Plat. 2	Plat. 3	Plat. 1	Plat. 2	Plat. 3
Energy	1.56	1.501	0.717	0.812	1.303	0.736	0.708	0.760	0.734
Deadline Miss	yes	yes	no	no	no	no	no	no	no

Rather interestingly, based on the experiments its safely to say that there is no “best DVS method”, and based on the application and the hardware used a careful chose must be made. However, our robust tool helps software designers to evaluate whether a DVS implementation is feasible for a given system, making it much easier and reliable to decide on a potential utilization of DVS.

## 6 Conclusions

The out-set of our work was to 1) identify what software parameters that are important, and 2) the influence of hardware parameters. Answering these questions has involved the design and implementation of a simulator with the capability to evaluate DVS schedules using a task-set specification and a user defined hardware model.

Using our simulator we have shown that software parameters have a non-trivial influence on DVS performance, especially an inaccurate estimation of WCET will for some DVS methods degraded the performance significantly. The hardware parameters have also been shown to influence DVS performance. They are, however, very depended on the task-set timing. Unfortunately, these parameters are, in most DVS evaluations, severely neglected, but we have shown that they must be considered in relation to the task-set specification. Based on our experiments we can now safely say that the following basic issues must be considered before using DVS:

- task-set specification (utilization, BWCR) related to DVS method
- WCET estimation for the task-set
- task-set timing related to transition time
- task-set timing related to speed steps

One design criteria for our simulator was the ability to replace parts of it. This is an important feature, since in this work, we have not considered the software overheads introduced by an RTOS, and the DVS itself. The hardware model only uses DVS-related components, e.g., speed steps, and transition times, but overheads from non-directly related hardware issues must also be considered. Such soft- and hardware related items can easily be represented in our simulator. Also we believe that issues like multi-level memory architectures and caches could be significant as they prolong the execution time of tasks.

For future work there is a huge effort evaluating and designing DVS methods that can handle more complex event structures, since many of today's applications include a mix of periodic, aperiodic and sporadic task patterns, the classical example being cellular phones. Another very important issue is the estimation of WCET. How is execution time analysed in dynamic software environments, and on processor architectures that are depended on its previous state. Some of these questions are currently being investigated in our labs.

## References

1. D. Shin, W. Kim, J. Jeon, J. Kim: "*SimDVS: An Integrated Simulation Environment for Performance Evaluation of Dynamic Voltage Scaling Algorithms*" Proceedings of Workshop on Power-Aware Computer Systems (PACS) (Feb. 2002) 98-113
2. W. Kim, D. Shin, H. S. Yun, J. Kim and S. L. Min: "*Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems*" Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) (Sep. 2002) 219-228
3. D. Shin, J. Kim, and S. Lee: "*Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications*" IEEE Design and Test of Computers, 18(23) (Mar. 2001) 20-30
4. F. Gruian: "*Hard Real-Time Scheduling Using Stochastic Data and DVS Processors*" Proceedings of International Symposium on Low Power Electronics and Design (2001) 46-51
5. Y. Shin, K. Choi, T. Sakurai: "*Power Optimization of Real-Time Embedded Systems on Variable Speed Processors*" Proceedings of International Conference on Computer-Aided Design (2000) 365-368
6. P. Pillai, K.G. Shin: "*Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems*" Operating Systems Review (ACM), 18th ACM Symposium on Operating Systems Principles (SOSP'01) (2002) 89-102
7. H. Aydin, R. Melhem, D. Mosse, P.M. Alvarez: "*Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems*" Proceedings of Real-Time Systems Symposium (2001) 95-105
8. W. Kim, J. Kim, S.L. Min: "*A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis*" Proceedings of the Design Automation and Test in Europe (DATE) (Mar. 2002), 788-794
9. T. Mudge: "*Power: A First Class Design Constraint for Future Architectures*" IEEE Computer, 32(4) (Apr. 2001) 52-58

# A Multi-level Validation Methodology for Wireless Network Applications\*

C. Drosos, L. Bisdounis, D. Metafas, S. Blionas, and A. Tatsaki

INTRACOM S.A.

19.5 Km Markopoulo Ave., GR-19002 Peania, Athens, Greece  
{cdro, lmpi, dmeta, sbli, atat}@intracom.gr

**Abstract.** This paper presents the validation methodology established and applied during the development of a wireless LAN application. The target of the development is the implementation of the hardware physical layer of the HIPERLAN/2 wireless LAN protocol and its interface with the upper layers. The implementation of the physical layer (modem) has been validated in two different levels. First, at the functional level, the modem was validated by a high-level UML model. Then, at the implementation level, a new validation framework drives the validation procedure at three different sub-levels of design abstraction (numerical representation, VHDL coding, FPGA-based prototyping). Using this validation methodology, the prototype of the HIPERLAN/2 modem has been designed and validated successfully.

## 1 Introduction

In wireless data communications there have been many standardization efforts in order to meet the increased needs of users and applications. In the 5 GHz band, there are the IEEE 802.1a [1] and the HIPERLAN/2 [2], both specified to provide data rates up to 54 Mbps for wireless LAN applications in indoor and outdoor environments. The market for these applications is growing rapidly and it is estimated to reach more than 61 million wireless products shipments by 2006 [3]. Both aforementioned standards operate in the same frequency band, and utilize orthogonal frequency division multiplexing (OFDM) for multicarrier transmission [4].

The purpose of this paper is to present a new multi-level validation methodology that was applied during the development of a HIPERLAN/2 modem. For the development of this modem an innovative flow was followed. This flow is based on the use of UML and its modeling capabilities for the high-level design and validation of the system. The flow is presented graphically in Fig. 1. Furthermore, the presented flow offers significant support for the validation of the hardware parts of the system, and more specifically, the implementation of the physical layer (modem) at three different levels of design abstraction (mathematical analysis, HDL design, FPGA prototype). These increased levels of support are achieved through a validation framework that integrates the validation of the modem with the validation of the high-level system model and its validation.

---

\* This work was partially supported by the IST-2000-30093 EASY project.

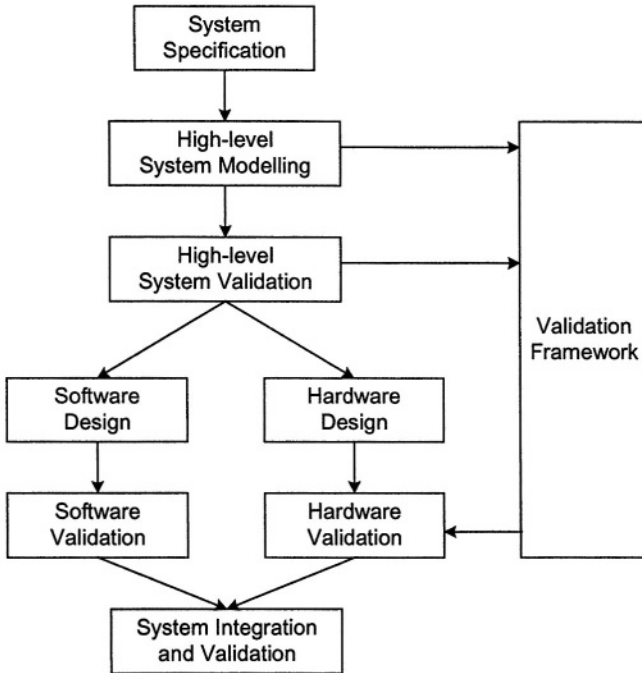


Fig. 1. Design and validation flow for the implementation of the HIPERLAN/2 system

Other proposed methodologies for telecommunication systems (such as the one presented in [5]) do not offer support for the validation of the modem's design at different levels of design abstraction (system model, mathematical analysis, HDL design, FPGA prototype), even if they propose UML for the modeling of the system, as in the design methodology presented in [6]. The use of UML for the modeling and validation of the modem, as it is proposed in this paper, boosts its development significantly and automates the validation process.

## 2 High-Level Modem Modeling

The major problem in the design of a complicated system, such as a wireless LAN modem, is the verification of the design and the early detection of design faults [7]. For this reason a UML based flow for the system design of the modem was followed in order to produce an executable system specification (virtual prototype) for early verification of its control functions. This virtual prototype is based on a UML model, which uses the UML-RT profile [8].

The high-level model of the system (including the functionality of the modem) can be used to early validate the design of the HIPERLAN/2 system. The validation scenarios are based on the captured system-level use case and sequence diagrams. The structure of the UML based system model is presented in Fig. 2.

Within the structure of the AP model there is a group of three objects that model the interface of the modem with the rest system, as well as the behavior of the mo-

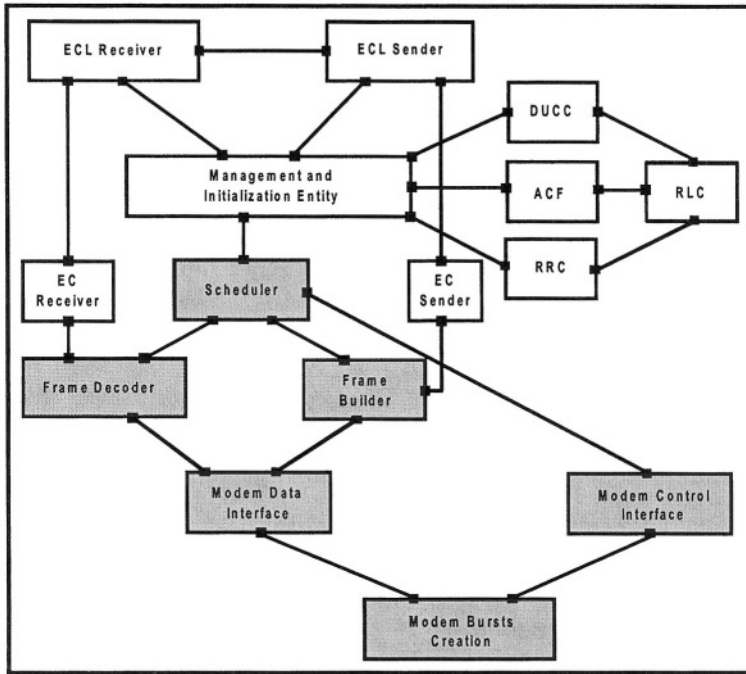


Fig. 2. Object structure of the high level model of the HIPERLAN/2 system

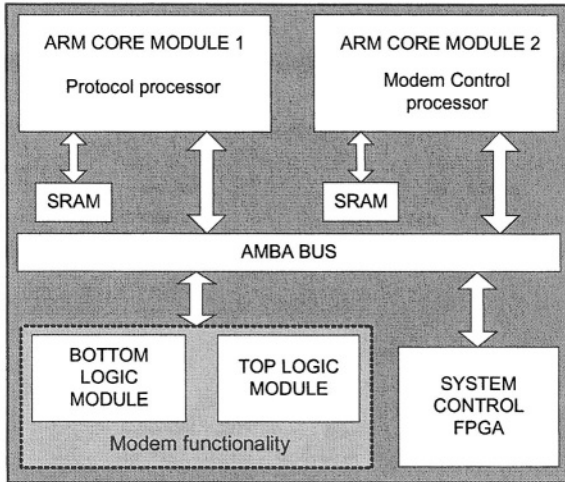
dem's control parts. These objects are the "modem data IF", the "modem control IF", and the "modem bursts creation". The first object models the behavior of the data manipulation parts of the modem's interface (memories and registers), the second object models the behavior of the control parts of the interface (commands accepted by the modem, interrupts), while the third object models the behavior of the modem (creation of broadcast, downlink and uplink data bursts).

The remaining objects inside the system structure (Fig. 2) are specific to the upper layers of the protocol stack and the custom implementation of the MAC layer and the error control mechanisms.

### 3 Modem's Hardware-Software Mapping and Implementation

An important task during the design of the modem is the mapping of its processes to the used instruction-set processors (i.e. software implemented processes) and to custom hardware (i.e. hardware implemented processes), as well as the determination of the interface of the processes implemented in software with those implemented in hardware. The mapping scheme that is used in this development is heuristic (based on our previous experience from similar systems' development). The architecture of the platform (ARM Integrator) in which the modem's prototype is implemented was selected in order to realize the produced mapping scheme of the overall system.

The prototype platform mainly consists of two ARM processor cores for the realization of the processes that were mapped for implementation in software, and FPGA



**Fig. 3.** Block diagram of the system mapping on the ARM Integrator

devices for the implementation of the hardware parts of the modem. The processor cores are responsible for the execution of the upper layers of the HIPERLAN/2 protocol stack and the software parts of the modem. The physical layer of the protocol (modem's data path and control units) was implemented onto the FPGA devices of the prototype platform. Details on the architecture of the overall system implementing the HIPERLAN/2 wireless LAN standard can be found in [9], while the mapping of the system to the various elements of the selected architecture is presented in Fig 3.

For the implementation of the interface between the modem and the HIPERLAN/2 DLC layer, a fully programmable hardware unit has been designed, capable of executing a set of suitable instructions. The selection of the instruction set of the modem is based on the results of alternative instructions sets using the high-level, UML based, virtual prototype of the system. The interface block of the modem is presented in Fig. 4.

The interface of the modem's hardware with the software parts of the modem includes a number of memory buffers, a number of interrupt signals and a command set. The command set of the modem includes suitable commands for transmission and reception of the various physical layer packets and commands for controlling the RF interface. Apart from the commands, the interface unit includes a number of storage buffers (e.g. command memory, transmit and receive data memories) and a series of interrupts (e.g. synchronization, end of receive, end of transmit), as acknowledgements to the protocol's requests.

## 4 Validation Framework

The validation of the design of the modem is performed in three different phases. The first one concerns the validation at the algorithmic level, where both receive and transmit algorithms of the modem are developed and verified at the numerical level. The second phase concerns the VHDL development environment and its simulator



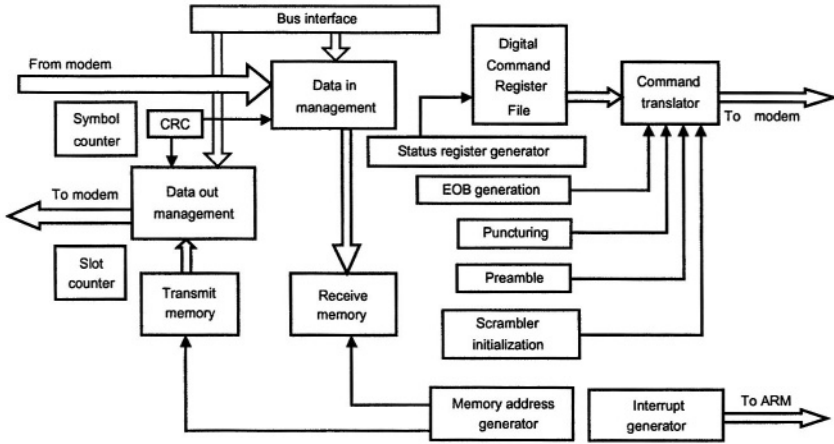


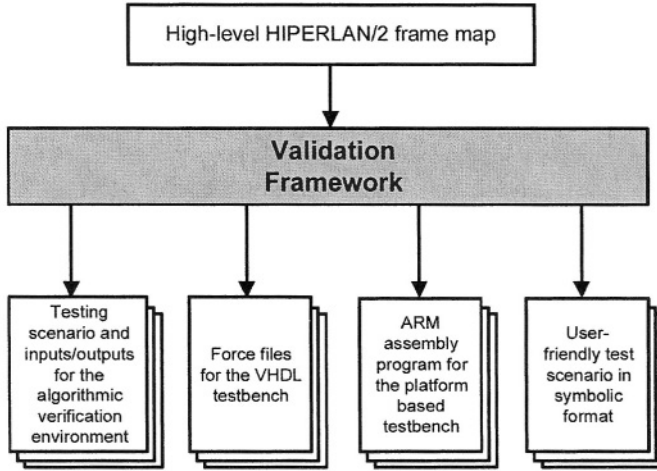
Fig. 4. Block diagram of the modem's interface

(modem design), while the third validation phase concerns the reconfigurable hardware and its accompanying platform (modem implementation).

Before the validation of the modem's algorithms, for the purposes of system validation, an executable specification model of the system has been developed, using UML. This model helps the validation of the specifications of the system early in the design process. An important artifact of the UML modeling is a set of system usage scenarios, captured formally using the UML's modeling techniques. These usage scenarios, accompanied with sequence diagrams formally presenting each one, are a helpful resource for the validation of the system's design at various levels of abstraction. This model is also used to validate the design of the system's software, using as input the formally captured system specifications in the form of sequence diagrams.

The novel approach to the validation of the hardware design of the system, and more specifically the modem, is to use the formal validation techniques offered by UML modeling for this purpose, and the developed UML model of the system. The idea is to use the same sequence diagrams and the same UML system's model for the validation of the hardware blocks of the modem, suitably modified to meet the nature and specific requirements of hardware validation. The way to achieve this is to create a methodology and a validation framework that takes as its input usage scenarios from the system level and produces validation patterns for all the three modem's validation phases. In order for the validation to be accurate, the transformation of the high-level usage scenarios to low-level test patterns for the validation of the modem has to be automatic and formal.

In order to automate and facilitate the method for generating test input for all the three phases of the modem's validation from the high-level system scenarios, a custom validation framework was developed. This framework provides an automatic connection of the whole system's validation with the validation of the modem at the various phases of its development process. The operation of the developed framework is presented graphically in Fig. 5. Its major advantage is that from a given high-level frame description it can produce test patterns for the three validation phases that are consistent, and perform the same validation scenario at the corresponding validation environment.



**Fig. 5.** Modem's validation framework

As input to the validation framework, we use a description of the map of the HIPERLAN/2 frame at a high-level format. These descriptions are captured during the execution of the UML model of the system, under the stimulus of the usage scenarios in the form of sequence diagrams. During the execution of these testing scenarios by the UML system model the outputs of the scheduler class of the system are logged. These logs contain the maps for a selected number of MAC frames, as these maps have been generated by the central scheduler of the system and its policies for allocating transmission resources. The validation framework is able to process these types of high-level frame information to produce test patterns for all three phases of the modem's validation.

The first phase of the validation of the modem concerns the validation of receive and transmit algorithms in a MATLAB environment. An ad-hoc testbench has been created inside this environment to validate the developed algorithms. This testbench can be driven by a simulation scenario file and data files containing data to be processed by the algorithms. All these types of files are generated automatically by the validation framework to efficiently validate the algorithms of the modem with the same operating scenarios as the ones used for the validation of the system's UML model.

The second validation phase supports the VHDL-based design of the modem. Inside the context of this phase, a VHDL-based testbench has been developed to help the verification task. The role of the validation framework during this phase is to produce input files and to support the operation of this VHDL-based testbench. More specifically, the framework produces force files that fill the memory blocks of the modem (configuration, transmit and control command memories) with valid contents, according to the high-level frame description. Furthermore, inputs and outputs of each algorithmic block are produced during the validation of the algorithms at the numerical level (MATLAB environment), which can also be used for the verification of the VHDL-based design, in direct comparison with the algorithmic results.

The last phase of the modem's validation concerns the FPGA-based prototype platform. The platform contains the reconfigurable blocks that realize the modem's cir-

cuitry and the system's processors. The validation phase at this platform is performed through the hardware-software interface of the modem, and is driven by the processor that is responsible for the modem control. The role of the validation framework during this phase is to produce automatically the processor source code that will drive the validation, by using the high-level frame description. The code must fill all modem's memories with valid data specific to the validation scenario and control the execution of this scenario.

In order to present the detailed operation of the validation framework and the functions that it performs in the context of the verification of the system's algorithms, results from its application to the verification process of the wireless LAN modem are given below. The modem verification case presented here is based on an in-filed usage scenario, suitably modified to meet the specific validation environments used for the development of the algorithms. The scenario used for the testing of the modem design is a part of the protocol's RLC association scenario, between the Access Point and the Mobile Terminal. A number of protocol frames are captured during this high-level usage scenario and are translated into suitable test vectors for the verification of the modem. One of these frames is presented in Fig. 6, showing the bursts comprising a protocol frame, along with the starting time slot, duration, number of data units for each burst.

The presented frame includes the broadcast burst (access point identity, capabilities and map of the current frame), the downlink burst (which contains a train of 5 short (SCH) and 12 long (LCH) packets), and the uplink burst (1 long and 10 short packets). Furthermore, the frame includes a random access burst, during which unassociated mobile terminals can perform their initial contact with the access point.

## 5 Validation Results

In the timing diagrams of Fig. 6, the signal "AP\_Tx\_IQ\_data\_out" and "MT\_Tx\_IQ\_data\_out" are the transmitter's output for the Access Point and the Mobile Terminal, respectively. The signals "AP\_Rx\_data\_out" and "MT\_Rx\_data\_out" are the receiver's output (the data that are inserted in the receive memory of the modem interface unit) for the Access Point and the Mobile Terminal, respectively. Also, the basic control signals for both Access Point and Mobile Terminal are shown ("data\_valid", enable and control commands for the receiver and start of transmission for the transmitter). Note, that analog interpretation of the data signals is used for better understudy of the results (data contained within the packets).

Fig. 6 illustrates the simulated signals for a complete frame (broadcast, downlink and uplink bursts). These signals were generated after the execution of the simulation scenario that was produced automatically by the validation framework for the input frame depicted in Fig. 6. The different bursts of the simulated frame are clearly presented in the same figure. For each burst of the frame, the packets that it contains are also displayed in the same figure. For the transmission of the user packets (LCHs), this example uses QPSK 3/4 physical mode. Similar test cases were also performed in order to cover all the physical layer modes [10] supported by the HIPERLAN/2 standard. The complete set of the performed test cases is given in the following table, where the type and the number of data units for each burst are listed.

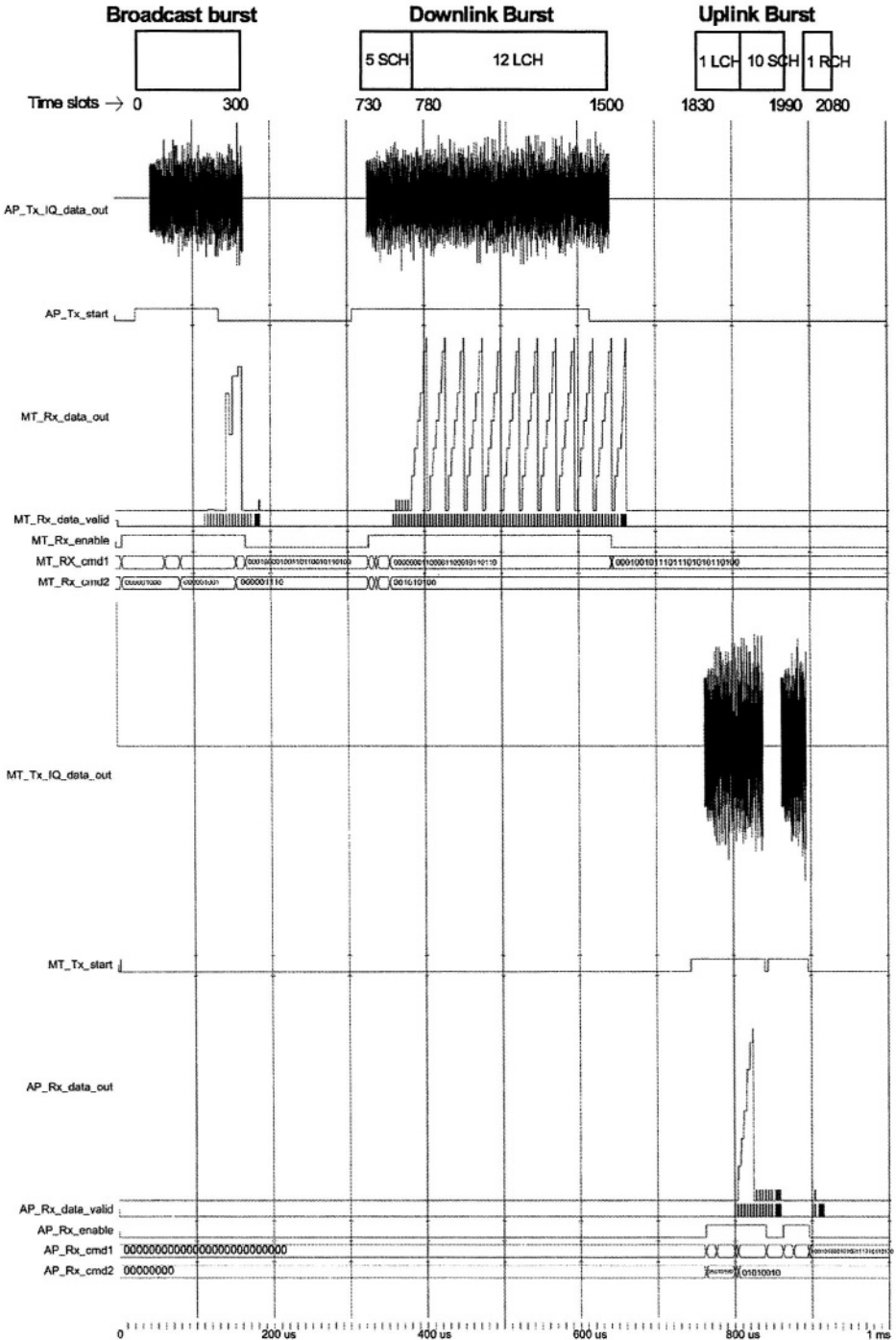


Fig. 6. Modem validation results

**Table 1. Modem's test cases**

Test case	Burst type	PDU types	Data bytes / PDU type (for each DLC packet)	PHY mode	Coding rate
1	Broadcast →	BCH, FCH, ACH	15/BCH, 27/FCH, 9/ACH	BPSK	3/4
	Downlink →	SCH, LCH	5/SCH, 12/LCH		
	Uplink	SCH, LCH, RCH	10/SCH, 1/LCH, 1/RCH		
2	Broadcast →	BCH, FCH, ACH	15/BCH, 27/FCH, 9/ACH	QPSK	1/2
	Downlink →	LCH	12/LCH		
	Uplink	LCH, RCH	1/LCH, 1/RCH		
3 (Fig. 6)	Broadcast →	BCH, FCH, ACH	15/BCH, 27/FCH, 9/ACH	QPSK	3/4
	Downlink →	SCH, LCH	5/SCH, 12/LCH		
	Uplink	SCH, LCH, RCH	1/LCH, 10/SCH, 1/RCH		
4	Broadcast →	BCH, FCH, ACH	15/BCH, 27/FCH, 9/ACH	16-QAM	9/16
	Downlink →	LCH	12/LCH		
	Uplink	LCH, RCH	2/LCH, 1/RCH		
5	Broadcast →	BCH, FCH, ACH	15/BCH, 27/FCH, 9/ACH	16-QAM	3/4
	Downlink →	LCH	12/LCH		
	Uplink	LCH, RCH	2/LCH, 1/RCH		
6	Broadcast →	BCH, FCH, ACH	15/BCH, 27/FCH, 9/ACH	64-QAM	3/4
	Downlink →	LCH	12/LCH		
	Uplink	LCH, RCH	12/LCH, 1/RCH		

## 6 Discussion and Conclusions

This paper presents the validation of the software and hardware parts of the baseband modem for a wireless LAN application. For the design of the modem, a flexible implementation scheme was followed, where some parts of the modem were implemented in hardware, for efficiency reasons, and some parts in software, for flexibility reasons. The specialized communication between the hardware and the software parts of the modem was designed and validated by using the high-level model of the system.

The functional validation of the modem and its interface was based on a high-level system model, developed using UML. This UML-based model of the HIPERLAN/2 system was validated using scenarios from real-life usage. The modem implementation was validated using an innovative validation framework. The framework uses as input a high-level frame format, as it has been produced by the simulation of the UML system model, and produces specific test patterns for the validation of the modem's algorithms, at three different phases of the development (algorithmic, HDL, FPGA-based prototyping). The developed framework can be efficiently used for designing/validating wireless network applications. However, the application of the proposed methodology is not limited to wireless network applications, but can be applied in any embedded system development that incorporates telecommunication services. Current trends in System-on-Chip design tend to adopt UML as high-level modeling language; therefore, the proposed validation environment can be applied seamlessly to embedded system designs that are based on the use of UML.

The main benefit of validation framework that was presented in this paper is that it integrates the validation of the modem's development with the validation of the rest of the system, which is performed at a high level of abstraction using UML modeling. The use of this validation framework helps in the easy and efficient identification and correction of the modem's design errors. Furthermore, the proposed validation frame-

work is based on the modeling of the system with UML, which is becoming a trend in today's embedded systems [11].

The prototype of the presented HIPERLAN/2 modem was implemented successfully on an FPGA-based platform (ARM Integrator). Furthermore, results from the co-simulation of the modem through the testing scenarios produced by the validation framework are also presented, along with sample outputs of the framework.

## References

1. The Institute of Electrical and Electronics Engineers (IEEE), 'Supplement to IEEE standard for information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer in the 5 GHz band', IEEE Std. 802.11a, (1999)
2. European Telecommunication Standards Institute (ETSI), Broadband Radio Access Network (BRAN): HIPERLAN Type 2: System Overview, TR 101 683, February 2000.
3. iSuppli Corp: "The future of wireless connectivity: Bluetooth and WLAN", Market analysis report, Q3 2002.
4. Van Nee, R., Prasad, R.: OFDM for mobile multimedia communications. Artech House, Boston, MA (1999)
5. Kneip, J., Weiss, M., Drescher, W., Aue, V., Strobel, J., Oberthur, T., Bole, M., Fettweis, G.: Single chip programmable baseband ASSP for 5 GHz wireless LAN applications, IEICE Trans. Electronics, Vol. 85 (2002) 359–367
6. Chen, R., Sgroi, M., Martin, G., Lavagno, L., Sangiovanni-Vincentelli, A., Rabaey, J.: Embedded System Design Using UML and Platforms, Proc. Forum on Specification and Design Languages, Marseille, France, 24-27 September, (2002)
7. Sangiovanni-Vincentelli, A., McGeer, P., Saldanha, A.: Verification of electronic systems, Proc. Design Automation Conf., Las Vegas, Nevada, 3-7 June (1996) 106-111
8. Selic, B., and Rumbaugh, J.: Using UML for Modeling Complex Real-Time Systems, Technical report, Rational Software (1998)
9. Bisdounis, L., Dre, C., Blionas, S., Metafas, D., Tatsaki, A., Ieromnimon, F, Macii, E., Rouzet, Ph., Zafalon, R., Benini, L.: Low-power system-on-chip architecture for wireless LANs, IEE Proc. Computers and Digital Techniques, Vol. 151 (2004) 2-15
10. Khun-Jush, J., Schramm, P., Wachsmann, U., Wegner, F.: Structure and performance of the HIPERLAN/2 physical layer, Proc. IEEE Vehicular Technology Conf., Houston, TX, 16-20 May (1999) 2667-2671
11. Martin, G.: UML for embedded systems specification and design: Motivation and overview, Proc Design Automation and Test in Europe Conf., Paris, France, 4-8 March (2002) 773-775

# SoftExplorer: Estimation, Characterization, and Optimization of the Power and Energy Consumption at the Algorithmic Level

Eric Senn, Johann Laurent, Nathalie Julien, and Eric Martin

L.E.S.T.E.R., University of South-Brittany, BP92116  
56321 Lorient cedex, France  
eric.senn@univ-ubs.fr

**Abstract.** We present SoftExplorer, a tool to estimate and analyze the power and energy consumption of an algorithm from the C program. The consumption of every loop is analyzed, and the influence of the data mapping is characterized. Several models of processor are available, from the simple RISC ARM7 to the very complex VLIW DSP TI-C67. Cache misses, pipeline stalls, and internal / external memory accesses are taken into account. We show how to analyze and optimize the power and energy consumption, and how to choose a processor and its operating frequency, for a MPEG-1 decoder. We also explain how to find the best data mapping for a DSP application.

## 1 Introduction

Traditional methods to estimate a processor's power consumption work mainly at the cycle-level or at the instruction-level. The tools Wattch and SimplePower [1,2] are based on cycle-level simulations. These simulations necessitate a low-level model of the processor's architecture; they are very time consuming for large programs. The Instruction-Level Power Analysis (ILPA) [3] relies on current measurements for each instruction and couple of successive instructions of the processor. The time to obtain a model is prohibitive for very complex architectures [4]. These methods perform power estimation from the assembly program with an accuracy from 4% for simple cases to 10% when both parallelism and pipeline stalls are effectively considered. Recent studies have introduced a functional approach [5,6], but few works are considering VLIW processors and the possibility for pipeline stalls [7]. As far as we know, only one unsuccessful attempt of algorithmic estimation has already been made [8].

A lot of optimizations can be conducted at the algorithmic level with a very strong impact on the system's power consumption [9]. To evaluate the impact of high-level transformations of the algorithm, we propose to estimate the application's consumption at the early stages in the design flow. We demonstrate that an accurate estimation of an algorithm's power consumption can be conducted directly from the C program without execution. Our estimation method relies on a *power model* of the targeted processor, elaborated during the *model*

*definition* step. This model definition is based on the *Functional Level Power Analysis* of the processor's architecture [10]. During this analysis, functional blocks are identified, and the consumption of each block is characterized by physical measurements. Once elaborated the model for the processor, a model of the algorithm is needed. This model is obtained during the *estimation process*, which consists in extracting the values of a few parameters from the code; these values are eventually injected in the power model to compute the power consumption. The estimation process is very fast since it relies on a static profiling of the code (3s for a FIR1024 or a FFT1024, 8s for the MPEG-1 decoder, 10s for the DWT512x512 [11]). Several targets can be evaluated as long as several power models are available in the library.

To perform estimation from the assembly code, only the two former models are needed. The model for the processor represents the way the processor's consumption varies with its activity. The model for the algorithm links the algorithm with the activity it induces in the processor. At the algorithmic level (that we also call "C-level"), a model for the compiler is also necessary: the *prediction model*. Indeed, depending on the compiler behavior, or merely on the options settled by the programmer during compilation, the assembly code will differ, and so will the processor's activity.

## 2 SoftExplorer

SoftExplorer can perform power and energy estimation both at the assembly and at the C level. Basically, SoftExplorer automatically performs the estimation process - it extracts parameters from the code - and computes the power consumption and execution time for the targeted power model. Table 1 presents the parameters for the power models included in SoftExplorer. Four power models have been developed so far, for the Texas Instruments' C62, C55, C67, and the ARM7.

**Table 1.** Power models' parameters

Parameters	C62	C67	C55	ARM7
$\alpha$	X	X		
$\beta$	X	X	X	
$\gamma$	X	X	X	
$\tau$	X	X		
$\varepsilon$	X	X	X	
PSR	X	X		
W	X	X	X	
F	X	X	X	X
MM	X	X	X	X
DM	X	X		
PM			X	



The parallelism rate  $\alpha$  assesses the activity between the fetch stages and the internal program memory controller of the processor. The processing rate  $\beta$  represents the utilization rate of the processing units (ALU, MPY).  $\gamma$  is the program cache miss rate.  $\tau$  is the external data memory access rate. The parameter  $\varepsilon$  stands for the activity rate between the data memory controller and the Direct Memory Access (DMA). PSR is the pipeline stall rate. These parameters actually depend on the algorithm itself, they are called *algorithmic parameters*.

The processor's consumption also depends on *architectural parameters*, which are rather related to the configuration of the processor, or more generally of the application. They are the clock frequency ( $F$ ), the memory mode for instructions ( $MM$ ), the data mapping ( $DM$ ), the DMA data width ( $W$ ), and the power management parameter  $PM$ , which indicates the units in sleep mode for low-power processors.

In SoftExplorer, once the target selected, the memory mode (mapped, cache, freeze, bypass) and the processor's frequency must be provided. At the C-level, the prediction model is also required for the C62, C55, or C67. We have defined four prediction models. The DATA (TIME\_optimal) model corresponds to the case where the compiler is settled to optimize the code for performance. The MIN (SIZE\_optimal) model corresponds to an optimization for size. The MAX (FULL\_parallel) model gives a maximum bound for the power consumption, and represents a situation where all the processing resources would be used restlessly. The SEQ (SEQuential) model stands for a situation where operations would be executed one after the other in the processor. That gives an absolute minimum bound for the power consumption (the MIN model gives a more realistic lower bound however). In the case of the two first prediction models, the data mapping is taken into account. Indeed, we have demonstrated that the number of external accesses, and the number of memory conflicts, are directly linked to the processor's processing and parallelism rates ( $\beta$  and  $\alpha$ ), and to the pipeline stall rate (PSR), which have a great impact on the final power consumption and execution time [12].  $\alpha$ ,  $\beta$ , and PSR are some of the power model's parameters which are automatically computed by the tool. For the ARM7, only the memory mode and operating frequency, are needed.

Then, the estimation can begin once the prediction type is chosen; indeed, SoftExplorer can perform three types of prediction:

**Coarse prediction:** There is no need for any further information from the user.

The C-code is parsed and the power consumption is computed with different values for the instruction cache miss rate  $\gamma$  and the PSR from 0 to 100 %. The result is a curve, or an area, displayed on the "C curves results" or the "C area results" page. Indeed, if the memory mode is *bypass* or *mapped*, then  $\gamma = 0$ , only the PSR varies, and a curve is computed. If the memory mode is *cache* or *freeze*, both the PSR and  $\gamma$  vary, and an area is drawn. We call these curves and areas *consumption maps*. The interest of the coarse prediction is to allow an estimation of an algorithm's power consumption very early in the design process. Indeed, even if the data mapping is not settled (the data mapping is not considered in this mode), the programmer

can have an idea of the power consumption, choose the processor, compare different algorithms, and be guided in the optimizations to be conducted on the code.

**Fine prediction:** In this mode, the data mapping file is used to determine the data access conflicts (internal and external) for each loop in the code. If  $\gamma = 0$ , which is the case for a large number of DSP applications, these conflicts permit to determine, for every loop, the precise number of pipeline stalls (i.e. the PSR) and the execution time. This local knowledge of every parameter in the power model of the target permits to compute, for each loop, the power consumption, the execution time, and the energy consumption, as well as the parallelism and processing rates. This fine analysis of the algorithm's consumption is very helpful to the designer. An accurate optimization of each portion of the code can be conducted that way.

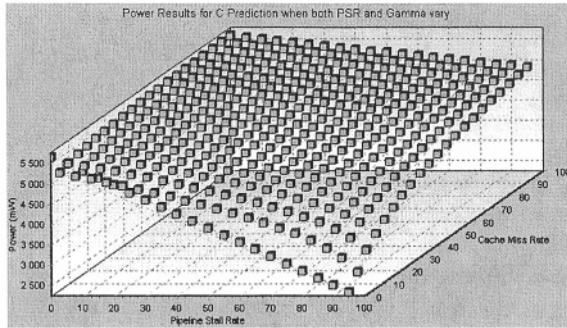
**Exact prediction:** The execution time, the cache miss rate and the pipeline stall rate are no more predicted by SoftExplorer, but are instead provided by the user. They can be obtained with the help of the targeted processor's development tools (e.g. TI's CodeComposer for the C6x). C-level profilers can also be used for an estimation of the execution time.

The precision of SoftExplorer was evaluated by comparing power estimations with measures for several representative DSP applications. The precision varies slightly from a processor to the other. For the C62, the maximal / average errors are 4% / 2.5% at the assembly level, and 8% / 4.2% at the C-level [10]. Energy estimation is less precise (average error is 11%, max error is 21%). This is due to the difficulty in estimating the execution time at the C-level, for very dynamic or control oriented algorithms. Indeed, in the case of dynamic loops, the number of iterations is not known by advance. Whenever a dynamic loop is encountered, the user must provide SoftExplorer with a value for the number of times the loop is iterated. A dynamic profiling is necessary; specific tools are used for this purpose. The inaccuracy of these tools is directly reported on SoftExplorer's results. Another approach for the user could be to provide the maximum limit for the number of iterations to get a maximum for the execution time and energy. Secondly, SoftExplorer's parser discards every control structure in the code. This is not a problem for data intensive applications - indeed power estimations are always very good - but this increases the error for the execution time when the algorithm includes a lot of control. Again, a dynamic profiling could be used, but with the same drawbacks than before.

## 3 Applications

### 3.1 Optimizing the Algorithm

We show, on an example, the results that are obtained with the three prediction types of SoftExplorer, and how optimizations can be conducted on the algorithm. The application is the MPEG1 decoder from *MediaBench*. Estimation is performed first in coarse mode for the C62. If the memory mode is mapped, a



**Fig. 1.** C area: the power is a function of PSR and  $\gamma$

curve that shows the evolution of the power consumption  $P$  with the PSR is plotted. The maximal value for  $P$  is 4400mW when PSR=0%, and its minimal value is 2200mW for PSR=100%. Figure 1 shows the evolution of the power consumption with both the PSR and  $\gamma$  (memory mode is cache). This time, the max/min values for  $P$  are 5592/2353mW. It can be observed that for the same PSR, the power consumption is always lower in mapped mode. Indeed, the max/min power consumption in freeze and bypass mode are respectively 5690/2442 and 5882/5073mW.

A fine prediction takes into account the data mapping. In mapped or bypass mode, the global power and energy consumptions are presented on the “results” page, with the execution time, and the global values of parameters  $\alpha$  and  $\beta$ . The power repartition in every functional part of the processor is also given (Figure 2). The DMA unit consumes no power for it is not used in this application. It is remarkable that a great part (47.4%) of the power consumption is due to the clock. The “loops” page displays the power consumption, execution time, and  $\alpha$  and  $\beta$  values, for each loop (Figure 3). These results are also presented in the form of a chronogram on the “graphics” page (Figure 4) with the power consumption per functional part.

As stated before, the designer gets a direct analysis of the power and energy consumption of every part of its algorithm, together with their intrinsic parallelism and processing rates. In this example, there are three small loops (loops 1, 4 & 7) that consume a lot of power, and two long loops (loops 2 & 5) that consume a lot of energy. In the small loops, the parallelism rate is 1 and the processing rate only 0.125. In order to lower the algorithm’s maximal power consumption, the programmer can investigate these loops and decrease the parallelism rate by rewriting the code. In the two long loops, the parallelism and processing rates are 0.5 and 0.375. Again, the designer can modify the code to increase this rates and to decrease the loops’ execution time, especially if some efforts are made to reduce the number of memory conflicts. The algorithm’s energy consumption can be reduced this way.

In cache or freeze mode, the variations of the power consumption, execution time, and energy consumption with  $\gamma$  are given on the “C curve” page.

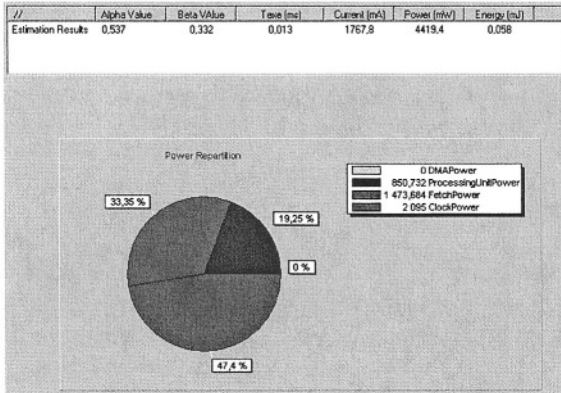


Fig. 2. The “Results” page also gives the power repartition

//	Loop Number	PLoss (mW)	TexteLoop (CPU Cycles)	NbitLoop	NbitPLoss	AlphaLoop	BetaLoop
Loop 1	1	5145.0	64	64	1	1.000	0.125
Loop 2	2	4429.5	1024	512	2	0.500	0.375
Loop 3	3	3045.5	120	64	2	0.500	0.188
Loop 4	4	5145.0	64	64	1	1.000	0.125
Loop 5	5	4429.5	1024	512	2	0.500	0.375
Loop 6	6	3045.5	120	64	2	0.500	0.188
Loop 7	7	5145.0	64	64	1	1.000	0.125
Loop 8	8	4129.5	120	64	2	0.500	0.250

Fig. 3. The “Loops” page presents the results for each loop

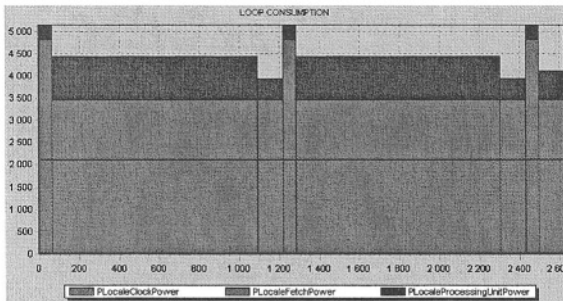


Fig. 4. A graphical display of the consumption per loop

An exact prediction is possible when the exact values for  $\gamma$ , PSR and  $T_{\text{exe}}$  are known. In our example, the TI’s development tool gives, after compilation,  $\gamma = 0$ , PSR=0.2 and  $T_{\text{exe}}=40\mu\text{s}$ . The power and energy estimations are displayed on the “results” page. The “loops” and “graphics” pages display local values for  $\alpha$ ,  $\beta$ , and the power consumption, assuming that pipeline stalls and cache misses are equally scattered in the algorithm.

### 3.2 Influence of the Data Mapping

We demonstrate on a simple example the influence of the data mapping on the power consumption and execution time. The algorithm that we use as a test

**Table 2.** Power and energy estimation with different mappings

Mapping	$\alpha$	$\beta$	Texe (ms)	Current (mA)	Power (mW)	Energy (mJ)	Conflicts	Tconflicts
1	0.02	0.01	6.9	877	2194	15.1	27601	441616
2	0.2	0.1	0.69	1165	2912	2.01	27601	27601
3	0.2	0.1	0.69	1165	2912	2.01	0	0
4	0.2	0.1	0.69	1165	2912	2.01	0	0

bench manipulates at the same time 3 images of size 100x100 (a,b,c) and 2 vectors of size 10 (e,f), in 3 successively and differently imbricated loop nests. Since the images and vectors are manipulated at the same time, their placement in memory has a strong influence on the number of access conflicts, and thus on the number of pipeline stalls. We show here that it is very quick and easy to try different placements, and to reach an optimal data mapping with the help of SoftExplorer. The results are presented on table 2.

In the first mapping, all the data structures are placed in the external memory. As a result, there are as much external accesses than accesses to the memory, and for every access the pipeline is stalled (during 16 cycles for the C6x). The relation between parameters  $\alpha$  and  $\beta$  and the power consumption is obvious. When the pipeline is stalled, the number of instructions that the processor executes in parallel ( $\alpha$ ) and the processing rate ( $\gamma$ ) decrease. As a result, the power consumption of the processor is reduced, but the execution time is lengthened and the energy consumption increases.

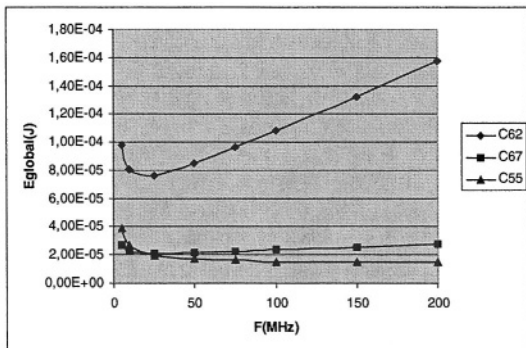
In the second mapping, all the structures are placed in the same bank in the internal memory (B0). There will be as much conflicts as before, but this time, the conflicts are internal. The C6x's pipeline is stalled during one cycle in case of an internal conflict. As a result, the time necessary to resolve all the conflicts, expressed in number of cycles, equals the number of conflicts itself.

The interest of the last mapping (4) is to give a minimum bound in term of number of conflicts since every structure in the algorithm is in a different bank. This solution will also give the higher power consumption and the smaller execution time. Indeed, since the pipeline is never stalled, the processor is used at its maximal capacity. The third mapping, achievable with a C6x, with (a,c,f) in bank B0 and (b,e) in bank B1, is as good as mapping 4 since it does not yield any conflict.

### 3.3 Finding the Right Processor / Frequency Couple

Even if it is easy to obtain the power consumption and the execution time of an algorithm with SoftExplorer, to actually find the right processor and its operating frequency is not straightforward. Indeed, the global energy consumed by the application depends on the energy consumed when the algorithm is executed, but also on the energy consumed when the processor is idle (equation 1).

$$E_{global} = P_{exe} \times T_{exe} + P_{idle} \times (T_{constraint} - T_{exe}) \quad (1)$$



**Fig. 5.** Energy vs Frequency for the MPEG-1 decoder

The timing constraint  $T_{constraint}$  is the maximum bound for the execution time. Over this limit, the application's data rate is not respected. Basically, if the frequency is high, the execution time is small and the active power (Pexe) increases. The idle time also increases with the frequency. On the other hand, as long as the execution time is lower than the timing constraint, it seems possible to slow down the processor to decrease Pexe. So, is it better to operate with a high or a low frequency? In fact, it actually depends on the application.

We pursue a little farther the analysis with our preceding example, the MPEG-1 decoder. This algorithm treats 4 macro-blocs of a QCIF image in one iteration. A QCIF image (88x72) contains 396 macro-blocs. Suppose a data-rate of 10 images/s, the timing constraint is  $T_{constraint} = 1.01ms$ . Then we use SoftExplorer to compute, at different frequencies, the execution time, power, and energy consumed by one iteration of the algorithm. Finally, we calculate with equation 1 the global energy consumed by the application at these frequencies. The results are presented on Figure 5 for the C55, C62 and the C67.

The two curves for the C62 and C67 present a minimum that gives the optimal operating frequency for this application: about 20MHz for the C62 and 40MHz for the C67. This minimum is 0.076mJ for the C62 and 0.021mJ for the C67, hence, the best couple processor/frequency among those two would be the C67 at 40MHz.

Whenever  $P_{idle}$  does not vary with the frequency, the resulting curve does not present a minimum anymore [11]; this is the case for the C55. For this processor, the frequency that gives the lower global energy consumption is the higher possible (200MHz). However, since the energy consumption is almost the same at 100MHz, this last frequency will be preferred for it implies a lower power consumption: cooling devices will be lighter in this case. In fact, the global energy consumption for the C55 at 100MHz is 0.015mJ. As a result, the couple C55 / 100MHz is definitely better than the two preceding.

We did not consider the wake-up time in equation 1. In fact, we measured that this wake-up time is very small before the execution time of the algorithm, and that the energy consumption involved is negligible. Moreover, to avoid waking-up

at each iteration, it is preferable to process a whole image with no interruption, and then to idle the processor until the next image. This decreases again the energy contribution of the waking-up. Of course, this can only be done if the application can bare a latency of one image. In a situation where the wake-up time  $T_{wu}$  could not be neglected, it is still possible to evaluate the global energy. Assuming that  $T_{wu}$  is counted in processor's cycles, and that the wake-up power  $P_{wu}$  is proportional to the frequency  $F$ , the wake-up energy  $E_{wu}$  is obviously constant. As a result, the curves that give the global energy in function of the frequency, are shifted of the value of  $E_{wu}$ . The method to find the best processor and frequency remains the same.

## 4 Conclusion

SoftExplorer is a tool that performs power and energy estimation both at the assembly level (from the assembly code) and at the algorithmic level (from the C-code). It is based on the Functional Level Power Analysis, which proves very fast and accurate for the estimation, as well as for the modelling of processors. Four power models are included. For DSP applications, and with elementary information on both architecture and data placement, our C-level power estimation method provides accurate results together with the maximum and minimum bounds of the power consumption. SoftExplorer appears very helpful for analyzing the power and energy consumption of every part of the algorithm. Every loop is characterized, its intrinsic parallelism and processing rates are exhibited, memory access conflicts are determined. Hot spots are detected; the designer is guided in the writing of the code, and in the mapping of data. We show how to use SoftExplorer to choose a processor and its operating frequency in order to minimize the application's overall energy consumption.

Future works include the development of new power models for the C64, the ARM9, the PowerPC and the OMAP. A generic memory model will be added to include the external memory consumption in our power estimation. Estimation of the execution time will be improved for control oriented applications, to increase SoftExplorer's accuracy for energy estimation at the C-level. Power and energy estimation will be investigated at the system level.

## References

1. D. Brooks, V. Tiwari, and M. Martonosi, "Watch: A framework for architectural-level power analysis and optimizations," in *Proc. International Symposium on Computer Architecture ISCA'00*, 2000, pp. 83–94.
2. W. Ye, N. Vijaykrishnan, M. Kandemir, and M. Irwin, "The design and use of SimplePower: A cycle accurate energy estimation tool," in *Proc. Design Automation Conf. DAC'00*, 2000, pp. 340–345.
3. V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *IEEE Trans. VLSI Systems*, vol. 2, pp. 437–445, 1994.

4. B. Klass, D. Thomas, H. Schmit, and D. Nagle, "Modeling inter-instruction energy effects in a digital signal processor," in *Proc. of the Power Driven Microarchitecture Workshop in ISCA'98*, 1998.
5. S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel, "An accurate and fine grain instruction-level energy model supporting software optimizations," in *Proc. Int. Workshop on Power And Timing Modeling, Optimization and Simulation PATMOS'01*, 2001, pp. 3.2.1–3.2.10.
6. G. Qu, N. Kawabe, K. Usami, and M. Potkonjak, "Function-level power estimation methodology for microprocessors," in *Proc. Design Automation Conf. DAC'00*, 2000, pp. 810–813.
7. L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria, and R. Zafalon, "A power modeling and estimation framework for VLIW-based embedded systems," in *Proc. Int. Workshop on Power And Timing Modeling, Optimization and Simulation PATMOS'01*, 2001, pp. 2.3.1–2.3.10.
8. C. H. Gebotys and R. J. Gebotys, "An empirical comparison of algorithmic, instruction, and architectural power prediction models for high performance embedded DSP processors," in *Proc. ACM Int. Symp. on Low Power Electronics Design ISLPED'98*, 1998, pp. 121–123.
9. M. Valluri and L. John, "Is compiling for performance == compiling for power?" in *Proc. of the 5th Annual Workshop on Interaction between Compilers and Computer Architectures INTERACT-5*, 2001, Mexico.
10. N. Julien, J. Laurent, E. Senn, and E. Martin, "Power consumption modeling of the TI C6201 and characterization of its architectural complexity," *IEEE Micro, Special Issue on Power- and Complexity-Aware Design*, Sept./Oct. 2003.
11. J. Laurent, N. Julien, E. Senn, and E. Martin, "Functional level power analysis: An efficient approach for modeling the power consumption of complex processors," in *Proc. Design Automation and Test in Europe DATE'04*, Paris, France, Mar. 2004.
12. E. Senn, N. Julien, J. Laurent, and E. Martin, "Power estimation of a c algorithm on a VLIW processor," in *Proceedings of WCED'02, Workshop on Complexity-Effective Design (in conjunction with the 29th annual International Symposium on Computer Architecture, ISCA '02)*, Anchorage, Alaska, USA, May 2002.



# Run-Time Software Monitor of the Power Consumption of Wireless Network Interface Cards

Emanuele Lattanzi, Andrea Acquaviva, and Alessandro Bogliolo

Information Science and Technology Institute, University of Urbino  
61029 Urbino, Italy  
{lattanzi,acquaviva,bogliolo}@sti.uniurb.it

**Abstract.** In this paper we present a new approach to power modeling and run-time power estimation for wireless network interface cards (WNICs). We obtain run-time power estimates by putting together four kinds of information: the nominal behavior of the card (taken from protocol and product specifications), its inherent power-performance properties (automatically characterized once for each card from digitalized current waveforms), the working conditions (characterized on the field by means of simple synthetic benchmarks) and the workload (observed at run time from the actual device). Experimental results show that our model provides run-time energy estimates within 3% from measurements.

## 1 Introduction

Wireless network interface cards (WNICs) are commonly used to grant network connectivity to mobile palmtop PCs. However, the actual usage of WNICs is limited by their power consumption, that is responsible of a large fraction of the energy budget of a pocket PC [1]. According to the IEEE 802.11b wireless LAN protocol [2], commercial network devices provide many low-power operating modes that either exploit the inherent burstiness of the workload or perform traffic reshaping to save power. Device drivers provide an application programming interface to allow the user to dynamically change the operating mode [3]. On the other hand, the energy efficiency provided by each operating mode is strongly affected by workload statistics and operating conditions, making it difficult to take optimal run-time decisions. Although accurate power/performance models have been developed for commercial WNICs [4], the accuracy they provide may be heavily affected by the lack of accurate information about traffic statistics and environmental conditions.

In general, power modeling and estimation entail the construction and characterization of a power model, its adaptation to the actual conditions, and the evaluation of the model under realistic workloads. In this paper we present a unified approach to power modeling and run-time power estimation for WNICs. The schematic representation of the approach is provided in Figure 1. First, we construct a parameterized state diagram for each operating mode to accurately describe the behavior of the device. Second, we characterize the model by fitting real-world power waveforms collected while running ad-hoc synthetic benchmarks. The characterization procedure is fully automated and makes use of pattern-matching algorithms to identify the regions of a power waveform corresponding to the state sequence under characterization.

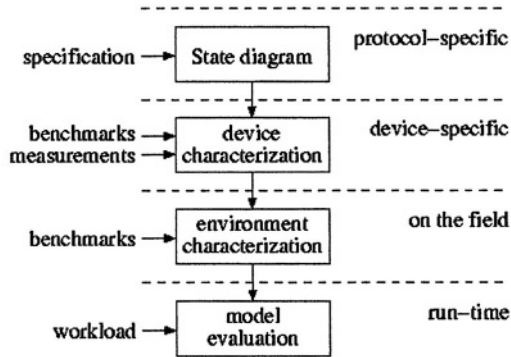


Fig. 1. Power modeling and estimation flow.

Timing and power parameters for each recognized state are then measured each time the state is found in the waveform and their average values are used to characterize the corresponding parameters of the model. State transitions are triggered either by self-events representing the autonomous evolution of the state of the WNIC during operation, or by external events representing workload elements. In addition, the time distribution of self-events may depend on environmental conditions (e.g., channel bandwidth and latency) and workload parameters (e.g., packet size).

In our approach, environmental conditions are estimated on the field by means of simple synthetic benchmarks and used to adjust the model, while workload information is taken directly from the device driver of a working WNIC. To this purpose, a pre-characterized power model is linked to the device driver of the WNIC, realizing a run-time software power monitor that provides the power waveform of the WNIC with a user-defined time resolution.

In summary, power estimates are obtained by putting together four kinds of information: the nominal behavior of the card (taken both from the protocol and from product specifications), its inherent power-performance properties (automatically characterized once for each card from digitalized current waveforms), the working conditions (characterized on the field by means of simple synthetic benchmarks) and the workload (observed at run time from the actual device).

We remark that the power monitor exploits only standard workload information made available by default by any commercial WNIC and runs as a stand-alone process on Linux PCs. In this paper we describe the application of the proposed approach to a Compaq WL110 WNIC [5]. In order to provide all the details of the proposed methodology within tight space limitations, we focus only on the operating modes exhibited by the WNIC under incoming UDP traffic. This is a significant case study both because incoming workloads are usually more difficult to model and characterize than outgoing ones and because the receiving part of the IEEE 802.11b specification [2] is complex enough to illustrate all the features of our approach.

Experimental results are provided showing that the power monitor has a negligible run-time overhead, while its cumulative accuracy is within 3% from measurements. Instantaneous accuracy depends on the user-specified time resolution used to trace the workload.

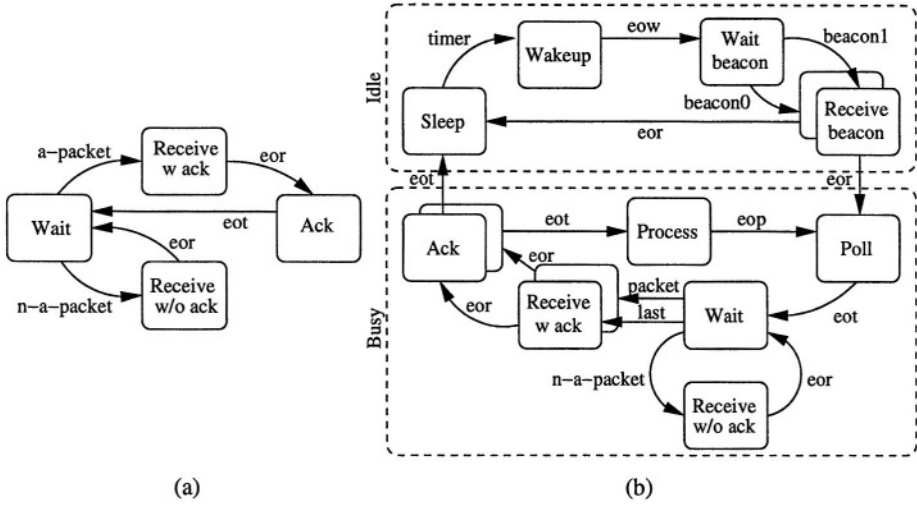


Fig. 2. State diagram of the WNIC working either in ON mode (a) or in PSP mode (b).

## 2 Model Building

In this section we build separate state diagrams for the operating modes of a WNIC implementing the IEEE 802.11b protocol [2]. The state diagrams are based both on the protocol specification and on the observation of experimental current profiles. Focusing only on the reception of UDP traffic, we describe two main operating modes: *always on (ON)* and *power-save protocol (PSP)*. The two operating modes can be viewed as macro-states of a top-level state diagram where state transitions are triggered by user commands.

### 2.1 ON Mode

The state diagram of the ON mode is shown in Figure 2a. The card waits for incoming packets that trigger transitions from wait to receive. Depending on the nature and on the correctness of the packet, the card may or may not send a MAC-level acknowledge to the base station. In particular, a positive acknowledge is required whenever a unicast packet is properly received, while no negative acknowledge is sent for corrupted packets, and no acknowledge is required for multi-cast or broad-cast packets. We call *a-packet* any packet that requires a positive acknowledge, *n-a-packet* any packet that will not be acknowledged.

Although the power state of the card during reception is independent of the nature of the received packet, in our state diagram we use two different receive states: *receive w ack*, that leads to the *Ack* state, and *receive w/o ack*, that leads back to the *Wait* state. State duplication is used only to represent the dependence of the acknowledge on the nature of the received packet.

## 2.2 PSP Mode

According to the 802.11b MAC-level protocol [2], the AP can perform traffic reshaping by buffering incoming packets to allow the WNIC to enter a low-power state. If the MAC-level power management is enabled, the WNIC goes to sleep for a fixed period of time as soon as no traffic is detected. While the NIC is sleeping, incoming packets are buffered by the AP. After expiration of the sleeping period, the card wakes up and listens for the *beacons* periodically sent by the AP. Beacons contain a *traffic indication map* (TIM) providing information about buffered packets, and are used to re-synchronize the WNIC to the AP. If there are buffered packets to be received by the client, the WNIC replies to the beacon by sending polling frames to get the back-log from the AP packet by packet. A positive acknowledge is sent for each properly received uni-cast packet.

The state diagram that represents the behavior described above is shown in Figure 2b. For readability, states are clustered into two subsets labeled Idle and Busy. The Idle part of the graph describes the behavior of the card when no traffic is received. The card stays in a low-power sleep state until a given timeout expires. Then it wakes up and waits for the beacon. We call *beacon1* (*beacon0*) a beacon if its TIM indicates that there is (there is not) buffered traffic for the card. If a *beacon0* is received, the card goes back to sleep as soon as the beacon has been completely received, otherwise it stays awake and enters the Busy sub-graph to get the buffered backlog from the base station.

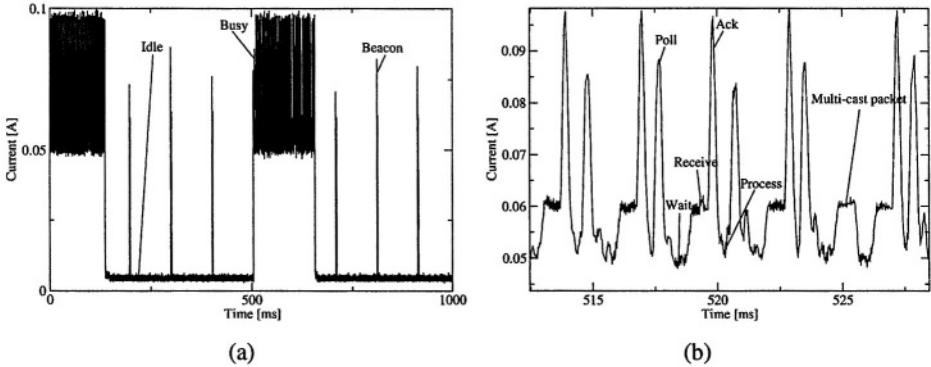
To get the buffered packets the card enters a 5-state loop that entails: sending a polling frame (*Poll*), waiting for a packet (*Wait*), receiving a packet (*Receive w ack*), sending a positive acknowledge (*Ack*) and preparing a new polling frame (*Process*). Each packet contains a *more* bit telling the card whether there are additional buffered frames (*more=1*) or not (*more=0*). The card exits the loop as soon as the last packet (i.e., a packet with *more=0*) is received.

Notice that, when in the wait state, the card is also sensitive to broad-cast and multi-cast packets that do not require any acknowledge. We use the same modeling strategy used for the ON-mode to model the reaction of the card to packets that do not require a positive acknowledge.

## 3 Model Characterization

All the states of the diagrams of Figure 2 need to be characterized for a specific device both in terms of power and in terms of performance (i.e., *permanence time*). Our characterization strategy is based on the automatic analysis of current waveforms collected during the execution of synthetic benchmarks that produce a known workload. The experimental setup used to collect current waveforms is described in Section 6.1. In this section we describe the model characterization strategy assuming that current waveforms are available.

Figure 3.a shows a current waveform obtained while receiving UDP packets in PSP mode. Idle and busy periods are clearly distinguishable. Figure 3.b shows a closer view of the busy region, with labels added to denote the phases of the 5-state loop traversed to get each buffered packet. As long as the states under characteriza-



**Fig. 3.** Current waveforms of a Compaq WNIC receiving UDP traffic in PSP mode.

tion can be recognized on a current waveform, both power and timing parameters can be taken from the waveform. There are, however, two practical issues that make the characterization process non trivial. First, some states have the same power consumption and are recognized only thanks to their position within a known state sequence. This is, for instance, the case of states *Wait* and *Process*, that are characterized by similar current values but are distinguishable because of the different position in the 5-state loop. Second, the noise of the traces and the inherent variability of the process under characterization make it impossible to characterize a power state from a single occurrence within a current profile.

To solve both issues we implemented an automated characterization tool that takes in input a current waveform and a template of the current profile generated by a specific state sequence under characterization. The tool searches within the current waveform all occurrences of the template and recognizes the portions of the profile associated with each state in the target sequence. Power and timing parameters are then computed for each state and averaged over all occurrences of the template in the trace. The standard deviation is also computed for each parameter in order to quantify the variations from the average values used as fitting parameters.

The template is specified in a text file containing the number of states in the sequence under characterization and the list of states (one per row). For each state, 7 parameters are specified: the name of the state, the minimum and maximum current values, the minimum and maximum permanence time, and the entering conditions, expressed in terms of threshold value and sign of the crossing slope. A state is exited when the next one is entered. Since the last state of the sequence has no next state, exit conditions are explicitly added in the last row of the template specification file. The template of the 5-state sequence corresponding to the reception of a uni-cast packet in PSP mode is shown in Figure 4 for a Compaq WNIC.

Notice that template specification is a manual and device-specific task. In practice, the user is asked to provide a first-cut characterization (i.e., a template) of the state sequence of interest based on a preliminary observation of the current profile. The template is then used by the tool to recognize in the current profile the regions to be used for the actual characterization. The output provided by the tool by applying the template of Figure 4 to a 2-second current waveform is:

Nstates	5					
//	minV	maxV	minT	maxT	Vth	S
//	[A]	[A]	[ms]	[ms]	[A]	[+-]
poll	0.06	0.11	0.1	0.6	0.065	+1
wait	0.04	0.06	0.2	1.2	0.06	-1
receive	0.05	0.07	0.2	2	0.057	+1
ack	0.06	0.11	0.1	0.6	0.065	+1
process	0.04	0.06	0	0.6	0.06	-1
EXIT					0.06	+1

Fig. 4. Example of template specification.

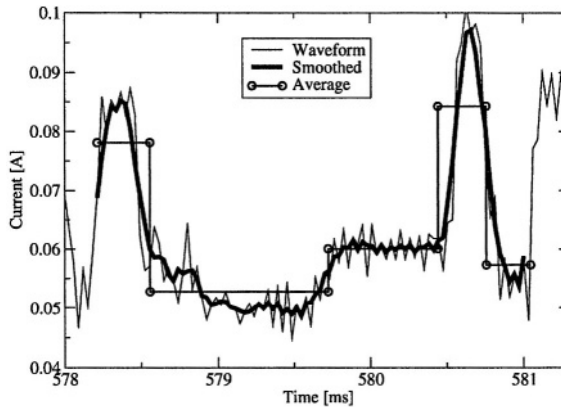


Fig. 5. Graphical output of the characterization tool.

State:	Current:	Time:
poll	V=0.078109+/-0.001593	T=0.343039+/-0.028596
wait	V=0.052796+/-0.001525	T=1.167395+/-0.118424
receive	V=0.060067+/-0.000662	T=0.719934+/-0.134619
ack	V=0.084236+/-0.000893	T=0.315303+/-0.020970
process	V=0.057344+/-0.002677	T=0.288382+/-0.141616

computed over 76 occurrences of the sequence of interest. The tool also provides a graphical user interface that allows the user to span the recognized occurrences of the pattern of interest. For each occurrence, three curves are plotted, as shown in Figure 5: the original current waveform, a smoothed waveform limited to the region of interest, and a step function representing the overall (average) characterization results (i.e., the waveform that will be provided by the model during simulation).

Notice that the waveform of Figure 3.b contains a region of activity due to an incoming multicast packet that doesn't require any acknowledge. The template-based characterization approach automatically neglects such region while characterizing the 5-state loop. On the other hand, the same waveform can be used together with a different template to characterize the *receive w/o ack* state. We used 4 different templates to characterize all the states of the ON and PSP operating modes: *receive with ack* when ON, *receive with ack* when PSP, *receive without ack*, *stay idle* in PSP.

The templates were applied to current waveforms collected when running synthetic benchmarks generating UDP packets of controlled size. The characterization was

repeated with different workloads and in different conditions in order to evaluate the effects of packet size and channel properties. Both in ON and PSP mode, the receive time depends on the ratio between the actual size of the packet and the channel bandwidth. In addition, the waiting time after a polling frame in PSP mode depends on the latency of the channel and on the size of the packet to be received. Both dependencies need to be characterized on the field to adapt the models to the actual working conditions and evaluated at run time for each packet.

## 4 Model Adaptation

Channel latency and bandwidth do not depend only on the WNIC. Rather, they depend on the base station, on the distance between the base station and the mobile client and on the environment. Hence, they cannot be characterized once and for all as properties of the card, but they need to be periodically re-evaluated on the field.

To this purpose we use a benchmark that sends bursts of packets of different sizes and measures the time taken by the client to receive each burst when the card is ON. The transmission time per packet is then obtained as the ratio between the burst time and the number of packets in the burst. The plot of the time per packet as a function of the packet size is a linear curve whose slope is the transmission time per byte, that is the inverse of the bandwidth. Notice that we characterize the bandwidth using UDP packets up to 1500 bytes in order to avoid protocol packetization.

The transmission time per byte can be directly used within the models of the WNIC to evaluate the permanence time of all receive states as a function of the size of the received packet.

In our model channel latency affects only the waiting time between a `Poll` and a `Receive` in PSP mode. Hence, we do not characterize the latency, but we directly characterize the dependence of waiting time from latency and packet size. This is done by running twice the benchmark described above, with the WNIC working in ON and PSP modes. The time-vs-size plot obtained in PSP mode differs from the previous one for two main reasons: first, it has a higher slope, second, it has a higher additive constant. The higher slope is due to the fact that packet size in PSP affects not only the receiving time, but also the waiting time between `Poll` and `Receive`. The higher additive constant accounts for the constant time per packet spent in `Poll` ( $T_{poll}$ ) and `Process` ( $T_{process}$ ) states, that are visited only in PSP mode, and for the constant contribution to the waiting time due to the transmission latency of the polling and data frames. If we denote by  $s_{PSP}$  and  $s_{ON}$  the slopes of the time-vs-size curves obtained in PSP and ON modes, and by  $c_{PSP}$  and  $c_{on}$  the corresponding constants, the estimated waiting time of the PSP model can be computed as:

$$T_{wait}(size) = (s_{PSP} - s_{ON}) \text{ size} + c_{PSP} - c_{on} - T_{poll} - T_{process} \quad (1)$$

## 5 Model Evaluation

Model evaluation requires detailed information about the actual workload. In particular, if we assume to know, for each received packet, the arrival time, the size, the nature (beacon0, beacon 1, unicast, multi/broad-cast, last) and the correctness, we may

construct a step-wise current waveform by simulating the finite state models with size-dependent permanence times.

An *event-driven* current monitor can be obtained calling the power model from the interrupt service routine of the device driver that reacts to each received packet.

## 5.2 Cycle-Accurate Evaluation

The model of the WNIC described in Section 2 has an interesting property: as long as the operating mode of the WNIC doesn't change, its overall energy consumption depends only on cumulative parameters, while it is independent of the arrival time and of the size of each packet.

This strong property, that doesn't hold in general for power managed systems, is due to the synchronous nature of the power save protocol, that periodically awakes the card independently of the traffic. In practice, we could evaluate the energy model periodically, based only on: the overall number of bytes received in the last period (needed to account for the overall receive energy and for the size-dependent contribution of the Wait state), the number of packets of each type (needed to account for the actual number of Poll, Ack and Process phases and for the size-independent contribution of the Wait state) and the operating mode (needed to account for the baseline energy consumption and to select the proper state model).

We implemented a cycle-accurate current monitor as a stand-alone process periodically accessing standard Linux interfaces [3] to obtain the information needed to evaluate the model. In particular, the standard `net_device` interface provided by the Linux kernel for all network devices makes available a data structure (`net_device_stats`) containing cumulative data for any type of packet handled by the WNIC since its device driver was initialized. In our case, we take from the data structure the following data: number of unicast received packets (`rx_packets`), number of corrupted packets (`rx_errors`), number of multicast received packets (`rx_multicast`) and total size of received packets (`rx_bytes`). In addition, we use the standard wireless interface to know the operating mode of the WNIC and its sleeping period while in PSP mode.

The model, evaluated using incremental data since last evaluation, provides the average power consumption computed over the last cycle. In addition, it uses cumulative information to construct a simulated current profile that provides a detailed view of the activity of the card, even if the times of the events within each cycle are reconstructed based on arbitrary assumptions. The time resolution (i.e., the evaluation period) is provided to the current monitor from command line. The minimum evaluation period is 10ms. Regardless of the time resolution, the model provides always the same accuracy at the boundaries of the evaluation periods.

A graphic user interface, based on the *Grace* plotting tool [6], can be activated to show run-time current profiles.



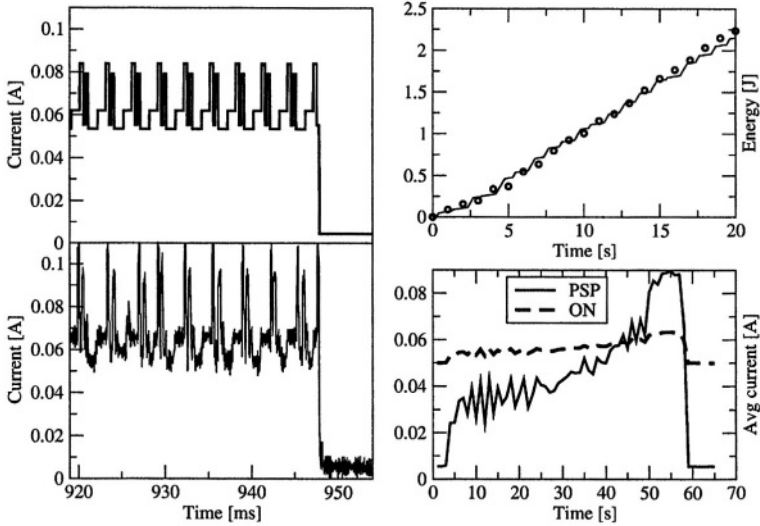


Fig. 6. Experimental results.

## 6 Experimental Results and Conclusions

To perform our experiments we used a Athlon 4 Mobile 1.2 GHz notebook running Linux kernel 2.4.21 [3]. The WNIC installed on the laptop was a COMPAQ WL110 [4], while the access point was a CISCO 350 Series base station [7]. The synthetic UDP benchmarks, the characterization tool and the power models were implemented in C, using the *Grace C* library [6] to provide graphical outputs.

The power consumption of the WNIC was measured using a *Sycard CardBus Extender* [8] that allowed us to monitor the time behavior of the supply current drawn by the card. The current waveforms were then digitized using a *National Instruments DAQ Board PCI 6024E* connected to a PC running *Labview 6.1*.

To validate the model we performed simultaneous current measurements and estimations while sending UDP packets to the card from a remote server. Three kinds of workloads were generated: a bursty workload with equally-spaced bursts of a fixed number of packets; a non-deterministic workload with packets of random size with exponential inter-arrival times; a non stationary workload with time-varying burst size, packet size and idle time. All benchmarks were executed with the WNIC either in ON or PSP mode, or switching between them during execution. Cumulative energy estimates were always within 3% from measurements, regardless of the time resolution used to evaluate the model. No performance degradation was observed.

Representative results are shown in Figure 6. Left-hand-side plots compare a measured current waveform (bottom) with the corresponding current profile obtained from the event-driven evaluation of the model (top). The top-right graph compares the cumulative energy obtained from measurements (solid line) with that obtained from the current model evaluated once per second (circles).

Finally, we observe that the energy models corresponding to different operating modes can be evaluated simultaneously to provide comparative information to be

used to take run-time power management decisions. As an example, we report in the bottom-right graph of Figure 6 the current profiles (averaged over a 1-second window) provided by the PSP and ON models of the WNIC simultaneously evaluated while applying a non-stationary workload to a card operated in PSP mode. The comparative results show that there are workload conditions that make PSP counterproductive, suggesting to switch off the power manager.

## References

1. Simunic, T., Vikalo, H., Glynn, P., De Micheli, G.: Energy efficient design of portable wireless systems. In Proc. of ISLPED (2000) 49-54
2. LAN/MAN Standards Committee of the IEEE Computer Society: Part 11: Wireless LAN MAC and PHY Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band. IEEE (1999)
3. Bovet, D., Cesati, M.: Understanding the Linux Kernel. O'Really & Associates, Sebastopol, CA (2001)
4. Acquaviva, A., Lattanzi, E., Bogliolo, A., Benini, L.: A Simulation Model for Streaming Applications over a Power-Manageable Wireless Link. In Proc. of European Simulation and Modeling Conference (2003)
5. HP:WL110.  
<http://h20015.www2.hp.com/content/common/manuals/bpe60027/bpe60027.pdf> (2004)
6. Grace Team: Grace User's Guide (for Grace-5.1.14).  
<http://plasma-gate.weizmann.ac.il/Grace/doc/UsersGuide.html> (2003)
7. Cisco System: Cisco Aironet 350 Series Access Points.  
[http://www.cisco.com/univercd/cc/td/doc/product/wireless/airo\\_350/accsspts/index.htm](http://www.cisco.com/univercd/cc/td/doc/product/wireless/airo_350/accsspts/index.htm) (2003)
8. Sycard Technology: PCCextend 140 CardBus Extender User's Manual.  
<http://www.sycard.com/docs/cextman.pdf> (1996)

# Towards a Software Power Cost Analysis Framework Using Colored Petri Net

Meuse N. Oliveira Júnior, Paulo R. Martins Maciel,  
Raimundo S. Barreto, and Fernando F. Carvalho

Centro de Informática (CIn)  
Universidade Federal de Pernambuco (UFPE)  
PO Box 7851, 50732-970, Recife-PE-Brazil  
{mnoj, prmm, rsb, ffc}@cin.ufpe.br

**Abstract.** This paper proposes a framework for software power cost estimation. This framework is conceived to be applied to embedded system design in which energy consumption is the main concern. The processor behavior is modeled in Colored Petri Net (CPN) so that to use CPN analyzing techniques for quantitative parameters. This approach allows to analyze consumption distribution along the program, using widespread CPN tool, as an open and retargetable evaluation environment. Such analysis helps designers to improve total consumption on either way: by software optimization and by software/hardware migration. A taxonomy is proposed in order to support that analysis. The main contribution includes the proposition of a method based on CPN for software power estimation that can be used as internal format for software power analysis tools.

## 1 Introduction

An embedded system may be represented by a dedicated processor surrounded by specific hardware systems, performing very specific tasks. The entire system can be implemented in a single chip, i.e. a System-on-a-Chip (SoC). In order to improve designs, methods for design space exploration play an important role, searching for optimizations of parameters such as performance and energy consumption. In power critical embedded systems, energy consumption may be regarded to the processor and to specific hardware systems. Despite of processor hardware optimization, processor consumption has a dynamic behavior that is defined by its software, meaning that the software power analysis is crucial. Most techniques for software analysis are based on instruction level simulation or on hardware simulations [5]. Instruction level simulation technique consists of characterizing processor consumptions according to individual instruction consumption (instruction base cost) and instruction combination consumption (inter-instruction cost) through physical measures. This characterization (consumption database) feeds an instruction level simulator. Then, the analysis is performed based on a simulator output pattern. Hardware simulation techniques are normally based on processor circuits descriptions, the processor consumption is computed from a mathematical circuit model. Some interesting frameworks have been implemented toward RTL-level and architecture-level simulators based on the SimpleScalar architectural research tool set

[1]. Unfortunately, frameworks based on hardware simulation are not flexible for modeling a large sort of devices due to their low-level abstraction models. Additionally, their time simulation are high [13], even though RTL-level simulation has hugely improved their performance. In addition to this, constructing an actual hardware-level simulator for power analysis implies having detailed information about implementation (synthesis) technology, that is not freely available. In contrast to processors, consumption associated with specific hardware systems (cores<sup>1</sup>) is defined only by its design description and the respective implementation technology (there is no software), hence the basic methods considered so far are based on hardware simulation paradigm. Some approaches have sped simulations up by system-level power models based on “instruction” concept [11]. The term “*Instruction*” is used as “*an action that, collectively with other actions, describe the ranges of possible behaviors of a core*” [11]. In other words, the above concept introduces a processor-like behavioral description. The action flow is termed as *Traces* and analyzed by simulation, in a similar way to processor instruction level simulation techniques. In the context of SoCs, it is very important to have an efficient system power estimation technique, such technique should gather processor, cores and interconnection evaluation [6].

Colored Petri Nets is a powerful formal description model that allows behavioral analysis based either on simulation or state analysis. This paper proposes an approach to model processors behavior based on Colored Petri Net [4] in order to use CPN tools, such as DesignCPN and CPNTool [4], as an open and retargetable evaluation environment. Initially, an instruction level model is proposed, but it is also possible modeling at either higher or lower abstraction level. In addition, CPN engines are able to deal with parallel instructions<sup>2</sup> allowing modeling complex processor architectures and concurrent *Core Traces*. Even considering the significant research effort in power estimation and low power design [3], any formal model based on Colored Petri Net has not been proposed to tackle such important problem.

## 2 Energy Consumption Concepts

Instructions consume energy, basically, through two ways. First, during the instruction execution, the processor reaches a finite set of state consuming an amount of energy. This energy cost is commonly called *instruction base cost* [7]. Second, according to operands, instructions performs register changes and/or memory accesses. This changes and accesses implies switching on hardware, i.e. dynamic consumption. During the program execution flow, the processor power consumption floating yields an average power consumption. It is possible to analyze the program flow under the consumption point of view to identify high consumption code sections in order to allow one to deal with them. The instruction characterization is determined while executing the same isolated instruction again and again [9]. Unfortunately, this procedure does not capture the consumption occurring between two consecutive instruction. In an actual program, the mix of instruction produces an additional energy cost. Empirical studies have shown

<sup>1</sup> The term *core* refers either processor or specific hardware system, but in this paper it will be used only to refer specific hardware system

<sup>2</sup> In both sense: core and processor instructions

that the energy cost of pairs of consecutive instructions is always different from the base cost sum [12]. The difference is termed as inter-instruction cost and it is accepted as a *circuit state overhead*. All points discussed until now seem intuitive provided that the processor executes only one instruction at a time. For pipelined processors the circuit state overhead can be modeled as spread out along pipeline stages. In the same way that exist a time mixing among instruction execution on pipeline, there exist a consumption mixing. This introduces a context data dependent variable at instruction base cost. During a pipeline stall there is consumption without association with a specific instruction. After all, the explanation above can be express by following equation [5]:

$$E_p = \sum_i (B_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) + \sum_h E_h$$

where  $E_p$  is the energy consumption associated to program  $p$ ,  $B_i$  is the base cost associated to instruction  $i$ ,  $N_i$  is the number of instruction  $i$  instances,  $O_{i,j}$  is the inter-instruction cost associated to instruction pair  $i, j$ .  $N_{i,j}$  is the number of consecutive  $i - j$  occurrences.  $E_h$  are others costs (pipeline stall and cache miss) associated to an internal context  $h$ . This particular work does not take into account pipelined architectures.

### 3 Nomenclatures and Definitions

For sake of simplicity, this paper assumes a non-pipelined processor architecture that does not execute preemptive processes<sup>3</sup>. Some definitions are necessary for better understanding the proposed method.

**Definition 1 (Execution Vector).** *Execution Vector is a vector where each component represents the number of instruction execution, enumerated by the instructions memory ordering.*

**Definition 2 (Consumption Vector).** *Consumption Vector is a vector where each component represents the respective instruction energy cost (base cost + inter-instruction cost), enumerated by the instruction memory ordering.*

From this moment on *Consumption Profile* and *Execution Profile* denote graphics representation of Execution and Consumption Vector respectively. Figure 1 depicts the Execution Profile for a nested-loop example. Analyzing the Execution Profile, it is possible to identify execution patterns such as:

**Definition 3 (Patch).** *Patch is a set of instructions that are located in consecutive addresses and are executed a same number of times.*

In the Figure 1, five patches are identified : from inst2 to inst6 (patch 1), from inst7 to inst9 (patch 2), from inst10 to inst14 (patch 3), from inst15 to inst18 (patch 4) and from inst19 to inst25(patch 5).

**Definition 4 (Loop-Patch).** *Loop-Patch represents a single loop. It consists of a patch in which the incoming (first) and outgoing (last) instruction are executed only once.*

<sup>3</sup> Interruption processes can be treated separately in order to simplify the analysis

There is no Loop-Patch in Figure 1.

**Definition 5 (Cluster).** *Cluster is a set of Patches joined together (aggregated) in consecutive addresses, in which the incoming (first) and outgoing (last) instruction are executed only once.*

In Figure 1 there is only one cluster: {patch 1, patch2, patch3, patch4, patch5}.

**Definition 6 (Bound-Patch Set).** *Bound-Patch Set is a set of Patches executed a same number of times and belong to the same Cluster.*

In Figure 1 there are two Bound-Patch Set: {patch1, patch5}, {patch2, patch4}.

**Definition 7 (Free-Patch).** *Free-Patch is a Patch present in a Cluster but not within the Cluster Bound-Patch Set.*

In the Figure 1 there is only one Free-Patch: patch3. The Execution Profile can also define metrics such as:

1. Instruction Consumption

$$I_i = (N_i \times B_i)$$

where  $I_i$  is the total consumption due to instruction  $i$ ,  $N_i$  is its number of executions and  $B_i$  is its energy cost (base cost + inter-instruction cost).

2. Patch Consumption

$$Pc_j = \sum_i (N_j \times B_i) = N_j \times \sum_i (B_i)$$

where  $Pc_j$  is the consumption of Patch  $j$ ,  $N_j$  is the Patch  $j$  execution number, and  $B_i$  the instruction  $i$  energy cost.

3. Cluster Consumption

$$Cc_k = \sum_{j,c} \sum_{i,p} (N_j \times B_i) = \sum_{j,c} (Pc_j)$$

4. Consumption Profile Vector

$$Cp_m = (Ev_m \bullet Cv_m)$$

where  $Ev_m$  is process  $m$  Execution Vector and  $Cv_m$  is its Consumption Vector.

Such definitions and metrics help the designer to figure out code structures and their energy consumption. For example, a Loop-Patch represents an isolated loop within the code. A Cluster represents consumption and time cost regions. A *Cluster* with *Bound-Patch* Sets such that its *Patches* have symmetric positions in the *Execution Profile* may represent a nested-loop (see Figure 1). Inspecting *Execution Profiles* and the *Consumption Profile* graphics, the designer is able to map consumption to code structures. Using such definitions, the designer can go to a graphic code representation and identifies the code structure and chooses an optimization strategy. In this work, Colored Petri Net is used as a formal, code behavioral and graphic model.

## 4 Colored Petri Net: An Overview

Petri nets are families of formal net-based modeling techniques, that model actions and states of systems using four basics entities: places, transitions, arcs and tokens. In the majority of models places are associated with local states and transitions with actions. Arcs represent the dependency among actions and states. An action occurs when

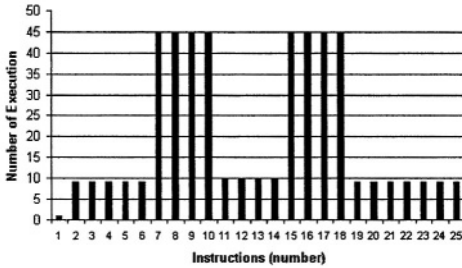


Fig. 1. Execution Profile Example

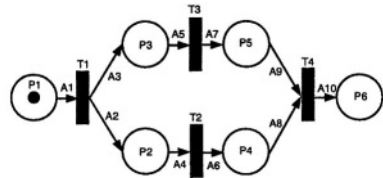


Fig. 2. Petri net example

a transition is “fired”, moving tokens from incoming places to outgoing places. A parallel process is described in Figure 2. Arcs describe which action (or actions) is possible from a given local state. In this figure, arc A1 links place P1 (state 1) to transition T1 (action 1), representing that action T1 requires local state P1. Arcs A2 and A3 connect transition T1 to places P2 and P3. It is easy to see two parallel paths (P3-T3-P5 and P2-T2-P4) as a representation of parallel processes. In order to accomplish the action T4 is necessary to reach state in with places P4 and P5 are marked. In order to represent possible local states, it is used a mark, called token. This simple net is known as place-transition net [8]. Place-transition nets are adequate to analyze some characteristics of system such as: repetitiveness, liveness and reachability, that means determinating whether a specific state is reachable. There are various extended Petri net models each one dealing with a specific modeling problem and distinct abstraction level. CPN is a high-level model that consider abstract data-types and hierarchy. CPN tools provide an environment for design, specification, validation and verification of systems [4]. Informally, Colored Petri Net is a Petri net with some modeling improvements: (i) Tokens express values and data structures with types (colors). (ii) Places have associated Type (color set) determining the kind of data (token) those places may contain. (iii) Transitions may express complex behavior by changing token value. (iv) Hierarchy can be handled at different abstraction levels . Transitions in a hierarchical net may represent more complex structures, where each transition (substitution transition)expresses another more complex net, and so on. A hierarchical net may describe complex systems by representing their behavior using a compact and expressive net. (v) Behaviors can be also described using high level program language.

In this way, it is possible to model a complex system using tokens for carrying sets of internal data values. Transitions represent actions that modify internal set of data values. The entire net represents the flow of changes into the system during its states evolution. The model can be analyzed by simulation, states and invariant analysis. State analysis means to study all possible system states, or an important subset, in order to capture system patterns. To handle with Colored Petri Net, and its analysis engines, there exist some widespread academic tools such as Design/CPN and CPNTool.

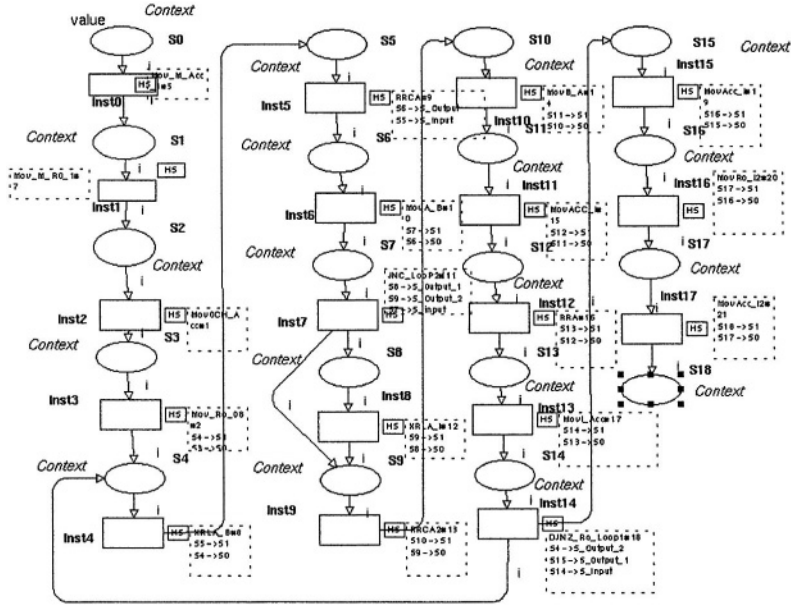


Fig. 3. Colored Petri net model for the CRC Case Study

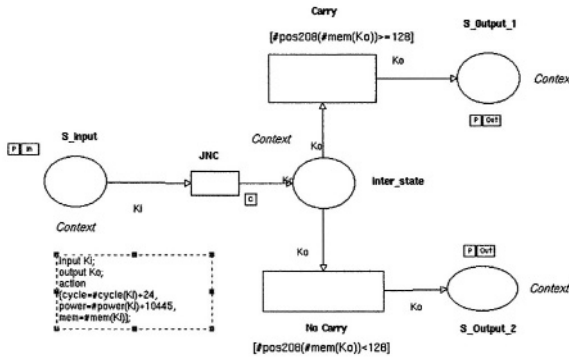


Fig. 4. Colored Petri net model for a branch instruction

### 5 Software Modeling

This work proposes to model the program flow as a safe Colored Petri Net where each possible processor state is modeled as a place, the internal context as a data structure within a token, and instructions as transitions that processes this token. The instruction behavior is defined by an associated CPN-ML code (a Standard ML Language subset) [4]. In contrast to previous CPN processor architecture models [2], where the processor is modeled based on architectural/RTL hardware model this approach models proces-



processor only by its instruction set. In fact, our approach deal with the processor as a *Core* where *instruction trace* is replaced by the program under analysis. In this way it is constructed a software model with expressly structural characteristic, the control and data flow are inherent to model description. Thus, the proposed model allows:(i) power characterization based on instruction-level power model,(ii) software flow analysis from the net structure (iii) space exploration based on Instruction Set Architecture (ISA). This approach allows architecture space exploration by CPN evaluation methods. The CPN engine is a general reconfigurable simulator, new instructions can be created and old one modified by edition of CPN-ML code. A new simulator is automatically created after each modification. In a general point of view, the CPN model behaves as a *Instruction Set Architecture Description Language*. In contrast to simulator based on non-formal model implemented using general purpose program language, the CPN model guarantees a formal way to model architectures and capture parameters. After be validated, a processor model analysis can be sped up by mapping CPN model to C++ applications. The good software engineering is take for granted due to model formalism. Based on the proposed model, a Colored Petri Net (CPN) models program flow so that the token movement illustrates how the processor context changes along the software execution possibilities. Standard techniques to capture transition invariants allows one to capture loops, consequently *Patches* and *Clusters*. By simulation and occurrence graph analysis (state analysis) it is possible to extract worst and best cases power costs. Inter-instruction effects and base cost are computed for every transition firing through CPN-ML code. The extraction of inter-instruction effect consists of evaluating flags (internal token values) that are changed after each transition firing. From two consecutive transitions (instruction), the second one identifies the first one and changes its instruction cost according to first one identity. Figure 3 shows the CRC (Cyclic Redundancy Check) code modeled in Colored Petri Net. Each transition represents an instruction, holding its behavior description on a CPN-ML code. When a transition fires, its respective CPN-ML code processes the token. This process implies that token value changes (register and memory change) according to instruction behavior. In fact, each instruction is modeled by a substitution transition. The Table 1 shows instruction-transition mapping. In Figure 3 transitions with two outgoing arcs represents conditional branch instructions. Figure 4 shows how JNC (Jump if Not Carry) instructions are modeled into a substitution transition. The instruction energy and time cost are processed into CPN-ML code associated to JNC transition and decision behavior is modeled by a choice net structure with transition guards.

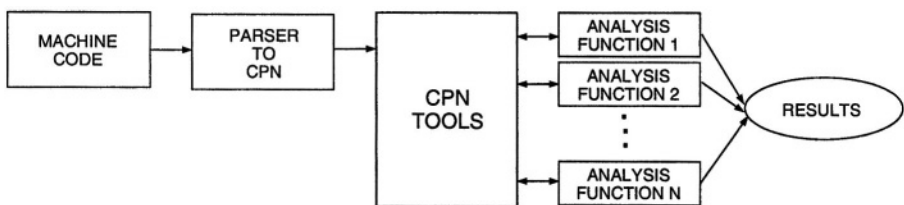


Fig. 5. The Proposed Framework

**Table 1.** Instructions base cost and transition mapping.

Instruction	Transition	Instruction Base Cost
mov 0AH,acc	1	47.5nJ
mov 0BH,r0	2	108.5nJ
mov 0CH,acc	3	47.5nJ
mov r0,#08H	4	59.5nJ
xrl a,b	5	44.3nJ
rrc a	6	51.4nJ
mov a,b	7	49.6nJ
jnc loop2	8	104.5nJ
xrl a,#0b1h	9	49.7nJ
rrc a	10	51.4nJ
mov b,a	11	47.5nJ
mov acc,0A	12	63.9nJ
rr a	13	49nJ
mov 0AH,acc	14	47.5nJ
djnz r0,loop1	15	49.1nJ
mov acc,0CH	16	63.9nJ
mov r0,0BH	17	108.5nJ
mov acc,0AH	18	63.9nJ

**Table 2.** Energy Distribution on Patches

Patch	Patch Energy	Percentile of Total Energy
1	270.54nJ	4.54%
2	2,214.21nJ	37.14%
3	248.62nJ	4.17%
4	2,996.00nJ	50.25%
5	231.54nJ	3.90%

## 6 Proposed Framework

In some methodology for embedded system design, such as platform based design on SoCs, loops can be unrolled and implemented in hardware to improve energy consumption and performance [10]. The partitioning can be implemented either from high-level source (C program source code) or machine-code (binaries). For partitioning analysis, both implementation are useful. On one hand, hardware/software partitioning at the machine-code level is a straightforward method, technologically reliable. On the other hand high-level source partitioning demands less hardware area on SoCs [10]. Using Colored Petri Net approach, *Patches* and *Clusters* identification process can be automatically obtained, and a complementary graphic inspection on the net can check their location and association with loops and nested-loops. The framework proposed here intends to approach this problem at processor instructions level (machine-code). The basic framework illustrated in Figure 5 consists of a CPN engine, a parser to translate machine-code to CPN model, and a set of specific functions for power and, performance evaluation. These functions return metrics, among then, *Consumption Profile Vector* (allowing best and worst case power consumption detection), *Patch Consumption* and *Cluster Consumption* (allowing Hard/Soft partitioning strategy evaluation). These functions, should be highlighted, estimate such metrics using a widespread general tool (a CPN Tool). Additionally, the natural Petri Net capacity for representing parallelism allows one to analyze processor architectures with instruction parallelism and concurrent traces on *Cores*.

## 7 Experiment and Results

This section presents a case study considering a AT89S8252 microcontroller and the behavioral description presented in Section 5. The instruction base cost was obtained from a set of basic measures that was performed using the basic technique described in [9]. From a single AT89S8252 microcontroller board, it was measured instantaneous current drawn by the processor during the execution of each instruction. For data acquisition it was used a TDS220 digital oscilloscope linked to PC desktop computer by serial port. The database file was formatted on spreadsheet format and analyzed. The same procedure was performed to analyze the case study total consumption and execution time. The CPN model is described in Figure 3. For sake of simplicity, only the basic instruction cost was take into account, and the analysis functions considered are those based on simulations without considering occurrence graph analysis. Table 1 shows the association between instructions, their respective transitions and their basic costs. The estimation error was calculated relating estimated and measured values. The execution time estimation error was 0.7%, a result that equals the physical instrumentation error. The total consumption error was 6,2%. Figure 7 shows the *CRC Execution Profile*, where it is possible to identify five patches: from inst1 to inst4 (patch 1), from inst5 to inst8 (patch 2), the inst9 (patch 3), from inst10 to inst15 (patch 4) and from inst16 to inst18(patch 5). One *Cluster*: {patch2, patch3, patch4}, two *Bound-Patch Set*: {patch1, patch5}, {patch2, patch4} and one *Free-Patch*: Patch 3 were identified. Inspecting the net (Figure 3), the *Cluster* is identified as a single loop, and the *Patch 3* as a conditional sequence. In this case, with only one instruction. Figure 6 shows the *Consumption Profile*, where a consumption concentration between instruction 5 and 15 is identified in the *Cluster* region, as expected. Table 2 shows *Patch* specific energy consumptions and their pencetile related to total energy consumption. Based on Table 2 and analyzing the program structure on net the designer may evaluate optimization strategies. Note that 50.25% of total energy is consumed on *Patch 4*, indicating it as best cadindate to hardware migration. This experiment shows the proposed model and framework as an analysis resource for embedded system design.

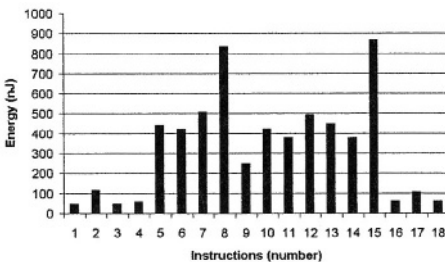


Fig. 6. Consumption Profile

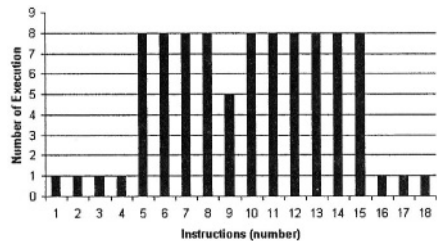


Fig. 7. Execution Profile

## 8 Conclusion

This paper proposed a new approach to model processors behaviors based on Colored Petri Net and a framework to use CPN tool as instruction level evaluation environment. The proposed approach allows to estimate power consumption distribution, therefore allowing designers to explore design spaces of hardware/software partitioning methods that takes into account power constraint. Besides, the proposed framework also make it possible very accurate performance estimation of software components, a very important aspect in hw/sw co-design strategies. Additionally, CPN capacity to describe and analyze concurrence will be exploited. As future works the model will be extended for being considered in: partitioning approach based on C source code, power estimation of pipelined architectures and Cores power estimation based on Trace concurrence.

## References

1. D. Brooks, V. Tiwari, and M. Martonosi. Watch: a framework for architectural-level power analysis and optimizations. In *Proc. of the 27th annual inter. symposium on Computer architecture*, pages 83–94. ACM Press, 2000.
2. F. Burns, A. Koelmans, and A. Yakovlev. Modelling of superscala processor architectures with design/CPN. In *Workshop on Practical Use of Coloured Petri Nets and Design/CPN, Aarhus, Denmark, 10-12 June 1998*, pages 15–30. Aarhus University, 1998.
3. W. Fornaciari, P. Gubian, and D. Sciuto. Power estimation of embedded systems: A hardware/software codesign approach. *IEEE Trans. on VLSI Systems*, pages 266–275, June 1998.
4. S. Christensen L. Kristensen and K. Jensen. The practitioner’s guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer: Special section on coloured Petri nets*, 2(2):98–132, 1998.
5. Th. Laopoulos, P. Neofotistos, C. Kosmatopoulos, and S. Nikolaidis. Current variations measurements for the estimation of software-related power consumption. *IEEE Instrumentation and Measurement Technology Conference*, May 2002.
6. S. Dey M. Lajolo, A. Raghunathan and L. Lavagno. Cosimulation-based power estimation for syste-on-chip design. *IEEE Trans. on Very Large Scale Integration (VLSI) System*, 10(3):253–266, June 2002.
7. V. Tiwari M. Lee, S. Malik, and M. Fujita. Power analysis and minimization techniques for embedded dsp software. *IEEE Trans. on Very Large Scale Integration Systems*, pages 123–135, March 1997.
8. T. Murata. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*, 77(4):541–580, April 1989.
9. N. Kavvadias S. Nikolaidis and P. Neofotistos. Instruction-level power measurement methodology. Technical report, Electronics Lab., Physics Dept., Aristotle University of Thessaloniki, Greece, March 2002.
10. G. Stitt and F. Vahid. Hardware/software partitioning of software binaries. In *Proc. of the 2002 IEEE/ACM inter. conf. on Computer-aided design*, pages 164–170. ACM Press, 2002.
11. F. Vahid T. Givargis and J. Henkel. Instruction-based system-level power evaluation of system-on-chip peripheral cores. *IEEE Trans. on Very Large Scale Integration Systems*, 10(6):856–863, Dec 2002.
12. V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. *IEEE Trans. on Very Large Scale Integration Systems*, 2(4):437–445, December 1994.
13. G. Yeap. *Practical Low Power Digital VLSI Design*. Kluwer Academic Publishers, 1998.

# A 260ps Quasi-static ALU in 90nm CMOS

F. Pessolano and R.I.M.P. Meijer

Philips Research, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands  
francesco.pessolano@philips.com

**Abstract.** A 32-bit ALU has been implemented in the baseline Philips-Motorola-ST 0.10um triple- $V_T$  CMOS technology. The ALU core has been designed with a combined dynamic/static design approach aiming at high-speed operation and standard cells based design. It runs at frequencies ranging from 3.8 GHz to 5.4 GHz (with nominal supply at room temperature) depending on the actual fabrication process corner.

## 1 Introduction

Arithmetic Logic Units (or ALUs) are important components in many modern ICs such as processors and re-configurable cores. It is, thus, important to be able to design high-speed ALUs while still using a standard semi-custom approach. We here describe a quasi-static 32-bit ALU fabricated in the baseline Motorola-Philips-ST 0.10um triple- $V_T$  CMOS technology. The core has been designed by means of high-speed static logic gates and dynamic flip-flops with nominal  $V_T$ . All gates have been designed as standard cells and the entire ALU core implemented in a semi-custom fashion. The core speed ranges from 3.85GHz (worst-case process corner) to 5.4GHz (best-case process corner) with nominal supply (1.0v) at room temperature (25C). The use of several high-speed techniques (selective pre-charge, path duplicating and logic partitioning) enabled the design of such high-speed complex arithmetic networks (e.g. adders and shifter) whilst still using static circuits.

This paper is organized as follows. Section 2 gives details on the organization of the ALU core. The two main high-speed strategies are discussed in section 3. Section 4 deals with more high-level high-speed “tricks”. Some conclusions are drawn in section V.

## 2 ALU Organization

The 32-bit ALU measures  $660 \times 224 \mu\text{m}^2$  (Fig.1 and Fig.2). It consists of two 32-bit input multiplexing registers (*IReg* in Fig.1), one 3-bit opcode register (*OpReg* in Fig.1), one 32-bit Han-Carlson adder [1] (*HC adder* in Fig.1), one 32-bit fully programmable shifter (*SHF* in Fig.1), a 32-bit multiplexing ALU loopback bus (*OMx* in Fig.1) and a 33-bit output register (*OReg* in Fig.1). The input registers allow selecting among the external inputs, their negated or the previously generated result transmitted through the loopback bus. The 32-bit adder and shifter operate in mutual exclusion

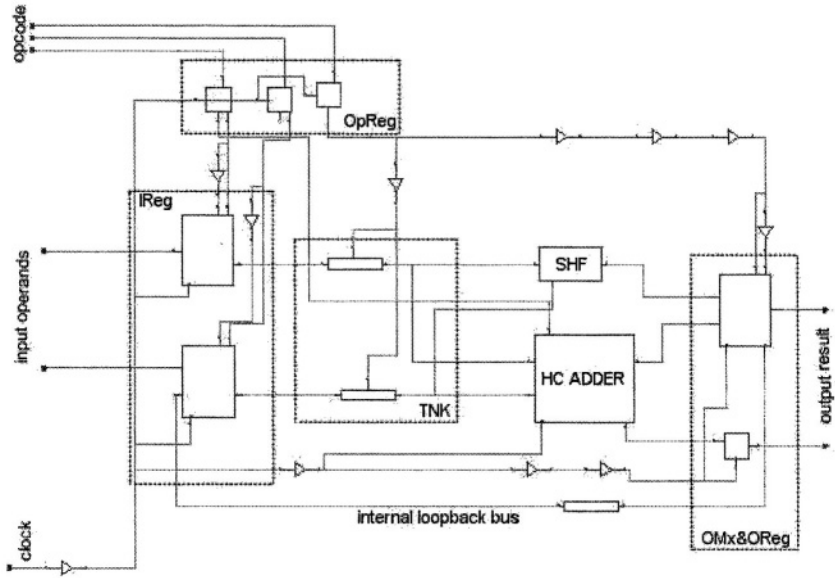


Fig. 1. Block-view of the 32-bit ALU core

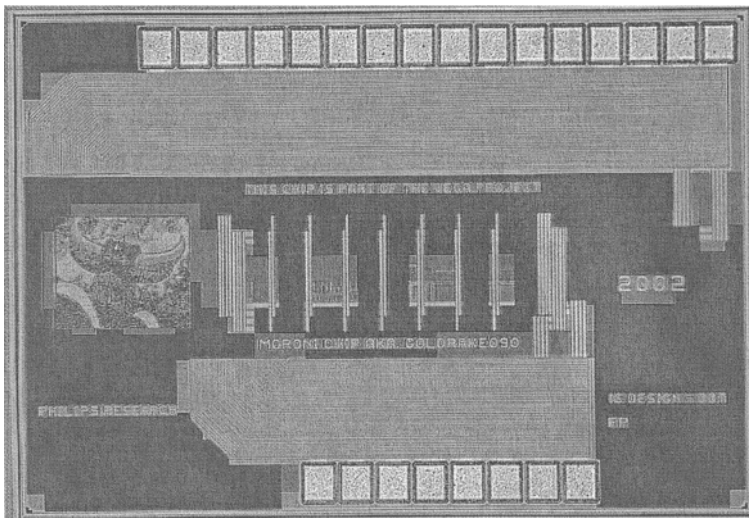


Fig. 2. Microphotograph of the ALU chip.

that is achieved by using tri-state buffers as part of the network broadcasting the operands to both units (*TNK* in Fig.1).

The multiplexing loopback bus receives the new results and transmits it to both the output register and the input register. The bus is  $160\mu\text{m}$  long and it has been partitioned in two chunks with a single level of stacked repeaters. The ALU can, thus,

execute addition, subtraction, shift and any (multi-cycle) sequence of these operations. The executed operation is selected by means of the 3-bit opcode register. The decode logic of the 3-bit opcode is distributed among the ALU components and it is performed “lazily” (as late as necessary). The entire design has been dimensioned so as to avoid logical paths of different lengths to have similar delays. In this way, the design offers good robustness with respect to process variations, even though it slightly reduces speed [2].

### 3 High-Speed Static Families

In order to achieve such a high-speed, the composing standard cells have been modified so as to improve not their individual speed, but their speed when used in a cascade. We have adopted several standard design strategies such as using only inverting gates and asymmetric dimensioning. Besides, we have adopted two other strategies: selective pre-charge and delay-balanced gates.

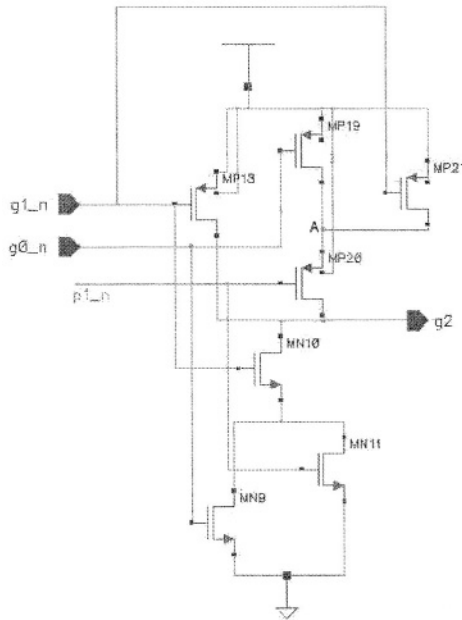


Fig. 3. Example of selective pre-charge in static gates.

#### 3.1 Selective Pre-charge

One of the key elements in making this core running at such high frequency is the adoption of selective pre-charge of unused paths in both dynamic flip-flops and static gates. A discharged internal floating node connected to the output node would drain current when the output is raised thus reducing the rise time. Selective pre-charging

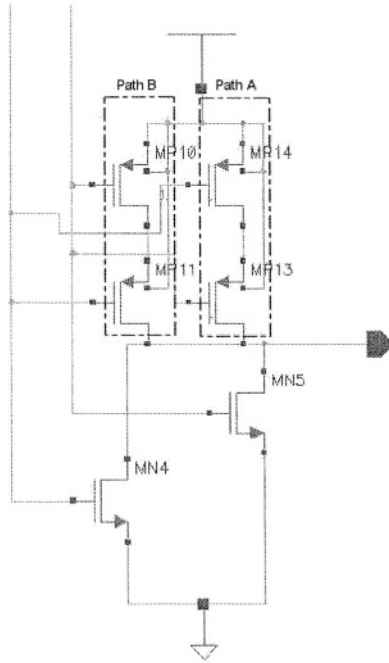


Fig. 4. Example of path replication for delay insensitivity.

uses some (or all) inputs of a gate in order to pull up/down internal floating nodes when connected to the output node so as to avoid such a speed reduction. The main side effect of this technique is that it introduces additional parasitic in this node. The amount of the parasitic can be minimized by means of smart cell layouts in most cases at the price of a larger cell area. For most simple cells, we have been able to reduce it to less than a minimum drain capacitance. This technique has been applied to several gates resulting speed improvements of 10% average (with peak of 40% in tri-state buffers).

Fig.3 gives a simplified example of a gate with selective pre-charge (by means of transistor MP21). We can easily identify at least one internal node (A) that might generate conflict with the output node when the input pattern is  $(/g1\_n, /g0\_n, p1\_n)$ . This node is, thus, pulled high by transistor MP21 when this input pattern occurs.

### 3.2 Delay-Balanced Gates

By delay-balanced gates, we mean gates whose delay is almost not dependent on the order of arrival of its inputs. This dependency is easily understood when looking at a simple stack of two transistors (e.g. path A or path B in A of Fig.4). If transistor MP11 is turned on last, its source will already be (fully or partially) charged, thus reducing the delay for charging the drain capacitance to the RC equivalent of MP11.

On the other hand, if MP10 is the last to be turned on, then the delay is equivalent to the delay of a distributed RC line (with two segments corresponding to the two transistors). The delay is obviously larger in the latter case.



In the design of regular structures (like data-paths or decoders), such dependency is undesirable as there could always be a particular order of arrivals for the inputs that makes each gate in the critical path slower. In the design of the ALU core (and specifically of its ADDER module), we have adopted a structural solution that makes the delay of a gate independent from the input arrival order. This is accomplished by simply splitting each stacked transistor configuration in two (or more) equal ones (Fig.4). Each of these new stacks of transistor is optimised for a different arrival order, thus making the over delay independent from it. The drawback of these methodologies are a slightly larger cell layout (10%) and, thus, parasitic. Nevertheless, the increase in parasitic does not correspond to an increase in gate delay such to make the overall gate slower than the original “delay sensitive” one. We have observed a speed improvement slightly below 9% due to the use of delay-balanced gates. We expect that this technique will be very beneficial when used for tool-based synthesis.

## 4 ALU Implementation Details

The ALU core adopts also strategies that go beyond simple circuitual solutions. We have also modified basic cells like flip-flops in order to augment their functionality as required by any generic ALU core. We have also studied different schemes for high-speed adder and shifter topologies, which are suitable with CMOS090 and beyond.

### 4.1 Multiplexed Registers

The core uses modified registers based on a typical tri-state inverter topology. Registers are modified so as to include special functionality such as multiplexing, demultiplexing and latching. In the multiplexing dynamic flip-flops, the selection of the input is performed while the new data is provided to the flip-flop itself – specifically in the second half of the clock cycle. This is accomplished by replacing the master latch in the flip-flop with two master latches each followed by a transmission gate (Fig.5). In this way, partial time stealing is possible that allows hiding of the multiplexing delay into the set-up time of the flip-flop itself. The penalty on the set-up time is limited to less than 5% with respect to the normal usage of the flip-flop. On the other hand, in the de-multiplexing flip-flop, the slave latch is replaced with two slave latches each followed by a transmission gate. The result is similar as for the multiplexing flip-flop. In latched flip-flops, the output of the master latch is also provided as an output (after being properly buffered). Such a gate is useful for control signals, as it makes possible stealing some timing budget from the previous stage (controlling the ALU core).

### 4.2 High-Speed Static Adder Design

The 32-bit Han-Carlson adder scheme (Fig. 6) has been selected for its shorter inter-gate interconnect and smaller area ( $300 \times 60 \mu\text{m}^2$ ), which result in smaller lateral capacitance for each composing gate. A first analysis proved this scheme to be sensibly faster than an equivalent Kogge-Stone structure [3]. The adder has been imple

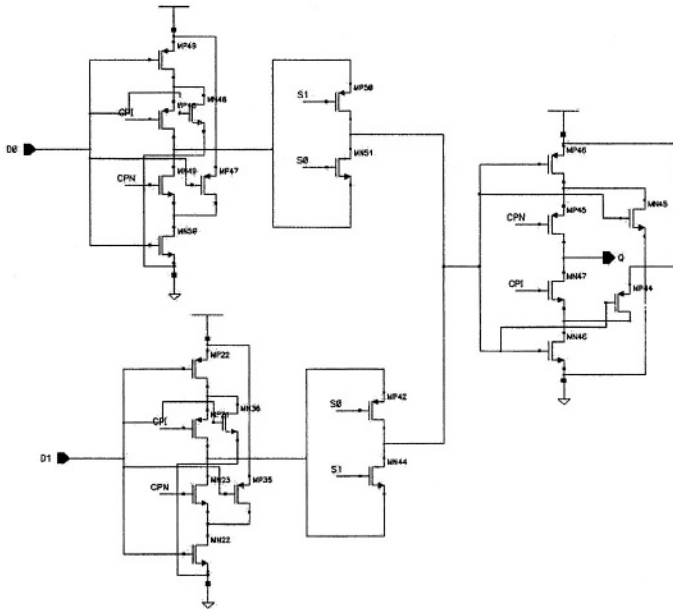


Fig. 5. Schematics for a dynamic multiplexing flip-flop.

mented by alternating inverting gates ( $/Q$  columns give inverted output and  $/D$  ones accept inverted inputs in Fig.5) so as to avoid intermediate inverters. This technique alone results in a 30% speed improvement. Every gate is static, adopts selective pre-charge and delay-balanced cells as discussed in the previous section. The 32-bit Han-Carlson adder has also been realised by alternating inverting single-rail dynamic and inverting static gates as described in [4] as a mean of comparison. This scheme resulted in a 15% improvement evaluation speed. However, this value does not include the penalty deriving by delay elements required for the distribution of a non-blocking clock in the adder as well as for a correct interfacing with other part of the ALU. Such penalties are likely to reduce the improvement to an insignificant percentage. Furthermore, the dynamic gates resulted having weaker driver strength, thus requiring over dimensioning and a more careful layout strategy.

### 4.3 High-Speed Pass Shifter Design

The 32-bit fully programmable shifter ( $300 \times 40 \mu\text{m}^2$ ) has been divided in two physical parts: a low-drive and a high-drive partition. The low-drive partition consists of the first two shifting levels (shifting one and two positions) where interconnects are sensibly shorter. In this case a pass-transistor gate with a state holder has been used (Fig.7). The state holder has been put outside the critical path as also described in [4]. The high-drive partition consists, instead, of the remainder of the shifter levels (shift four, eight and sixteen positions), where interconnect load is about four times higher. In this case a high-drive multiplexer based on tri-state has been used (Fig.8). A state holder has been also used as amplifier to reduce slew time of transitions. Also in this case the state holder has been put outside the critical path.

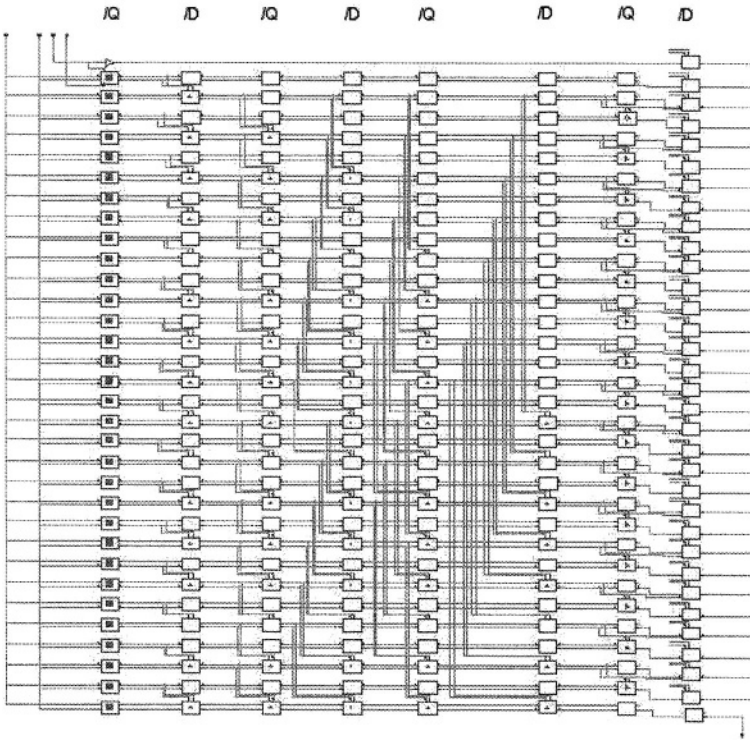


Fig. 6. 32-bit Han-Carlson adder schematics.

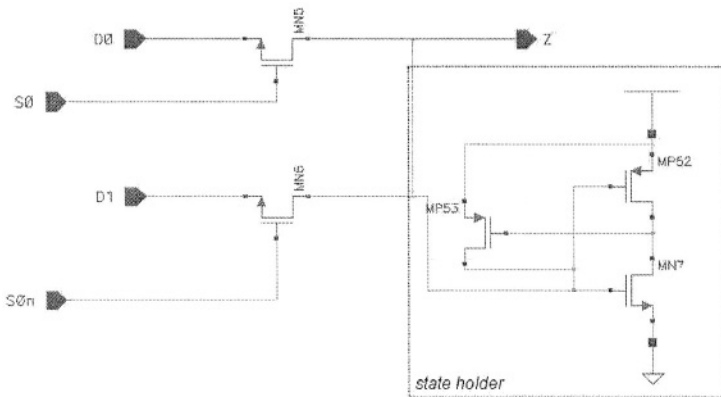


Fig. 7. Low-drive short interconnect pass-logic multiplexer schematics.

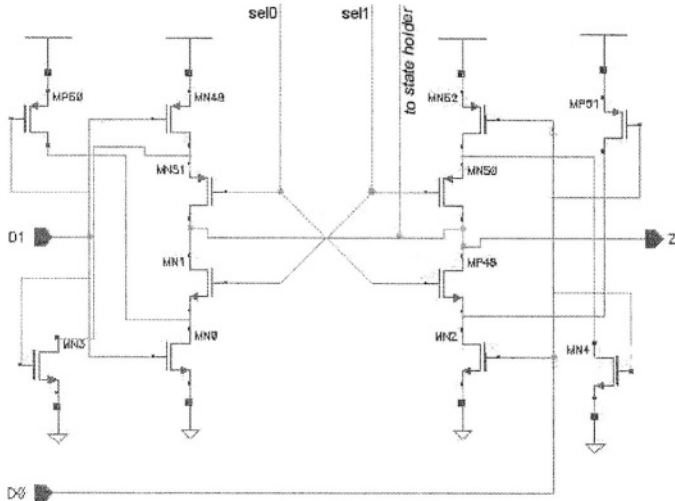


Fig. 8. High-drive long interconnect TG multiplexer schematics.

#### 4.4 De-coupling Strategy for High-Speed

More than 40% of the core area consists of de-coupling cells, which are required to avoid bounce on the supply lines. The cells have been designed in a standard cell fashion and using both types of MOS transistors. Two layout implementations have been done where the difference can be found in the layout orientation of the transistors. One implementation is optimized to achieve a maximum capacitance per area ratio at the expense of the discharge performance. Here, the channel length of both transistors varies with cell width, while their channel width stays fixed. The other implementation is optimized to achieve an improved discharge performance at the expense of the maximum capacitance per area ratio. For this case, both transistors are rotated such that their channel width varies with cell width, while now their channel length stays fixed. Both implementations of the de-coupling cell have been used in this high-speed design.

A de-coupling cell has been attached to each individual gate and flip-flop within the ALU core. The cell has been characterized for an improved discharge performance, which makes it more suitable for circuits in the GHz range. This has been accomplished by choosing the optimal implementation for a required capacitance value.

De-coupling cells have also been used to fill-up the empty spaces and the surrounding of the ALU core. These cells are mainly used to balance the effect of the logic outside the ALU core required for testing and measurement. Since this will be done at low frequencies, the discharge performance is not that much of an issue here. Therefore, the cell has been characterized for the maximum capacitance per area ratio.

## 5 Conclusions

An ALU core with cycle time below 260ps has been described. The core has been fabricated in the latest Philips-Motorola-ST 0.10um **triple- $V_T$**  CMOS technology by means of a standard semi-custom approach with improved static cells and dynamic flip-flops. The obtained ALU performs well when compared to similar design adopting a full dynamic design style [4] designed in comparable (in terms of performance) technology. Measurement and testing structure, not here discussed, have been also included in the final design so as to perform at speed testing.

## References

- [1] T. Han, D.A. Carlson, "Fast Area Efficient VLSI Adders", 8<sup>th</sup> Symposium on Computer Arithmetic, pp. 49-56, May 1987
- [2] X. Bai et al., "Uncertainty-aware circuit tuning", Proc. of 39<sup>th</sup> Design Automation Conference, New Orleans, June 2002
- [3] P.M. Kogge, H.S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations", IEEE Transactions on Computers, Vol. 22, Nr.8, pp. 786-793, Aug. 1973
- [4] M. Anders et al., "A 6.5GHz 130nm Single-Ended Dynamic ALU and Instruction-Scheduler Loop", ISSCC Digest of Technical Papers, pp.410-477, Feb 2002

# Embedded EEPROM Speed Optimization Using System Power Supply Resources

Jean-Michel Daga, Caroline Papaix, Marylene Combe,  
Emmanuel Racape, and Vincent Sialelli

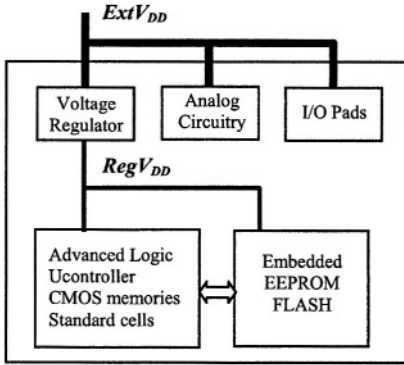
ATMEL, Zone Industrielle  
13106 Rousset Cedex, FRANCE

**Abstract.** Decreasing power supply with advanced technologies makes memory design and speed optimization more difficult. This is especially true for EEPROM and FLASH memories requiring internally high voltage supplies for programming. In addition to the well known techniques such as multi threshold voltage used to manage speed versus leakage on SRAM memories, increasing power supply resources appears to be a cost effective solution to improve speed, and decrease power. This concept already applies on most of systems on chips where 3.3V is used for analog design, while low power supply compatible with DSM devices is used for digital design. We propose to extend this approach to the design of EEPROM memories. The dual supply concept (3.3/1.8V) has been used to design a 256KB EEPROM memory on a 0.18 $\mu\text{m}$  process with embedded memory, resulting in 25ns worst case random access time, while reducing by 3.7x the size of the charge pumps, compared to the single 1.8V supply approach.

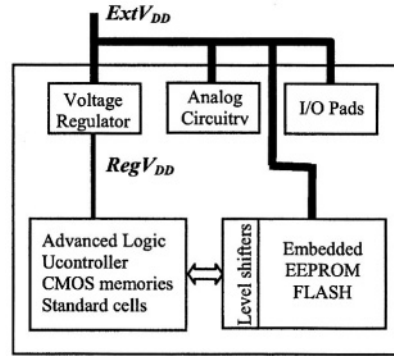
## 1 Introduction

Today, a large amount of Intellectual Property is embedded on advanced Systems on Chips resulting in several million of gates on a single chip. Because of the aggressive pressure for cost reduction, the race to technology shrinking has been pursued following the Moore's law over 30 years. In order to maintain acceptable power consumption and reliability with advanced technologies, supply voltage has been reduced from 5V with 1 $\mu\text{m}$  technology down to 1.8V with 0.18 $\mu\text{m}$  technology, and 1.2V at the current 130nm node. However, such supply voltage shrinking is not observed at system level. Actually more than 90% of the systems on chips using a 0.18 $\mu\text{m}$  technology are 3.3V compliant, 5V compliant or tolerant. The power supply management on such SOCs is described in Fig. 1. The external power supply  $\text{ExtV}_{\text{DD}}$  (3.3V or 5V) is used to supply the I/Os and most of the analog circuitry. A voltage regulator generates the internal regulated power supply  $\text{RegV}_{\text{DD}}$  (1.8V for a 0.18 $\mu\text{m}$  logic), which is used to supply all the advanced logic including the micro-controller, CMOS memories, glue logic etc.

For many reasons such as noise immunity, precision requirements, device limitations, the analog circuitry does not take the same benefit as the logic parts from the scaling of the technology. In practice, most of the analog circuitry is done using 3.3V dedicated devices, in order to maintain the same level of design quality and perform-



**Fig. 1.** First option for power supply management of advanced SOCs



**Fig. 2.** Second option for power supply management of advanced SOCs

ance whatever the technology node is. Until now, the CMOS logic and memories have been shrunk, resulting in 50% of area saving at each new generation, while improving performances. At the 130nm node and below, it becomes more difficult to improve delay performances without increasing the leakage current in a prohibitive way for portable applications.

Design optimization is done using low speed, high threshold voltage devices where delay is not critical, or in very large SRAM arrays for example, in order to save leakage [1]. High speed, low threshold devices are used to improve speed on critical path. Multi VT is an extensively used solution to the leakage problem, at the expense of additional process steps and cost, as well as overall system speed decrease. Better power delay compromises should be also obtained by managing different power supplies. For example, 1.5V should be used on SRAM arrays to improve speed, while maintaining device reliability.

Regarding now embedded EEPROM or FLASH memories, there are several options. First option, currently used on advanced systems on chips is to supply the memory using the regulated power supply as in Fig. 1. This option makes it possible to use advanced CMOS logic for the design of the memory logic, resulting in improved density and speed on these parts [2]. Unfortunately, lowering the power supply of the EEPROM or FLASH memory results in several issues during memory writing and reading. Because the high voltage potentials used for program and erase operation do not shrink, the sizes of the charge pumps increase. Moreover, because the memory cell current during read operation greatly depends on the word line voltage, boosting the word line above the regulated voltage during read is commonly used to provide a good functionality [3]. However, boosting is time and current consuming. A second option, illustrated in Fig. 2 is to supply directly the EEPROM or FLASH memory with the external supply voltage. That way, charge pump sizes can be reduced, and no boosting during read is necessary. This option has two main drawbacks: the logic parts of the memory must be designed using thick oxide devices, because the thin oxide devices cannot support the external supply. This makes control logic, pre-decoding and output data-path larger and slower compared to the low voltage option. The second drawback is that level shifters are necessary at the inputs and outputs of the memory, to allow communication with the rest of the circuit supplied with the regulated voltage.

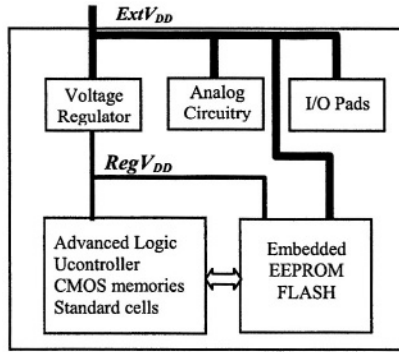


Fig. 3. Proposed approach for power supply management on advance SOCs

We propose here a third option described in Fig. 3, taking benefit of the available external supply and regulated supply for the EEPROM memory. The low voltage, logic parts of the memory use thin oxide devices and are supplied by the regulated internal supply, while the external supply voltage is directly supplied to the charge pump for programming, to the word line and bit line decoding during the read operation. This dual supply approach is the optimal solution for speed and area enhancement, making the use of optimal low voltage, high-speed devices for decoding and sensing design possible, while avoiding internal boosting delays during read, and over-sizing of the write charge pump. The dual supply memory is described on section 2. Memory simulation aspects are addressed on section 3, silicon results on a 256KB EEPROM memory are reported on section 4, and a conclusion of this work is given on section 5.

## 2 Dual Supply EEPROM Memory Description

### 2.1 General Description and Power Supply Management

A description of the dual supply EEPROM memory is given in Fig. 4. It is composed of a memory array including the memory cells, a X decoding including the pre-decoding and word line drivers, a Y decoding including the pre-decoding, the select driver and the bit line select modules. A module including the sense amplifiers (SA) and data out path is connected to the bit line select. The column latch module [4] function is twofold: store the data that will be programmed in parallel in the array, and drive the corresponding bit lines of the cells that are written to the high voltage. The charge pump [5] is used to generate the high voltage necessary to program the memory cells. Vmctrl block at the output of the charge pump supplies the external supply or the high voltage to the Vmrw node during read and program operation respectively. In addition, the control logic part is necessary to manage functional and test modes, as well as the writing delays. Three different powers are supplied to the different parts of the memory: the external power supply,  $ExtV_{DD}$ , the regulated power supply,  $RegV_{DD}$ , and the programming voltage that is generated internally. The control logic, Y and X pre-decoding, sense amplifiers and data out logic are supplied by the



regulated supply voltage. The charge pump and select driver are directly supplied by the external supply.

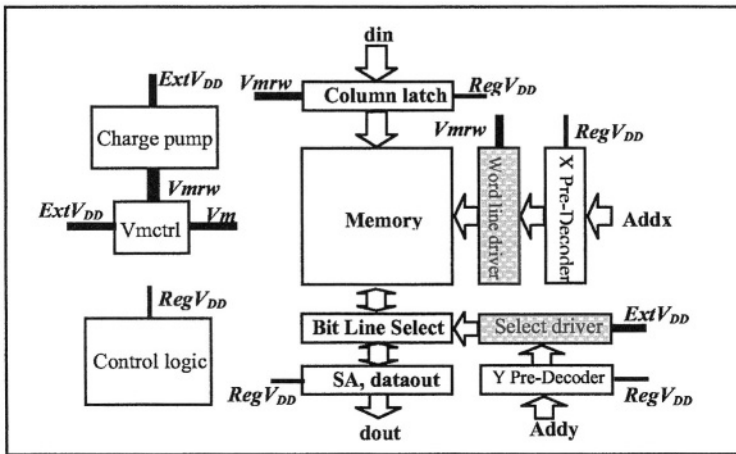


Fig. 4. General description of the dual supply EEPROM memory

## 2.2 Detailed Description

**Vmctrl block.** The Vmctrl block function at the output of the charge pump is as follows: during the read operation it provides the external supply  $ExtV_{DD}$  to the word line driver, during the write operation it provides the high voltage to the column latches and the word line driver as well. The electrical schematic of this block is given in Fig. 5. At the beginning of the read operation, Vmrw node potential is equal to  $ExtV_{DD} - V_t$ , where  $V_t$  is the threshold voltage of the path transistors of the charge pump. During the read operation, the read input is driven to high value, and the external supply  $ExtV_{DD}$  is directly connected to the Vmrw node through P3 device, and can be supplied to the word line driver. During the write operation, the read input goes low, and P3 is turned OFF. Because P3 is OFF, there is no DC path between the high voltage and Vext.

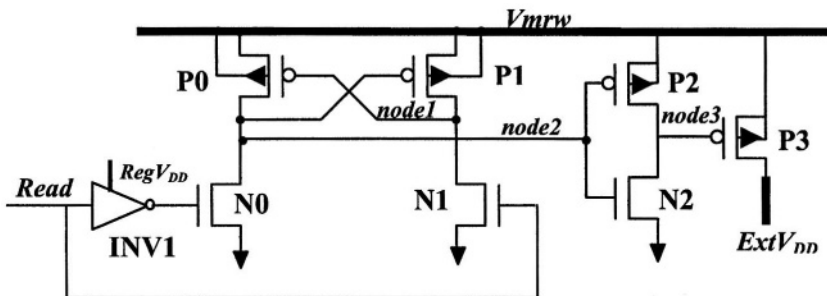


Fig. 5. Schematic description of the Vmctrl block

**Word line driver.** The signal  $V_{mrw}$  is provided to the word line driver which function is twofold: connect the word line to the high voltage during write, to the external voltage  $ExtV_{DD}$  during read. Word line driver schematic is given in Fig. 6. Transistors (P10, P11, N10, N11) act as a level shifter to drive the input of transistor P12 to low if the word line is selected (in1 input high) or  $V_{mrw}$  if the word line is not selected (in1 input low). If the word line is selected, the high voltage is transmitted to the word line through P12 during write operation. Transistor N12 is a low threshold voltage  $V_{thN12}$  N transistor which function is to speed up the word line rising delay in read. During the read operation,  $ctrl1$  signal is set to  $RegV_{DD}$ , N12 starts conducting. The word line rises to  $RegV_{DD} - V_{tN12}$  thanks to N12 and P12 supplying current in parallel. When the word line rises from  $RegV_{DD} - V_{tN12}$  to  $ExtV_{DD}$ , N12 is OFF, and P12 only drives current.

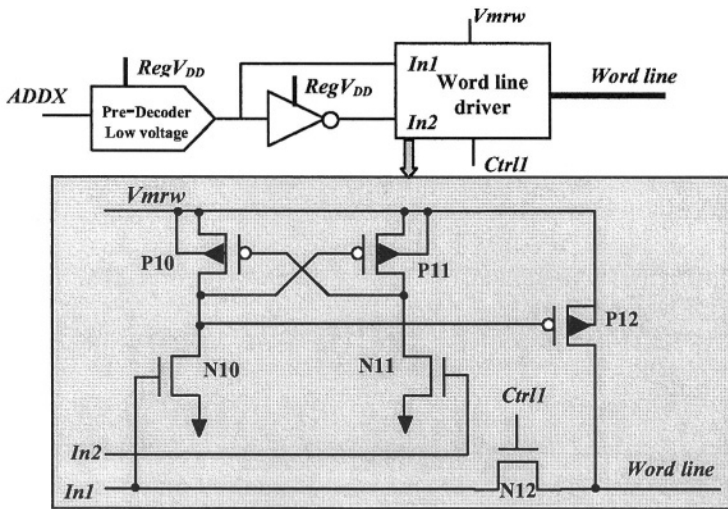


Fig. 6. Word line driver description

**Bit line select.** The bit line selection for read is done thanks to the bit line select block, which detailed description is given in Fig. 7. There are two serial transistors between the sense line connected to the sense amplifier, and the bit line connected to the memory cells. The devices connected to the bit lines and controlled by the  $yr_{dh\_i}$  signals have their drain connected to the high voltage during write operation. Thus, these devices are thick oxide, large effective length, poor gain devices. The transistor connected to the sense line, and controlled by the  $yr_{dl}$  signal, is not connected to the high voltage. So a thin oxide, high drive device is used. During read operation, the bit line must be precharged through the bit line select transistors in a minimum amount of time. Because of their intrinsic poor drive, the high voltage devices limit the speed of the precharge. A solution to increase their speed is to increase their width, but this is very expensive in terms of area. The other solution proposed here is to drive their gate to  $ExtV_{DD}$ . The select driver block of Fig. 8 is used to generate the  $yr_{dh\_i}$  signals. The low voltage signal  $yr_{dhlv\_i}$  at the output of the pre-decoding block is shifted to  $ExtV_{DD}$  thanks to a level shifter.

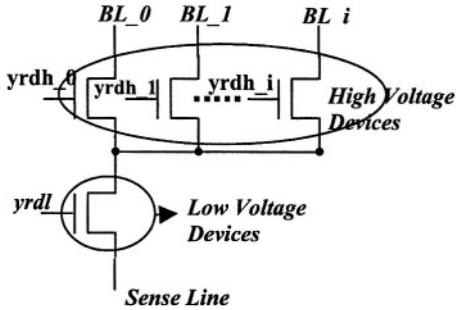


Fig. 7. Bit line select description

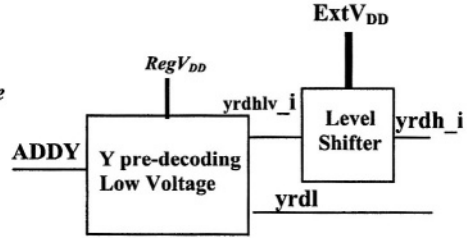


Fig. 8. Select driver description

### 3 Memory Simulation

Extensive simulations have been done to validate the memory design. First of all, the charge pump has been optimized. Then, the access time has been simulated on extracted critical path, considering all the read sequences that should occur when accessing to the memory in a fully asynchronous way.

#### 3.1 Charge Pumps Simulation

Driving the charge pump using ExtV<sub>DD</sub> instead of RegV<sub>DD</sub> results in a big area saving. It has been demonstrated [6] that the charge pump efficiency is inversely proportional to the power supply, and to the number of stages. Using ExtV<sub>DD</sub>=3.3V instead of RegV<sub>DD</sub>=1.8V on a 0.18μm 2Mbits EEPROM memory, the charge pump area has been reduced by 3.7. This has been obtained while keeping constant the charge pump output voltage and supplied current.

#### 3.2 Read Access Time Simulation

Current sensing is used for memory read [7]. The memory cell current flows across the bit line and is sensed by the current sense amplifier. When the memory cell current is above the trip point of the sense amplifier, a '0' value is read. When it is under the trip point, a '1' value is read. The '0' is obtained after programming the cell, and turning its threshold voltage to negative value, resulting in a read current within 10 to 20uA. The '1' is obtained after erasing the cell and turning its threshold voltage to a positive value, then resulting in no cell current when reading [8].

Accurate simulation of the access time is mandatory but not easy at all. The access time depends on a number of parameters such as bit line and word line capacitance and resistance, cross coupling, cell current and noise, sense amplifier characteristics, decoding speed [9]. It depends also on the read sequence, for example reading a 0 after a 0 on the same bit line results in a different access time than reading 0 after 0 on two different bit lines and so on. At the end, the worst case delay is obtained when the worst case read sequence is applied, using the worst case conditions for process, temperature and power supply (P, V, T). Memory design optimization for speed will

consist on balancing the different delays obtained using the different sequences in order to achieve the best performances.

### Read 0 (ON Cell) Optimization

The read '0' access time is the delay necessary to bias the memory cell in such conditions that its current is above the sense trip point, in addition to the sense amp current to voltage conversion delay. The biasing delay is the time necessary to drive the word line and the bit line to the correct values. The word line driver is shown in Fig. 6. The bit line precharge is controlled by the sense amp circuitry as shown in Fig. 10, and is clamped to around 1V to avoid any kind of read disturb [10].

When optimizing the read '0' delay, decoding and word line rising speed are the key parameters. At the beginning of the read, the bit line current is first dominated by the precharge current, then by the cell current. In order to get a smooth transition without any glitch to '1' value, it is desirable to have the cell current rising above the sense trip point before the end of the precharge. So in this case, precharge speed is not critical. The bit line current when reading an ON cell is illustrated in Fig. 9a.

In single supply memories, boosting the word line using an internal voltage doubler is an efficient approach to improve read '0' speed [11]. Unfortunately, boosting is time consuming, and this is critical if we target very fast read access time. Assuming for example a 2Mbits EEPROM in 0.18 $\mu\text{m}$  technology, 1.8V single supply operation, it is 11ns to drive the word line to 2V, 30ns for the same word line to reach 2.5V using an optimized voltage doubler. This delay has been reduced to 5ns and 9ns respectively using an external 3.3V supply voltage.

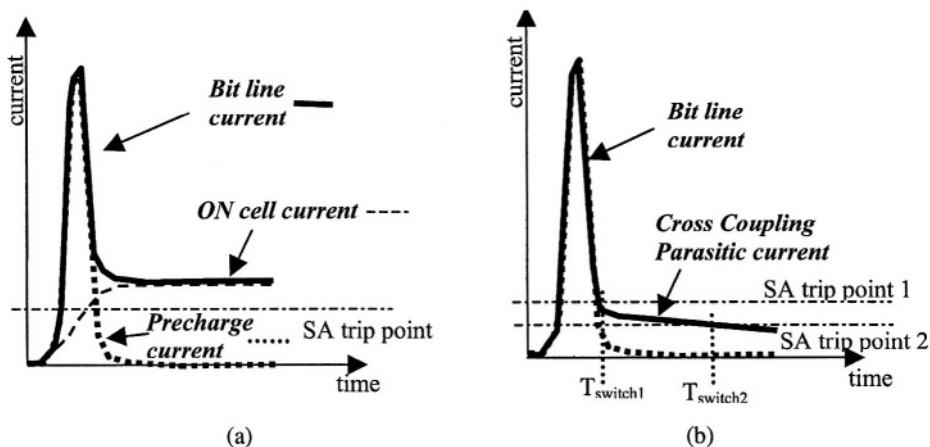


Fig. 9. (a) BL current when reading an ON cell ('0'), (b). BL current when reading an OFF cell('1') surrounded by ON cells (Fig. 10)

### Read 1 Optimization

The read '1' access time is the delay necessary to cancel the current flowing across the bit line, in addition to the sense amp current to voltage conversion delay. In the ideal case, the bit line current when reading a '1' cell is equal to the precharge current, that cancels when the bit line has been driven to approximately 1V thanks to the

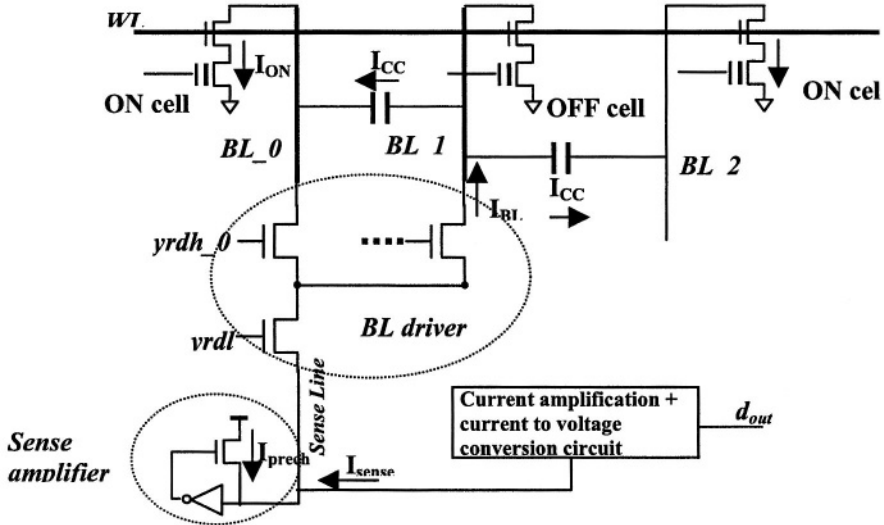


Fig. 10. Sense amplifier to memory cell read circuitry and currents

sense amp circuitry illustrated in Fig. 10. So, bit line precharge speed optimization is critical. This is achieved by correct sizing of the sense amplifier precharge circuitry, as well as BL select circuitry and driver optimization using the dual supply concept described in Fig. 7 and Fig. 8. Anyway, depending on the read sequence, there are some constraints due to the cell environment making the switching operation different from the ideal case. For example, if an ON cell conducting current has been previously read on the same bit line, it is mandatory to cancel its current before the end of the precharge. This is obtained by accurate sizing of the word line discharge circuitry. More difficult to simulate and overcome is the parasitic current induced by the bit line to bit line coupling capacitance. This is illustrated in Fig. 10. Assume that the OFF cell connected to the bit line 1 is read after reading the ON cell connected on bit line 0, and that the cell connected to the bit line 2 is also an ON cell. At the beginning of the read operation, bit line 0 is to 1V approximately, bit line 1 and bit line 2 are to 0V. The precharge occurs on bit line 1. In this case, bit line 1 rises very fast to 1V. Due to the coupling capacitance the bit line 2 is also increased to a lower value. At the end of the precharge operation, it is expected to have zero current on the bit line 1, allowing sense amplifier switching to 1. Unfortunately, the cells connected to bit line 1 and 2 have their word line to 'H' value, and they drive current. Because bit line 1 and 2 are not driven by the precharge circuitry, they decrease slowly to 0V thanks to the ON cell current. This variation on bit line 0 and 2 results in a current flowing from bit line 1 which is maintained to 1V to bit line 0 and 2 across the coupling capacitances. This current flows through bit line 1, and is seen by the sense amplifier that does not switch to 1 as long as the current is above the trip point. This is illustrated in Fig. 9b. Assuming  $C_{BL}$  is the bit line capacitance,  $C_{CC}$  the cross coupling capacitance and  $I_{ON}$  the current of the ON cell, the current flowing across the coupling capacitance can be approximated by:

$$I_{CC} = (1 / (1 + C_{BL} / C_{CC})) * I_{ON}$$

This current can cause big perturbations on advanced processes where  $C_{cc}$  is on the same order of magnitude as  $C_{bl}$ . This is a good example of the read access time sensitivity to the cell environment and read sequence. Assuming that OFF cells are now connected to bit line 0 and 2, the cross coupling current won't exist and no parasitic effect will perturb the ideal read 1 case.

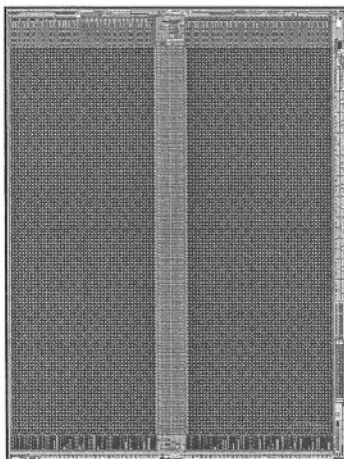
Using this approach together with the Word line driving to  $ExtV_{dd}=3.3V$  the access time of the 2Mbits EEPROM memory has been decreased to 25ns. 40ns is obtained using a boost strategy on a single  $RegV_{dd}=1.8V$  supply memory [11].

### 4 Results

A 256KB EEPROM with page erase granularity has been designed using 3.3V and 1.8V supplies. This memory has been processed on the 0.18 $\mu m$  CMOS plus embedded EEPROM of ATMEL. The layout of the memory is shown in Fig. 11. Access time has been measured on a dedicated test chip using different patterns, such as checkerboards (CKB), all '00' or all 'FF'. Sense amplifier trip point was tunable, using programmable options. The measured access time on different patterns is reported on Table 1 for different sense amplifier tuning conditions. As expected from simulations, small sensitivity to the sense trip point has been measured when reading 'FF'. An important delay increase is observed on CKB when decreasing the sense amplifier trip point, due to the previously described parasitic effect. This measurements have been done at 1.6V/3.0V power supplies operation.

**Table 1.** Measured access time versus sense amplifier trip point

Sense trip point ( $\mu A$ )	4.5	6.5	10
Access time on CKB (ns)	43	27	21
Access time on FF (ns)	23	22	20
Access time on 00 (ns)	20	22	24



**Fig. 11.** Layout of the 256KB EEPROM

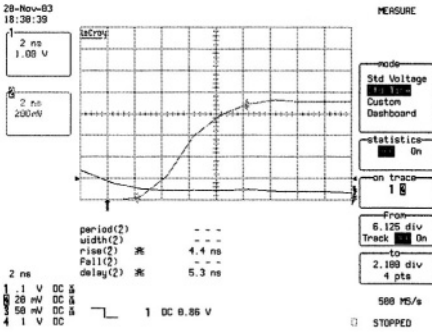


Fig. 12. Micro-probing of the BL in read

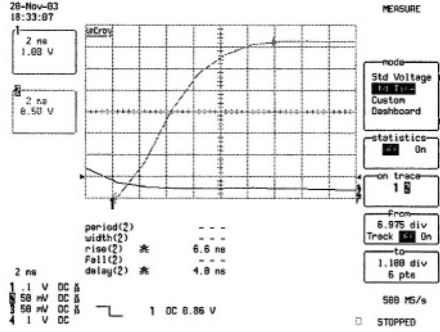


Fig. 13. Micro-probing of the Word line in read

Micro-probing of the critical nets during read operation has been performed. Bit line rising curve during precharge is reported in Fig. 12. As can be observed, it is about 10ns for the

bit line to reach its clamped value at 950mV. Fig. 13 shows the word line rising operation at 3.6V. It is less than 10ns for the word line to reach 3.5V. This good balancing between word line rising delay and BL precharge allows optimal read conditions.

## 5 Conclusions

The dual supply concept has been successfully used to improve EEPROM memories speed, power and area. 25ns access time has been achieved on a 256KB memory, to be compared to the 40ns obtained for the same memory size using 1.8V only [11]. Size of the block has been reduced according to the charge pumps area reduction of 3.7x. Finally power has been reduced, during programming because efficiency of the pump is improved linearly when increasing the power supply, in read because no more voltage doubler is necessary to improve speed. This approach is cost effective because the power supply resources are already available at system level, where 3.3V on dedicated devices is commonly used for analog design, even at the 130nm technology node. In addition to the already design or technological solutions implemented to deal with deep submicron issues such as leakage current, increasing power supply resources appears to be a very promising solution, not only for EEPROM/FLASH memories, but for all types of memories as well.

## References

- [1] K. Itoh, “Low-Voltage Embedded-RAM Technology: Present and Future”, IFIP International Conference on VLSI, Montpellier France, December 2001.
- [2] JM. Daga et al, “Design Techniques for EEPROMs Embedded in Portable Systems o Chips”, Design and Test of Computers Journal, IEEE, January-February 2003, pp. 68-75.

- [3] T. Tanzawa and S. Atsumi, "Optimization of Word Line Booster Circuits for low-Voltage flash Memories", *J. Solid-State Circuits, IEEE*, vol. 34, no. 8, Aug. 1999, pp. 1091-1098.
- [4] Tah-Kang J. Ting et al, "A 50-ns CMOS 256K EEPROM", *J. Solid-State Circuits, IEEE*, vol. 23, no. 5, Oct. 1988, pp. 1164-1170.
- [5] J. F. Dickson, "On-Chip High-Voltage Generation in NMOS Integrated Circuits Using an Improved Voltage Multiplier Technique", *J. Solid-State Circuits, IEEE*, vol. 11, no. 3, June 1976, pp. 374-378.
- [6] C. Papaix and JM Daga, "High Voltage Generation For Low Power Large VDD Range Non Volatile Memories", *Proc. PATMOS*, September 2001, Yverdon Switzerland.
- [7] C. Papaix and JM Daga, "A New Single Ended Sense Amplifier for Low Voltage Embedded EEPROM", *IEEE MTDT*, July 2002, Bandol France.
- [8] P. Cappelletti et al, "Flash Memories", Kluwer Academic Publishers, 1999.
- [9] Nobuaki Otsuka and Mark A. Horowitz, "Circuit Techniques for 1.5-V Power Supply Memory", *IEEE J. Solid-State Circuits*, vol. 32, no. 8, pp. 1217-1230, August 1997.
- [10] Bruce Euzent et al, "Reliability Aspects of a Floating Gate EEPROM", *Proc. IRPS*, pp. 11-16, 1981
- [11] JM. Daga et al, "A 40ns random access time low voltage 2Mbits EEPROM Memory for embedded applications", *IEEE MTDT*, July 2003, San Jose USA.



# Single Supply Voltage High-Speed Semi-dynamic Level-Converting Flip-Flop with Low Power and Area Consumption

Stephan Henzler<sup>1</sup>, Georg Georgakos<sup>2</sup>, Jörg Berthold<sup>2</sup>, and  
Doris Schmitt-Landsiedel<sup>1</sup>

<sup>1</sup> Munich University of Technology, Theresienstr. 90, 80290 Munich, Germany  
henzler@tum.de,

<sup>2</sup> Infineon Technologies AG, Coporate Logic, Balanstr. 73, D-81541 Munich, Germany

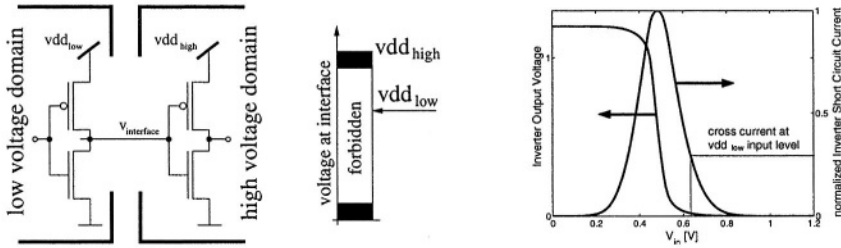
**Abstract.** A novel semi-dynamic level-converting flip-flop is proposed and compared to standard level shifter circuits concerning speed, power, integration and soft-error-rate. The flip-flop operates at one single supply voltage which allows most compact design as well as reduced place and route effort. In addition to its superior delay and power consumption properties, it offers simple implementation of robust multi-vdd low power systems with minimum area overhead and easy integrability into standard design flows.

## 1 Introduction

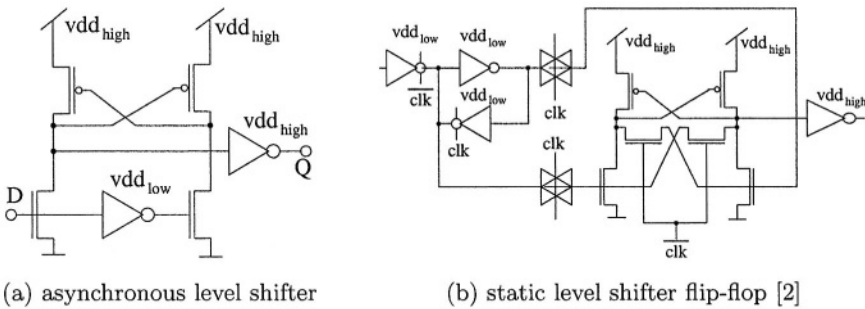
In deep sub-micron CMOS technologies, power consumption will become one of the major limitation for low-power mobile as well as high performance systems. Exponentially increasing transistor count per chip as well as technologies with higher leakage currents require sophisticated circuit strategies. One promising approach is to use different supply voltages in different circuit blocks or in different paths within one single block in order to reduce dynamic power consumption. If the transistor threshold voltage is held constant gates working with the lower supply voltage will have lower speed. Gates working with the higher supply voltage have better speed performance at the cost of increased power consumption. Hence the high supply voltage should only be used where the higher speed is absolutely required.

In circuit blocks with a high switching activity dynamic power is still the major contributor to energy consumption. Hence it is beneficial to reduce this power component by reducing the supply voltage while the switching speed is held constant due to simultaneously reduced transistor threshold voltage.

If there are used multiple supply voltages in static CMOS logic, the interface between different voltage domains is critical: A logic high level out of the domain with the lower supply voltage may fall within the forbidden input voltage region of the following stage (see Fig. 1). This may result in an undefined output level but in each case the static cross currents of the destination gates are greatly increased by not being switched fully. Therefore level converting circuits



**Fig. 1.** At the interface of the voltage domains, input voltage may fall into forbidden voltage range of subsequent gates. Additionally static cross currents are increased dramatically.



(a) asynchronous level shifter

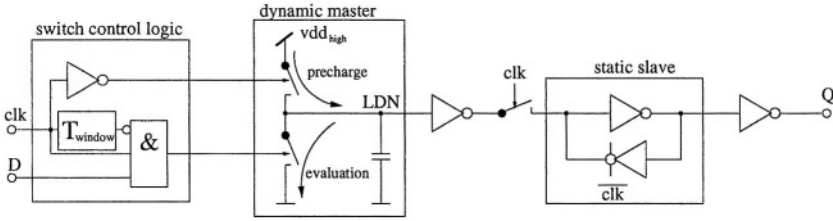
(b) static level shifter flip-flop [2]

**Fig. 2.** CMOS realization of conventional static level-converters

have to be inserted at the interface. In dynamic CMOS logic a non-full swing signal does not cause static cross currents. Hence additional level shifter cells are not required because the level conversion is done within the dynamic gate. Unfortunately, dynamic circuit styles are barely suited for reliable low-power mobile applications in undefined varying environment. In new deep sub-micron technologies this fact is worsened by increasing leakage currents and increasing soft-error susceptibility due to decreased device feature size. For such applications the superior robustness of single-ended static CMOS logic is exploited and hence sophisticated level shifter circuits are required. Fig. 2 shows two conventional level shifter circuits: The asynchronous level shifter in Fig. 2(a) may be inserted anywhere within the logic, whereas Fig. 2(b) shows a combination of the level shifter and a flip-flop circuit.

The major drawback of both of these level shifters is the fact that they cause additional delay and power consumption. Thus the lower supply voltage can only be used in paths that are sufficiently shorter than the critical path, because the power and delay penalty of the level shifter have to be gained first, before the use of multiple supply voltages is beneficial at all. Consequently the requirements for a good level shifter are low power consumption and minimum propagation delay.

A serious drawback of conventional shifter cells in static single ended CMOS is that they need both supply voltages within the cell. This does not only complicate the power routing but also increases the total cell area because of ESD and latch-up design rules.



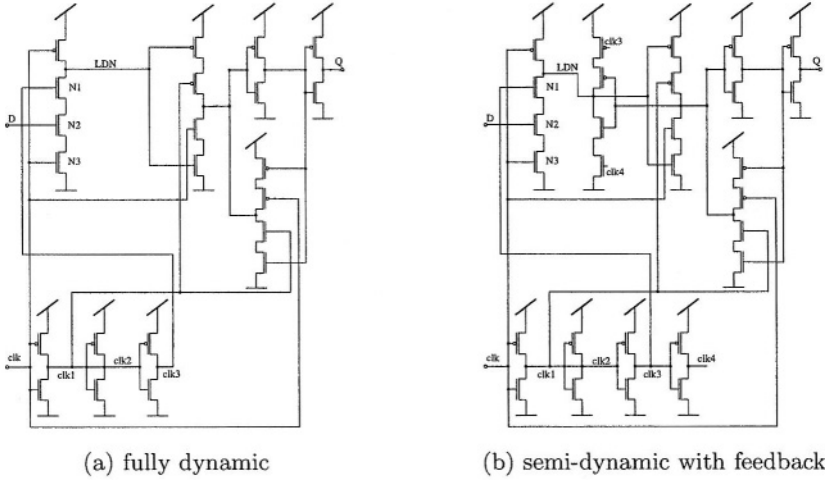
**Fig. 3.** Principle of dynamic edge-triggered level-conversion-flip-flop. As the logic decision node (LDN) is high ohmic after the evaluation phase the amount of charge stored on its node capacitance is not critical for correct operation. Insensitivity to leakage currents can be achieved by the proposed semi-dynamic design.(see below)

## 2 Principle of Dynamic Level Conversion

The major drawbacks of conventional level shifter designs result from the positive feedback and the usage of both supply voltages within the shifter cell. Therefore we propose a new principle of level conversion which omits feedback during the level decision. In contrast to [1] where the data signal has to be inverted this approach uses only one single supply voltages. The idea is to precharge the logic decision node (LDN) to the high supply voltage  $vdd_{high}$  and to discharge this node on a rising clock edge if the input is logically high (Fig. 3). Only one single n-channel device is affected by the input (or only a p-channel device in the complementary case), and no cross currents occur. A simple possibility to control the precharge and the evaluation phase is to apply the timing of a flip-flop: During the inactive phase of the clock ( $clk = 0$ ) the LDN is precharged to  $vdd_{high}$ . When the clock rises, the precharge path is cut-off and the evaluation path is sensitized: If the input level is high at this moment, the LDN will be discharged to ground. In order to get an edge triggered flip-flop, the evaluation path has to be desensitized a few moments after the rising edge of the clock signal. As shown in Fig. 3 this may be achieved by a delay element. Following the LDN, a conventional static latch stores the data when the clock returns to zero again.

A simple realization of the concept is given in figure 4 (a). The time window during which the evaluation path is sensitized is defined by an inverter delay chain. The switch control logic is realized by the three serially connected transistors N1, N2 and N3. A two stage switch control logic can be used to avoid three serially connected transistors for better scaling properties. It is important to notice that the input is connected to only one n-channel transistor. Therefore no cross currents occur, and no constraints have to be made for the input voltage except that it has to be sufficiently larger than the threshold voltage of transistor N2. Sufficiently large means that the capacitance of the LDN can be discharged during the time  $T_{window}$  which enables the evaluation path according to

$$\tau_{discharge} < T_{window} \approx T_{hold} \ll \frac{1}{f_{clk}} \quad (1)$$



**Fig. 4.** CMOS realization example of dynamic level shifter flip-flops. The semi-dynamic architecture combines the speed advantage of dynamic circuits with the reliability advantages of static CMOS logic.

where  $\tau_{discharge}$  is the dominant time constant for discharge of the capacitor via the three transistors N1 to N3,  $T_{hold}$  is the hold time of the flip-flop and  $f_{clk}$  is the clock frequency of the system. The upper bound is important for the flip-flop to appear edge triggered and the lower one to work at given  $vdd_{low}$ .

### 3 Robust Semi-dynamic Architecture

In the design described above, data is represented by the charge stored on the LDN while the clock signal is high. On the falling clock edge, data is transferred into the static slave and the LDN is precharged to  $vdd_{high}$  again. This design ensures optimum speed and area efficiency, but the charge on the LDN is affected by leakage currents, noise, cross coupling and the injection of charge generated by radiation (soft error rate, SER). If one or more of these disturbances become serious, the error probability of the flip-flop will increase. Especially in deep sub-micron technologies this becomes noticeable because of their growing leakage currents and shrinking intrinsic capacitances.

Hence there is a minimum clock frequency  $f_{min}$  for which the flip-flop like each dynamic circuit can be used. For slower operation the charge representing the information on the LDN is corrupted by gate and subthreshold currents even if there is no external disturbance. For ultra low-power applications strong reduction of the clock frequency or even clock gating in conjunction with a reliable signal processing is important. Thus, to exploit the benefits of dynamic level conversion under these nonideal conditions, we propose the semi-dynamic level shifter flip-flop given in Fig. 4(b). Compared to the first flip-flop there is a feedback protecting the LDN against disturbances immediately after the level

decision has been done. As the feedback is only active after the evaluation phase fast signal propagation due to dynamic level conversion is achieved. Due to the feedback and the internal self-timed structure ( $T_{window}$  is generated within the flip-flop) there is no minimum operation frequency and data is stored reliably even with clock gating. This is the main advantage over true-single-phase-clocked latches/flipflops which are widely used in dynamic circuits. As they store the information dynamically they are neither suited for low-power applications nor for noisy operation environment.

In [1] an alternative precharge level converter flip-flop is proposed. Performance and design aspects of this circuits with respect to the flip-flop proposed in this paper are discussed in section 9.

## 4 Performance

The signal delay of a CMOS gate depends on the supply voltage and the transition time of the input signal. Since the slave and the interface circuit between master and slave consist of conventional CMOS gates, the delay of these gates depends on the transition time of the logic decision node (LDN). In the case of a logical high level at the input, the latter time depends on the discharge velocity of the LDN, determined by the current through the transistor stack N1, N2 and N3. On the rising edge of the clock, transistor N3 switches on and discharges the small capacitance at its drain node. Transistor N2 has the smallest overdrive, thus the total current is mainly limited by this device. Hence at the beginning of the discharge process nearly the whole LDN potential drops across drain and source of transistor N2 which therefore operates in saturation region. According to the alpha-power-law [6] the discharge current can be approximately modeled by

$$I_2 \approx \frac{1}{2} \beta_n (v_{dd_{low}} - v_{tn})^\alpha \quad (2)$$

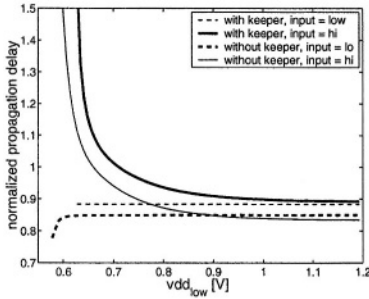
Thus the slope dependent delay component of the interface gate evolves corresponding to

$$d_{interface} \propto \frac{1}{V_{LDN}} \propto \frac{1}{(v_{dd_{low}} - v_{tn})^\alpha} \quad (3)$$

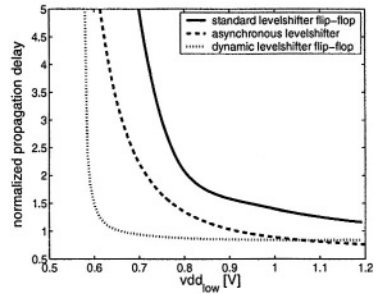
and the propagation delay of the flip-flop increases with decreasing  $v_{dd_{low}}$ . It should be mentioned that the input voltage level only affects the discharge velocity of the LDN. Hence the effect on the total delay of the flip-flop is small over a wide range of input voltages.

In the case of a logic zero at the input, the LDN is not discharged and the LDN potential is already valid when the interface switches on. Therefore the propagation delay of a logical zero is independent of the value of  $v_{dd_{low}}$ .

Fig. 5 shows simulation results of the two flip-flop versions for an industrial 90nm CMOS technology. The delays are normalized to the delay of a static standard (transmission gate) flip-flop at  $v_{dd_{high}}$  with equal drive strength. As expected, the clk-to-Q delay for a logic zero at the input is independent of



**Fig. 5.** Clk-to-Q delay dependence of the two dynamic level shifter flip-flops on the minor supply voltage. The delays are normalized to the delay of a static standard (transmission gate) flip-flop.



**Fig. 6.** Delay dependence of different level-converting circuits on the lower supply voltage with a high supply voltage  $vdd_{high} = 1.2V$ .

the  $vdd_{low}$  potential, whereas for a high input, this quantity increases slightly with decreasing  $vdd_{low}$  potential. This means that these level shifter cells, like conventional level shifters, have to be characterized for each input value.

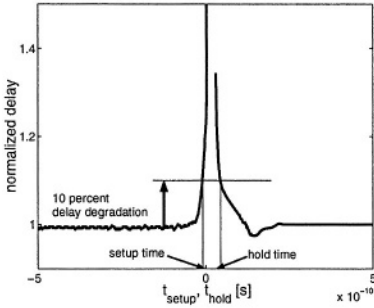
Another result given by Fig. 5 is the higher delay of the feedback protected version. Yet, the delay penalty of the feedback is less than 10% and is approximately independent of the value of  $vdd_{low}$ . Finally it is worth mentioning that over a wide range of  $vdd_{low}$  the clk-to-Q delay of the level shifter flip-flops is smaller than the delay of static standard flip-flops due to the dynamic evaluation principle. Therefore, there is no timing limitation concerning the applicability of the dynamic level shifter flip-flops and they may substitute all flip-flops in a design, whether the level shifter operation is needed or not.

Fig. 6 shows the normalized propagation delay of the classical asynchronous level shifter, the conventional level shifter flip-flop and the new semi-dynamic version versus the lower supply voltage. It can be seen that the latter not only is the fastest level-converting circuit but also has the weakest delay dependence on the  $vdd_{low}$  potential. This is beneficial because a multi- $vdd$  system using these flip-flops is less sensitive to  $vdd_{low}$  variations and hence the cost for robust systems decreases.

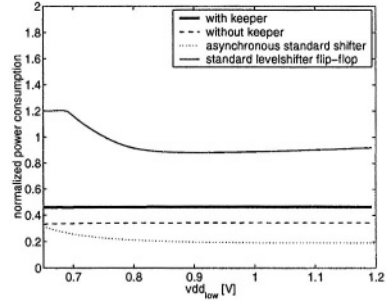
In flip-flops, two important quantities are the setup and hold times. Fig. 7 shows the dependence of the delay on these quantities. The setup time is small because the evaluation phase of the master starts with the rising edge of the clock. It can be seen that setup and hold behaviour of the level shifter flip-flops do not impose any limitations to their applicability.

## 5 Power Consumption

Fig. 8 shows the normalized energy dissipation of the different level shifters. Concerning the dynamic level shifters, energy losses due to precharging the LDN are



**Fig. 7.** Setup and hold behaviour of the semi-dynamic level converter flip-flop with feedback at  $vdd_{low} = 0.8V$  and  $vdd_{high} = 1.2V$ . The delay is normalized to the delay of a standard transmission gate flip-flop.



**Fig. 8.** Power consumption of different level shifter circuits during two switching cycles, normalized to power consumption of a static standard flip-flop at  $vdd_{high}$

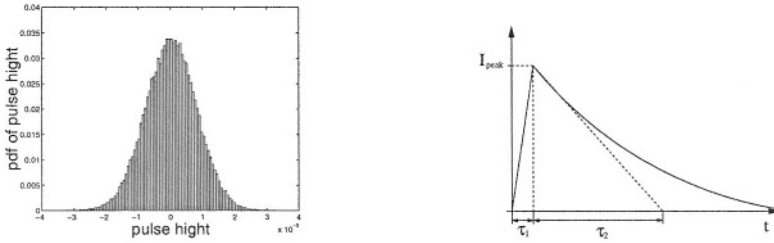
independent of the  $vdd_{low}$  value. As the whole dynamic level shifter circuit operates at  $vdd_{high}$ , the energy dissipation is nearly independent on the input voltage. On the other hand, level shifter circuits using feedback (Fig. 2) suffer from signal contentions and therefore from long input voltage dependent switching times and large cross currents. Therefore, the dynamic level shifters are preferable also from power aspects.

## 6 Performance Improvements

As described above, at very low input voltages the total clk-to-Q delay of the circuit is limited by the discharge process of the logic decision node (LDN). The current discharging the LDN strongly depends on the threshold voltage of the transistors N1, N2 and N3. To increase circuit speed at very low input voltages, it is suggested to use low or zero threshold voltage devices, if the process allows this option, or to apply a forward body bias. Static power consumption is only slightly affected by this because only one single current path in the whole circuit contains the leaky transistors. As the evaluation time  $T_{window}$  is small the reliability of the semi-dynamic flip-flop is not affected because the feedback compensates for increased leakage currents.

## 7 Impact of Single Supply Voltage

The most challenging task in multi-vdd CMOS design is the integration into standard design flows and the implementation of the system without significant area and wiring overhead: If there are large distances between the gates of different voltage domains, interconnect capacity and therefore dynamic losses increase. Additionally a robust design concerning latch-up and ESD requires



**Fig. 9.** Shape and intensity statistics of the soft error pulses used to model soft error events within the different flip-flop circuits. The time constants  $\tau_1$  and  $\tau_2$  represent typical values for the considered technology.

sufficient large distances between the wells of different supply voltage domains. Single supply voltage level shifter cells enable minimum cell area consumption, because the whole circuit can be placed into one single well, as well as minimum place and route efforts. Moreover it is possible to implement the whole level shifter circuit within the regular standard cell pitch. In [3] a double height architecture is proposed. This may complicate the flow integration which can result in significant area overhead. Therefore the proposed semi-dynamic level shifter circuit is straight-forward in implementing multi-vdd schemes within industrial design flows.

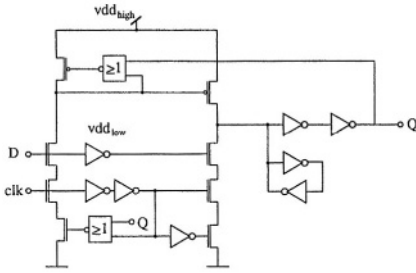
## 8 Soft Error Modeling and Analysis

Soft error rate, probably the most serious deep sub-micron effect for reliable systems, has been examined for the new flip-flop architectures. Therefore we used Monte Carlo analysis in which the current pulse shown in Fig. 9 has been injected into an arbitrary node of the circuit. During different phases of the clock cycle, the dynamic level converter flip-flops are differently susceptible to soft error events. Thus the time when the pulses occur is uniformly distributed over the clock period. According to Fig. 9 the pulse height is given by a gaussian distribution. The time constants describing the pulse as well as the variance of the pulse height represent typical values of the considered technology. A static standard flip-flop is used as reference. The following table shows the probability of bit errors in a 500 MHz System at temperature of  $100^\circ\text{C}$  under the test scenario described above:

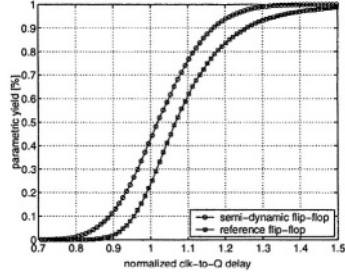
standard	full dynamic	with feedback
0.1975	0.2399	0.1898

These probabilities do not describe soft error rates directly because radiation flux and critical area are not considered here. However, the probabilities represent a good comparison of the soft error susceptibility of the different designs.

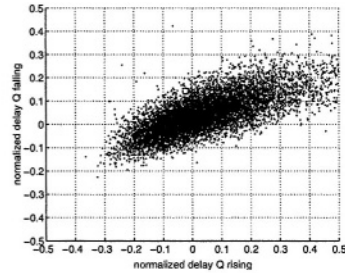
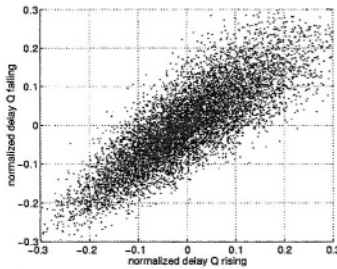




**Fig. 10.** Static self-precharge level converter flip-flop [1]. As the precharge device is turned-off only after the signal has propagated through the whole circuit, signal delay is increased.



**Fig. 11.** Parametric yield  $Y(d)$  of the semi-dynamic levelshifter flip-flop and the static precharge levelshifter flip-flop. The delay is normalized to the medium delay of the semi-dynamic flip-flop. For a given maximum delay the yield of the semi-dynamic circuit is higher.



**Fig. 12.** Variation of the propagation delays of the two flip-flop circuits under  $6\sigma$  device parameter variations. The delays are normalized to the medium delay of the semi-dynamic flip-flop. The medium delays of the semi-dynamic flip-flop are smaller as well as the deviations from this values than for the self precharge flip-flop.

It can be stated that there are no reliability drawbacks of the level converting flip-flops if the feedback protected version is used. The error probability of the full dynamic version increases if clock frequency decreases, because the LDN is more seriously affected by leakage currents and therefore more susceptible to charge injections.

## 9 Self-Precharging Architecture

During the previous sections the beneficial properties of the semi-dynamic level shifter flip-flops have been shown. In [1] a static precharge level-converting flip-flop (SPFF) has been proposed (cf. Fig 10). This section compares the two approaches: The most serious drawback of the SPFF in the single-ended version that is relevant here is the necessity of an inverted data signal (cf. sec 7).

In order to achieve a fair and realistic comparison of the two circuits we first optimized the circuit delay for a rising and falling transition. Then we used Monte-Carlo simulation to investigate these delays under a  $6\sigma$  variation of global (process) and local (mismatch) variation of transistor parameters. Fig. 12 shows the scattering plots of the clk-to-Q delays for the two circuits. It can be seen that the rising and falling delay of the semi-dynamic level converter flip-flop are well correlated. For the SPFF the correlation is weak because rising and falling signals propagate along different paths through the circuit. This results in reduced yield if mismatch becomes relevant. The medium delay of the semi-dynamic circuit is smaller compared to the SPFF which is a result of the dynamic level decision. In order to compare the delay performances of the two circuits under parameter variation we calculated the speed dependent yield  $Y$  defined by

$$Y(d) = \frac{\text{number of samples with } \max(d_{\text{rise}}, d_{\text{fall}}) < d}{\text{number of all samples}} \quad (4)$$

The resulting yield function is given in Fig. 11. As one can see the majority of samples of the semi-dynamic level converter are faster than the SPFF. This results in the higher yield for a given worst case delay. In the SPFF circuit the precharge transistor is active until the data signal has propagated through the circuit. Hence there is a competition between the three transistor stack discharging the precharged node  $x$  and the precharge device. This results in a larger delay and a smaller yield for a given worst case delay  $d$ .

## 10 Conclusion

Two novel dynamic level-converting master-slave flip-flops operating at one single supply voltage have been proposed. It has been shown that compared to standard solutions, these flip-flops have significantly lower delay and an input voltage independent power consumption. The soft error rate of the feedback protected version is not worse than that of standard flip-flops, because the circuit behaves static again immediately after the level conversion. Finally, the superior integrability into design flows and minimum area consumption have been outlined.

## References

1. Mahmoodi-Meimand, H., Roy, K.: Self-Precharging Flip-Flop (SPFF): A New Level Converting Flip-Flop. ESSCIRC (2002)
2. Augsburg, S., Nikolic, B.: Combining Dual-Supply, Dual-Threshold and Transistor Sizing for Power Reduction. ICCD (2002)
3. Ishihara, F., Sheikh, F., Nikolic, B.: Level Conversion for Dual-Supply Systems. ISLPED (2003)
4. Chandrakasan, A., Sheng, S., Broderon, R.: Low-Power CMOS Digital Design. JSSC, Vol. 27, No. 4 (1992)
5. Gonzales, R., Gordon, B., Horowitz, M.: Supply and Threshold Voltage Scaling for Low Power CMOS. JSSC, Vol. 32, No. 8 (1997)
6. Sakurai, T., Newton, A. R.: Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. JSSC, Vol. 25, No. 2, (1990) 584–594

# A Predictive Synchronizer for Periodic Clock Domains

Uri Frank and Ran Ginosar

VLSI Systems Research Center,  
Technion—Israel Institute of Technology, Haifa 32000, Israel  
ran@ee.technion.ac.il

**Abstract.** An adaptive predictive clock synchronizer is presented. The synchronizer takes advantage of the periodic nature of clocks in order to predict potential conflicts in advance, and to conditionally employ an input sampling delay to avoid such conflicts. The result is conflict-free synchronization with minimal latency. The adaptive predictive synchronizer adjusts automatically to a wide range of clock frequencies, regardless of whether the transmitter is faster or slower than the receiver. The synchronizer also avoids sampling duplicate data or missing any input.

## 1 Introduction

Large systems on chip (SoC) typically contain multiple clock domains. Inter-domain communications require data synchronization, which must avoid metastability while typically facilitating low latency, high bandwidth, and low power safe transfer. The synchronizer must also prevent missing any data or reading the same data more than once.

Communicating clock domains can be classified according to the relative phase and frequency of their respective clocks [1][7][14]. *Heterochronous* or *periodic* domains operate at nominally different frequencies, *plesiochronous* domains have very similar clock frequencies, *multi-synchronous* domains have the same clock frequency but a slowly drifting relative phase, and *mesochronous* domains have exactly the same frequency.

The simplest solution for inter-domain data transfer is the two-flip-flop synchronizer [1][12][13]. The main problem with that synchronizer is its long latency: Typically, a complete transfer incurs waiting about one to two clock cycles at each end. Although it is a very robust solution, it is sometimes misused or even abused in an attempt to reduce its latency [9].

Another commonly used synchronizer is based on dual-clock FIFO [1][8]. In certain situations, especially when a complete data packet of a pre-defined size must be transferred, this may be an optimal solution. Another advantage is that synchronization is safely contained inside the FIFO, relieving designers of the communicating domains of this delicate design task. The main drawback of FIFOs is their one-to-two cycle latency that is incurred when the FIFO is either full or empty, and that scenario is highly typical with periodic clock domains where the clock frequencies are different.

Mesochronous synchronizers are described in [1][3][4][14]. A rational clocking synchronizer for a special case of periodic domains in which the two clocks are related by the ratio of two small integers is described in [6]. Another synchronizer for a limited case of periodic domains is described in [5]. A plesiochronous synchronizer is proposed in [2]. It incorporates an exclusion detection controlling a multiplexer that selects either the data or a delayed version of it. While having a low latency and a “duplicate and miss” algorithm for the plesiochronous case, it is inapplicable to periodic domains. Another predictive synchronizer for plesiochronous domains is presented in [11]. It predicts the transmit clock behavior compared to the receive clock in advance. Using this data an “unsafe” signal is produced to control data latching.

Dally & Poulton [1] suggest a predictive synchronizer for *periodic* clock domains in which two versions of the data are latched and selected according to the output of a phase comparator that compares the two clocks. The circuit is non-adaptive, requiring advanced knowledge of the two frequencies, and does not handle missed or duplicate data samples.

We investigate *Adaptive Predictive Synchronizers* for low-latency bridging periodic domains where the two frequencies are unknown in advance at design time and may also change from time to time. Such synchronizers are also suitable for other types of domain relationships. In particular, predictive synchronizers are designed for high performance minimal latency transfer of data almost every cycle (of the slower clock). The synchronizers must prevent missing any data or sampling the same data more than once.

In Section 2 we describe the problem and introduce the circuits of the predictive synchronizer. Different components of the synchronizer are presented in Section 3, and miss and duplicate handling are discussed in Section 4. Section 5 analyzes the adaptation time of the synchronizer.

## 2 Synchronizer Overview

Consider high bandwidth data sent from a transmitter clock domain to a receiver domain. The data lines change state simultaneously with the rising clock of the transmitter domain, and the transmitter clock is sent to the receiver together with the data, serving as a “data valid” or “ready” signal (this is also termed “source synchronous” data transfer). Metastability may happen at the receiving end if the receiver (sampling) clock rises simultaneously with the transmitter clock.

The *Two-Way Adaptive Predictive Synchronizer* (Fig. 1. A Two-Way Predictive Synchronizer with Miss and Duplicate Protection) synchronizes bi-directional communications between two periodic clock domains (‘Left’ and ‘Right’). The synchronizer receives the two clocks, and manages safe data transfers both ways. It produces SEND and RECV control outputs to both domains, indicating when it is safe to receive and send new data on both sides, avoiding data misses and duplicates due to mismatched clock frequencies.

Fig. 2 shows the internal structure of the two-way predictive synchronizer. Since it is symmetric, in the following we will mostly consider only one half of it, the part that moves data from Left to Right. We adopt the term “Local Clock” for the receiver’s clock (the Right Clock in this case) and “External Clock” for the sender’s (Left) clock. In the following we describe the synchronizer circuits in a top-down manner.

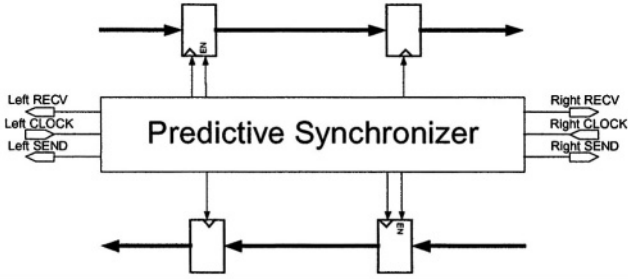


Fig. 1. A Two-Way Predictive Synchronizer with Miss and Duplicate Protection

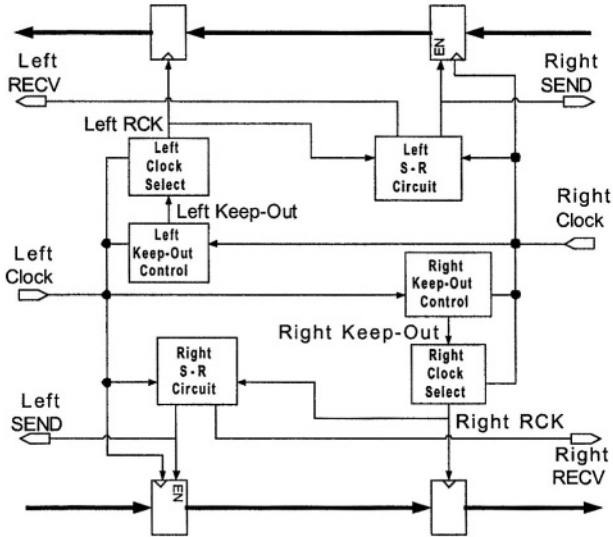


Fig. 2. Architecture of the two-way predictive synchronizer

One way of avoiding synchronization failures is to delay the sampling (local) clock when the input changes simultaneously with it. In order to decide if we need to delay, we track the rising edges of both the local and external clocks. If the rising edges occur “close” to one another (within a predetermined range  $d$ ) we delay the sampling clock by a predetermined amount  $T_{KO}$  to “keep out” of the danger zone. We can realize such a scheme by using the keep-out signal as a selector to a multiplexer on the clock (Fig. 3).

Thanks to the periodic nature of the clocks, we can reduce the number of cycles needed for synchronization by predicting the relative clock timing one cycle in advance. Let  $T_{LOCAL}$  and  $T_{EXT}$  be the clock periods of the receiver (local) and transmitter (external) clocks, respectively. Let’s assume that we have a conflict at time zero. In order to have a second conflict, an integral number of cycles of the local clock must span the same time as an integral number of cycles of the external clock, namely there should exist some  $K$  and  $N$  such that:

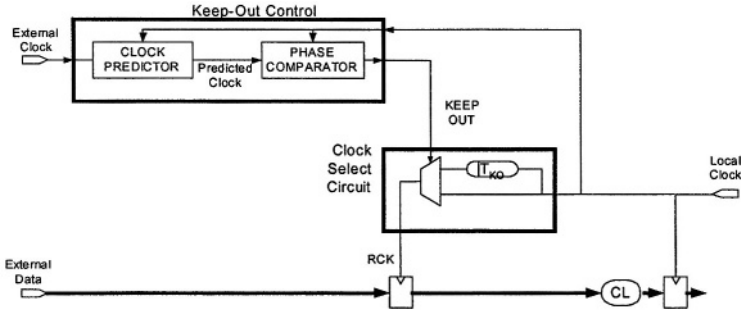


Fig. 3. The Keep-Out Control, sampling clock selector, and the receiver's sampling register

$$\begin{aligned}
 N \times T_{local} &= K \times T_{ext} \\
 (N - 1)T_{local} &= KT_{ext} - T_{local} = KT_{ext} - \frac{K}{N}T_{ext} = \Delta + \left( K - \left\lceil \frac{K}{N} \right\rceil \right) T_{ext} \\
 \Rightarrow \Delta &= \left\lceil \frac{K}{N} \right\rceil T_{ext} - T_{local}
 \end{aligned}$$

This means that if we delay the external clock signal by  $\Delta$  we will get a conflict after  $N-1$  cycles of  $T_{local}$  instead of  $N$ . The predictor (contained in the “Keep-Out Control”) is shown in Section 3. We could similarly predict the clock more than one cycle in advance, but the further in advance we try to predict, the larger the cumulative error (resulting from skew, drift, and jitter in the prediction circuits).

The predicted external clock is phase-compared to the local clock. The phase comparator detects a conflict when the rising edges of the two clocks happen at less than a certain delay  $d$  between them. The detection is used as a selector to the clock delay multiplexer (Fig. 3). Fig. 4 shows waveforms of a conflict and its resolution (when the sampling clock is delayed). Combinational logic can be placed right after the sampling register of the receiver, as can be seen in Fig. 3. The maximal allowed combinational delay is  $T_{TKO}$ , and thus the incurred synchronization latency is only  $TKO$ , a small portion of the cycle time.

### 3 Keep-Out Control

The Keep-Out Control (Fig. 3) consists of the clock predictor and the phase comparator. A conceptual clock predictor is shown in Fig. 5[1]. The “ $T_{LOCAL}$  delay” block is an adaptive delay line tuned to the cycle time of the local clock. The “programmable delay” block is another adaptive delay line. The feedback circuit adjusts it so that the two inputs to the conflict detector rise at approximately the same time. Once adjusted, the “predicted clock” output provides a copy of the external clock one local cycle time in advance.

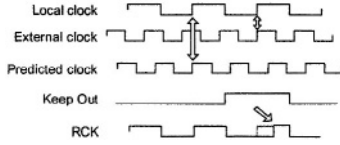


Fig. 4. A conflict is predicted one cycle in advance, delaying input sampling

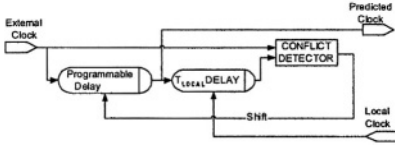


Fig. 5. Adaptive clock predictor

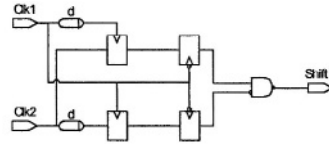


Fig. 6. Conflict Detector

The programmable delay line consists of a simple digital tapped inverter chain [4] [15]. The delay can be either increased or decreased one step each cycle. A four-flip-flop circuit is used to construct both a basic conflict detector (Fig. 6) and a two-way conflict detector (Fig. 8). These two conflict detectors allow about one half cycle time for any metastability in the first sampling stage to resolve. We show below (Section 3.1) that it is an extremely safe conflict detector.

The conflict detector (Fig. 6) detects simultaneous rising edges (within  $d$  delay of each other). The “ $T_{LOCAL}$  delay” block (Fig. 7) is a simplified digital DLL, consisting of a conflict detector and two programmable delays. This circuit starts with a minimal delay and increases the delay until it is equal to a full cycle. The flip-flop provides a loop delay (of two local clock cycles) until the lower programmable delay line has had time to adjust to a new value and the conflict detector has responded to that new value. Once the lower delay line has converged to  $T_{LOCAL}$ , its programming code is copied to the upper delay line.

This circuit, however, does not track  $T_{LOCAL}$  very well: If the local clock cycle decreases, the delay line would have to be reset and tuned from the beginning. Hence we employ a two-way conflict detector (Fig. 8), which operates bi-directionally to either decrease or increase the delay. It detects which clock rises first (“ $a < b$ ” in the figure means “ $a$  rises before  $b$ ” and “ $a \sim b$ ” means that they rise simultaneously). If the delayed clock rises before the non-delayed one, then the delay is increased, and vice versa.

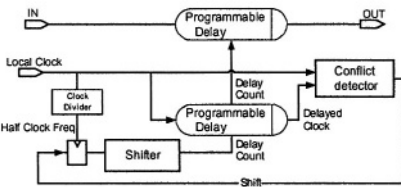


Fig. 7. The adjustable  $T_{LOCAL}$  Delay Line

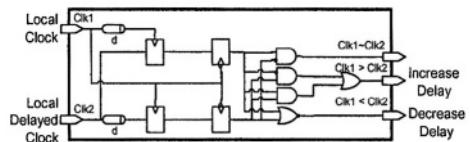


Fig. 8. Two-way conflict detector

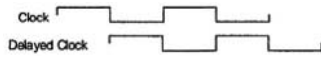


Fig. 9. 180° Phase shift

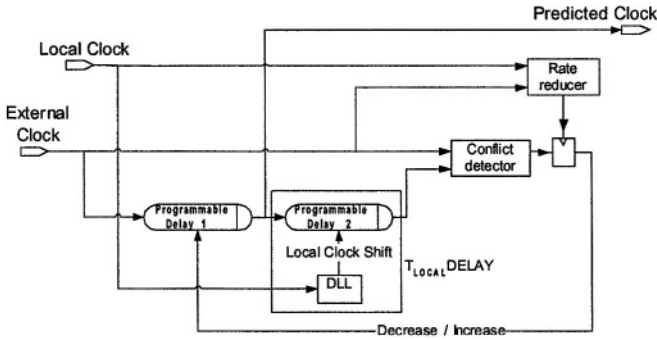


Fig. 10. Adaptive clock predictor

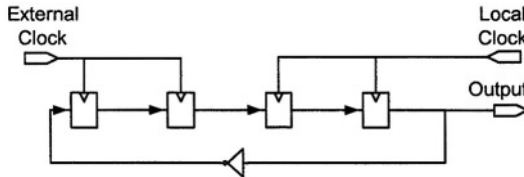


Fig. 11. Rate Reducer

To understand the (180°) phase shift signal in Fig. 8, consider the scenario in Fig. 9. When the two clocks have precisely inverted phases, the  $T_{LOCAL}$  block cannot decide whether to increase or decrease the delay. We resolve that ambiguity by deciding to always increase the delay in that case.

The  $T_{LOCAL}$  DLL (Fig. 7) converges to the cycle time of the local clock as follows. The minimum step by which the delay line can be increased or decreased is  $q$ . The DLL starts from a small delay and increases the delay by steps of size  $q$  up to the complete local cycle time, taking  $T_{LOCAL} / q$  steps. Each step takes two local clock cycles, and thus the total convergence time of the DLL is  $(T_{LOCAL} / q) \times 2 \times T_{LOCAL}$ .

We now examine the clock predictor circuit in more detail (Fig. 10). Programmable Delay 1 is started at zero delay, and the delay is increased progressively until the two inputs to the conflict detector concur. The loop delay in this case (the delay between successive steps of delay adjustment) must be the maximum of the two clock cycles. Since it is unknown in advance which clock is slower (the external or the local clock), the rate reducer (Fig. 11) waits for at least one cycle of each, synchronizing to each clock in turn by means of two flip-flops.

The delay introduced by the rate reducer between successive adjustments of Programmable Delay 1 (two passes around the circle in the rate reducer) is  $4T_{LOCAL} + 4T_{EXTERNAL}$  (in the worst case). Programmable Delay 1 must be tuned to a total



delay of  $\varepsilon = \lceil K/N \rceil \times T_{EXTERNAL} - T_{LOCAL}$  (as in Section 2). Since each time the delay is increased only by  $q$ , the number of steps required to tune Programmable Delay 1 is  $\frac{(\lceil K/N \rceil \times T_{EXTERNAL} - T_{LOCAL})}{q}$ . As the delay between successive steps is determined by the rate reducer, the total tuning time for Programmable Delay 1 is

$$\frac{(\lceil K/N \rceil \times T_{EXTERNAL} - T_{LOCAL})}{q} \times 4(T_{LOCAL} + T_{EXTERNAL})$$

The total adaptation time comprises the above expression plus the DLL convergence time,  $(T_{LOCAL} / q) \times 2 \times T_{LOCAL}$ . This adaptation time is further analyzed in Section 5.

### 3.1 Metastability of the Conflict Detector

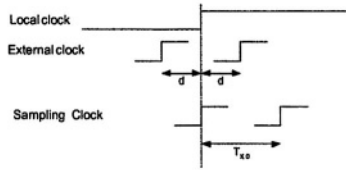
We note that, in a standard SoC (namely a digital IC based on standard cells and designed with standard EDA tools) the shortest clock cycle is typically about 160 FO4 inverter delays [16]. The nominal FO4 inverter delay depends mostly on the process technology (e.g. about 30ps for 0.13µm logic CMOS technology). Thus, the shortest high phase (with a 50% duty cycle clock) is about 80 inverter delays long. The lower bound is determined as follows. When the two clocks differ by about  $d$ , the conflict detector can enter metastability and take a long time to resolve. The MTBF of such a conflict detector can be determined as follows. Assume  $\tau$  is one inverter delay,  $W$  is two inverter delays, and  $F_b = F_c$  (conflict is possible every cycle when the two clocks are about equal frequency), then [12]:

$$MTBF = \frac{e^{T/2\tau}}{WF_c F_d} = \frac{e^{80}}{2 \times \frac{1}{160} \times \frac{1}{160}} \times \tau \approx 10^{39} \text{ inverter delays} \approx 10^{20} \text{ years}$$

That very safe MTBF (quoted in years) scales quite well over a number of process technologies. In fact, even if the time reserved for metastability resolution is halved to about 40 FO4 inverter delays, the MTBF would still safely exceed 10,000 years, an acceptable goal for most SoCs.

The conflict resolution delay  $d$  should be large enough to accommodate local jitter and delay variations inside the conflict detector. For instance,  $d=10$  FO4 inverter delays could be used. The “keep out” delay  $T_{ko}$  must be longer than  $d$  and shorter than one half cycle of the fastest possible clock. For example,  $T_{ko}=20$  inverter delays could be used.

Thus, while it is unlikely that any conflict detector has not resolved by the time it is used, it may have resolved to an unexpected value. Let us consider that situation for each of the conflict detectors. Note that in each half of the predictive synchronizer (Fig. 2) only the Keep-Out Control receives both clocks. Inside the Keep-Out Control, any one of the three conflict detectors may be affected. The effect of bad resolution of the conflict detectors in the  $T_{LOCAL}$  DLL and in the loop that adjusts Programmable Delay 1 (Fig. 10) is merely a potential extension of the convergence time. In the case of the phase comparator that generates Keep-Out (Fig. 3), note that it could become

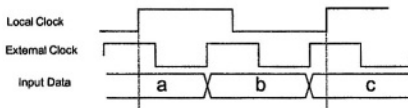


**Fig. 12.** Cases of incorrectly resolved Keep-Out: Input data is sampled either at local clock or after  $T_{KO}$  delay; in either case, it is safely separate from the rising edge of the external clock.

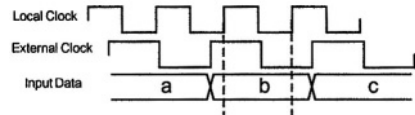
metastable when the local and external clocks are about  $d$  apart. However, a wrong value of Keep-Out is inconsequential in this case, and the data is sampled safely whether Keep-Out is asserted or not. The different cases are exemplified in Fig. 12.

## 4 Misses and Duplicates

When the transmitter clock is faster than the receiver's, the transmitter cannot send new data every cycle or else some data values will be missed by the receiver (Fig. 13). Conversely, when the receiver uses a faster clock, it cannot sample the input every cycle or else it will sample the same data more than once (Fig. 14).



**Fig. 13.** Miss Condition

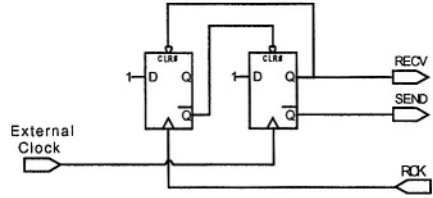
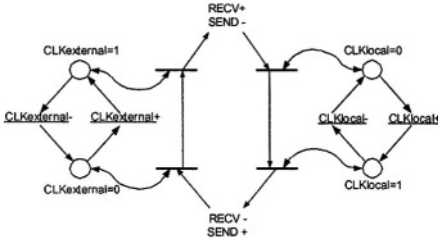


**Fig. 14.** Duplicate Condition

The complete two-way predictive synchronizer (Fig. 1) provides control signals to avoid missed and duplicate data. The SEND signal (generated on the receiver side) guarantees (when low) that a fast sender will keep its output unchanged until it is sampled, and the RECV signal, generated by the receiver, stops (when low) a fast receiver from using the same data more than once. Figure 15 describes the algorithm that generates SEND and RECV in terms of a Signal Transition Graph (STG). RECV is set upon a rising edge of the external clock (new data is available) and reset by a rising edge of the local clock (new data has been received, ready to receive the next one). SEND is simply the opposite of RECV. The STG is implemented, for instance, by the circuit of Fig. 16. Note that RCK is employed instead of the Local Clock; otherwise, that circuit would have been subject to metastability. RCK is now guaranteed to never coincide with the External Clock. Note also that this circuit cannot be synthesized directly from the STG by tools such as PetriPy [17].

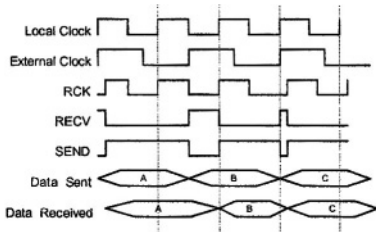
An example waveform of a fast receiver is given in Fig. 17 and a fast sender scenario is shown in Fig. 18.

Although metastability cannot occur in the S-R circuit, it can occur in the conflict detector leading to a wrong keep-out signal resulting in a wrong delaying of RCK. This can happen as shown before in Fig. 12 when the clocks are  $d$  time apart. The case when the external clock rises  $d$  time before the local clock is inconsequential because the relative ordering of the two clocks is unaffected (RCK succeeds the external clock).

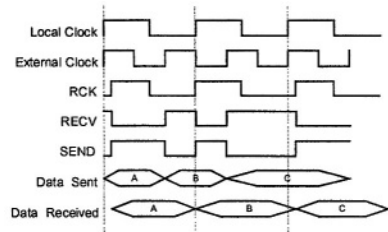


**Fig. 15.** SEND and RECV Control STG (double arrows are probe arcs: the transition happens if the place holds a token)

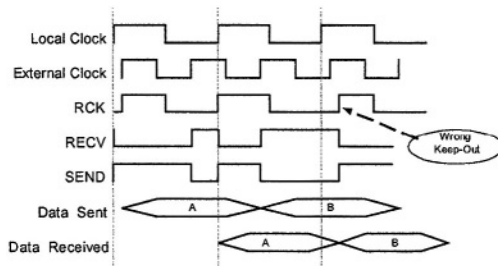
**Fig. 16.** Duplicate and Miss Control Circuit



**Fig. 17.** Fast Receiver Waveforms



**Fig. 18.** Fast Sender Waveforms



**Fig. 19.** Wrong Keep-Out on Fast Sender

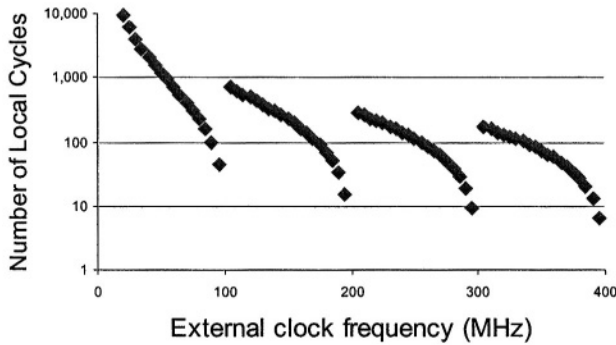
We now look at the case when the external clock rises after the local clock. In the case of a fast sender, RECV and SEND are extended (Fig. 19) and sampling the data by the receiver is delayed until the next cycle (Fig. 18). The worst case may happen when both clocks have the same frequency, and (when the external clock lags the local clock by  $d$  time) Keep-Out happens to oscillate every cycle—the resulting data transfer rate is effectively cut in half (but misses and duplicates are avoided).

## 5 Analysis

As shown in Section 3, the total adaptation time is

$$\frac{\left(\left\lceil \frac{K}{N} \right\rceil \times T_{\text{EXTERNAL}} - T_{\text{LOCAL}}\right)}{q} \times 4(T_{\text{LOCAL}} + T_{\text{EXTERNAL}}) + \frac{T_{\text{LOCAL}}}{q} \times 2 \times T_{\text{LOCAL}}$$

The chart in Fig. 20 shows the adaptation time, measured in cycles of a 100MHz local clock as a function of the external clock frequency, where the delay resolution  $q=100\text{ps}$ . Note that this is a very simplistic analysis; minor modification of the circuits could enable binary-search type of convergence, which would reduce total adaptation time by a logarithmic factor.



**Fig. 20.** Adaptation time of the predictive synchronizer (100MHz local clock, 100ps digital delay resolution)

## 6 Conclusions

A two-way adaptive predictive synchronizer for SoC with multiple clock domains has been presented. The synchronizer takes advantage of the periodic nature of clocks in order to predict potential conflicts in advance, and to conditionally employ an input sampling delay to avoid such conflicts. The result is conflict-free synchronization with almost zero latency (much less than one cycle). The adaptive predictive synchronizer adjusts automatically to a wide range of clock frequencies, regardless of whether the transmitter is faster or slower than the receiver. The synchronizer also avoids sampling duplicate data or missing any input. Adaptation to changing clock frequencies have been shown to require anywhere from tens of cycles to ten thousand cycles, depending on the relative frequencies.

## References

- [1] W.J. Dally and J.W. Poulton, "Digital Systems Engineering," Cambridge University Press, 1998.
- [2] L.R. Dennison, W.J. Dally and D. Xanthopoulos, "Low-latency plesiochronous data retiming," *Proc. 16th Conf. Adv. Res. in VLSI*, pp. 304-315, 1995.
- [3] Chakraborty, M.R. Greenstreet, "Efficient self-timed interfaces for crossing clock domains," *Proc. 9th IEEE Int. Symp. Asynchronous Circuits and Systems (ASYNC'03)*, pp. 78-88, 2003.
- [4] Y. Semiat and R.Ginosar, "Timing Measurements of Synchronization Circuits," *Proc. 9th IEEE Int. Symp. on Asynchronous Circuits and Systems (ASYNC'03)*, 2003.
- [5] J. Gandhi, "Apparatus for fast logic transfer of data across asynchronous clock domains" USA Patent 6,172,540, 2001.
- [6] L.F.G. Sarmenta, G.A. Pratt, S.A. Ward, "Rational Clocking," *Proc. ICCD*, pp.217-228, 1995.
- [7] D.G. Messerschmitt, "Synchronization in Digital System Design," *IEEE J. Selected Areas in Communication*, 8(8), 1990.
- [8] T. Chelcea and S.M. Nowick, "Robust Interfaces for Mixed-Timing Systems with Application to Latency-Insensitive Protocols," *Proc. ACM/IEEE Design Automation Conference*, 2001.
- [9] R. Ginosar, "Fourteen Ways to Fool Your Synchronizer," *Proc. 9th IEEE Int. Symp. on Asynchronous Circuits and Systems (ASYNC03)*, 2003.
- [10] Kessels, Peeters, Kim, "Bridging Clock Domains by synchronizing the mice in the mousetrap," *Proc. PATMOS*, 2003.
- [11] W.K. Stewart, S.Ward, "A solution to a special case of Synchronization Problem," *IEEE Trans. Comp.*,37(1), 1988.
- [12] C. Dike and E. Burton, "Miller and Noise Effects in a Synchronizing Flip-flop," *IEEE J. Solid-State Circuits*, 34(6), pp. 849-855, 1999.
- [13] D. J. Kinniment, A. Bystrov, and A. Yakovlev, "Synchronization Circuit Performance," *IEEE J. Solid-State Circuits*,37, pp. 202-209, 2002.
- [14] R. Ginosar and R. Kol, "Adaptive Synchronization," *Proc. ICCD*, 1998.
- [15] S.W. Moore, G.S. Taylor, P.A. Cunningham, R.D. Mullins, and P. Robinson, "Self-Calibrating Clocks for Globally Asynchronous Locally Synchronous Systems," *Proc. ICCD*, 2000.
- [16] International Technology Roadmap for Semiconductors (ITRS), 2001.
- [17] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems*, Vol. E80- D, No. 3, pp. 315-325, 1997.

# Power Supply Net for Adiabatic Circuits

Jürgen Fischer, Ettore Amirante, Agnese Bargagli-Stoffi, Philip Teichmann,  
Dominik Gruber, and Doris Schmitt-Landsiedel

Institute for Technical Electronics, Technical University Munich  
Theresienstrasse 90, D-80290 Munich, Germany  
juergen.fischer@tum.de  
<http://www.lte.ei.tum.de>

**Abstract.** Adiabatic circuits are introduced for ultra-low-power applications. This work presents the impact of the power net on the ideal trapezoidal power supply waveform. An oscillator consisting of four 90-degree shifter is simulated in a  $0.13\mu\text{m}$  CMOS technology providing a conversion efficiency of 85%. The trapezoidal waveform is distorted after a long line because harmonics are inhibited and the signal is delayed. A disturbed phase relationship does not affect energy dissipation and a phase margin larger than 30 degrees is provided. For high frequencies the impact of phase delay on the energy dissipation can be neglected.

## 1 Introduction

The technology progress accompanied by the system on a chip (SoC) approach leads to an increasing number of transistors per chip. As a consequence, the dynamic power dissipation increases even in modern technologies with reduced supply voltage. For battery-operated or mobile systems, this increase represents a great problem because the capacity of batteries does not increase in the same way. Additionally, the leakage losses due to the low threshold voltages will become a major concern especially for mobile systems. The dynamic energy dissipation per switching event in static CMOS exhibits a fundamental limit of  $\frac{1}{2}CV_{DD}^2$ . Adiabatic circuits are able to break this limit due to their energy efficient switching operation. Therefore, they use a trapezoidal power supply.

In this paper, the Efficient Charge Recovery Logic (ECRL) [1,2] and the Positive Feedback Adiabatic Logic (PFAL) [3,4] are considered. Both families provide a large energy saving factor over static CMOS at operating frequencies of about 100MHz in a  $0.13\mu\text{m}$  CMOS technology. These frequencies are suitable for digital signal processing. Most evaluations of adiabatic circuits were performed neglecting the clocked power supply line. Using the transmission line theory, the attenuation and phase delay of the trapezoidal power supply are characterized leading to a maximum line length. Additionally, the parasitics of the line make a contribution to the energy dissipation. The considered adiabatic logic families require a four phase power supply. For a whole adiabatic circuit, it is important to know how the phase delay influences the functionality and energy dissipation. To the best of our knowledge no results were presented about this matter. Beside

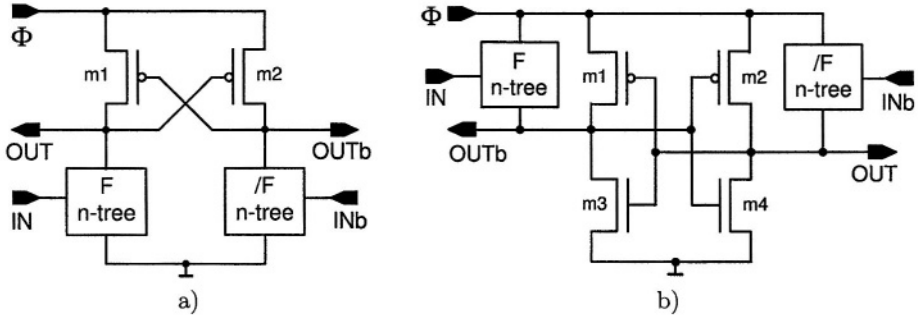


Fig. 1. General schematic of a) ECRL and b) PFAL gates.

the parasitics and their direct impact on the energy dissipation, also the impact of phase delay will be characterized.

## 2 Adiabatic Logic Families

The most promising adiabatic logic families are the Efficient Charge Recovery Logic (ECRL) and the Positive Feedback Adiabatic Logic (PFAL). In figure 1 the general schematics are shown. Both families are dual rail encoded and utilize only n-channel transistors to evaluate the logic function  $F$  as well as the complementary function  $/F$ . Whereas in ECRL two cross-coupled p-channel transistors act as memory unit, in PFAL two cross-coupled inverters are used.

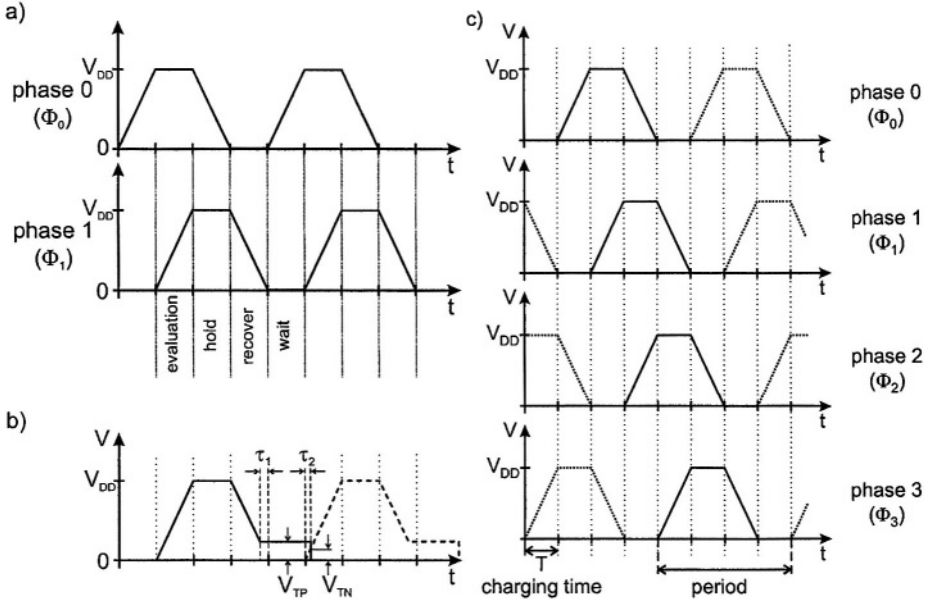
The clocked power supply ideally uses a trapezoidal waveform as shown in figure 2 a). During the evaluation phase the output which represents a logic value of 1 is charged. In the hold phase, the output signals are held at a constant voltage, thus the succeeding stages gain valid input signals. When the output signals  $OUT$  and  $Outb$  are evaluated by the succeeding stages the charge is recovered. This recovery process is performed as long as the p-channel transistor is conducting. As shown in figure 2 b) the output signal remains at a constant value when the power supply falls below  $|V_{TP}|$ . The recovery process is stopped  $\tau_1$  before the phase ends, with

$$\tau_1 = \frac{|V_{TP}|}{V_{DD}} \cdot T_{charge} \tag{1}$$

The generator can be designed in an easier way if a symmetric waveform is provided. Therefore a wait phase is inserted as fourth phase. As the input signals are a quarter period in advance with respect to the supply voltage, the whole adiabatic circuit is supplied by a four phase power clock system, as shown in figure 2 c).

In [5] three main sources of energy dissipation in adiabatic logic circuits are determined. At high frequencies, the adiabatic losses are dominant

$$E_{adiab} = \frac{RC}{T_{charge}} CV_{DD}^2, \tag{2}$$



**Fig. 2.** a) Trapezoidal power supply  $\Phi_1$  with the four phases evaluate, hold, recover and wait. The input signal  $\Phi_0$  is a quarter period in advance. b) Alternating output signals with incomplete recover process. c) Four phase power supply for a whole adiabatic system.

where  $R$  is the resistance of the charging path,  $C$  the load capacitance,  $V_{DD}$  the peak of the clocked supply voltage and  $T_{charge}$  the charging time. When the voltage goes below the threshold voltage  $V_{TP}$  of the p-channel device, the transistor cuts off and the recovery process is not carried out completely. The output node voltage remains at the threshold voltage. Toggling the output values this residual charge is discharged instantaneously comparable to the switching in static CMOS. This source is observable at medium frequencies:

$$E_{V_{TP}} = \frac{1}{2} C V_{TP}^2 \quad (3)$$

The residual charge is discharged when the new input signals switch the logic function block on. This happens when the power supply is above the threshold voltage of the n-channel devices  $V_{TN}$  at the time

$$\tau_2 = \frac{V_{TN}}{V_{DD}} \cdot T_{charge} \quad (4)$$

At low frequencies, the dissipation due to leakage mechanisms must be taken into account. The largest off-current  $I_{off}$  flows during the hold time  $T_{hold}$  when the maximum voltage  $V_{DD}$  is applied to the gate:

$$E_{leak} \propto V_{DD} I_{off} T_{hold} \quad (5)$$



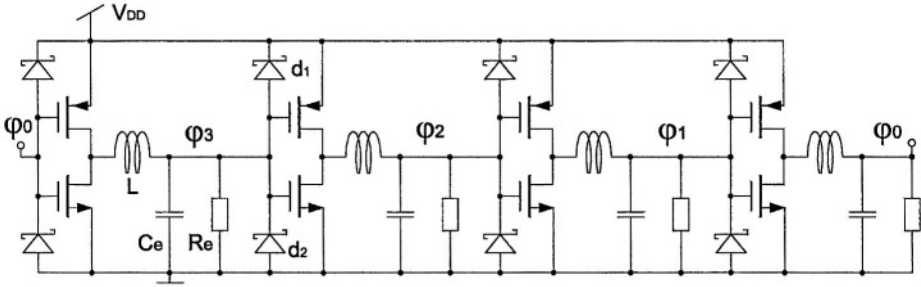


Fig. 3. Schematic of the 4-phase shifter oscillator

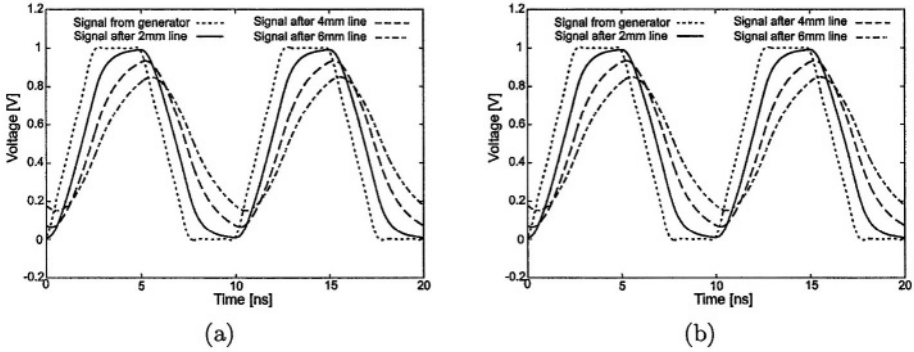
### 3 Power Clock Generator

A high efficiency low-power four-phase power clock generator is proposed in [6]. This generator is realized as a ring of four low-power 90-degree shifters (figure 3). Each shifter consists of a CMOS inverter and a LC resonant circuit. The energy oscillates between the reactive elements. The DC power supply provides only the energy which is dissipated on the resistance and diodes. Each stage of the adiabatic circuit is represented by its equivalent load consisting of a resistance  $R_e$  and a capacitance  $C_e$ . Schottky diodes are used for output amplitude regulation because their low  $V_\gamma$  allows to achieve a moderate energy dissipation. The inductors are external to the chip because of the higher quality factor achievable. We simulated this power clock generator in a  $0.13\mu\text{m}$  CMOS technology at a frequency of 20 MHz using the design methodology proposed in [6]. The conversion efficiency amounts to 85%. With a 30% variation of the value of a single inductance the phase variation amounts to less than 10 degrees. For the further investigation, a 10 degrees shift is taken into account as maximum deviation resulting from an unbalanced generator.

### 4 Characterization of the Clocked Power Line

In real adiabatic circuits, the influences of interconnect lines, in particular the clocked power lines have to be considered. Compared to static CMOS, the current in the ground line does not give rise to a large contribution to the energy dissipation. When a capacitance is charged during the evaluation phase, current is flowing also in the ground node. This current does not need to be derived from the external ground, because somewhere nearby another capacitance is discharged during the recover phase providing the necessary current. As a result, the charge is only transported over a short line of the ground grid on the chip and the impact of the ground line can be neglected.

In this paper, a clocked power line is assumed with a resistance load per unit length  $R' = 75\text{k}\Omega\text{m}^{-1}$ , a capacitive load per unit length  $C' = 400\text{pFm}^{-1}$  and an inductive load per unit length  $L' = 1.4\mu\text{Hm}^{-1}$ . These are typical values



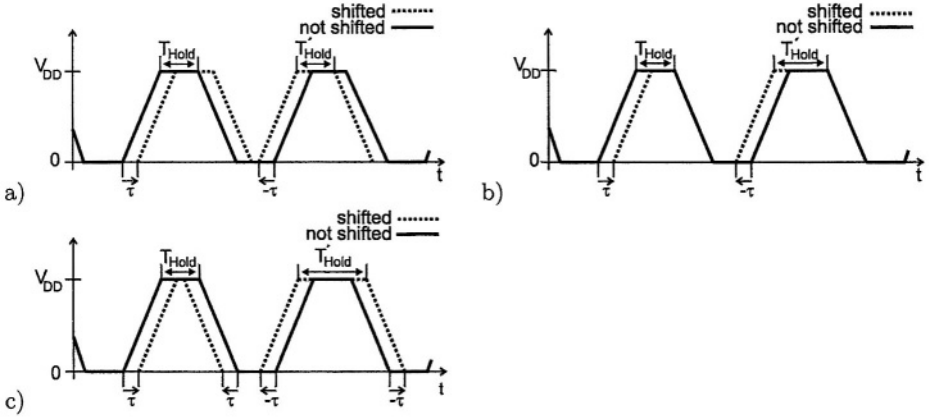
**Fig. 4.** a) Trapezoidal signal from the generator and the distorted signals after different line lengths (without load). b) Energy over time of 256 1bit-PFAL-full-adder without the power line, with moderate line length (1mm) and with a long line (4mm).

of a  $0.13\mu\text{m}$  process: Simulating an unloaded  $0.8\mu\text{m}$  wide line with different lengths, the trapezoidal waveform can still be observed with a 2mm long line. Longer lines inhibit the harmonics and the fundamental frequency remains with a definite delay and a definite attenuation (figure 4). Thus a length of 4mm is chosen as limit in this simulation. For a 2mm long line a phase delay of about 45 degrees is observed. Simulating a loaded line, 256 1bit-PFAL-full-adder are used, whose output nodes are terminated with a capacitive load of 1fF. They are divided in 4 blocks, which are distributed with constant distance over the  $0.8\mu\text{m}$  wide clocked power line. In figure 4 b) the increase of energy dissipation with longer lines is shown. Without considering the clocked power line, the adder field dissipates 2.4pJ per cycle. If the 4 blocks are distributed over a 1mm long line, the energy dissipation per cycle increases by 41% to 3.2pJ. The simulation with a length of 4mm exhibits an increase by a factor 3.3 to 8.0pJ.

The clocked power line network of adiabatic circuits has to be compared to the global clock distribution network of static CMOS. This network dissipates 10-50% of the total chip power in static CMOS. In [7] a distributed oscillator is proposed which reduces this part of the energy dissipation. The distributed oscillator consists of clock sectors with area up to  $2.5\text{mm} \times 2.5\text{mm}$ . This approach can be adopted for the power supply net in adiabatic logic system, as the used line length is in the scope. On the one hand, using sinusoidal waveforms as power supply slightly increases the energy dissipation in the logic. On the other hand, the energy conversion factor of the power clock generator advances. Altogether, the energy dissipation of the power supply net in adiabatic circuits can be decreased in the same way as the dissipation of the global clock network in static CMOS by using a distributed resonant grid.

## 5 Impact of the Phase Delay

Until now only the impact of the additional parasitics is taken into account. In real circuits the right phase correlation can not be guaranteed. Therefore, we

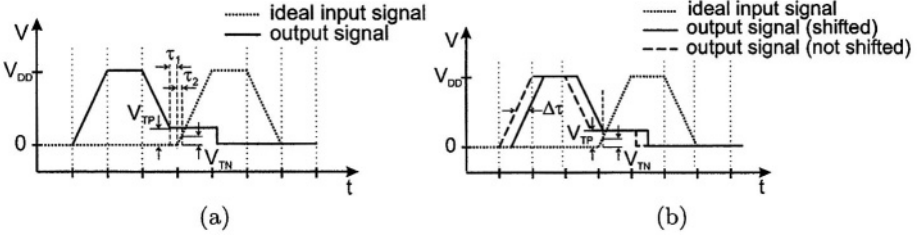


**Fig. 5.** Considered variations of the phase correlation: a) shift of a whole phase b) delay of the evaluation phase c) variation of the hold phase.

considered in our simulations three possible variations of the four phase power clock system deriving from an unbalanced 4-phase generator (section 3) or from the properties of the power line. First, a whole phase is shifted with respect to the adjacent phases (figure 5 a). Second, the delay of the evaluation phase is changed (figure 5 b). This occurs for example if a large load is charged and the loading current results in a large voltage drop over the charging resistance. Third, the hold time is varied (figure 5 c). A variation of the charging time  $T_{charge}$  is not performed because the charging time is related to the frequency. This variation can be derived from a superposition of a frequency variation and a variation of the hold time. As the charging time is held constant, the adiabatic loss  $E_{adiab}$  (eqn. 2) and the frequency independent dynamic losses  $E_{V_{TP}}$  (eqn. 3) also remain constant. Therefore, the impact of phase delay on energy dissipation is expected to be lower at high frequencies. At low frequencies, the leakage losses are important. They are associated with the hold time  $T_{Hold}$ . Two variations of the phase correlation alter the hold time (figure 5 b and c). After a delay of the evaluation phase the hold time is changed by  $\tau$  or the hold time is directly varied by  $2\tau$ . The new reduced hold time  $T'_{Hold}$  will lead to a decreased energy dissipation.

In case a) the maximum tolerable delay  $\tau_{max}$  is reached when the function blocks are turned on by the input signals while the output charge is still recovered. For example when an ECRL gate (figure 1) recovers the charge from the output node OUT, the function block F must not be turned on, otherwise a short circuit current will flow. As mentioned in section 2 the recover process is stopped at a time  $\tau_2$  before the wait phase starts and the function blocks are switched on at a time  $\tau_1$  after the wait phase starts (figure 6a). Therefore the maximum delay  $\tau_{max}$  is the sum of both times (figure 6b)

$$\tau_{max} = \tau_1 + \tau_2 = \frac{|V_{TP}| + V_{TN}}{V_{DD}} \cdot T_{charge} \tag{6}$$



**Fig. 6.** Toggling input and output signals a) without delay and b) with maximum tolerable delay.

Normalizing the delay  $\tau$  and  $\tau_{max}$  to the charging time  $T_{charge}$ , the delay can be expressed in degrees:

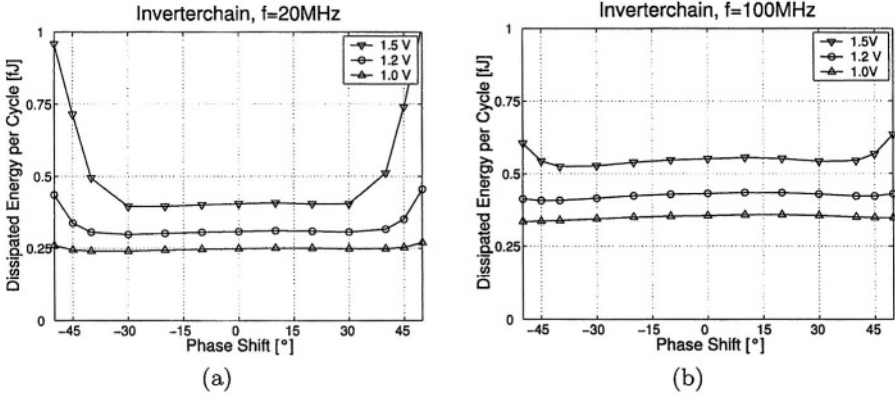
$$\alpha = \frac{\tau}{T_{charge}} \cdot 90^\circ$$

$$\alpha_{max} = \frac{|V_{TP}| + V_{TN}}{V_{DD}} \cdot 90^\circ$$

In the considered  $0.13\mu\text{m}$  CMOS technology the threshold voltages of the p- and the n-channel device amount to  $V_{TP} = -300\text{mV}$  resp.  $V_{TN} = +260\text{mV}$ . This leads to a maximum delay phase  $\alpha_{max} = 42^\circ$  for  $V_{DD} = 1.5\text{V}$ . Decreasing the maximum supply voltage  $V_{DD}$  will increase the maximum delay phase.

## 6 Simulation Results

To prove this theoretical approach simulations are performed with ECRL and PFAL inverter chains in a  $0.13\mu\text{m}$  CMOS technology. All variations were applied to one inverter stage. Figure 7 shows the simulation result delaying one whole phase in an ECRL inverter chain for the frequencies 20MHz and 100MHz. These two figures are typical for all variations and confirm the analysis performed in the previous sections. As the charging time  $T_{charge}$  is not varied the adiabatic loss remains constant. A shift of a whole phase does not alter the hold time and no significant change of the leakage current occurs until the calculated maximum delay of  $\alpha_{max} = 42^\circ$ . Therefore, only slight deviations of the energy dissipation can be observed. Exceeding this point the large short circuit current is flowing and the energy dissipation rises. Shifting a whole phase means that on the one side two phases will get closer together whereas on the other side the distance between two phases will be increased. Therefore the deviation does not depend on the algebraic sign of the phase shift. If the peak of the supply voltage  $V_{DD}$  is lowered to 1.2V and 1.0V, two effects appear. According to eqn. 2 scaling down the supply voltage will decrease the regular energy dissipation as it is shown in figure 7. This effect was already proposed in [8]. Beyond this the maximum delay is increased by decreasing  $V_{DD}$ . At 100MHz the impact on the energy dissipation is lower because the adiabatic loss gets more dominant, whereas the variations of the phase delay mainly affect the leakage losses.



**Fig. 7.** Energy dissipation of an ECRL inverter chain shifting one whole phase. Also the frequency and maximum supply voltage is varied.

**Table 1.** Relative deviation of the energy dissipated in PFAL and ECRL inverter chains applying several kind of phase delays with  $V_{DD} = 1.5V$ .

Relative deviation of energy dissipation [%]		PFAL		ECRL		
		frequency	phase shift -10°/+10°	phase shift -30°/+30°	phase shift -10°/+10°	phase shift -30°/+30°
delay of whole phase	1MHz		-0.8/+1.0	-2.3/+23.2	+2.1/+0.6	+20.4/+20.4
	100MHz		-0.6/-0.3	+3.9/+12.1	-0.7/+0.7	-4.4/-1.6
delay of evaluation phase	1MHz		+1.4/-1.4	+4.3/-4.3	+4.1/+4.0	+21.6/+10.6
	100MHz		+1.2/-1.1	+4.2/-3.1	+0.9/-0.8	+2.5/-2.0
variation of hold phase	1MHz		+3.9/+3.7	+31.8/-10.9	+8.7/-6.0	+52.5/-11.8
	100MHz		+2.1/+1.7	+19.6/-3.3	+2.4/-2.4	+3.4/-8.6

In table 1 all relative deviations of the energy dissipation due to the considered phase delays are summarized for the frequencies 1MHz and 100MHz applying a supply voltage  $V_{DD}=1.5V$ . A phase shift of 10 degrees represents the maximum deviation resulting from an unbalanced generator. The second phase shift of 30 degrees takes into account different line lengths of two communicating stages. Also in this case, the value is held below the maximum tolerable phase delay. First, we regard low frequencies where the impact of leakage currents can be observed. Shifting the whole phase at 1MHz maximum deviation amounts to about 20% for both families. A delay of the evaluation phase mainly affect ECRL, which results in maximum relative deviation of +21.6%. Increasing the hold time, which is represented by a negative shift, leads to an increased leakage current. At -30 degrees the energy dissipation has raised by +52.5% (ECRL). On the other hand a positive shift leads to reduced hold decreasing the energy

dissipation (-11.8%). At high frequencies (e.g. 100MHz) the impact of the leakage currents is smaller as it can be observed in table 1. Except of the large variations in the PFAL inverter chain, all variation of the phase delay show only slight deviations of the energy dissipation.

As a result, in the design long distances between communicating cells have to be avoided. Then the impact of phase delay on the energy dissipation of adiabatic circuits can be neglected for the frequencies of interest.

## 7 Conclusions

This work dealt with the impact of supplying adiabatic circuits with trapezoidal waveforms. Therefore a four phase power clock generator was proposed showing a conversion efficiency of 85% in the considered  $0.13\mu\text{m}$  CMOS technology. Simulations were performed with 256 1bit-PFAL-full-adders without and with considering the clocked power line. Advantageous for adiabatic circuits is the fact that the current in the ground line does not give rise to a large contribution to the energy dissipation, because the current in the ground line is not derived from the external ground but from adjacent stages which are in the complementary phase. The power dissipation of the global clock network in static CMOS amounts to 10-50% of the total chip power. The simulations for the power supply net of adiabatic circuits showed a comparable increase of the energy dissipation. The power line length should be kept reasonably short, which could be achieved by using a resonant distributed clocked power network. Beside the impact of the clocked power line on the energy dissipation, also the impact of phase delay in adiabatic circuits was characterized. Three possible variations were considered. First, a whole phase was shifted. Second, the evaluation phase was delayed and last the hold time was varied. Considering the different phase delay, noticeable deviations of the energy dissipation were mainly observed at 1MHz. At the frequencies over 100MHz which are interesting for digital signal processing only slight deviations were found. All of the three variations exhibited the same maximum tolerable phase delay  $\alpha_{max}$ . Exceeding this maximum, the clocked power supply and ground are short circuited because the logic blocks are turned on while the recovery process was still in progress.  $\alpha_{max}$  amounts to  $42^\circ$  for  $V_{DD} = 1.5V$ . Decreasing the supply voltage  $V_{DD}$  the maximum phase delay was increased and the regular energy dissipation was scaled down.

Scaling down the supply voltage enables less energy dissipation accompanied by a more robust behavior against mismatch of the phase correlation. The impact of phase delay on the energy dissipation of adiabatic circuits can be neglected for the frequencies of interest.

**Acknowledgements.** This work is supported by the German Research Foundation (DFG) under the grant SCHM 1478/1-3.

## References

1. Kramer, A., Denker, J. S., Flower, B., Moroney, J.: 2nd order adiabatic computation with 2N-2P and 2N-2N2P logic circuits. Proceedings of the International Symposium on Low Power Design, pp. 191–196, 1995.
2. Moon, Y., Jeong, D.: An Efficient Charge Recovery Logic Circuit. IEEE Journal of Solid-State Circuits, Vol. 31, No. 4, pp. 514–522, 1996.
3. Vetuli, A., Di Pascoli, S., Reyneri, L. M.: Positive feedback in adiabatic logic. Electronics Letters, Vol. 32, No. 20, pp. 1867–1869, 1996.
4. Blotti, A., Di Pascoli, S., Saletti, R.: Simple model for positive-feedback adiabatic logic power consumption estimation. Electronics Letters, Vol. 36, No. 2, 116–118, 2000.
5. Fischer, J., Amirante, E., Randazzo, F., Iannaccone, G., Schmitt-Landsiedel, D.: Reduction of the Energy Consumption in Adiabatic Gates by Optimal Transistor Sizing. Proceedings of the 13th International Workshop on Power And Timing Modeling, Optimization and Simulation, PATMOS'03, Turin, Italy, September 2003, pp. 309-318.
6. Bargagli-Stoffi, A., Iannaccone, G., Di Pascoli, S., Amirante, E., Schmitt-Landsiedel, D.: Four-phase power clock generator for adiabatic logic circuits. Electronics Letters, Vol. 38, No. 14, pp. 689–690, 2002.
7. Chan, S. C., Restle, P. J., Shepard, K. L., James, N. K., Franch, R. L.: A 4.6GHz Resonant Global Clock Distribution Network. Digest of Technical Papers, IEEE International Solid-State Circuits Conference, pp. 342-343, February 2004.
8. E. Amirante, A. Bargagli-Stoffi, Fischer, J., Iannaccone, G., Schmitt-Landsiedel, D.: Adiabatic 4-bit Adders: Comparison of Performance and Robustness against Technology Parameter Variations. Proceedings of the 45th IEEE International Midwest Symposium on Circuits and Systems, MWSCAS'02, Tulsa, OK, USA, Vol. III, pp. 644-647, August 2002.

# A Novel Layout Approach Using Dual Supply Voltage Technique on Body-Tied PD-SOI

Kazuki Fukuoka<sup>1</sup>, Masaaki Iijima<sup>1</sup>, Kenji Hamada<sup>1</sup>,  
Masahiro Numa<sup>1</sup>, and Akira Tada<sup>2</sup>

<sup>1</sup> Faculty of Engineering, Kobe University  
1-1 Rokko-dai, Nada-ku, Kobe 657-8501, Japan,  
numa@kobe-u.ac.jp

<sup>2</sup> LSI Product Technology Unit, Renesas Technology Corp.  
4-1 Mizuhara, Itami, Hyogo 664-0005, Japan

**Abstract.** This paper presents a novel layout approach using dual supply voltage technique. In Placing and Routing (P&R) phase, conventional approaches for dual supply voltages need to separate low supply voltage cells from high voltage ones. Consequently, its layout results tend to be complex compared with single supply voltage layout results. Our layout approach uses cells having two supply voltage rails. Making these cells is difficult in bulk due to increase in area by n-well isolation or in delay by negative body bias caused by sharing n-well. On the other hand, making cells with two supply voltage rails is easy in body-tied PD-SOI owing to separation of transistor bodies by trench isolation. Since our approach for dual supply voltages offers freedom for placement as much as conventional ones for single supply voltage, existing P&R tools can be used without special operation. Simulation results with MCNC circuits and adders have shown that our approach reduces power by 19 % and 25 %, respectively, showing almost the same delay with single supply voltage layout.

## 1 Introduction

Power consumption of LSI has been increasing due to increase in the number of transistors. Low power techniques are important, particularly for mobile applications.

The total power consumption  $P$  in CMOS LSI is given by

$$P = P_{\text{dyn}} + P_{\text{sc}} + P_{\text{leak}} \quad (1)$$

where  $P_{\text{dyn}}$  is the dynamic power,  $P_{\text{sc}}$  is the short circuit power, and  $P_{\text{leak}}$  is the leakage power. Among them,  $P_{\text{dyn}}$  occupies a large part of  $P$  in CMOS LSI.

The dynamic power  $P_{\text{dyn}}$  is given by

$$P_{\text{dyn}} = p_t f C_L V_{\text{DD}}^2 \quad (2)$$

where  $p_t$  is the switching probability,  $f$  is the clock frequency,  $C_L$  is the load capacitance, and  $V_{\text{DD}}$  is the supply voltage. As shown in Eq. (2), lowering  $V_{\text{DD}}$  is effective to reduce power consumption.



On the other hand, lowering  $V_{DD}$  causes increase in propagation delay  $t_{pd}$ , which is approximately given in [1] by

$$t_{pd} = \frac{kC_L V_{DD}}{(V_{DD} - V_{th})^\alpha} \quad (3)$$

where  $\alpha \cong 1.3$ ,  $k$  is a constant value, and  $V_{th}$  is the threshold voltage. In order to avoid increase in  $t_{pd}$  by lowering  $V_{DD}$ ,  $V_{th}$  should be lowered together. However, lowering  $V_{th}$  leads to increase in  $P_{leak}$ . Hence, reducing power is difficult to achieve only by lowering  $V_{DD}$  in recent years.

The dual supply voltage technique [2] is one of the techniques for reducing  $P_{dyn}$ . Its basic idea is that normal supply voltage ( $V_{DDH}$ ) is provided for gates on critical paths and low supply voltage ( $V_{DDL}$ ) is provided for those on non-critical paths. Although  $t_{pd}$  of non-critical paths increases by using  $V_{DDL}$ ,  $t_{pd}$  of the whole circuit is kept unchanged if  $t_{pd}$  of the non-critical paths using  $V_{DDL}$  is shorter than that of the critical paths. This technique does not need to lower  $V_{th}$ , which is effective to avoid increase in  $P_{leak}$ .

However, most of conventional dual supply voltage techniques place cells using  $V_{DDL}$  ( $V_{DDL}$  cells) in  $V_{DDL}$  rows, and those using  $V_{DDH}$  ( $V_{DDH}$  cells) in  $V_{DDH}$  rows. Thus, freedom for placement is suppressed due to this constraint. Furthermore, wire lengths tend to be long by separating  $V_{DDL}$  cells from  $V_{DDH}$  ones.

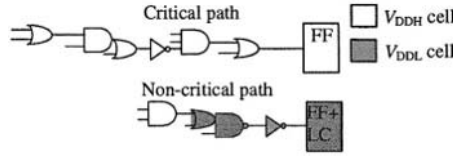
We propose a novel layout approach using dual supply voltage technique without losing freedom for placement. Our technique is based on cells having two supply voltage rails. Making these cells is difficult in bulk due to increase in area by n-well isolation or in delay by negative body bias which is caused by sharing n-well between different supply voltages. We make it possible by employing body-tied PD-SOI.

This paper is organized as follows. In Section 2, we discuss conventional dual supply voltage techniques and issues with them. In Section 3, we propose a novel layout approach using dual supply voltage technique. Section 4 presents simulation results using SPICE. Finally, Section 5 gives conclusion.

## 2 Background

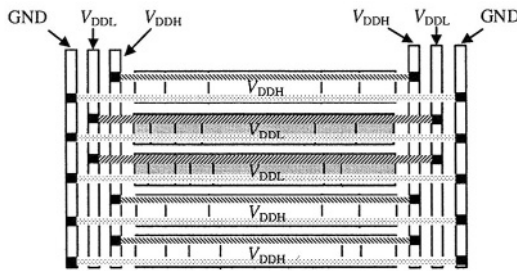
In this section, we discuss conventional dual supply voltage techniques in detail.

As mentioned in Section 1, most of dual supply voltage techniques employ  $V_{DDH}$  cells for critical paths and  $V_{DDL}$  cells for non-critical paths. An output of  $V_{DDH}$  cell can be connected to an input of  $V_{DDL}$  cell, however, an output of  $V_{DDL}$  cell should not be connected to an input of  $V_{DDH}$  directly. A voltage level conversion (LC) cell is inserted between an output of  $V_{DDL}$  cell and an input of  $V_{DDH}$  one. Since insertion of LC cells leads to performance degradation, it is required to minimize the number of LC cells. For this purpose, level conversion is limited only once on a path from flip-flop (FF) to FF by using FF with LC cells as shown in Fig. 1.



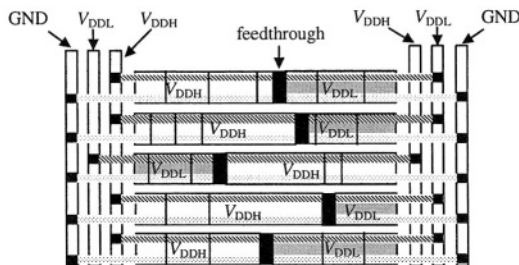
**Fig. 1.** A typical circuit using dual supply voltage technique.

For using dual supply voltage technique, designers need to separate cells in  $V_{DDH}$  and  $V_{DDL}$  rows. Normal standard cells have a supply voltage rail and a ground rail. As shown in Fig. 2, cells using the same supply voltage, either  $V_{DDH}$  or  $V_{DDL}$ , can be placed in a row. In this layout approach, wire lengths tend to be long between  $V_{DDH}$  cells and  $V_{DDL}$  ones, and freedom for placement is constrained in comparison with layout for single supply voltage.



**Fig. 2.** Layout using conventional technique.

Figure 3 shows an improved layout approach [3]. In this approach,  $V_{DDH}$  and  $V_{DDL}$  cells can be placed in each row. However, it is needed to partition a row into  $V_{DDH}$  part and  $V_{DDL}$  one, and freedom for placement is constrained in comparison with layout for single supply voltage.



**Fig. 3.** Layout using improved technique [3].

Furthermore, conventional layout approaches using dual supply voltages have another problem in placing LC cells. Figure 4 shows an FF with LC cell used in [4], which has two supply voltage lines. However each cell is placed in either  $V_{DDH}$  row or  $V_{DDL}$  low. In conventional approach, an FF with LC cell is placed in  $V_{DDH}$  row, and  $V_{DDL}$  is provided from  $V_{DDL}$  row by a narrow signal line. In this approach, there is a possibility of IR drop due to high resistance compared with a thick supply voltage rail. In particular, it will appear as an important problem in deep submicron era.

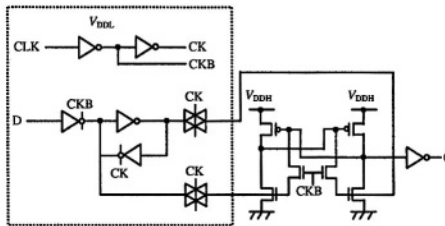


Fig. 4. FF with LC cell [4].

Finally, we discuss influence of interconnect delay. Logic gates are separated in  $V_{DDH}$  and  $V_{DDL}$  before P&R. After routing, delay of non-critical path may increase more than that of critical path by interconnect delay. In particular, interconnect delay becomes dominant in sub-100nm node. It is difficult to estimate interconnect delay before P&R. Hence there is a possibility that timing constraint is not satisfied by using  $V_{DDL}$  cells. Conventional dual supply voltage approach employs such a synthesis process as shown in Fig. 5 [4]. To avoid influence of interconnect delay, conventional approach performs synthesis in two-pass. Although this approach is effective to replace an existing layout, long TAT is needed for a new design. Furthermore, moving cells for separating rows needs rerouting, and may sometimes need retry of floor planning due to change of cell assignment from  $V_{DDL}$  to  $V_{DDH}$ . Therefore, using conventional dual supply voltage technique has been difficult in deep submicron era.

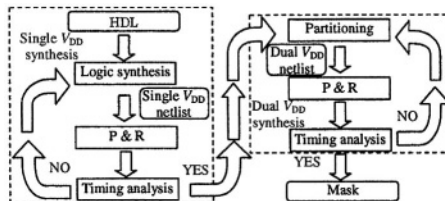


Fig. 5. Conventional synthesis process.

### 3 Proposed Layout Approach

In this section, we propose a novel layout approach for dual supply voltage technique in body-tied PD-SOI to solve the problems with conventional techniques.

#### 3.1 Proposed Cells

Our basic idea is using cells having two supply voltage rails as in [3]. By using such cells, designers place them in the same way as single  $V_{DD}$  layout as shown in Fig. 6. To use this approach, however, we need to solve the problem with body(well) contact.

Designers normally connect the body of pMOS to  $V_{DD}$ . In dual supply voltage technique, the body contact of pMOS is connected to  $V_{DDH}$  in  $V_{DDH}$  cells, and to  $V_{DDL}$  in  $V_{DDL}$  cells. For this purpose, two approaches have been proposed. One is using n-well isolation as shown in Fig. 7(a), which suffers from large area penalty due to wide well-to-well spacing. The other is using the shared-well approach [5] as shown in Fig. 7(b), with which area is smaller than Fig. 7(a). However, since all bodies of pMOS are connected to  $V_{DDH}$ , negative body bias is fed to pMOS of  $V_{DDL}$  cells, which increases gate propagation delay. This approach is used in dynamic logic circuit which contains a few number of pMOS. It is difficult to apply the shared-well approach to static CMOS.

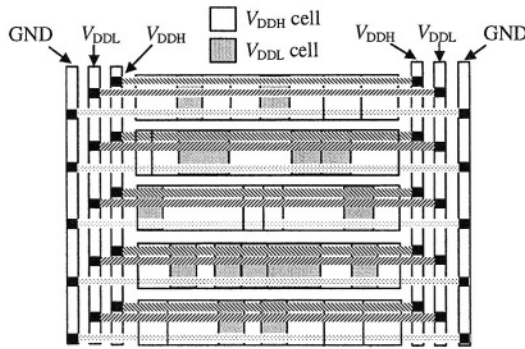


Fig. 6. Layout using proposed approach.

We solve the problem with body contact by employing body-tied PD-SOI. In SOI, body of each transistor is separated by trench isolation [6]. Large space is not needed between a part connected to  $V_{DDL}$  and that connected to  $V_{DDH}$ , which is different from well isolation approach. Furthermore, each body is free from negative bias. Therefore, such a cell layout as shown in Fig. 8 can be achieved without area penalty and speed degradation.

Furthermore, it has been difficult to apply dual supply voltage technique to datapath circuits due to separation of rows [5]. Our layout approach can also

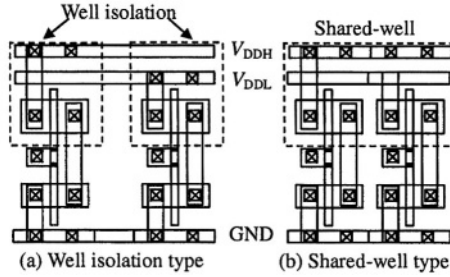


Fig. 7. Body contact with conventional approaches.

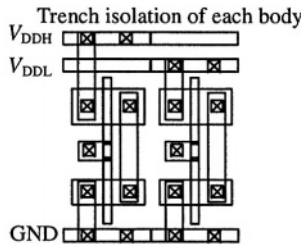


Fig. 8. Proposed cell using body-tied PD-SOI.

be applied to datapath circuits owing to freedom for placement like the single supply voltage layout.

Although our approach can be used also in floating PD-SOI, it suffers from floating body effect. In particular, history effect [7] makes timing analysis more difficult. Thus, we employ body-tied PD-SOI.

### 3.2 FF with LC Cell

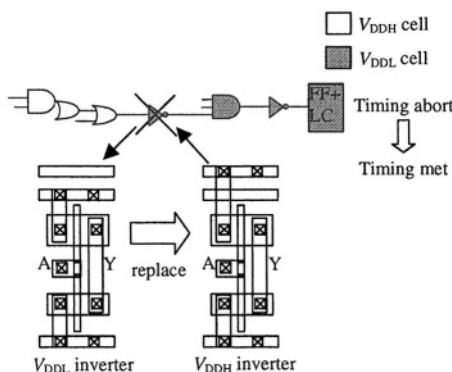
Since an FF with LC cell using our approach has two supply voltage rails, it does not need a narrow signal line connected to  $V_{DDL}$ . The FF with LC cell can be placed in the same way as an FF in single supply voltage layout. We employ FF with LC cells shown in Fig. 4. Table 1 shows performance of the FF with LC cell compared with a normal FF for single supply voltage. “FF+LC” represents FF with LC and “FF” indicates normal FF for single supply voltage layout. Both FF cell and FF with LC cell consume the maximum power when D signal changes. The maximum power is reduced by 16 %, since  $V_{DDL}$  is provided to the master latch. The average power consumption is reduced by 28 %, since level converter does not work when D signal does not change. Delay of FF with LC is longer than that of normal FF due to level conversion. This delay overhead is added to the delay in the next stage. Area overhead is 59 %, which comes from two supply voltage rails and increase in the number of transistors due to level conversion in the slave latch.

**Table 1.** Performance of FF with LC cell.

Type of FF	Supply Voltage	Power [ $\mu$ W]		Delay [ns] (CLK-Q)	Area [ $\mu$ m <sup>2</sup> ]
		max.	ave.		
FF+LC	Dual	16.5	10.8	0.32	113
FF	Single	19.7	15.0	0.20	71
Ratio	-	0.84	0.72	1.60	1.59

### 3.3 ECO Approach

As mentioned in Section 2, delay of non-critical paths using  $V_{DDL}$  cells may be longer than that of the critical path due to interconnect delay. In our approach, we make  $V_{DDL}$  cells to be pin-compatible with  $V_{DDH}$  ones. In case delay of path using  $V_{DDL}$  does not meet timing constraint,  $V_{DDL}$  cells are replaced by  $V_{DDH}$  ones in the ECO process as shown in Fig. 9. Our approach does not need to retry P&R more than once.



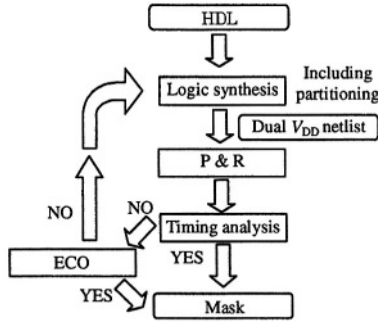
**Fig. 9.** Cell replacement for timing constraint.

### 3.4 Layout Process

Finally, we propose a synthesis process using our approach. Figure 10 shows the proposed synthesis process using dual supply voltage technique. The process is performed in one-pass, and ECO process is added after timing analysis. Since our approach is able to avoid influence on interconnect delay owing to freedom for placement and ECO approach, no P&R is retried for paths using  $V_{DDL}$ . Therefore, our approach requires shorter TAT than conventional approach.

## 4 Simulation Results

In this section, we present some SPICE simulation results with the proposed approach in 0.18  $\mu$ m PD-SOI technology. We have set the supply voltages,



**Fig. 10.** Proposed synthesis process.

$V_{DDH} = 1.8$  V and  $V_{DDL} = 1.2$  V, respectively. Value of  $V_{DDL}$  has been determined based on the relationship that  $V_{DDL} = 0.6 \sim 0.7 V_{DDH}$  for minimum power consumption [8]. We have used RC parameters extracted from layout for simulation, and applied our layout approach to ten MCNC benchmark circuits [9] and three types of 16-bit adders (Ripple carry adder: RCA, Carry select adder: CSLA, and Carry skip adder: CSKA).

We have evaluated power, delay, and area. All results have been compared with single supply voltage ( $SV_{DD}$ ) layout. In synthesizing circuits, we have defined the critical path of the whole circuit as the maximum delay path, and have replaced  $V_{DDL}$  cells by  $V_{DDH}$  ones not to exceed the critical path delay. We have evaluated the power at 100 MHz, using 100 random input vectors.

#### 4.1 Results with MCNC Benchmark Circuits

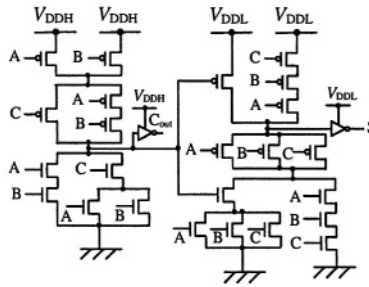
The results with MCNC benchmark circuits are shown in Table 2. Results with delay are almost the same with single  $V_{DD}$  layout in spite of area penalty. Power is reduced by 19 % in average, and by 29 % at the maximum with circuit “lal”. Area increases by 22 %. Most of area overhead depends on the ratio of the cell height,  $\text{prop.}/SV_{DD}$ . The number of grids for  $SV_{DD}$  cell and proposed one are 9 and 11, respectively. If  $\text{prop.}/SV_{DD}$  ratio is small, area penalty would be reduced.

#### 4.2 Results with Adders

We have designed a full adder cell as shown in Fig. 11. This full adder cell contains blocks for carry generation and sum generation. The carry generation block is usually critical for the overall delay of adder. In conventional dual supply voltage techniques, it is impossible to make a cell as shown in Fig. 11, because transistors using different supply voltages cannot be placed in a cell. Such a cell can also be implemented with our approach. We have confirmed that our approach can be applied to datapath circuits through the implementation of such adders.

**Table 2.** Delay, power, and area with MCNC circuits.

Circuits	Delay [ns]			Power [ $\mu$ W]			Area [ $\mu$ m <sup>2</sup> ]		
	SV <sub>DD</sub>	prop	Prop SV <sub>DD</sub>	SV <sub>DD</sub>	prop	Prop SV <sub>DD</sub>	SV <sub>DD</sub>	prop	Prop SV <sub>DD</sub>
b9	0.62	0.64	1.03	143.0	109.6	0.77	1429	1725	1.21
cc	0.57	0.57	1.00	63.4	50.0	0.79	741	906	1.22
cmb	0.43	0.44	1.02	49.0	38.2	0.78	653	792	1.21
cu	0.41	0.41	1.00	42.6	37.7	0.88	595	728	1.22
lal	0.74	0.76	1.03	90.8	64.6	0.71	1125	1380	1.23
ldd	0.87	0.87	1.00	103.6	90.7	0.88	1080	1315	1.22
pcl	0.66	0.72	1.09	45.6	37.6	0.82	635	776	1.22
pm1	0.36	0.36	1.00	45.7	40.5	0.89	529	647	1.22
x2	0.52	0.54	1.04	64.0	56.8	0.89	622	760	1.22
z4ml	0.61	0.61	1.00	33.7	29.3	0.87	326	399	1.22
Ave.	-	-	1.02	-	-	0.81	-	-	1.22



**Fig. 11.** Schematic of proposed full adder.

**Table 3.** Delay, power, and area with adders.

Circuits	Delay [ns]			Power [ $\mu$ W]			Area [ $\mu$ m <sup>2</sup> ]		
	SV <sub>DD</sub>	prop	Prop SV <sub>DD</sub>	SV <sub>DD</sub>	prop	Prop SV <sub>DD</sub>	SV <sub>DD</sub>	prop	Prop SV <sub>DD</sub>
RCA	2.37	2.37	1.00	251.9	172.6	0.69	1200	1536	1.28
CSLA	1.34	1.34	1.00	420.7	296.1	0.70	2408	3067	1.27
CSKA	0.88	0.89	1.01	315.5	268.3	0.85	3607	4409	1.22
Ave.	-	-	1.00	-	-	0.75	-	-	1.25

Table 3 shows the results with three types of adders. Power is reduced by 25 % without speed degradation. In particular, power of RCA is reduced by 31 %. However, area penalty is 25 %. Our future work is to improve the proposed technique in order to reduce area penalty.



## 5 Conclusion

In this paper, we have proposed a novel layout approach using dual supply voltage technique for body-tied PD-SOI. The proposed technique uses cells having two supply voltage rails, which is difficult in bulk due to increase in area by n-well isolation or in delay by negative body bias caused by sharing n-well between different supply voltages. As a result of SPICE simulation, we have confirmed that average power is reduced by 19 % for MCNC circuits and by 25 % for adders without speed degradation, respectively. Our future work is to improve the proposed technique in order to reduce area penalty.

## References

1. T. Sakurai and A. R. Newton "Alpha-power law MOSFET model and its application to CMOS inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 584-594, Apr. 1990.
2. K. Usami and A. M. Horowitz "Clustered voltage scaling technique for low-power design," Proc. *ISLDP '95*, pp. 3-8, 1995.
3. C. Yeh, Y. Kang, S. Shieh and J. Wang, "Layout techniques supporting the use of dual supply voltages for cell-based designs," Proc. *36th DAC*, pp. 62-67, 1999.
4. M. Takahashi, M. Hamada, T. Nishikawa, H. Arakida, T. Fujita, F. Hatori, S. Mita, K. Suzuki, A. Chiba, T. Terazawa, F. Sano, Y. Watanabe, K. Usami, M. Igarashi and T. Ishikawa "A 60-mW MPEG4 video codec using clustered voltage scaling with variable supply-voltage scheme," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1772-1780, Nov. 1998.
5. Y. Shimazaki, R. Zlatanovici and B. Nikolic "A shared-well dual supply voltage 64-bit ALU," Digest of technical paper, *ISSCC*, 2003.
6. Y. Hirano, S. Maeda, T. Matsumoto, K. Nii, T. Iwamatsu, Y. Yamaguchi, T. Ipposhi, H. Kawashima, S. Maegawa, M. Inuishi and T. Nishimura "Bulk-layout-compatible 0.18- $\mu\text{m}$  SOI-CMOS technology using body-tied partial-trench-isolation (PTI)," *IEEE Trans. Elec. Dev.*, vol. 48 no. 12 pp. 2816-2822, Dec. 2001.
7. D. Suh and J. G. Fossum "A physical charge-based model for non-fully depleted SOI MOSFET's and its use in assessing floating-body effects in SOI CMOS circuits," *IEEE Trans. Elec. Dev.*, ED-42 pp. 728, 1995.
8. M. Hamada, M. Takahashi, H. Arakida, A. Chiba, T. Terazawa, T. Ishikawa, M. Kanazawa, M. Igarashi and K. Usami "A top-down low power design technique using clustered voltage scaling with variable supply-voltage scheme," Proc. *CICC '98*, pp. 495-498, 1998.
9. S. Yang "Logic synthesis and optimization benchmarks user guide version 3.0," MCNC, 1991.

# Simultaneous Wire Sizing and Decoupling Capacitance Budgeting for Robust On-Chip Power Delivery\*

Jingjing Fu<sup>1</sup>, Zuying Luo<sup>1</sup>, Xianlong Hong<sup>1</sup>, Yici Cai<sup>1</sup>,  
Sheldon X.-D. Tan<sup>2</sup>, and Zhu Pan<sup>1</sup>

<sup>1</sup> Department of Computer Science and Technology, Tsinghua University,  
Beijing, 100084, P.R.China

{fujingjing00,pz01}@mails.tsinghua.edu.cn,

{luozy,hxl-dcs}@mail.tsinghua.edu.cn

<sup>2</sup>Department of Electrical Engineering, University of California at Riverside,  
CA 92521, USA

stan@ee.ucr.edu

**Abstract.** In this paper, we present an efficient method to simultaneously size wire widths and decoupling capacitance (decaps) areas for optimizing power/ground (P/G) networks modeled as RLC linear networks subject to reliability constraints. We formulate the problem as a nonlinear optimization problem and propose an efficient gradient-based non-linear programming method for searching the solution. We apply a time-domain merged adjoint network to efficiently compute the gradients and a novel equivalent circuit modeling technique to speed up the optimization process. The resulting algorithm is very efficient and experimental results show that the new algorithm is capable of optimizing P/G networks modeled as RLC networks with million nodes within 10 hours in modern workstations.

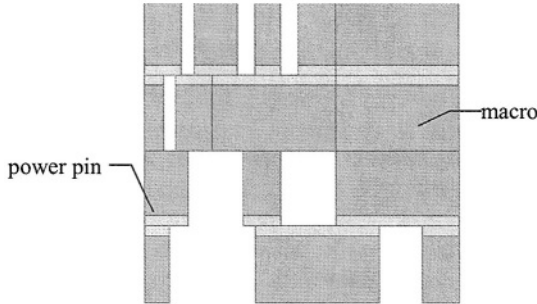
## 1 Introduction

Signal integrity in VLSI is emerging as a limiting factor in the nano-regime VLSI chip designs as technology scales. This is especially true on global networks like P/G networks where noise margins have been reduced greatly in today's advanced VLSI designs due to decreasing supply voltages and the presence of excessive voltage drops coming from resistive and inductive effects. In order to reduce IR drops, wire sizing is typically employed [4][5][6][7]. As for dynamic voltage fluctuations on a P/G network, adding decap is the most efficient way to reduce such noises [8][9][18].

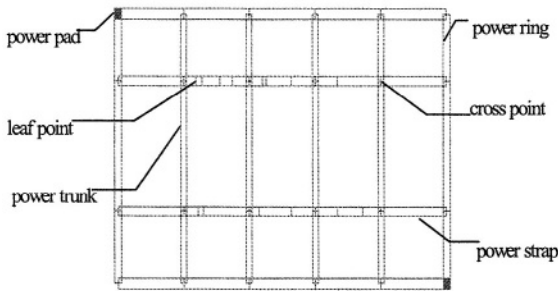
Because wires consume the routing resource while decaps cost spare die area, algorithm that can simultaneously optimize both of them is needed to reach the best trade-off between routing resource and die area as demonstrated in [10], which used a geometric multi-grid based method. But the geometric multi-grid approach was only applied to mesh-structured P/G grids modeled as resistor-only networks with decaps where capacitive and inductive parasitics on wire segments are ignored. Also the

---

\* This work was supported by National Natural Science Foundation of China (NSFC) 90307017, 60176016 and 60121120706 and National Natural Science Foundation of USA (NSF) CCR-0096383.



**Fig. 1.** An instance of standard cell layout



**Fig. 2.** Structure of power network

geometric multi-grid simulation method [10] is less flexible to deal with non-mesh-structured circuits [17], which are very common in various ASIC designs. Another problem with the multi-grid method is that it can easily lead to optimistic solutions [14][17], which makes this method less reliable.

In this paper, we focus on standard-cell like ASIC layout structures as shown in Fig.1 and try to optimize the areas of P/G networks as the objective [6][7][9]. In the figure, the white space is the spare space for adding decaps to reduce dynamic noise. Fig.2 shows a power network of the standard-cell ASIC layout shown in Fig.1. Power trunks in the power network will be sized for reducing IR drop.

The P/G networks are modeled as lumped RLC circuits. To clearly describe our work, we make the following assumptions:

- 1) The package is predominantly inductive, and is modeled by serially connected inductors through power supply sources to their contact points on the power grid.
- 2) The power grid is modeled as a RLC network where R and L are the parasitical resistance and inductance of wires and C is the capacitance consisting of parasitical capacitance and decoupling capacitance (both built-in and added-on).
- 3) Connections between top and low levels are fine enough that we ignore the resistor of the connection.
- 4) The nonlinear devices or modules are modeled as time-varying current sources connected between each node on power grid and ground.

The problem we are concerned in this paper is how to efficiently optimize P/G networks by simultaneously sizing wires and decaps subject to the reliability and design rules for P/G grids, which consist of constraints due to dynamic voltage fluctuation.

tuations, electro-migration and other design rules. The advantages of our approach are:

- 1) The approach can optimize P/G networks modeled as RLC networks.
- 2) It is a very general approach and can be applied to any P/G network structures.
- 3) The new method is based on a very efficient and general nonlinear programming framework, which can deal with both convex and concave cases [7] and can be scaled to handle circuits with millions of nodes.
- 4) Gradients are efficiently computed using time-domain merged adjoint network combined with equivalent circuit modeling techniques.

This paper is organized as follows. Section 2 formulates the wire sizing and decap budgeting problem. The efficient non-linear programming technique and the equivalent circuit modeling method are described in Section 3. Experimental results are presented in Section 4. Section 5 concludes the paper.

## 2 Problem Formulation

For the similarity of power and ground networks, we will only describe the algorithm for power networks in this paper. Let  $G = \{N, B, M\}$  be a power network with  $n$  nodes  $N = \{1, \dots, n\}$  and  $b$  RLC branches of power trunks  $B = \{1, \dots, b\}$  and  $m$  nodes that are permitted to connect tunable decaps  $M = \{1, \dots, m\}$ . Each branch  $(p, q)$  in  $B$  connects two nodes  $p$  and  $q$  with current flowing from  $p$  to  $q$ . For a node  $p$ ,  $v_p(t)$  is the node voltage.

### 2.1 Objective Function

The objective is to minimize the combined area used by power grid wires and decaps subject to design rules and power network integrity related constraints:

$$\min A = \sum_{i \in B} (l_i w r_i) + \eta \sum_{j \in M} (w c_j \times H) \tag{1}$$

where  $l_i$  and  $w r_i$  are the length and width of branch  $i$ ,  $H$  is the fixed height of standard cells and  $w c_j$  is the width of the decap connected to node  $j$ . And  $\eta$  is the weighted factor for total decap area. Designers can properly set the value of  $\eta$  to set up the priorities of the resources of the routing and die area.

### 2.2 The Constraints

**1) The voltage drop constraints.** To ensure the correct and reliable logic operation, the voltage drops from the power pads to the leaf nodes should be restricted. To efficiently measure the dynamic voltage drop, we follow the voltage drop noise metric definition in [9] and reformulate the voltage drop constraints as follows:

$$s_i = \int_0^T \max(V_{\min} - v_i(t), 0) dt = \int_{t_1}^{t_2} (V_{\min} - v_i(t)) dt, \tag{2}$$

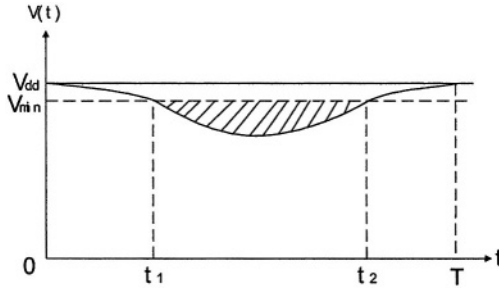


Fig. 3. Illustration of dynamic voltage drops

where  $[t_p, t_2]$  is the time interval in which the constraint is violated as shown in Fig.3. There may exist several such intervals in one clock cycle.

**2) The electro-migration constraints.** Electro-migration effects on a power grid wire segment set an upper bound on the current density of the wire segment as the routing layer has fixed thickness. This constraint for branch  $(p, q)$  is expressed as  $|i_{p,q}(t)| \leq w_{p,q} \times \sigma$ , and can be re-written as  $|v_p(t) - v_q(t)| \leq \rho l_{pq} \sigma$ , where  $\sigma$  is the maximal current density,  $\rho$  is the sheet resistance, and  $l_{pq}$  is the length of branch  $(p, q)$ . Due to dynamic nature of  $v_{p,q}(t)$ , we reformulate the constraint as below:

$$\begin{aligned}
 t_{p,q} &= \int_0^T \max \left[ \left( |v_p(t) - v_q(t)| - \rho l_{p,q} \sigma \right), 0 \right] dt \\
 &= \int_{t_1}^{t_2} \left( |v_p(t) - v_q(t)| - \rho l_{p,q} \sigma \right) dt
 \end{aligned}
 \tag{3}$$

where  $[t_p, t_2]$  is the time interval in which the constraint is violated. Similarly, there may exist several such intervals in one clock cycle.

**3) Minimal wire width constraints.** The wire width cannot be smaller than the minimal width of metal line  $w_{min}$ .

**4) Decap area constraints.** Because we essentially re-distribute the spare space around cells, the total width of all decaps added in a row should be limited by the width of total spare area in this row:

$$dw_{ir} = \max \left( \sum_{jc \in ND(ir)} w_{ir,jc} - rw_{ir}, 0 \right), ir \in NR,
 \tag{4}$$

where  $NR$  is the row set defined as  $\{1, 2, \dots, N_{row}\}$  and  $N_{row}$  is the row number of the cells in the placement;  $ND(ir)$  is the decap position set of  $ir^{th}$  row defined as  $\{1, 2, \dots, nr_{ir}\}$  and  $nr_{ir}$  is the number of nodes where decaps can be attached in the  $ir^{th}$  row;  $w_{ir,jc}$  is the width of the decap at row  $ir$  and position  $jc$ ; and  $rw_{ir}$  is the width of total spare area in row  $ir$ .

**5) Decap maximum width constraints.** The width of a decap in row  $ir$  should be bounded by the total width of spare area provided by this row.

$$ew_{ir,jc} = \max(w_{ir,jc} - rw_{ir}, 0), ir \in NR, jc \in ND(ir),
 \tag{5}$$

### 3 Solution Based on Non-linear Programming

The resulting problem is a nonlinear minimization problem as voltages and currents are nonlinear functions of the width vector of wires and decap areas. As a result, we propose to use a gradient-based non-linear programming method to solve the optimization problem as gradients can be efficiently computed in time-domain.

Specifically, at the beginning, all the decap widths are set to be zero while all the wire widths are assigned as the minimum wire width  $w_{min}$ . We then analyze the network to get the node voltage waveforms and the branch currents, and identify the constraint violations. In each optimization iteration, we use conjugate gradient method to update wire and decap widths. This process stops when all the constraints are satisfied or no improvement can be made.

#### 3.1 Formulation of Penalty Function

The first step is to transform the original constrained problem into a sequence of unconstrained problems. The transformation is accomplished by adding to the objective function a penalty term that gives a high cost to the constraint violations. We adopt a penalty function as below:

$$f = A + \alpha \cdot \left( \sum_{i \in N} s_i^2 + \sum_{(p,q) \in B} t_{pq}^2 + \sum_{ir \in NR} dw_{ir}^2 + \sum_{jc \in ND(ir)} ew_{ir,jc}^2 \right), \tag{6}$$

where  $\alpha$  is the penalty parameter.  $A$ ,  $s_i$ ,  $t_{p,q}$ ,  $dw_{ir}$  and  $ew_{ir,jc}$  are defined in equations (1), (2), (3), (4) and (5) respectively. Let us set the penalty term  $p_t$  as

$$p_t = \alpha \cdot \left( \sum_{i \in N} s_i^2 + \sum_{(p,q) \in B} t_{pq}^2 + \sum_{ir \in NR} dw_{ir}^2 + \sum_{jc \in ND(ir)} ew_{ir,jc}^2 \right), \tag{7}$$

We then rewrite the penalty function as  $f=A+p_t$ . Meanwhile, we transform the original constrained problem into a problem of minimizing the penalty function (8),

$$\begin{aligned} \min f &= A + p_t \\ &= A + \alpha \cdot \left( \sum_{i \in N} s_i^2 + \sum_{(p,q) \in B} t_{pq}^2 + \sum_{ir \in NR} dw_{ir}^2 + \sum_{jc \in ND(ir)} ew_{ir,jc}^2 \right), \end{aligned} \tag{8}$$

In this optimization problem, the variable is the width vector  $W=[wr_1, wr_2, \dots, wr_b, wc_1, wc_2, \dots, wc_m]^T$  that consists of two parts: the wire width vector  $Wr=[wr_1, wr_2, \dots, wr_b]^T$  and the decap width vector  $Wc=[wc_1, wc_2, \dots, wc_m]^T$ .

#### 3.2 Optimization Scheme

Given an initial penalty parameter  $\alpha$ , we minimize the penalty function. We then increase the value of the penalty parameter  $\alpha$  for the next minimization iteration such that the contribution of the penalty with respect to the actual cost is maintained at a

constant level. Such dynamic penalty factor adjustment can help speedup of the convergence of the optimization process. The process continues until all the constraints are satisfied or no improvement can be found. The solution procedure can be described briefly as below:

1. Set an initial value of penalty parameter  $\alpha$ ; initial wire and decap width vector  $W^{(0)}=[w_{r_1}, w_{r_2}, \dots, w_{r_b}, w_{c_1}, w_{c_2}, \dots, w_{c_m}]^T=[w_{min}, w_{min}, \dots, w_{min}, 0, 0, \dots, 0]^T$ ; and error bound,  $\epsilon_b > 0$ .
2. Solve unconstrained minimization problem (8), obtain current width vector  $W^{(k)}$ .
3. If  $p_i^{(k)} < \epsilon_b$ , then stop, else update penalty parameter  $\alpha$ , set  $k = k + 1$  and turn to step 2.

### 3.3 Time-Domain Merged Adjoint Network Method

In order to efficiently compute the gradients used in conjugate gradient method, we directly compute the required gradient vector used in the objective function instead of individual gradients of each wire and decap as traditional method did [9]. By using the convolution method for node voltage sensitivity calculation presented in [9], the final resulting method is a time-domain merged adjoint network method, which was described in detail in [7] [18]. The gradient of penalty function  $f$  with respect to the width vector  $W$  can be expressed as following:

$$\nabla f(W) = \left[ \frac{\partial f}{\partial w_{r_1}}, \dots, \frac{\partial f}{\partial w_{r_i}}, \dots, \frac{\partial f}{\partial w_{r_b}}, \frac{\partial f}{\partial w_{c_1}}, \dots, \frac{\partial f}{\partial w_{c_j}}, \dots, \frac{\partial f}{\partial w_{c_m}} \right]^T, \quad (9)$$

$i \in B, j \in M$

Using time-domain merged adjoint network method, we only need to simulate the power network twice. The first simulation is on time period from 0 to  $T$  with original time-varying current sources in original power network. The second simulation is on time period from  $T$  to 0 with merged current sources in the adjoint network. Then, voltages got from these two simulations are convolved to obtain the gradients. Such two-simulation approach can significantly speed up the optimization process as the computing cost is independent of violation nodes.

### 3.4 Equivalent Circuit Modeling for Transient Simulation

Although many efficient simulation methods have been proposed in the past [1]-[3][11]-[16], the regular structures of RLC P/G networks with standard-cell layouts are not explicitly exploited. It was shown that there exist many regular structures in the P/G networks of standard-cell layouts as shown in Fig.4. For such a RLC chain circuit, we can reduce it into a simple resistor-only equivalent circuit at each time step and thus speed up the transient simulation of the P/G networks.

In our method, we combine equivalent circuit modeling with preconditioned conjugate gradient method (PCG)[7][16] to perform the transient analysis. Specifically, at each time step, the numerical integration by using companion models in Norton’s form is performed and the original RLC circuit will become a resistor-only circuit.

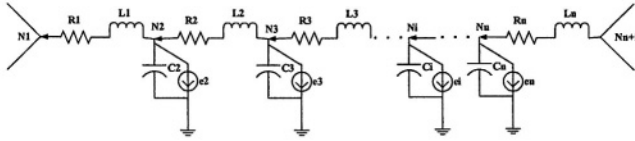


Fig. 4. A series RLC chain in a P/G network.

We then repeatedly apply the Y-Δ network transformation to reduce one node at a time until only the terminal nodes are left. After this, PCG algorithm solves the reduced network. Once the voltage at terminal nodes are solved, all the voltages at intermediate nodes of original circuits can be back solved using the superposition principle.

The new equivalent modeling technique can be viewed as a special pre-ordered Gaussian elimination process where no-fills are generated. By combining it with PCG, we take advantage of both direct (via Gaussian elimination) and iterative approaches.

### 4 Experimental Results

The proposed optimization algorithm has been implemented in C and C++ programming languages. All the experimental results are obtained on SUN UltraSparc workstation V880 with 750MHz CPU and 2GB memory.

We tested our program with some real industry standard-cell circuits with pre-placement information in LEF/DEF format. Those circuits have complexities ranging from 744 nodes to 1.6 million nodes. To demonstrate the efficiency of our algorithm we compare the experimental results with the results from method named two-step optimization method that first size the wire widths and then tune decap size. Table 1 shows the parameters of each circuit and the experimental results of the two-step method. In table 1, the violation nodes are nodes on the power grid that violate the voltage drop constraints or the electro-migration constraints. Column 4 and 5 in table 1 show the routing resources and space chip areas used by the two-step optimization method respectively. The last column shows the running time.

Table 1. Experimental results of two-step method

Name of circuits	#nodes	#violation nodes	Area of power grid (μm <sup>2</sup> )	Area of decaps (mm <sup>2</sup> )	Time(s)
Test1	744	91	6006.00	0.0057	110.23
Test2	3741	665	16828.50	0.0350	341.78
Test3	32112	3683	66452.40	0.1745	1315.63
Test4	112392	10755	125239.68	2.6465	2549.37
Test5	321120	11496	87264.00	3.2290	9485.69
Test6	1618026	612132	368288.00	3.0256	20499.89



**Table 2.** Comparison of new algorithm with the two-step method

Name of circuits	$\eta$ (wire/decap)	Area of power grid ( $\mu\text{m}^2$ )	Area of power grid ratio	Area of decaps ( $\text{mm}^2$ )	Area of decaps ratio	Time(s)
Test1	0.70/0.30	4938.51	82.23%	0.0051	90.52%	28.79
Test2	0.62/0.38	13503.65	80.24%	0.0321	91.63%	736.18
Test3	0.43/0.57	52500.77	79.01%	0.1861	106.61%	1382.81
Test4	0.60/0.40	44202.21	35.29%	0.9596	36.26%	4297.58
Test5	0.76/0.24	87264.47	100.00%	2.7587	85.44%	17058.32
Test6	0.10/0.90	194976.00	52.94%	2.1878	72.31%	22252.80

Table 2 shows the experimental results of the proposed optimization algorithm and the comparison between these two algorithms. In column 2,  $\eta$  is the weighted factor for routing resource and total decap area. Designers can properly set the value of  $\eta$  to set up the priorities of the resources of routing and die area. In Table 2, for each circuit, the power grid area and decap area obtained from two-step method are normalized to 1 and are compared with the new method in terms of normalized ratios in column 4 and 6. From column 4, 6 and 7 we can find that the new algorithm is more area efficient although it takes more time to achieve this. It shows that our algorithm has the capability of optimizing very large scale circuit instance. For instance it takes 6.2 hours to optimize the largest circuits instance (Test6) with 16 millions nodes. After the optimization, all the violations are gone. Although we note that if the optimization problem is over constrained, we still may end up with violations after optimizations.

## 5 Conclusion

In this paper, we have proposed an efficient algorithm to optimize P/G networks modeled as RLC circuits by simultaneously sizing wires and tuning decap sizes subject to the reliability and design rule constraints of VLSI on-chip P/G grids. Our algorithm utilizes gradient-based nonlinear programming algorithm to search for the best solution and is flexible enough to handle any P/G circuits with capacitive and inductive parasitics. By applying time-domain merged adjoint network method combined with a novel equivalent circuit modeling technique, gradients used in our non-linear programming framework can be efficiently computed, which is key to the overall efficiency of the new algorithm. Experimental results show that by simultaneously sizing the areas of decaps and areas of power grid wire segments, the new algorithm is more area efficient compared with a two-step algorithm where wire sizing and decap allocation are done sequentially. Comparing with the existing methods, the new method can simultaneously perform the wire sizing and decap budgeting on P/G networks modeled as RLC networks with millions of nodes in a reasonable time for the first time.

## References

1. H. H. Chen, D. D. Ling. "Power supply noise analysis methodology for deep-submicron VLSI chip design." Proc. 34th ACM/IEEE Design Automation Conf., pp. 638-643,1997.
2. G. Bai, S. Bobba and I. N. Hajj, "Simulation and optimization of the power distribution network in VLSI circuits", Proc. IEEE/ACM International Conf. on Computer-Aided Design., pp. 481-486, 2000.
3. H.-H. Su, K. H. Gala, and S. S. Sapatnekar. "Fast analysis and optimization of power/ground networks." Proc. IEEE/ACM International Conf. on Computer-Aided Design., pp. 477-482, 2000.
4. S. Chowdhury and M. A. Breuer, "Minimal area design of power/ground nets having graph topologies," IEEE Trans. on Circuits and Systems, pp.1441-1451, December. 1987
5. T. Mitsuhashi and E.S. Kuh, "Power and Ground Network Topology Optimization for Cell Based VLSIs", Proc. 29th ACM/IEEE Design Automation Conference, pp. 524-529, 1992
6. X.-D. Tan and C.-J. Shi. "Reliability-constrained area optimization of VLSI power/ground networks via sequence of linear programming." Proc. 36th ACM/IEEE Design Automation Conf., pp. 78-83,1999.
7. X.-H. Wu, X.-L. Hong, Y.-C. Cai, and et al. "Area minimization of power distribution network using efficient nonlinear programming techniques." Proc. IEEE/ACM International Conf. on Computer-Aided Design., pp. 153-157, 2001.
8. S.-Y. Zhao, K.K. Roy, C.-K. Koh. "Decoupling capacitance allocation for power supply noise suppression." Proc. IEEE/ACM International Symp. on Physical Design, pp.66-71,2001.
9. H.-H. Su, K.K. Roy, S.S. Sapatnekar, S.R. Nassif. "An algorithm for optimal decoupling capacitor sizing and placement for standard cell layouts. " Proc. IEEE/ACM International Symp. on Physical Design, pp.68-75,2002.
10. K. Wang and M. Marek-Sadowask, "On-chip power supply network optimization using multigrid-based technique", Proc. ACM/IEEE Design Automation Conf., pp. 113-118, June, 2003.
11. X.-D. Tan and C.-J. Shi. "Fast power-ground network optimization using equivalent circuit modeling," Proc. 38th ACM/IEEE Design Automation Conf., pp. 550-554, 2001.
12. A. Odabasioglu, M. Celik, and L. T. Pilleggi, "PRIME: Passive reduction-order interconnect macromodeling algorithm," IEEE Trans. On Computer-Aided Design, vol. 17, no. 8, pp.645-654. 1998
13. A Dharchoudhury, R. Panda, D. Blaauw, R. Vaidyanathan and et al "Design and analysis of power distribution networks in power PC microprocessors", Proc. ACM/IEEE Design Automation Conf., pp. 738-743, June, 1998.
14. J. N. Kozhaya, S. R. Nassif, and F.N. Najm. "A multigrid-like technique for power grid analysis. IEEE Trans. Computer-Aided Design, vol. 21, no.10, pp. 1148-1160, Oct. 2002.
15. M. Zhao, R. V. Panda, S. S. Sapatnekar and et al, "Hierarchical analysis of power distribution networks", Proc. ACM/IEEE Design Automation Conf., pp. 150-155, June, 2000
16. T. Chen and C. C. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative method", Proc. ACM/IEEE Design Automation Conf., pp. 559-562, June, 2001
17. S.R.Nassif, and J.N.Kozhaya. "Fast power grid simulation", Proc. ACM/IEEE Design Automation Conf., pp. 156-161, June, 2000.
18. J. Fu and Z. Luo and X. Hong and Y. Cai and S. X.-D. Tan and Z. Pan "A fast decoupling capacitor budgeting algorithm for robust on-chip power delivery", Proc. Asia South Pacific Design Automation Conf. (ASPDAC), pp. 505-510, Jan. 2004.

# An Efficient Low-Degree RMST Algorithm for VLSI/ULSI Physical Design\*

Yin Wang<sup>1</sup>, Xianlong Hong<sup>1</sup>, Tong Jing<sup>1</sup>, Yang Yang<sup>1</sup>,  
Xiaodong Hu<sup>2</sup>, and Guiying Yan<sup>2</sup>

<sup>1</sup> Dept. of Computer Science and Technology, Tsinghua Univ., Beijing 100084, P. R. China  
{wang-y01, yyys99}@mails.tsinghua.edu.cn,  
{hxl-dcs, jingtong}@tsinghua.edu.cn

<sup>2</sup> Institution of Applied Mathematics, Chinese Academy of Sciences,  
Beijing 100080, P. R. China  
xdhu@public.bta.net.cn, yangy@mail.amss.ac.cn

**Abstract.** Motivated by very/ultra large scale integrated circuit (VLSI/ULSI) physical design applications, we study the construction of rectilinear minimum spanning tree (RMST) with its maximum vertex degree as the constraint. Given a collection of  $n$  points in the plane, we firstly construct a graph named the bounded-degree neighborhood graph (BNG). Based on this framework, we propose an  $O(n \log n)$  algorithm to construct a 4-BDRMST (RMST with maximum vertex degree  $\leq 4$ ). This is the first 4-BDRMST algorithm with such a complexity, and experimental results show that the algorithm is significantly faster than the existing 4-BDRMST algorithms.

## 1 Introduction

In recent years, the very/ultra large scale integrated circuit (VLSI/ULSI) has profoundly advanced, which enables us to design a single chip with more and more functions and many more transistors. Meanwhile, the needs for small chip size propel fabrication technology into the nanometer era. The shrinking of geometries calls for great concerns for interconnect effects. For high performance VLSI/ULSI physical design, interconnect optimization research should play an active role.

Rectilinear minimum spanning tree (RMST) construction is fundamental to the interconnect optimization in physical design. RMST is frequently used as the wire length/approximation delay estimation in the whole chip floorplanning and placement phase. Since Hwang [10] proved that RMST is a  $3/2$  approximation of the rectilinear Steiner minimal tree (RSMT), many Steiner tree heuristics use RMST as a backbone for RSMT in the process of routing. Thus, RMST needs highly efficient solutions.

Many physical design algorithms construct a RMST as a framework for later processing. They require RMST with a low maximum vertex degree because their time complexity often grows up exponentially with respect to the maximum vertex degree.

---

\* This work was supported in part by the NSFC under Grant No.60373012 and No. 60121120706, the SRFDP of China under Grant No.20020003008, and the Hi-Tech Research and Development (863) Program of China under Grant No.2002AA1Z1460.

These include the algorithm of Georgakopoulos and Papadimitriou [6] and some Steiner tree approximations [12], as well as VLSI global routing algorithms [1,17]. Thus, low-degree RMST construction is important to such applications. This paper mainly focuses on the construction of a low-degree RMST.

The remainder of this paper is organized as follows. In Section 2, we define the bounded degree rectilinear minimum spanning tree (BDRMST) and the bounded-degree neighborhood graph (BNG), and discuss their properties. In Section 3, the algorithm of constructing the BNG is described. In Section 4, a proof of the algorithm's correctness is given. Then, Section 5 shows experimental results. Finally, Section 6 concludes the paper.

## 2 Preliminaries

### 2.1 The BDRMST

In a given set of points in the plane and an integer  $d \geq 2$ , find a minimum-cost rectilinear spanning tree with maximum vertex **degree**  $\leq d$ . We call this rectilinear spanning tree a *bounded degree rectilinear minimum spanning tree* ( $d$ -BDRMST).

Existing spanning tree algorithms can be classified into two categories. Some of them efficiently find a minimum-cost spanning tree [2,9,11,14,18,23], but they do not guarantee a bound on the maximum vertex degree. Others indeed construct bounded-degree spanning trees [3,13,19], but they do not guarantee a minimum-cost connection.

Finding a RMST with bounded vertex degrees is usually hard. In fact, finding a 2-BDRMST is equivalent to solving the traveling salesman problem (TSP), which is known to be NP-hard [5]. Papadimitriou and Vazirani [16] showed that finding a Euclidean 3-BDRMST is also NP-hard. However, they noticed that a Euclidean 5-BDRMST could be found in polynomial time. Later, Robins and Salowe [20] showed that each point set in the rectilinear plane has a 4-BDRMST. They pointed out that the 4-BDRMST could be computed in polynomial time. Shortly after, Griffith et al. [8] proposed a polynomial time algorithm to compute a 4-BDRMST as a step in their Batched 1-Steiner heuristic (BIS). The running time of their algorithm is  $O(n^2)$ . There are two limitations to their algorithm. Firstly, they use a linear-time neighbor searching method. Searching the neighbors of  $n$  points takes  $O(n^2)$  time. Secondly, in the point-adding phase, they use a linear-time dynamic minimum spanning tree (MST) maintenance algorithm, and adding  $n$  points takes  $O(n^2)$  time. Even though Fredrickson's data structure [4] can reduce the time complexity of the point adding stage to  $O(n \log n)$ , the neighbor searching stage will still cost  $O(n^2)$  time. It is impractical due to its complicated description and large hidden constants. This scheme can not improve the running time further.

The quadratic complexity of the 4-BDRMST construction does not slow down BIS because BIS itself is an  $O(n^3)$  algorithm. But in physical design applications, it needs a sub-quadratic Steiner heuristic based on low-degree RMST. An  $O(n^2)$  BDRMST will become the bottleneck. So, we try to solve the 4-BDRMST problem in an efficient way.

In order to provide a framework for BDRMST construction, we construct a subgraph of the Delaunay triangulation of which the maximum vertex degree  $\leq 4$ . We call this graph the *bounded-degree neighborhood graph* (BNG). Based on this framework, we propose an  $O(n \log n)$  algorithm to construct a 4-BDRMST. To our knowledge, this is the first 4-BDRMST algorithm proved with such a time complexity. We have implemented the algorithm and compared it with existing algorithms. Experimental results show that our algorithm is much faster than the 4-BDRMST algorithm of Griffith *et al* [8]. Meanwhile, it is much faster than many typical algorithms that compute ordinary RMSTs.

## 2.2 The BNG

Since the MST problem on a weighted graph is well studied [2,7,14,18], the idea to construct a metric MST by first constructing a graph on the point set are proposed in many metric MST algorithms [9,22,23]. But they cannot output MSTs with bounded degree. Griffith *et al.*'s 4-BDRMST algorithm does not construct a graph, but it actually utilizes an implicit graph in the neighborhood relations. Similarly, our 4-BDRMST algorithm is based on a graph named the BNG.

Before we define the BNG, we need to mention the *uniqueness property* observed by Robins and Salowe [20].

**Definition 1.** Given a point  $p$ , a region  $R$  has the uniqueness property with respect to  $p$  if for every pair of points  $u, w \in R$ , we have  $\|wu\| < \max (\|wp\|, \|up\|)$ . A partition of space into a finite set of disjoint regions is said to have the uniqueness property if each of its regions has the uniqueness property.

We call a partition with the uniqueness property a *unique partition*, and a region of a unique partition a *unique region*. Fig.1 illustrates a kind of unique partition in the rectilinear plane that have the uniqueness property. This is called a *diagonal partition*. Note that it has 8 unique regions (4 two-dimensional “wedges” and 4 one-dimensional “half-lines”).

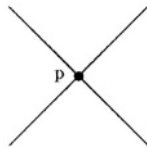


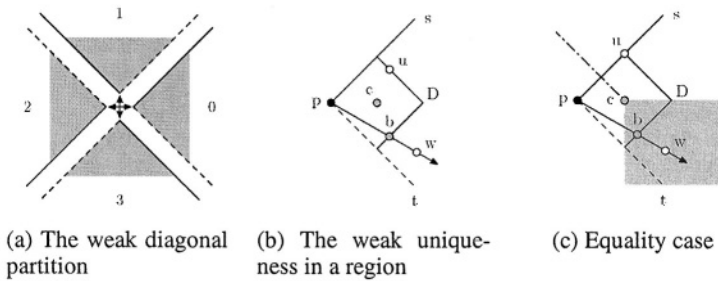
Fig. 1. Partition that has the uniqueness property (the diagonal partition)

Consider the MST algorithms on graphs. The *cycle property* of the MST states that, an edge with the longest weight in any cycle can be safely deleted. So the uniqueness property implies that, if we are going to construct a MST on the point set, we can connect points to only the nearest neighbors in a unique region.

The diagonal partition is rather strong in the sense that it ensures  $\|wu\| < \max (\|wp\|, \|up\|)$ . If we use this partition, the maximum degree of the MST would be 8. In fact,  $\|wu\| \leq \max (\|wp\|, \|up\|)$  is sufficient for the MST construction. So we define the *weak uniqueness property* as follows.

**Definition 2.** Given a point  $p$ , a region  $R$  has the weak uniqueness property with respect to  $p$  if for every pair of points  $u, w \in R$ , we have  $\|wu\| \leq \max(\|wp\|, \|up\|)$ . A partition of space into a finite set of disjoint regions is said to have the weak uniqueness property if each of its regions has the weak uniqueness property.

Similar to the diagonal partition, we represent an exploded view of a *weak diagonal partition* in Fig.2(a). This can be thought as if we “perturb” each half-line of the diagonal partition clockwise into a wedge. We prove the following theorem similar to Robins and Salowe’s Lemma 6 [20]. The only difference is that equality can occur in our partition.



**Fig. 2.** The weak diagonal partition

**Theorem 1.** Given a point  $p$  in the rectilinear plane, each region of the weak diagonal partition has the weak uniqueness property. (Due to the paper length, we do not give the proof here.)

Note that if two points are in a weak diagonal partition of another point (we call it the center), the two points cannot form an angle  $\geq 90$  degrees with the center. This property will be used to prove several other properties later.

Now we consider a graph on a point set in the rectilinear plane. We do a weak diagonal partition at each of its vertices. If the center is connected to only the nearest neighbor in each unique region, we call this graph a *weak unique graph*. It is obvious that the maximum vertex degree in a weak unique graph is 4. Then we can define BNG.

**Definition 3.** The BNG is a connected sub-graph of the Delaunay triangulation that is a weak unique graph.

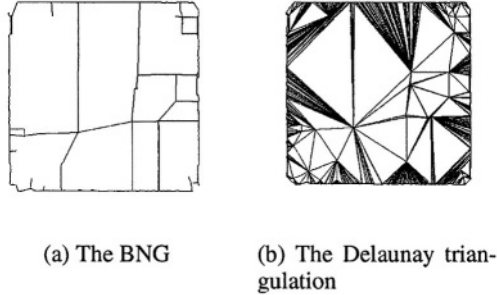
Now we discuss some properties of the BNG.

**Property 1.** The vertex degree of the BNG is bounded by 4.

This is obvious. So we are guaranteed to find a 4-BDRMST on it if it contains one. We will prove in Section 4 that the BNG actually contains at least one MST. We have the following property.

**Property 2.** The BNG has at most  $2n$  edges, where  $n$  is the number of vertices in the BNG.

On uniformly distributed random point sets, the number of edges is normally below  $1.5n$ . On a point set extracted from a real circuit design, the number of edges is



**Fig. 3.** A bounded-degree neighborhood graph and the corresponding Delaunay triangulation on a point set of size 830 extracted from a real circuit

usually smaller. Because the MST algorithms on graphs usually run in time  $O(m \log n)$ , where  $m$  is number of edges, this property leads to a significant speed up of the MST construction. Fig.3 compares a BNG and a Delaunay triangulation on the same point set. The Delaunay triangulation usually has about  $3n$  edges. Since the BNG is a sub-graph of the Delaunay triangulation, we have the following property.

**Property 3.** *The BNG is planar.*

Thus an  $O(n)$  MST algorithm [2] can be applied on it to achieve great efficiency.

### 3 The Algorithm and Its Complexity

We construct the BNG by *pruning* a Delaunay triangulation. At each point  $p$ , we do a weak diagonal partition. In each region, if there are more than one edge adjacent to  $p$ , we say that there is a *conflict*. We then delete longer edges, leaving the shortest one. When there are two edges of equal length, we delete the edge with a larger Euclidean length.

At last, some Delaunay edges remain. The resulting sub-graph is the BNG. We call this process a *uniqueness pruning process*. The BNG can be constructed by putting a fully constructed Delaunay triangulation through the uniqueness pruning process, but we need not construct a Delaunay triangulation in full before we start the BNG construction. We can prune the Delaunay edges on-the-fly, as described in detail below.

We have a two-dimensional pointer array PART of size  $4n$ . We record the shortest edge in region  $r$  of the point  $p$  in PART [p][r]. At the beginning of the Delaunay triangulation, each element of the array is a null pointer. As soon as the Delaunay triangulation process reports an edge, we examine it with a procedure DIAGONAL\_TEST. The procedure tests both end points of the edge. Centered at an end point  $p$ , we do a weak diagonal partition. If there is already an edge recorded in PART [p][r], a conflict occurs. We then delete the longer between the two: the edge recorded in PART [p][r] and the edge we are testing. And we record the shorter in PART [p][r]. In case of ties we delete the edge with a larger Euclidean length. Only the edges not deleted at the end are included in the BNG. Our pruning procedure DIAGONAL\_TEST can be specified formally as follows.

We can integrate this procedure into any  $L1$  Delaunay triangulation algorithm [15, 21]. Due to the advantages of the plane-sweep algorithm, we choose the algorithm of Shute et al. [21]. The `DIAGONAL_TEST` procedure runs in constant time for each Delaunay edge and takes  $O(m)$  time as a whole, where  $m$  is the number of Delaunay edges. Since the Delaunay triangulation algorithm takes  $O(n \log n)$  time, the overall time complexity is  $O(n \log n)$ . We need  $O(n)$  space to record the shortest edges in the unique regions and the Delaunay triangulation algorithm also takes  $O(n)$  space. So the overall space requirement is  $O(n)$ . A careful study reveals that since our algorithm only needs an array of pointers of size  $4n$ , and the Delaunay edges are tested before they are stored, many Delaunay edges are left out of the storage. Since a Delaunay edge takes much more space than a pointer, we actually need less space than the Delaunay triangulation algorithm.

We can apply any RMST algorithm on the resulting BNG to get an 4-BDRMST. Since the BNG is planar, we can apply an efficient  $O(n)$  algorithm as described in [2] on it. Thus we get an  $O(n \log n)$  time algorithm to compute the 4-BDRMST.

**Algorithm 1** `DIAGONAL_TEST`

**Require:** A two dimensional array `PART` of size  $4n$ . The input is an Delaunay edge  $e$ ;

```

for both of the end point  $s$  of  $e$  do
  if  $e$  is already deleted then
    Return;
  end if
   $r \leftarrow$  the region number of  $s$  in which  $e$  lies;
  if PART [ $s$ ][ $r$ ] = NULL then
    PART [ $s$ ][ $r$ ]  $\leftarrow e$ ;
    else if WEIGHT( $e$ ) > WEIGHT(PART [ $s$ ][ $r$ ]) then
      Delete( $e$ );
    else if WEIGHT( $e$ ) < WEIGHT(PART [ $s$ ][ $r$ ]) then
      Delete(PART [ $s$ ][ $r$ ]);
      PART [ $s$ ][ $r$ ]  $\leftarrow e$ ;
    else
      Between the choice of  $e$  and PART [ $s$ ][ $r$ ], delete the edge
      with a longer Euclidean length, store the other in PART
      [ $s$ ][ $r$ ];
    end if
  end for

```

## 4 Proof of Algorithm Correctness

The main result of this section is Theorem 2, showing that the BNG contains at least one 4-BDRMST. We will derive it from several smaller facts.

It is obvious the uniqueness pruning process will not delete a potential RMST edge in a triangular cycle, but we are not convinced yet that the pruning process does not incidentally delete a cut edge from the Delaunay graph.

The uniqueness pruning process can process other graphs as well, but not every graph can be pruned in this manner to be used for RMST construction. In the weak



diagonal partition of Fig.2(b), without loss of generality, we assume that  $\|up\| \leq \|wp\|$ . By the weak uniqueness property we have  $\|wu\| \leq \|wp\|$ . Then the pruning process might delete the edge  $wp$ . It can cause no harm if we are constructing an RMST on a complete graph, since there is always an edge between  $u$  and  $w$ . But this is not always true for other graphs. What if  $wp$  is the only edge between  $w$  and  $p$ ? If this is the case and we simply delete  $wp$  from the graph, then  $w$  might be isolated and there can be no spanning trees in the graph. We call such a configuration that there are only two edges between three points a *broken triangle*. If there is a broken triangle in a unique region, the uniqueness pruning process may cut the graph into components. We will show that this cannot happen if we use the Delaunay graph.

**Lemma 1.** *The uniqueness pruning process cannot isolate a point from a broken triangle of the L1 Delaunay triangulation.*

**Proof.** In the Delaunay triangulation, the only case when broken triangles appear is when the bisector of  $up$  and  $wp$  do not intersect in a Voronoi point (See Fig.4. we also depicted the weak diagonal partition on  $p$  in the figure.) This situation only happens on the periphery of the Delaunay triangulation. To form parallel bisectors,  $pu$  and  $pw$  must form an angle  $\geq 90$  degrees, this is a fundamental characteristics of the bisectors in the L1 metric. Thus  $w$  and  $u$  both cannot lie in a region of a weak diagonal partition centered at  $p$ . Thus in a Delaunay triangulation there can be no conflicts between the edges of the broken triangles and no edges will be deleted. So the pruning process cannot isolate a point from a broken triangle.

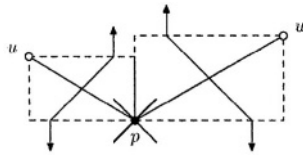


Fig. 4. Parallel bisectors in L1 Delaunay triangulation

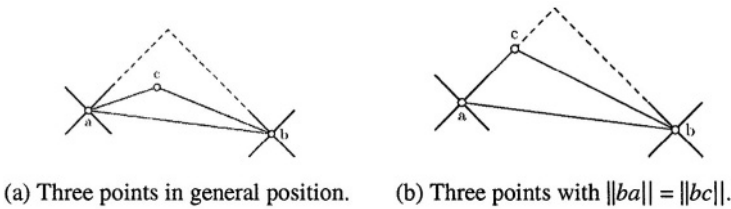


Fig. 5. Two conflicts in one triangle

Next we show that the pruning process cannot isolate a point from a full triangle.

**Lemma 2.** *The uniqueness pruning process cannot isolate a point from a triangle in any connected graph.*

**Proof.** To isolate a point from a triangle, we must delete two edges. Now we will show that the pruning process can delete only one edge from a triangle. Consider the situation depicted in Fig.5(a). First we assume that  $ac$  and  $bc$  are both in general position, that is, they are not on a half-line. So they cannot be of equal length as  $ab$ . With-

out losing generality, we assume that  $ac$  and  $ab$  lie in some weak diagonal region with respect to  $a$ , and  $bc$  and  $ba$  lie in some weak diagonal region with respect to  $b$ . Here is two con-flicts in the triangle  $\Delta abc$ . Note that the angle  $\angle acb$  is larger than 90 degrees and edges  $ca$  and  $cb$  cannot be both inside any weak diagonal region. So there can be at most two conflicts and  $ab$  must be the longest edge in the triangle  $\Delta abc$ . In both conflicts, only  $ab$  will be considered for deletion by the pruning process and no points will be isolated.

Now we consider the case of ties, as in Fig.5(b). We have  $\|ba\| = \|bc\|$ . If we choose to delete  $bc$  when pruning on  $b$ , later the process would delete  $ab$  when pruning on  $a$ , and  $b$  will be isolated. So we must choose to delete  $ba$  when pruning on  $b$ . Remember that in the pruning process, we choose to delete the edge opposite to the obtuse angle in case of ties. So we will choose to delete  $ab$ , and no points can be isolated.

**Theorem 2.** *The bounded-degree neighborhood graph contains a 4-BDRMST.*

**Proof.** Since an  $L1$  Delaunay triangulation contains only broken triangles and full triangles, following Lemmas 1–2, we can see that an  $L1$  Delaunay triangulation is still connected after the uniqueness pruning process. And according to the cycle property of the RMST we can see that the uniqueness pruning process preserves the RMST contained in a  $L1$  Delaunay triangulation.

Since the BNG is constructed by pruning the rectilinear Delaunay triangulation, it contains at least one RMST. Consider our weak diagonal partition scheme, this RMST must be a 4-BDRMST.

## 5 Experimental Results and Discussions

We have implemented the algorithm using C programming language. It uses Shute et al.'s plane-sweep Delaunay triangulation algorithm to feed the Delaunay edges to the pruning process, resulting in a BNG. We then use Kruskal's algorithm [14] to find an RMST on it. All generated RMSTs are correct and has the maximum degree bounded by 4.

### 5.1 Typical Existing Algorithms

We compare our algorithm to several other RMST algorithms. Among the programs, only Robins's can generate 4-BDRMSTs, others can only output ordinary RMSTs. All programs are implemented using the C programming language, compiled with the same GCC compiler and run on the same machine: a Sun Fire 880 workstation with 8GB RAM.

- Prim's algorithm [18]. We use the implicit complete graph on the point set as the input to Prim's algorithm. We did not use Kruskal's algorithm because it needs to sort the  $O(n^2)$  edges at the beginning and is impractical for large inputs.
- An  $O(n \log n)$  time algorithm that first computes octant nearest neighbors for each point using the divide-and-conquer algorithm of Guibas and Stolfi [9], then finds an RMST on this graph using Prim's algorithm.

- A RMST code by Dr. L. Scheffer, combining Prim’s algorithm with on-the-fly computation of octant nearest neighbors via quad-tree based rectangular range searches.
- A 4-BDRMST algorithm implemented by Gabriel Robins for the Batched 1-Steiner Heuristic as described in [8]. We also include a version of our program that construct an ordinary RMST directly on the Delaunay triangulation.

**Table 1.** Running time on random point sets

Input	Prim	Guibas	Scheffer	Robins	Delaunay	BNG
2000	0.16s	0.02s	0.05s	0.06s	0.03s	0.03s
4000	0.63s	0.05s	0.12s	0.29s	0.06s	0.06s
8000	2.73s	0.14s	0.35s	1.45s	0.13s	0.11s
10000	5.55s	0.19s	0.50s	2.81s	0.16s	0.14s
20000	33.63s	0.47s	1.46s	18.61s	0.36s	0.29s
40000	154.16s	1.16s	4.05s	80.97s	0.78s	0.64s
80000	-	2.74s	9.24s	329.43s	1.75s	1.46s
100000	-	3.59s	19.23s	518.10s	2.30s	1.91s

**Table 2.** Running time on point sets extracted from real circuit designs

Input	Prim	Guibas	Scheffer	Robins	Delaunay	BNG
337	<0.01s	<0.01s	<0.01s	<0.01s	<0.01s	<0.01s
830	0.01s	<0.01s	0.02s	0.01s	<0.01s	<0.01s
1944	0.10s	0.01s	0.03s	0.05s	0.02s	0.01s
2437	0.16s	0.02s	0.04s	0.08s	0.03s	0.03s
2676	0.23s	0.03s	0.05s	0.11s	0.04s	0.03s
12052	6.33s	0.20s	0.33s	80.97s	0.21s	0.18s
22373	44.99s	0.53s	0.72s	25.43s	0.43s	0.35s
34728	102.52s	0.76s	1.08s	62.08s	0.71s	0.58s

**Table 3.** Comparison of edge numbers for Delaunay triangulation and BNG on random point sets

Input	Delaunay	BNG
2000	5842 (2.92n)	2889 (1.44n)
4000	11777 (2.94n)	5779 (1.44n)
8000	23679 (2.95n)	11621 (1.45n)
10000	29637 (2.96n)	14501 (1.45n)
20000	59505 (2.98n)	29140 (1.46n)
40000	119282 (2.98n)	58227 (1.46n)
80000	239003 (2.99n)	116572 (1.45n)
100000	298894 (2.99n)	145924 (1.46n)

**Table 4.** Comparison of edge numbers for Delaunay triangulation and BNG on point sets extracted from real circuits

Input	Delaunay	BNG
337	954 (2.83n)	409 (1.21n)
830	1931 (2.32n)	884 (1.07n)
1944	5717 (2.94n)	2890 (1.48n)
2437	7181 (2.94n)	3681 (1.51n)
2676	7915 (2.95n)	3861 (1.44n)
12052	35920 (2.98n)	18413 (1.53n)
22373	66730 (2.98n)	33012 (1.48n)
34728	103759 (2.99n)	51698 (1.49n)

## 5.2 Discussions

Table 1 shows the results of running the programs on input data using randomly generated points, with sizes ranging from 2000 to 100000. Note that the timing results for small input sets may be inaccurate due to the system timer’s resolution. The last rows are results from large random inputs, but in practice we seldom meet input sets larger

than 30000. We list them there only to illustrate the growth of running time of our algorithm.

Table 2 shows the running time on point sets extracted from real circuit designs. The data shows that our program is hundreds of times faster than Robins's with a medium size point set. When compared with ordinary RMST algorithms, our program takes only about half the time as the program that implements Guibas and Stolfi's  $O(n \log n)$  algorithm, which shows that our algorithm has very small hidden constants. Our 4-BDRMST program also runs faster than ordinary RMST program based on Delaunay triangulation. Thus our algorithm shows its advantages in both BDRMSTs and ordinary RMSTs.

Table 3 compares the number of edges of the rectilinear Delaunay triangulation and the BNG on random point sets.

Table 4 compares the number of edges of the rectilinear Delaunay triangulation and the BNG on point sets extracted from real circuit designs. The data shows that the BNG has much fewer edges than the Delaunay triangulation. This results in faster RMST construction while using less space.

## 6 Conclusions

We have proposed an  $O(n \log n)$  algorithm for 4-BDRMST construction based on the BNG. To our knowledge, this is the first 4-BDRMST algorithm of such a time complexity. Experimental results show that our algorithm is significantly faster than the existing 4-BDRMST algorithms. It is even faster than the best-known algorithms that compute ordinary RMSTs. It is helpful for high performance physical design.

## References

1. T.Barrera, J.Griffith, et al. Toward a Steiner engine: Enhanced serial and parallel implementations of the iterated 1-steiner algorithm. In *Proc. GLSVLSI*, pp.90, MI, 1993.
2. D.Cherton and R.E.Tarjan. Finding minimum spanning trees. *SIAM Journal on Computing*, 5(4): 724–742, Dec. 1976.
3. S.P.Fekete, S.Khuller, M.Klemmstein, et al. A network-flow technique for finding low-weight bounded-degree trees. In *Proc. of International Integer Programming and Combinatorial Optimization IPCO Conference. Jun 1996. Vancouver, Canada*, volume 1084, pp. 105–117.
4. G.N.Fredrickson. Data structures for on-line updating of minimum spanning trees. *SIAM Journal on Computing*, 14:781–798, 1985.
5. M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP Completeness*. W. H. Freeman, 1979.
6. G. Georgakopoulos and C. H. Papadimitriou. The 1-steiner tree problem. *Journal of Algorithms*, 8: pp.122–130,1987.
7. R.L.Graham and P.Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7:43–57, 1985.
8. J.Griffith, G.Robins, J.S.Salowe et al. Closing the gap: Near-optimal steiner trees in polynomial time. *IEEE Trans. on CAD*, 13(11): 1351–1365, Nov. 1994.
9. L.J.Guibas and J.Stolfi. On computing all north-east nearest neighbors in the  $L_1$  metric. *Information Processing Letters*, 17, 1983.

10. F.K.Hwang. On steiner minimal trees with rectilinear distance. *SIAM journal on Applied Mathematics*, 30:104–114, 1976.
11. F.K.Hwang. An  $O(n \log n)$  algorithm for rectilinear minimal spanning trees. *Journal of the ACM*, 26(2):177–182, Apr. 1979.
12. A.B.Kahng and G.Robins. A new class of iterated Steiner tree heuristics with good performance. *IEEE trans. Computer-Aided Design*, 11:893–902, 1992.
13. S.Khuller, B.Raghavachari, and N.Young. Low degree spanning trees of small weight. *SIAM Journal on Computing*, 25(2):355–368, 1996.
14. M.Kruskal. On the shortest spanning subtree of a graph, and the traveling salesman problem. *Proc. Amer. Math Soc.*, 7:48–50, 1956.
15. D.T.Lee and C.K.Wong. Voronoi diagrams in  $L_1(L_1)$  metric with 2-dimensional storage applications. *SIAM Journal of Computing*, 9:200–211, 1980.
16. C.H.Papadimitriou and U.V.Vazirani. On two geometric problems relating to the traveling salesman problem. *Journal of Algorithms*, 5:231–246, 1984.
17. B.T.Preas and M.J.Lorenzetti. *Physical Design Automation of VLSI Systems*. Benjamin/Cummings, Menlo Park, CA, 1988.
18. A.Prim. Shortest connecting networks and some generalizations. *Bell Syst. Tech J.*, 36:1389–1401, 1957.
19. R.Ravi and J.onemann. A matter of degree: Improved approximation algorithms for degree-bounded MSTs. In *Proc. ACM Symposium on Theory of Computing*, 2000.
20. G.Robins and J.S.Salowe. Low-degree minimum spanning tree. *Discrete and Computational Geometry*, 14:151–165, Sept. 1995.
21. G.M.Shute, L.L.Deneen, and C.D.Thomborson. An  $O(n \log n)$  plane-sweep algorithm for  $L_1$  and  $L_1$  delaunay triangulations. *Algorithmica*, 6:207–221, 1991.
22. A.C.-C.Yao. On constructing minimum spanning trees in  $k$ -dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, Nov. 1982.
23. H.Zhou, N.Shenoy, and W.Nicholls. Efficient spanning tree construction without delaunay triangulation. *Information Processing Letters*, 81(5), 2002.

# Wirelength Reduction Using 3-D Physical Design

Idris Kaya, Silke Salewski, Markus Olbrich, and Erich Barke

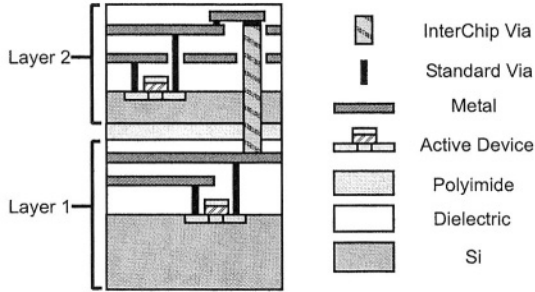
University of Hannover, Institute of Microelectronic Systems,  
Appelstr. 4, 30167 Hannover, Germany  
{kaya,salewski,olbrich,barke}@ims.uni-hannover.de

**Abstract.** While the feature size of integrated circuits decreases with every technology node, the impact of interconnect delay on the total delay increases. Thus, minimizing the wirelength becomes one of the most important tasks in physical design of high performance circuits. In this paper we present a 3-D design flow for vertical integrated circuits. Our floorplanning and placement results show reductions of both total wirelength and lengths of the longest nets up to 50%. Thus, we demonstrate the capability of significant interconnect delay reduction using vertical integration.

## 1 Introduction

With increasing circuit complexities and decreasing dimensions of electronic devices in integrated circuits interconnect delay becomes a critical part of the total delay. Repeater insertion can basically handle this problem but only at the expense of additional active area and power consumption. In the past the challenging demands for new interconnect and process technologies were met by the introduction of new materials as copper and low  $\kappa$  dielectrics. However, for the long term it is believed that only new process technologies will deliver the solution, amongst them the vertical integration [1]. Multiple layers of active devices are stacked on top of each other and are connected by short vertical interconnects. An overview on existing vertical integration technologies is presented in [2]. In this paper, we concentrate on a technology that is fully CMOS compatible [3] and is known as InterChip Via (ICV) technology. The basic architecture is shown in Figure 1 [4]. ICVs ensure vertical interconnections between chip layers and connect the topmost metal layer of the bottom chip layer with the topmost metal layer of the next chip layer. Therefore, it is impossible to place active elements at ICV positions. Investigations show that published ICV-dimensions are commensurate with average dimensions of standard cells. In consequence, ICVs can be treated like standard cells during placement. However, ICVs differ electrically from standard vias as published in [3]. We consider this by a special wirelength contribution of the ICVs.

Vertical integration is not only a research area for academia but for industry as well [5]. Nevertheless, commercial EDA tools for 3-D design are not available



**Fig. 1.** InterChip Via technology

at all. In academia only few investigations have been done on floorplanning, placement or routing [6,7,8,9,10,11,12].

In this paper we present a floorplanning and a placement tool for vertical integration. Because vertical interconnects are considered during placement, routing can be done in 2-D with a common router. Our experimental results demonstrate the efficiency of our design flow and show significant wirelength and longest netlength reduction for benchmark circuits using vertical integration.

## 2 3-D Floorplanning

### 2.1 Discrete 3-D Floorplan Representation

Vertical integration technologies build 3-D ICs by stacking multiple layers of active devices. Therefore, modules are placed on discrete layers and the related floorplan has to represent this *discrete* 3-D structure. In consequence, we use a 2-D data structure for each layer. For the 2-D data structure we choose slicing trees (which are equivalent to polish expressions) [13] because they are the simplest and smallest structures for floorplanning. Moreover, since the problem is getting more complex with the new degree of freedom, the reduction of the solution space caused by the slicing structure is appropriate. Hence, the discrete 3-D floorplan is represented by a forest of slicing trees.

### 2.2 A Genetic Algorithm for 3-D Slicing Floorplanning

We choose a genetic algorithm [14] because this is a very flexible algorithm. It preserves all possibilities for solving problems coming up in this new field. A genetic algorithm mimics the natural process of evolution. In general, it evolves a gene string by crossover, mutation and selection operations. This stochastic method can be used to solve different kinds of problems and has been applied to the floorplanning problem in several previous approaches [15,16,17]. Since our 3-D floorplan is not represented by a gene string, we have to customize the mutation and crossover operations to be suitable for the chosen data representation.

**Mutation.** We propose four different mutation operations: change a cut direction, rotate a subtree, reshape a subtree and interchange two subtrees. All of these make incremental changes to a randomly chosen individual. The first three operate on a single slicing tree. Therefore, they are not affected by the extension to a discrete 3-D floorplan. Moreover, they retain the partitioning of the modules to layers, whereas the interchange of two subtrees can involve subtrees from two different slicing trees or layers, respectively. Hence, this operation optimizes the assignment of modules to layers and thus is indispensable in our algorithm.

**Crossover.** The crossover operation creates two offsprings by combining portions of two parent individuals. This is done by interchanging subtrees between the two parent individuals. To obtain feasible floorplans the interchanged subtrees need to have an identical set of leaves.

### 2.3 Cost Function

The individuals in a population are selected for crossover, mutation and reproduction depending on their fitness that represents their quality accordingly to the cost function

$$cost = \alpha \cdot area(FP) + \beta \cdot wirelength(FP).$$

The parameters  $\alpha$  and  $\beta$  allow a balancing of the terms for wirelength and area, which is described in the following. The area of a 2-D floorplan is given by the rectangle enclosing all modules. The area of each layer in a 3-D IC can be defined equivalently. Due to restrictions in the process technology the area of a 3-D IC is not determined by the sum of these areas. Although each layer of active devices can be processed independently from the others when using ICV technology [3] the sequential bonding process of the wafers prohibits an increase in area from each layer to its overlying layer. Therefore, we define the total area of a 3-D IC by

$$area = \sum_{i=1}^{\#layers} (\max_{j=1\dots i} \{W_j\} \cdot \max_{j=1\dots i} \{H_j\})$$

with  $W_j$  and  $H_j$  denoting width and height of the enclosing rectangle on layer  $j$ , respectively.

The wirelength of each net is estimated by the bounding box half perimeter, the total wirelength is therefore given by

$$wirelength = \sum_{i=1}^{\#nets} \left[ (\max_{j \in M_i} \{x_j\} - \min_{j \in M_i} \{x_j\}) + (\max_{j \in M_i} \{y_j\} - \min_{j \in M_i} \{y_j\}) \right. \\ \left. + k \cdot ((\max_{j \in M_i} \{z_j\} - \min_{j \in M_i} \{z_j\})) \right]$$

where  $M_i$  is the set of modules connected to net  $i$ . The coefficient  $k$  rates the properties of the inter-layer vias. Comparing the electrical properties of ICVs to those of standard interconnects and assuming the  $z$ -position of a module is set to the ordinal number of its corresponding layer leads to  $k = 20$ .



### 2.4 Experimental Results

We applied our algorithm to the well-known MCNC benchmark circuits to figure out the performance of our algorithm. All modules are assumed to be soft with their aspect ratios allowed to vary from 0.1 to 10. The pin positions given in the benchmarks are scaled to fit the actual shapes of the modules. In all experiments we used 5000 generations and a population size of 100 for our genetic algorithm. This leads to runtimes of a few minutes on a SUN Ultra 10 with 400 MHz. For each benchmark circuit we performed 100 runs and determined the average values. In Table 1 the averaged results are compared to results presented in recent publications [18,19,20]. Although the main application of our algorithm is the floorplanning on multiple layers, the outcome for conventional 2-D floorplans is appreciable, since the results are comparable to those of algorithms that are particularly implemented for the 2-D case.

To determine the efficiency of our algorithm when floorplanning is applied to multiple layers, we compare our results to those published in [8], though we can hardly assess those values because there is neither a statement on runtimes nor on statistical bases. Therefore, besides average results, Table 2 shows our best values out of 100 runs. Whereas our average results are competitive to those of [8], our best cases outperform [8] apart from one exception. Both approaches demonstrate a significant wirelength reduction.

Unfortunately, in [8] only results for the largest benchmark circuits ami33, ami49, and payout are presented, and floorplanning is solely performed for one and two layers. In Table 3 we present our results for the complete set of benchmark circuits utilizing up to five layers. We observed a significant wirelength reduction with an increasing number of layers while the area remains almost constant.

**Table 1.** Comparison with 2-D floorplanning tools.

	3-D Floorplanner		Guo [18]		Murata [19]		Hong [20]	
	Area	WL	Area	WL	Area	WL	Area	WL
apte	48.6	339	63.3	330	46.6	421	47.9	203
xerox	21.1	402	23.8	478	19.5	534	20.3	504
hp	9.26	120	9.91	167	8.87	157	NA	NA
ami33	1.24	62.3	1.34	50.9	1.16	51.3	1.23	56.7
ami49	44.0	882	45.5	673	36.0	850	39.4	870

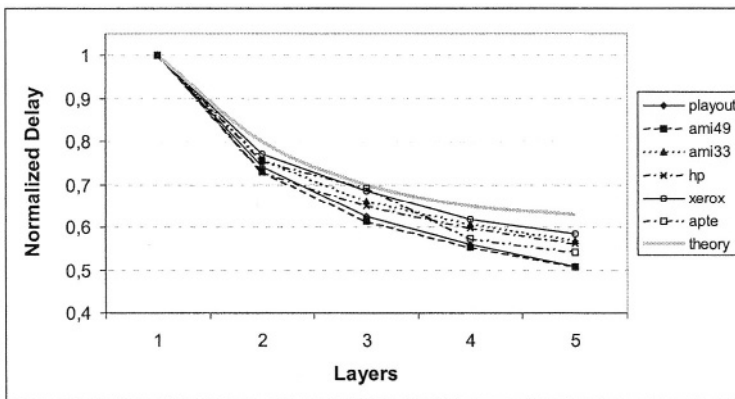
**Table 2.** Comparison of discrete 3-D floorplanning tools

	ami33				ami49				payout			
	Area		WL		Area		WL		Area		WL	
	2-D	3-D	2-D	3-D	2-D	3-D	2-D	3-D	2-D	3-D	2-D	3-D
Average	1.237	1.206	62.32	47.85	44.04	43.97	881.5	641.6	104.3	106.8	5387	3995
Best	1.197	1.167	52.83	41.59	41.50	42.06	731.0	561.7	99.54	103.9	4408	3247
Deng [8]	1.316	1.282	81.35	64.71	44.10	43.20	894.1	625.8	120.8	117.0	5166	3163

**Table 3.** Results on MCNC benchmarks for multi-layer floorplanning

	apte		xerox		hp		ami33		ami49		payout	
	Area	WL	Area	WL	Area	WL	Area	WL	Area	WL	Area	WL
<b>1 layer</b>	48.6	339	21.1	402	9.26	120	1.24	62.2	44.0	882	104	5.39
<b>2 layers</b>	48.6	262	21.1	314	9.42	94.0	1.21	47.9	44.0	642	107	4.00
<b>3 layers</b>	48.4	245	20.8	284	9.47	89.0	1.20	39.0	43.1	549	106	3.37
<b>4 layers</b>	47.6	213	20.4	265	9.40	85.0	1.19	36.2	42.9	487	107	3.03
<b>5 layers</b>	47.3	208	20.2	261	9.29	84.2	1.18	34.7	42.9	444	107	2.81

To survey the benefit of increasing the number of layers in more detail, we also investigated the effect of vertical integration on the longest nets. Floorplanning regards the global wires of the design that mostly appear in the critical paths. Thus, the delay of the longest net strongly affects the performance of the chip. In Figure 2 we show the normalized delay of the longest net as a function of number of layers for all benchmarks. For simplification, we neglected the impact of interconnect capacitances on interconnect delay. Moreover, all wires are rated by the same resistance per unit length since they are all global wires. Therefore, the delay is assumed to be proportional to wirelength. Figure 2 shows that for all benchmark circuits the delay is decreased by vertical integration. However, the reduction becomes smaller with further increase in number of layers. This outcome confirms the prediction of theoretical considerations, [2] which is also shown in Figure 2, for practical applications.

**Fig. 2.** Normalized delay as a function of active devices layers

### 3 3-D Placement

In the 2-D case we consider cells as standard-cells with same height and different width. In our 3-D standard-cell approach we consider standard-cells to be cuboids having identical height ( $y$ -direction), identical thickness ( $z$ -direction) and variable width ( $x$ -direction). The entire placement volume has width  $W$ , height  $H$  and thickness  $T$ . We investigate vertical integration with several layers. For all designs, the total placement area remains constant.

#### 3.1 3-D Force Directed Placement

Force directed placement is an algorithm for placing cells or modules. Basics are described in [21]. In this section we extend the standard 2-D algorithm to the third dimension. In a standard-cell approach cells are modelled as masses and connections between them as springs that lead to attracting forces between connected cells.

The objective function corresponds to the potential energy of the mass-spring system and can be written in matrix notation as follows:

$$E_{cost}(\mathbf{p}) = \frac{1}{2} \cdot \mathbf{p}^T \cdot \underline{\mathbf{C}} \cdot \mathbf{p} + \mathbf{d}^T \cdot \mathbf{p} + \text{const.} \quad (1)$$

$\mathbf{p}$  is the placement vector. The rules for assigning  $\underline{\mathbf{C}}$  and  $\mathbf{d}$  are shown in [21], but for 3-D we have to consider the  $z$ -related part.

The minimum of (1) is calculated by solving the following matrix equation:

$$\underline{\mathbf{C}} \cdot \mathbf{p} + \mathbf{d} = 0 \quad (2)$$

Solving (2) leads to a placement with a global minimum of the objective function but also with many overlaps. We aim to achieve a regular distribution of all cells in the layout volume. That is the reason for introducing a repelling force  $\mathbf{e}$ . Instead of (2) we now solve (3) with numerical solution methods.

$$\underline{\mathbf{C}} \cdot \mathbf{p} + \mathbf{d} + \mathbf{e} = 0 \quad (3)$$

The vector  $\mathbf{e}$  is calculated in step  $k$  as follows:

$$\mathbf{e}_k = \mathbf{e}_{k-1} + \mathbf{f}(x, y, z) \quad (4)$$

$$\mathbf{f}(x, y, z) = \frac{c_1}{2\pi} \iiint_{-\infty}^{\infty} D(x', y', z') \frac{\mathbf{r} - \mathbf{r}'}{|\mathbf{r} - \mathbf{r}'|^3} dx' dy' dz' \quad (5)$$

The second term differs in the exponent in the denominator from the 2-D function.  $D(x, y, z)$  describes the local density at point  $(x, y, z)$  (refer to [21]).

Solving (5) directly is too time consuming. However, (5) is a convolution which can be processed conveniently by transforming the problem into the frequency domain, multiplying and transforming the result back into the original

domain. In this way repelling force fields can be obtained quickly using FFT. From that, repelling forces for each cell can be calculated. Solving (3) leads to the next placement vector.

Our placer can deal with two different optimization criteria, the reduction of total wirelength (Mode 1) and number of ICVs (Mode 2), respectively.

The first mode focuses on small wirelength without considering vertical interconnects. Our 3-D algorithm ensures the reduction of the total wirelength automatically.

Currently, ICVs have a non-negligible area and different electrical characteristics from vias between metal layers. That is the reason for considering the number of ICVs in the second mode. The aim is to get a small number of ICVs. The  $z$ -dimension is weighted excessively by enlarging the thickness of all elements to ensure this objective. Then, the algorithm tends to place connected standard cells in one layer instead of placing them in different layers. Due to the fact that a regular distribution of standard cells is another major objective, a certain number of ICVs remains necessary. Comparisons of the electrical properties of ICVs to those of standard interconnects lead to  $20\ \mu\text{m}$  for the contribution of ICVs to total wirelength. In Mode 2 we ensure that no problems in placing ICVs can emerge because we place standard cells and ICVs simultaneously. If it is conceivable that an ICV is needed between two cells, our placer adds an ICV and places it in the placement volume. After the insertion of ICVs the placement is carried on for some iterations until the placement is checked again for possible deletion or necessary insertion of ICVs.

The legalized placement is attained by mapping standard cells and ICVs to the nearest layer in the first step. Afterwards, all cells and ICVs are assigned to standard cell rows. Then they are sorted in  $x$ -dimension and placed in rows. In a final step an intra-row optimization is performed.

### 3.2 Experimental Results

We use the MCNC benchmarks to show the efficiency of our algorithm. For comparisons between 2-D and 3-D placements, each of them fully legalized and design-rule compliant, we use the total wirelength calculated with the bounding box estimation. Due to the fact that base areas are not explicitly given for all benchmarks we only present relative wirelength reductions, always referring to the 2-D case.

Mode 1 shows remarkable reduction of total wirelength. Total wirelength reductions up to 47% (*biomed*, 4 layers) can be obtained with our force-directed discrete 3-D placer in Mode 1. Refer to Table 4 for other benchmark results with up to 4 layers. Comparisons show that our 3-D placer outperforms published wirelength reductions for two layers in [7] (refer to Table 4).

Reduced total wirelength means better timing behavior in general. But longest wires are most critical. It is essential that longest wires are as short as possible to reduce interconnect delay and to meet timing constraints. Therefore, we analyse the longest nets in 2-D and 3-D. Results show that average longest netlength reductions between 26% (2 layers) and 47% (4 layers) are achievable.

**Table 4.** Total wirelength reduction (TWLR) and longest netlength reduction (LNLR) for MCNC benchmarks without considering ICVs (Mode 1)

Design	# Cells	Case	Deng [7]	2 Layers	3 Layers	4 Layers
primary1	833	TWLR	23.0%	22.6%	29.7%	37.4%
		LNLR	20.3%	16.2%	30.0%	39.2%
struct	1952	TWLR	14.1%	21.2%	36.2%	40.5%
		LNLR	31.7%	30.2%	41.9%	50.5%
primary2	3014	TWLR	16.0%	18.3%	36.6%	42.1%
		LNLR	21.3%	16.2%	30.1%	39.0%
industry1	3085	TWLR	16.5%	24.5%	34.2%	41.9%
		LNLR	28.3%	30.8%	44.3%	51.8%
biomed	6514	TWLR	16.2%	26.8%	37.4%	47.3%
		LNLR	29.2%	30.3%	40.0%	50.2%
industry2	12637	TWLR	09.3%	19.0%	27.4%	32.5%
		LNLR	28.2%	34.0%	46.1%	51.6%
Average		TWLR	15.9%	22.1%	33.6%	40.3%
		LNLR	26.5%	26.3%	38.7%	47.1%

**Table 5.** Number of ICVs, total wirelength reduction (TWLR) and longest netlength reduction (LNLR) for MCNC Benchmarks considering ICVs (Mode2)

Design	# Layers	3-D Placer			hMetis	
		TWLR	LNLR	# ICVs	1% Dev.	10% Dev.
primary1	2	13.9%	16.2%	146	101	79
	3	24.6%	30.0%	304	221	194
	4	32.4%	39.2%	475	384	324
struct	2	07.7%	21.2%	317	268	223
	3	25.7%	39.0%	618	549	452
	4	27.7%	43.0%	921	834	732
primary2	2	08.7%	16.2%	474	248	196
	3	19.0%	30.1%	795	517	488
	4	26.3%	39.0%	1212	1040	970
industry1	2	26.1%	23.8%	515	575	450
	3	35.4%	37.6%	1340	1317	1033
	4	43.1%	44.7%	2243	2096	1738
biomed	2	16.4%	22.4%	2327	1421	1298
	3	28.8%	33.4%	2716	2763	2301
	4	32.6%	42.8%	3343	4247	3767
industry2	2	16.9%	33.0%	1731	800	522
	3	20.4%	36.6%	3302	1608	1380
	4	21.5%	39.0%	4738	3037	2426

3-D integration shows great benefits for reducing delay and meeting timing constraints. Comparisons with published longest netlength reductions for 2 layers [7] show that our 3-D placer is competitive. Our placer shows better results than Deng's [7] for the more important larger designs (*industry1*, *biomed* and *industry2*). Refer to Table 4 for details.

In the second mode wirelength reduction, compared to 2-D, is still significant. The highest wirelength reduction (43%) is obtained for *industry1* in a 4-layer design. This result is not comparable with the wirelength reduction attained in Mode 1 because in Mode 2 standard cells are not distributed regularly in the placement volume. Refer to Table 5 for details. In [6] total wirelength reductions of 7% to 17% aiming at a small number of ICVs are listed for 3-D designs with up to 5 layers. Our placer shows average total wirelength reductions of 17% for 2 layers, 25% for 3 layers and 30% for 4 layers, respectively.

Mode 2 ensures a small number of ICVs. We compare the necessary number of ICVs with the state-of-the-art partitioning tool hMetis with a deviation of 1% and 10%, respectively. Our placer shows numbers for ICVs that are in proximity to hMetis. These results are quite good since our 3-D placer is not designed for partitioning. Comparisons between Mode 1 and 2 show that we achieve average ICV-number reductions of 48%, 57% and 55% for 2, 3, and 4 layers, respectively.

As far as longest nets are concerned, in Mode 2 longest net reductions of 22% for 2 layers, 35% for 3 layers and 41% for 4 layers are achievable. Thus, even when the number of ICVs is minimized, the timing behavior of all designs can be improved remarkably.

## 4 Conclusions

In this paper we present new 3-D physical design tools that consider technological constraints derived from an existing technology for vertical integration. Using several layers of active devices we demonstrate the efficiency of our design flow and observe significant wirelength reductions for all circuits, design stages and optimization modes. Even for 2 layers our experiments show netlength reductions up to 34% and for 4 layers we achieve up to 52% netlength reduction. Therefore, our flow is well suited for high performance and timing critical designs.

## References

1. ITRS: The international technology roadmap for semiconductors. In: <http://public.itrs.net/Files/2001ITRS/Home.htm>. (2001)
2. Banerjee, K., Souri, S.J., Kapur, P., Saraswat, K.C.: 3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. Proc. of the IEEE **89** (2001) 602–633
3. Ramm, P., Bonfert, D., Gieser, H., Haufe, J., Iberl, F., Klumpp, A., Kux, A., Wieland, R.: Interchip Via technology for vertical system integration. In: Proc. IITC. (2001) 160–162
4. Kleiner, M.B., Kühn, S.A., Ramm, P., Weber, W.: Thermal analysis of vertically integrated circuits. In: Proc. IEDM. (1995) 487–490

5. INFCC200208.133e: Infineon presents SOLID, a world first 3D chip integration technology. Press Release (2002)
6. Das, S., Chandrakasan, A., Reif, R.: Design tools for 3-D integrated circuits. In: Proc. ASPDAC. (2003) 53–56
7. Deng, Y., Maly, W.P.: Interconnect characteristics of 2.5-D system integration scheme. In: Proc. ISPD. (2001) 171–175
8. Deng, Y., Maly, W.P.: Physical design of the “2.5D” stacked system. In: Proc. ICCD. (2003) 211–217
9. Enbody, R.J., Lynn, G., Tan, K.H.: Routing the 3-D chip. In: Proc. DAC. (1991) 132–137
10. Goplen, B., Sapatnekar, S.: Efficient thermal placement of standard cells in 3D ICs using a force directed approach. In: Proc. ICCAD. (2003) 86–89
11. Ohmura, M.: An initial placement algorithm for 3-D VLSI. In: Proc. ISCS. Volume 6. (1998) 195–198
12. Tanprasert, T.: An analytical 3-D placement that reserves routing space. In: Proc. ISCAS. Volume 3. (2000) 69–72
13. Wong, D., Liu, C.: A new algorithm for floorplan design. In: Proc. DAC. (1986) 101–107
14. Holland, J.: *Adaption in Natural and Artificial Systems*. The University of Michigan Press (1975)
15. Cohoon, J., Hegde, S., Martin, W., Richards, D.: Floorplan design using distributed genetic algorithms. In: Proc. ICCAD. (1988) 452–455
16. Rebaudengo, M., Reorda, M.: Gallo: A genetic algorithm for floorplan area optimization. *IEEE Trans. CAD* **15** (1996) 943–951
17. Lin, C., Chen, D., Wang, Y.: An efficient genetic algorithm for slicing floorplan area optimization. In: Proc. ISCAS. Volume 2. (2002) 879– 882
18. Guo, P., Cheng, C., Yoshimura, T.: An o-tree representation of non-slicing floorplan and its applications. In: Proc. DAC. (1999) 268–273
19. Murata, H., Kuh, E.: Sequence-pair based placement method for hard/soft/pre-placed modules. In: Proc. ISPD. (1998) 167–172
20. Hong, X., Huang, G., Cai, Y., Gu, J., Dong, S., Cheng, C., Gu, J.: Corner block list: An effective and efficient topological representation of non-slicing floorplan. In: Proc. ICCAD. (2000) 8–12
21. Eisenmann, H., Johannes, F.M.: Generic global placement and floorplanning. In: Proc. DAC. (1998) 269–274

# On Skin Effect in On-Chip Interconnects

Daniel A. Andersson, Lars “J” Svensson, and Per Larsson-Edefors

Department of Computer Engineering, Chalmers University of Technology,  
SE-412 96 Göteborg, Sweden.

{daan,larssv,perla}@ce.chalmers.se

**Abstract.** We investigate the influence of skin effect on the propagation delays of on-chip interconnects. For long wires, designed in the LC regime, on the top metal layer in a contemporary process, we find that the skin effect causes an extra delay by 10%. The impact of the skin effect on delay is furthermore rapidly increasing with increased interconnect width.

## 1 Introduction

The operating frequencies of high-performance digital chips are currently well into the gigahertz region. As a result, it is necessary to take transmission-line properties into account when modeling long interconnects on large chips. These properties affect both signal propagation within the interconnect and signal reflections at its transmitting and receiving ends.

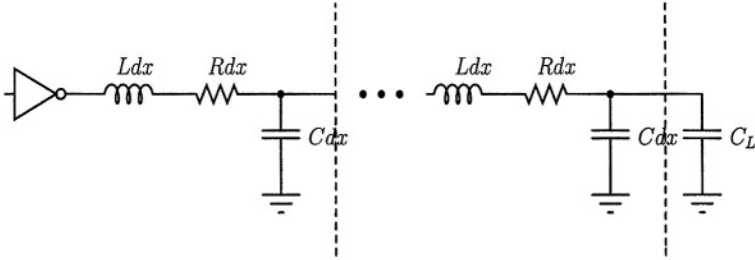
For gigahertz operating frequencies, resistance and inductance per unit length of an interconnect vary with the signal frequency content. This phenomenon is known as the *skin effect* [7]: The current magnitude is large at the surface of the conductor, but smaller in its interior. At high frequencies, therefore, the effective conductor cross-sectional area is reduced.

The skin effect affects the high-frequency transmission-line properties of long conductors. Signal attenuation will be higher due to the increased resistance, and the frequency dependence of resistance and inductance will introduce an additional, frequency-dependent phase shift. Furthermore, signal reflection coefficients at transmitter and receiver will be frequency dependent due to the varying characteristic impedance of the transmission line.

We are not aware of any detailed investigations of skin-effect influence on the performance of on-chip digital interconnects. Printed-circuit-board signal traces are wider than on-chip wires, so the skin effect is noticeable at lower frequencies; its influence has been described by Dermer *et al* [2]. Also, several authors [1,4] have addressed high-frequency on-chip interconnect behavior, with transmission-line properties except the skin effect. Cao *et al* [5] have considered frequency-dependent effects in RLC interconnects, and concluded that frequency-dependent resistance and inductance do not matter for the signal propagation delay; but they present analyses only for a small number of cases, with a very limited parameter range. In contrast, our analysis, as presented in this paper, will show that the skin effect may affect the delay significantly for certain design cases pertaining to on-chip digital interconnects.



The remainder of this paper is organized as follows. We will describe our simulation setup in Section 2. In Section 3, we compare results from our setup with previously published data, without including the skin effect. Next, the skin effect is included in the calculations in Section 4. The discussion in Section 5 concludes the paper.



**Fig. 1.** A transmission line with a driver and a capacitive load,  $C_L$ , which represents a receiver.

## 2 Simulation Setup

### 2.1 Transmission-Line Model

The transmission-line setup used in this paper is shown in Fig. 1. Values for the incremental transmission-line parameters ( $Ldx$ ,  $Rdx$ , and  $Cdx$ ) are derived from the structure shown in Fig. 2. The numerical values correspond to the top metal layer of the  $0.18\text{-}\mu\text{m}$  node of the Berkeley Predictive Technology Model (BPTM) [8]. Transmission line lengths ( $l$ ) and widths ( $w$ ) are varied; in all cases. The spacing ( $s$ ) and the height ( $h$ ) from the ground plane are assumed fixed at  $2\text{ }\mu\text{m}$  and  $8\text{ }\mu\text{m}$ , respectively. The thickness ( $t$ ) of the interconnect is fixed and defined by the  $0.18\text{-}\mu\text{m}$  node used. The actual values for  $Ldx$  and  $Cdx$  are calculated with Eq. 1 and 2, respectively [8].

$$C_{coup} = \epsilon \left( 1.44 \frac{t}{s} \left( \frac{h}{h + 2.059s} \right)^{0.0944} + 0.7428 \left( \frac{w}{w + 1.592s} \right)^{1.144} + \right. \\ \left. + (1.158 \left( \frac{w}{w + 1.874s} \right)^{0.1612} \right) \left( \frac{h}{h + 0.9801s} \right)^{1.179} \right) \quad (1)$$

$$L_{wire} = \frac{2 \cdot \mu \cdot l}{\pi} \left( \ln \left( \frac{2l}{w + t} \right) + 0.5 + 0.2235(w + t) \right) \quad (2)$$

The calculated capacitance represents only the capacitance to adjacent wires. This assumption simplifies our analysis since it isolates the influence of the skin effect.

The driver stage is represented as a voltage source in series with a linear source resistance ( $R_{source}$ ). The receiver stage is represented as a linear capacitance ( $C_L$ ). These linear components are used to model the salient behavior of CMOS inverters. We assume that the driver and the receiver are equally sized, and that the gate capacitance and the output resistance for a minimum-sized inverter are  $1.9\text{ fF}$  and  $8\text{ k}\Omega$ , respectively [1] at a  $0.18\text{-}\mu\text{m}$  technology.

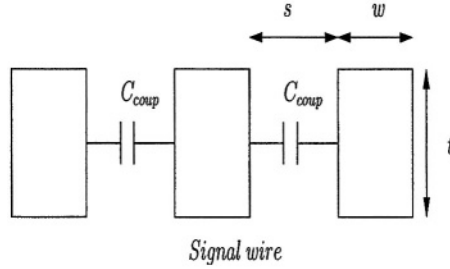


Fig. 2. The structure of the case we are simulating.

### 2.2 Transfer Function

We use the Telegrapher’s equation to model signal propagation through the transmission line [1,4]. Our calculations closely follow Dermer’s approach [2]: we generate a time-domain input signal (with a rise time of 10 ps), convert it to the frequency domain, and use a voltage transfer function to derive the frequency-domain output signal. We model the frequency dependence of resistance and inductance using formulae suggested by Arabi *et al* [3], as is further described below. The frequency-domain output signal is then converted back into the time domain.

Our chosen voltage transfer function, shown in Eq. 3, has also been used by Ismail *et al* [4]:

$$\frac{V_{out}}{V_{in}} = \frac{2}{\left(\frac{R_{source}}{Z_0} + 1\right) \left(\frac{Z_0}{Z_L} + 1\right) e^{\gamma l} + \left(\frac{R_{source}}{Z_0} - 1\right) \left(\frac{Z_0}{Z_L} - 1\right) e^{-\gamma l}} \quad (3)$$

Here  $\gamma = \sqrt{(j\omega L + R)(j\omega C)}$ . Furthermore,  $Z_L = 1/j\omega C_L$ , while  $Z_0$  represents the characteristic impedance, which is calculated from the interconnect parameters.

### 2.3 Frequency Dependence

In the general case, the interconnect resistance of Eq. 3 consists of two parts: the DC resistance  $R_{dc}$  and an additional term  $Z_{skin}$ , which is due to the skin effect. It is helpful to imagine all current to be confined to a thin “skin” layer at the surface of the conductor. The thickness of this layer, the *skin depth*, has an inverse relationship with frequency. For a given conductor, the skin effect becomes noticeable at frequencies where the skin depth is on the order of the conductor cross-sectional diameter.

The total resistance is given by  $R = R_{dc} + Z_{skin}$ . Here  $R_{dc}$  is given by  $R_{dc} = \rho l / (w \cdot t)$  where  $\rho$  is the electrical resistivity of the interconnect material. The following expression for  $Z_{skin}$  [3] includes both the frequency-dependent resistance and inductance:

$$Z_{skin} = Z_{re} + Z_{im} \quad (4)$$

$Z_{re}$  and  $Z_{im}$  are modeled with Eq. 5 and Eq. 6, respectively:

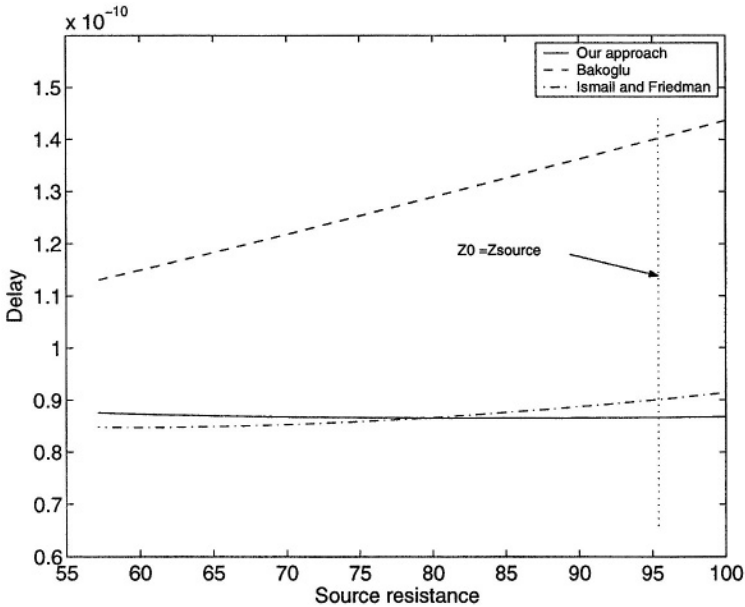
$$Z_{re} = \frac{R_{dc}}{\sqrt{\omega_s}} \sqrt{\omega} \quad (5)$$

$$Z_{im} = \frac{jR_{dc}}{\sqrt{\omega_s}} \sqrt{\omega} \tag{6}$$

The real part represents the resistive influence, whereas the imaginary part represents the inductance. The inductance must be adjusted together with the resistance for the resulting model to obey the laws of causality [6]. The parameter  $\omega_s$  is the frequency where the skin effect becomes noticeable.

### 3 Performance Without Skin Effect

Three design parameters determine the signal delay in the transmission-line setup in Fig. 1: the interconnect width, the interconnect length, and the driver source resistance. It would be desirable to model the delay directly as a function of these parameters. When using such a modeling function, it is important to note its range of validity:



**Fig. 3.** The delay using RC (Bakoglu), RLC (Ismail and Friedman) and numerical models (our approach) for an interconnect of length 4 mm and width 4  $\mu\text{m}$ .

Fig. 3 shows two delay estimates as a function of driver source resistance, and also the corresponding values produced with the simulation setup described in Section 2. (The skin effect is not taken into account in any of these graphs.) The range of source resistance values was chosen to include the characteristic impedance of the driven transmission line, since the LC regime requires  $Z_0 > R_{source}$ . The Bakoglu model [9] is shown in Eq. 7:

$$t_d = R_{source0} (C_{L0} + \frac{C_{wire}}{W/W_0}) + R_{wire} (0.4C_{wire} + 0.7C_{L0}(W/W_0)) \tag{7}$$

Here  $C_{L0}$  and  $R_{source0}$  is the receiver gate capacitance and driver source resistance, respectively. The interconnect capacitance is denoted  $C_{wire}$  and  $R_{wire}$  its resistance.  $W$  denotes the width of the driver transistor. The subscript “0” denotes the minimum size, so  $W/W_0$  is the upsizing factor of driver and receiver. From Eq. 7, it is clear that Bakoglu’s model does not take inductance into account at all. Not surprisingly, it overestimates the propagation delay across the source resistance range. It is included here only as a point of reference and will not be discussed further. The model of Ismail and Friedman [4] does include the inductive effects. It yields good estimates of the signal delay for source resistances smaller than the characteristic impedance of the transmission line.

When transmission line losses are insignificant, the ratio of the source resistance  $R_{source}$  to the characteristic impedance  $Z_0$  will determine the peak voltage value at the receiver input. A small source resistance will result in a voltage overshoot at the receiving end. The final voltage at the receiver will be approached in an oscillatory manner, as many reflected waves arrive. Such oscillations and overshoots may cause undesired electrical effects in the receiver, including unnecessary power dissipation.

For larger source resistance where  $R_{source}$  is approaching  $3 \times Z_0$ , the first incident wave will barely push the receiver input above 50% of its final voltage, and the 50%-50% delay increases rapidly when  $R_{source}$  approaches this limit.

Thus, unnecessarily low  $R_{source}$  values will cause overshoots, whereas unnecessarily high values will cause a large propagation delay.

## 4 Performance with Skin Effect

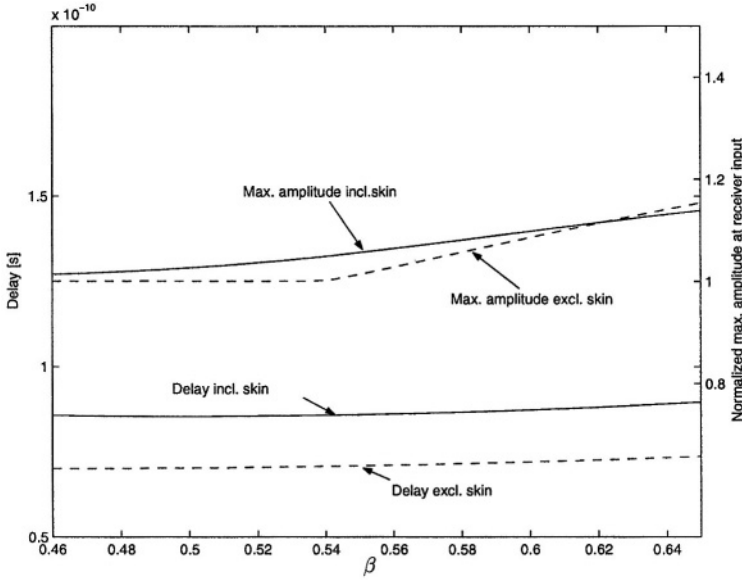
We will now investigate how delay and overshoot are affected by the skin effect. Fig. 4 presents the corresponding curves for the case where the skin effect is included, and the case where it is not. On the x-axis, we choose to have voltage division  $\beta = Z_0/(R_{source} + Z_0)$  which makes it easy to see where the overshoot occurs.

Clearly, the delay is increased significantly for the source resistance values of interest. Also, the skin effect serves to reduce the overshoot for small values of the source resistance, which could improve the efficiency of the driver in terms of power. However, this will not be treated further in this article.

The case in Fig. 4 only represents a driver-interconnect configuration where the interconnect parameters are fixed and the driver capability is altered. A question that still remains is how the delay is affected by the skin effect when both interconnect and driver parameters are changed. By varying the three parameters, i.e. driver and receiver transistor upsizing, interconnect width and length, we are able to emulate several cases where the reflections, the duration of the reflections and the effects from the signal propagation are significantly different.

Fig. 5 shows the percentage by which the skin effect will increase the signal delay compared to simulations of our model *without* frequency dependent  $R_{wire}$  and  $L_{wire}$ . The delay increase (in percentage) is plotted against i) the upsizing factor and ii) the interconnect length. From this it is possible, for a certain design scenario, to find the extra delay that the skin effect will incur on the total delay.

From a given design space of the upsizing factor and interconnect length, we wish to predict with what percentage the delay will increase as we change the interconnect width.



**Fig. 4.** The delay and overshoot versus  $\beta$  (the voltage division) with and without the skin effect for a  $1 \mu\text{m}$  wide and 4 mm long interconnect.

In order to choose a reasonable relation between the upscaling factor and interconnect length we use the modes of operation [10] that can occur for a driver-interconnect configuration. The delay for the RC and LC mode can be described with Eq. 8 and 9, respectively.

$$t_d = R_{source} (c_{wire}l + C_L) + \frac{r_{wire}c_{wire}l^2}{2} \tag{8}$$

$$t_d = Z_0 c_{wire}l \tag{9}$$

Here,  $r_{wire}$  is the interconnect resistance per unit length,  $c_{wire}$  is the interconnect capacitance per unit length according to Eq. 1, and  $l$  is the interconnect length. The delay formula in Eq. 8 is equivalent to the one in Eq. 7, but presented this way to show the dependence on length. Also, the term  $R_{wire}C_L$  is not included in Eq. 8 according to the approach used in [10].

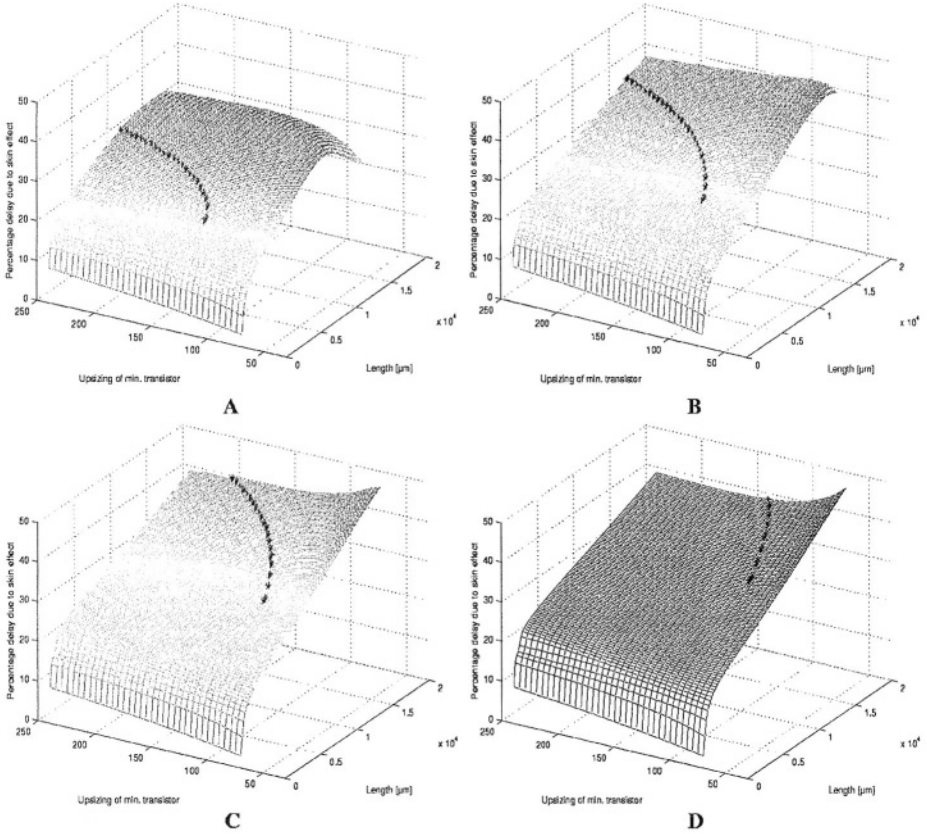
Depending on the parameters, there will exist regions where either Eq. 8 or 9 is defining the delay. These regions are defined by Eq. 10, 11 and 12.

$$\text{Driver delay: } R_{source} > Z_0 \tag{10}$$

$$\text{LC delay: } R_{source} < Z_0 \text{ and } l < l_{crit} \tag{11}$$

$$\text{RC delay: } R_{source} < Z_0 \text{ and } l > l_{crit} \tag{12}$$

$$l_{crit} = \frac{2Z_0}{R_{wire}}$$



**Fig. 5.** The percentage difference in delay caused by the skin effect for different values of the interconnect width. The graphs corresponding to the widths  $1 \mu\text{m}$ ,  $1.5 \mu\text{m}$ ,  $2 \mu\text{m}$  and  $2.7 \mu\text{m}$  are shown in A, B, C and D, respectively. The black curve represents the limit between the LC (to the left of the curve) and RC mode (to the right of the curve), which is the chosen design limit for interconnects in this article. The limit to the area where the driver delay is the dominating term is not included in the graphs since it is only relevant to short interconnects (where the skin effect has limited impact on delay).

In the region defined by Eq. 10, the source resistance in Eq. 8 is more significant than interconnect properties and thereby the driver delay is dominating the total delay. By dominating, we mean that one of the terms  $R_{source}(c_{wire}l + C_L)$ ,  $0.5r_{wire}c_{wire}l^2$  and  $Z_0c_{wire}l$  (in Eq. 8 and 9) is larger than the other two. As the characteristic impedance becomes larger than the source resistance, the importance of the driver delay decreases. Now, the terms in Eq. 8 and 9 that include interconnect properties are dominating, which introduces the regions of Eq. 11 and 12. The limit between these two regions depends on when the quadratic term in Eq. 8 is dominating, which is defined by  $l_{crit}$ .

By using the boundary between Eq. 11 and Eq. 12 we will have a reasonable limit that gives a minimum driver size for an interconnect of a certain length. Going below

this limit is highly undesirable since the quadratic delay term will dominate and thus the delay will increase quadratically with length. This limit is plotted in each graph of Fig. 5 on top of the mesh representing the delay increase. To the left of this curve, the driver-interconnect is in LC mode and to the right it is in RC mode. In Fig. 5, we do not bother about plotting the limit to the area where the driver delay is the dominating term, since we are only interested in the limit between RC and LC. From Fig. 5 it can be seen how this limit changes with interconnect width. Following the limit between LC and RC mode, this clearly indicates that the influence of the skin effect is rapidly increasing as the interconnect width is increased.

## 5 Conclusion

In this article we show that the skin effect can be evident in long on-chip interconnects. We have outlined in what design situations in the LC regime the signal propagation delay will be significantly affected by the skin effect. Our results, based on the 0.18- $\mu\text{m}$  Berkeley Predictive Technology Model interconnect parameters, indicate that the skin effect causes a minimum of 10% extra delay. The impact of the skin effect on delay is furthermore rapidly increasing with increased interconnect width. This is in contrast to Cao *et al* [5], who state that the skin effect will increase the delay only by approximately 1%.

## References

- [1] K Banerjee and A Mehrotra, "Inductance Aware Interconnect Scaling", Proc. of Intl Symp. on Quality Electronic Design, March 18–21, 2002.
- [2] G Dermer and C Svensson, "Time Domain Modeling of Lossy Interconnects", IEEE Trans. on Advanced Packaging, vol. 24, no. 2, May 2001, pp. 191–6.
- [3] T R Arabi, A T Murphy, T K Sarkar, R F Harrington and A R Djordjevic, "On the Modeling of Conductor and Substrate Losses in Multiconductor, Multidielectric Transmission Line Systems", IEEE Trans. on Microwave Theory and Techniques, vol. 39, pp. 1090–1237, July 1991.
- [4] Y I Ismail and E G Friedman *On-Chip Inductance in High Speed Integrated Circuits*, Kluwer Academic Publishers, 2001.
- [5] Y Cao, X Huang, D Sylvester, T-J King and C Hu, "Impact of On-Chip Interconnect Frequency-Dependent  $R(f)L(f)$  on Digital and RF Design", Proc. of ASIC-SOCC, Sept. 2002.
- [6] R N Bracewell, *The Fourier Transform and its Applications*, 3rd ed., McGraw-Hill, 2000.
- [7] B Kleveland, Q Xiaoning, L Madden, T Furusawa, R W Dutton, M A Horowitz and S S Wong, "High-Frequency Characterization of On-Chip Digital Interconnects" IEEE J. of Solid-State Circuits, vol. 37, no. 6, June 2002.
- [8] <http://www-device.eecs.berkeley.edu/~ptm>. Device Research Group of the Department of E.E. and C.S., University of California at Berkeley, 2004.
- [9] H B Bakoglu, *Circuits, Interconnects, and Packaging for VLSI*, Addison-Wesley, 1990.
- [10] A Deutsch et al., "When are Transmission-Line Effects Important for On-Chip Interconnections", Trans. on Microwave Theory and Techniques, vol.45, pp.1836–1846, Oct. 1997.

# A Low and Balanced Power Implementation of the AES Security Mechanism Using Self-Timed Circuits

D. Shang, F. Burns, A. Bystrov, A. Koelmans, D. Sokolov, and A. Yakovlev

School of EECE, University of Newcastle upon Tyne, NE1 7RU, U.K.

{delong.shang, f.p.burns, a.bystrov, albert.koelmans,  
danil.sokolov, alex.yakovlev}@ncl.ac.uk

**Abstract.** The hardware implementation of AES algorithm as an asynchronous circuit has a reduced leakage of information through side-channels and enjoys high performance and low power. Dual-rail data encoding and return-to-spacer protocol are used to avoid hazards, including data-dependent glitches, and in order to make switching activity data-independent (constant). The implementation uses a coarse pipeline architecture which is different from traditional micropipelines. The pipeline stages are complex and have built-in controllers implemented as chains of David cells (special kind of latches), whose behaviour is similar to fine-grain pipelines. A highly balanced security latch is designed. The design is partly speed-independent; in a few places it uses well localised and justified relative timing assumptions. The security properties of the system are evaluated by extensive simulation and by counting switching activity.

## 1 Introduction

Cryptosystem designers usually focus on implementing rigorous algorithms to protect data from logic attacks. The symmetric block cipher Rijndael [1] was standardized by the National Institute of Standards and Technology as Advanced Encryption Standard (AES) [2] in November 2001. AES, the successor of the Data Encryption Standard (DES), is the preferred algorithm for implementation of cryptographic protocols that are based on a round-based symmetrical cipher. It is defined for a block size of 128 bits and key lengths of 128, 192, and 256 bits. According to the key length, these variants of the AES are called AES-128, AES-192, and AES-256. As in OpenCore [3], we focus on demonstrating an implementation of the AES-128, simply called AES in this paper.

So far, all reported AES implementations were designed as synchronous circuits using single-rail [4]. In our recent work we derived secure dual-rail circuits from single rail netlists [16], without changing the clocked context of the circuit. This work presents a compact design with power savings due to self-timed implementation. The area savings are achieved by using a partly Speed Independent (SI) approach, having local timing assumptions thanks to the inherent regularity of the AES logic. Our main contributions are therefore as follows: (1) a pipelined AES architecture for self-timed implementation with a combination of distributed and centralised control, (2) a new secure latch designed to work with automatic reset to spacer for power balancing, and a new self-timed circuit implementation method based on partially speed-independence. Finally, the new AES implementation is compared with a synchronous high security AES implementation showing gains in speed and power.



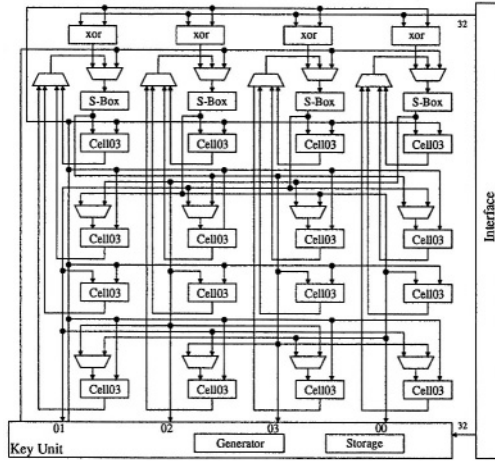


Fig. 1. Low cost AES architecture.

## 2 AES Architecture

Currently, several hardware implementation methods have been designed and published. A reasonable hardware architecture with low cost is shown in Figure 1, in which (compared to the architecture used in the OpenCore AES), we only used four S-Boxes (The S-Boxes are very expensive; they take more area and consume much of the total circuit power [9]). Although some extra multiplexers are needed, the overall circuit is much smaller.

In order to implement the algorithm, the S-Boxes are reused inside the data unit and shared by the data unit and key unit. In the data unit, this architecture has some performance penalties because the four S-Boxes cannot deal with 128-bit inputs simultaneously. In the key unit, no performance penalties exist because the data unit and key unit use the S-Boxes at different times.

Because four S-Boxes can only deal with 32-bit data at a time, it is unnecessary to request the 128-bit inputs simultaneously. 32-bit inputs are enough each time. In addition, the ShiftRows, MixColumns, and others function units having abilities to deal with 32-bit data, are enough as well. Compared to OpenCore AES, the resulting new architecture saves on a large number of hardware resources.

## 3 Self-Timed Implementation

Based on the above architecture, a self-timed circuit was implemented having the properties of low power and high security. Pipelining techniques were used to improve performance. The S-Boxes were designed especially for low power consumption requirements, and dual-rail self-timed circuits were used to reduce side-channel information leakage. Secure latches were designed as symmetrical well balanced (in terms of delay and power consumption) circuits supporting the return-to-spacer protocol.

	stage 1	stage 2	stage 3	stage 4	stage 5	stage 6
	SubBytes	ShiftRows	MixCol	MixCol2	MixCol3	MixCol4
time 0	①					
time 1	②	①				
time 2	③	②	①			
time 3	④	③	②	①		
time 4	⑤	④	③	②	①	
time 5		⑤	④	③	②	①

Fig. 2. The diagram of the pipeline used in the AES.

### 3.1 Pipelining

Pipelining increases throughput and it is used throughout the design. For this we allow multiple processes to overlap in time. This technique is implemented by placing registers in between the stages [10].

Four S-Boxes are used in our design and, hence, the size of the datapath is limited to 32-bits. As explained in Section 1, the algorithm is round-based. There are three types of rounds: initial round, normal round and final round. Each round is divided into stages. For example, the normal round is divided into four stages: SubBytes, ShiftRows, MixColumns, and AddRoundKeys. In each round, 32-bit data should pass through the stage four times and then a 128-bit intermediate result data is generated and passed into the next round. Because all stages are independent, this mechanism is possible to implement as a pipeline. The mechanism is arranged as shown in Figure 2.

In Figure 2, the encircled numbers 1, 2, 3, and 4 stand for intermediate data used in the data unit; they are passed through all stages; and the encircled number 5 stands for the data used in the key unit; it is only passed through the SubBytes and the ShiftRows stage. The input of the pipeline is 32-bit and the output of the pipeline is 128-bit data. This is because according to the AES algorithm, the MixColumns uses all intermediate data (4\*32-bits) obtained from the previous stage to produce 128-bit data, which is used in the AddRoundKeys stage. So in Figure 2, the MixColumns stage is used four times to show that a 128-bit intermediate data word is generated in each round.

After the MixColumns stage, the 128-bit intermediate data has been obtained and is used to generate a new intermediate data. In order to improve the performance, the datapath between MixColumns output and the AddRoundKeys stage has 128-bits.

Our AES uses pipelining of the SubBytes stage, ShiftRows stage, and the MixColumns stage. In most self-timed implementations, the control of the pipeline is often implemented using a micropipeline [11] controller. We implement the AES pipeline differently, using a direct mapping design flow [12]. The controller in the pipeline is implemented using DC circuits [12]. In the implementation, we designed several DC controllers: the input controller, SubBytes controller, ShiftRows controller, MixColumns controller and so on. All the controllers are synchronised using guards (control variables). A Petri net (PN) model of part of the controller (dealing with Text) is shown in Figure 3, in which only the Inputs, SubBytes, ShiftRows, MixColumns controllers are modelled (the dotted parts are not included). It works as follows: after the first stage (Inputs) reads a Text, it passes a token (Text) to the second stage (SubBytes) and then the Inputs are ready to read the second, the third and forth Texts and pass tokens to the S-Boxes; the S-Boxes are fired by the tokens; They read the token sequentially and compute them to generate intermediate data (token) for the third stage (ShiftRows); the ShiftRows work

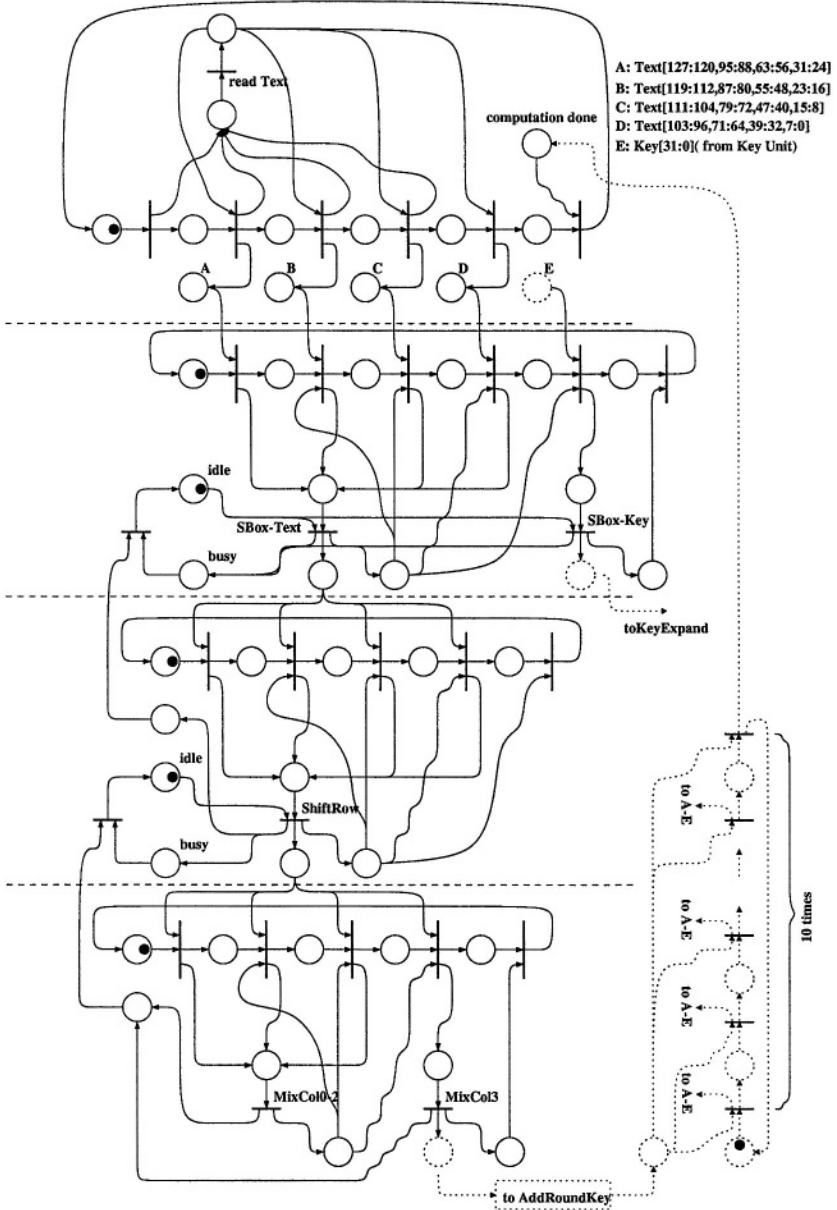


Fig. 3. PN model of the control.

similar to the SubBytes but pass tokens to the MixColumns; the MixColumns read the data (32-bit data in four times) and finally generate a 128-bit data for the AddRoundKeys. The PN is mapped to hardware circuits.

### 3.2 Low Power Design

In [9], power consumption of S-boxes was investigated. It was found that the S-Boxes in the SubBytes component consume most of the total power. For instance, 75% is consumed by the S-Box in a one round/cycle loop. It was also found that the power consumption of the S-Boxes was strongly influenced by the number of dynamic hazards. The two characteristics, differences of signal arrival time at each gate and propagation probability of signal transitions (which are the main reasons for creating or propagating dynamic hazards), are significantly different among the S-Boxes.

#### A. Differences of signal arrival time at each gate

Dynamic hazards occur when signal arrival times are different between multiple inputs of a gate. If multiple gates are connected serially and some of the internal gates generate hazards, then the hazards propagate into the circuit path, and extra power is consumed.

#### B. Propagation probability of signal transitions

The kinds of logic gates used influences the propagation of hazards. In particular, the use of XOR gates increases power consumption, because the probability of propagating a signal transition is 1, i.e. all hazards can be propagated from inputs to outputs, while the probabilities are 0.5 in other gates such as AND and OR. The use of many XORs is therefore a major reason for the large power consumption.

In [9], a low power S-Box architecture: a multi-stage PPRM (Positive Polarity Reed-Muller [13]) architecture was proposed, in which a low power consumption S-Box was implemented. The S-Box is fully computed using a three-stage PPRM architecture rather than using a look up table which is used in the OpenCore AES. The equations can be found in [9]. However, the equations were designed for synchronous circuit implementations. In our case we plan to implement the AES using dual-rail return-to-spacer asynchronous circuits. Figure 4 shows the dual-rail asynchronous AND, OR, and XOR gates.

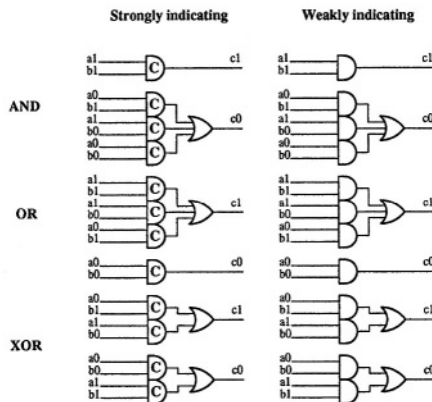


Fig. 4. Dual-rail implementations of some useful gates

In the figure, the implementations on the left are *strongly indicating* and on the right are *weakly indicating* [14] (there are other kinds of implementations such as early propagation, not covered in this paper). A strongly indicating function block waits for all of its inputs to become valid before it starts to compute and produce valid outputs, and it waits for all of its inputs to become empty before it starts to produce empty outputs. A weakly indicating function block may produce some outputs based on a subset of the inputs, but it should retain at least one empty output until all inputs have assumed valid values. The strongly indicating function blocks are good for low power designs. Because we use the return-to-spacer technique, after a computation every input returns to spacer (00). As a result, the outputs return to spacer (00) as well. When new inputs arrive, if not arriving at the same time, at least one input is still a spacer. In the weakly indicating implementation, one input is spacer and the output is still spacer due to the AND gate preventing the other inputs propagating to the outputs, so no extra power is consumed. Therefore, the weakly indicating implementations are suitable for low power implementation. On the other hand, as strongly indicating function blocks have a fixed computation delay, weakly indicating function blocks may offer some benefits to obtain better average performance.

One major disadvantage is that the above AND and OR dual-rail gates are asymmetric, which means that they are not good for balancing the power consumption of the circuits. In order to balance power, we use identical gates in the complementary pairs.

In the AES algorithm, the main operations are XOR. The same technique is used to design the other parts of the AES to guarantee low power consumption and power balancing. Due to using weakly indicating dual-rail gates in the implementation, a timing assumption is introduced, that is, before the function units are used again, they should return to spacer. This is a reasonable assumption, because, compared to a round cycle, the reset timing is short. So there is enough time to reset the function units to spacer.

### 3.3 High Security Latch Design

Dual-rail asynchronous circuits have high security properties because a dual-rail circuit has a symmetry between ‘0’ and ‘1’ actions that reveals little through its power consumption and electromagnetic emissions. Obviously, our AES implementation based on the dual-rail asynchronous circuits is basically secure.

However, a standard dual-rail latch is unsafe because, although it is symmetrical with respect to its two values, it still reveals whether a new value is the same as its predecessor or not. One solution to solve this problem is to propagate the spacer through latches, removing all traces of the old value before the new one is stored [15]. A new return-to-spacer latch was designed, as shown in Figure 5 (left side).

The basic idea of the latch is as follows: before writing a new value, the latch is reset automatically; then the new value is latched. In Figure 5 (left side), the top part is a modified latch and the bottom part is a controller. When the “req” signal arrives, it resets the latch to the spacer (00), and then enables the latch to save the new value. This means that the latch, regardless of whether the new value is the same as the old one, consumes the same amount of power. The simulation result shown in Figure 5 (right side) presents this point. We write 0,0,1, and 0 to the latch in an experiment. The top current waveform shows the power signature is the same. As a result, the profile of the power consumption

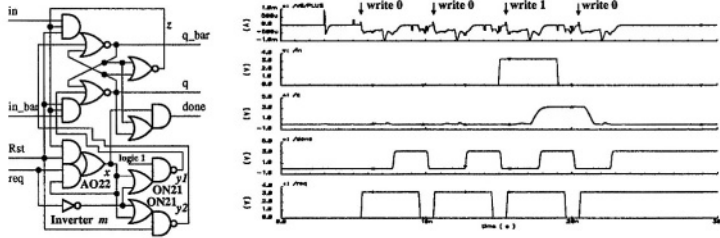


Fig. 5. Possible implementation of latches with spacer.

Table 1. The simulation results of the secure latch.

	req+ -> (q,q_bar)+	req+ -> done+
speed	1.7ns	2.2ns

is constant. So the latch is suitable for security designs. The performance measurements of the latch (for CMOS technology 0.35um) are shown in Table 1.

The STG specification of the latch is shown in Figure 6, in which the arrow line connected from req- to x- is dotted. This means that this is not a real connection. It just stands for the m+ happening earlier than the x- happening. This is one timing assumption in the latch that is the bottom inverter is faster than the AO22 gate. The other one is that two ON21 gates have same delay.

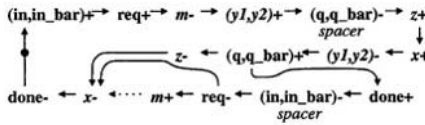


Fig. 6. The STG specification of the latches.

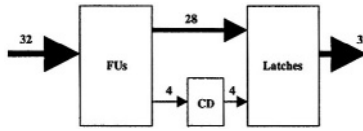
For other parts of the AES, due to the use of the return-to-spacer technique, it is easy to see that they consume the same amount of power during each round. This prevents power consumption information leakage.

### 3.4 Self-Timed Implementation

In self-timed design, the datapath is traditionally implemented using either fully SI circuits or bundled-data (BD) circuits. In SI circuits, dual-rail encoding (or some others such as 1-of-4) is used plus completion detection (CD) logic, which guarantees that it is hazard-free and fully independent of its delay. In BD, single-rail encoding is used with scaled (the worst case) timing assumption. However, both of them have some advantages and disadvantages. For example, the dual-rail implementation with secure latches is power balanced. It is good for security design. However it takes a large amount of hardware resources to implement the CD logic; the BD implementation is reasonable

for size. However, it is not power balanced [15] and it is time consuming to calculate the timing assumptions (delays) in the implementation.

Our design requirements make us look for a hybrid solution. Firstly for security, we cannot use the BD implementation. Secondly, the size of the datapath in the AES is 32-bits and fully SI circuits have a large penalty on the size of the circuits due to large CD. However SI implementation is good for security, therefore it is included as part of our design. So our method selects part of the datapath and implements it in SI circuits. We also try to benefit from logic regularity as in BD. For example, 32-bits of data, after passing through function units, are written to latches. 4-bits of the total 32-bits are passed through CD logic and the others are implemented normally as shown in Figure 7.



**Fig. 7.** A datapath implementation example.

Here a timing assumption is used: before the part of the datapath implemented by SI circuits settles down, the other normal datapath should settle down as well. Theoretically this is a reasonable assumption, because this part of the datapath takes more time than the normal datapath due to the CD logic.

We implemented the AES using the above techniques. It saves on a large number of hardware resources compared to the fully SI implementation and reduces delay calculating work compared to bundled-data implementations.

## 4 Results, Conclusions, and Future Work

The AES was implemented in 0.35 $\mu$ m CMOS technology. The complete circuit was simulated using the Cadence toolkit. The digital simulation results show the circuit working as expected. In addition, we compared our implementation with a dual-rail synchronous AES with return-to-spacer security design [16]. The comparison results are shown in table 2.

The performance and power consumption (measured in terms of switching activities) of the asynchronous AES are 33% and 28% better than the synchronous one respectively, although there is a modest area penalty (18%).

We also analysed side-channel leakage attack of the AES through the power consumption. No matter what the testing patterns are, the AES consumed the same power

**Table 2.** Comparison results

	performance	size (mm <sup>2</sup> )	switching activity
dual-rail asynchronous AES	800 ns	1,897	608,370
dual-rail synchronous AES	1200 ns	1,609 (without clock tree)	843,780

(the same switching activity occurred). This is very important, which means that the design is data-independent and has no power side-channel leakage.

The simulation results show that the design using partly SI circuits is a reasonable method to implement a big asynchronous design. However, it introduces problems when laying out the circuits because generally the layout tools cannot guarantee balancing of the two-wire routes for dual-rail logic. In the future, we plan to investigate a reasonable asynchronous design methodology.

The design is implemented using self-timed circuits, which remove the global clock completely. It is good for security design. However, the variability of the timing of self-timed circuits is a weakness that could be exploited by alternative attack techniques. We also need to investigate this. Furthermore, some optimization will be done in the future to improve the performance.

In the future, we intend to implement the AES automatically based on our design flow, in which the main part of the automation tool has been developed in the C language under the Linux system.

**Acknowledgements.** This work is supported by the Engineering and Physical Sciences Research Council (EPSRC) at Newcastle upon Tyne University (Project BESST GR/R 16754, Project STELLA GR/S 12036). We got many useful discussions and help from Dr. Fei Xia, Mr. Julian Murphy and others .

## References

1. J. Daemen and V. Rijmen, *The Design of Rijndael*, Springer-Verlag, 2002.
2. <http://csrc.nist.gov/publications/fips/nps197/fips-197.pdf>.
3. [http://www.opencores.org/projects/aes\\_core](http://www.opencores.org/projects/aes_core).
4. S. Managard, M. Aigner, and S. Dominikus, A Highly Regular and Scalable AES Hardware Architecture, *IEEE Transactions on Computer*, Vol. 52, No. 4, April 2003.
5. R. Anderson, Why Cryptosystems Fail, *Communications of ACM*, 37(11), pp. 32-40, 1994.
6. R. Anderson, and M. Kuhn, Tamper Resistance: A Cautionary Notice, Proc. 2nd USENIX Workshop on Electronic Commerce, Oakland California, 1996.
7. Erwin Hess, Norbert Jansen, Bernd Meyer, and Torsten Schuetze, Information Leakage Attacks Against Smart Card Implementations of Cryptographic Algorithms and Countermeasures: A Survey, <http://www.math.tu-dresden.de/~schuetze/reports/leakage.pdf>.
8. <http://www.cs.man.ac.uk/amulet>.
9. Sumio Morioka and Akashi Satoh, An Optimized S-Box Circuit Architecture for Low Power AES Design, in *the Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems 2002 (CHES 2002)*, San Francisco Bay, USA, August 2002.
10. David A. Patterson and John L. Hennessy, *Computer Organization & Design: the Hardware/Software Interface*, 2nd edition, ISBN 1-55860-491-X, Morgan Kaufman Publishers, Inc., San Francisco, California, 1997.
11. Ivan E. Sutherland, Micropipelines, *Communications of the ACM*, Vol 32(6), pp. 720-738, June 1989.
12. D. Shang, F. Burns, A. Koelmans, A. Yakovlev and F. Xia, Asynchronous System Synthesis Based on Direct Mapping using VHDL and Petri nets, Accepted for publication, IEE Proc. of CDT.



13. T. Sasao, 1993, AND-EXOR expressions and their optimization, in Sasao editor: Logic Synthesis and Optimization, pp. 287-312, Kluwer Academic Publishers.
14. C. L. Seitz, 1997, System timing, in Introduction to VLSI systems (Carver Mead and Lynn Conway), chapter 7, pp. 218-262, Addison Wesley.
15. L. A. Plana, P. A. Riocreux, W. J. Bainbridge, A. Bardsley, J. D. Garside, and S. Temple, SPA: A Synthesiable Amulet Core for Smartcard Applications, *In the Proceedings of 8th International Symposium on Asynchronous Circuits and Systems, Manchester, U.K.*, April 2002.
16. D. Sokolov, J. Murphy, A. Bystrov and A. Yakovlev, Improving the Security of Dual-rail Circuits, submitted to CHES 2004.

# A Power Consumption Randomization Countermeasure for DPA-Resistant Cryptographic Processors

Marco Bucci<sup>1</sup>, Michele Guglielmo<sup>2</sup>, Raimondo Luzzi<sup>1</sup>, and  
Alessandro Trifiletti<sup>3</sup>

<sup>1</sup> Infineon Technologies Austria AG, Babenbergerstrasse 10, A-8020 Graz  
(AUSTRIA)

{bucci.external, raimondo.luzzi}@infineon.com

<sup>2</sup> Vitrociset S.p.A., Via Tiburtina 1020, I-00156 Rome (ITALY)  
m.guglielmo@vitrociset.it

<sup>3</sup> Dept. Electronic Eng., University of Rome, Via Eudossiana 18, I-00185 Rome  
(ITALY)

trifiletti@mail.die.uniroma1.it

**Abstract.** Attacks based on a differential power analysis (DPA) are a main threat when designing cryptographic processors. In this paper, a countermeasure against DPA is presented and evaluated on a case study simulation. It can be implemented, using a standard digital technology, by applying a straightforward transformation to the original design, without an actual redesign. A methodology to perform a DPA in simulation is presented which can be exploited to test the resistance of a cryptographic processor during its design flow. By using the above methodology, the proposed countermeasure shows a  $30dB$  attenuation of the signals exploited by the DPA.

**Keywords:** Differential power analysis, DPA, power analysis, countermeasures, chipcards, cryptography.

## 1 Introduction

In several applications, cryptographic devices are used not only to perform encryption and decryption operations, but also to physically protect various secret data such as cryptographic keys and user PINs. Basically, these devices must be able to use their internal secret data without make them externally visible. This feature is essential to protect a device from cloning since the secret data are, actually, its identity. It is worthwhile to notice that, sometimes, the device can operate in a very hostile environment. As an example, in pay-tv systems, the owner of the access control device (i.e. the chipcard) is also its potential attacker. Moreover, in such applications, some of the secret data are owned by the system so that, if they are compromised, a large scale attack consisting in a massive cloning of the access control device is possible.

New challenges in this field are the so called side-channel attacks, i.e. attacks based on the information leakage from the cryptographic device due to physical emissions such as power consumption, EM-signals and execution timing. These attacks employ statistical analyses that allow to extract useful information from extremely noisy signals.

In particular, power analysis techniques are a main concern when designing cryptographic co-processors for chipcard integration since, due to physical constraints, adequate shielding and power consumption filtering cannot be employed.

Differential power analysis (DPA), introduced by P. Kocher et al. in 1998 [1], constitutes a real threat to chipcard security and attacks using DPA have been shown to be successful at breaking many cryptographic algorithms [2,3,4]. As a result, a plethora of countermeasures has been proposed in the recent years to prevent power analysis attacks on chipcards. On the algorithmic (software) level, random process interrupts [5] and random masking of intermediate variables [6] are widely exploited techniques. These are platform-dependent countermeasures and, usually, a substantial processing-time overhead is needed.

Hardware countermeasures can be classified according to the involved abstraction level during the design flow. System-level techniques include adding noise to the device power consumption [7], duplicating logics with complementary operations [8], active supply current filtering with power consumption compensation, passive filtering, battery on chip and detachable power supply [9]. Observe that some of mentioned techniques have a pure theoretical interest since, due to technological and cost constraints, they cannot be employed to design tamper resistant chipcards. Gate-level countermeasures include circuitual techniques which can be implemented using logic gates available in a standard-cell library, e.g. random masking, state transitions and Hamming weights balancing. At last, the transistor-level approach to prevent DPA is based on the adoption of a logic family whose power consumption is independent of the processed data. In literature, many differential and dynamic logic families are available which are suitable to design DPA resistant cryptographic hardware [10,11]. Although this technique is widely recognized as the most powerful DPA countermeasure, its cost in term of design time is usually very high since a full-custom approach is required.

From this short overview, it follows that choosing the right countermeasure to guarantee the wanted security level is a main issue to address. Unfortunately, most DPA countermeasures are based on heuristic remarks and their effectiveness can be verified only once a prototype has been designed and manufactured.

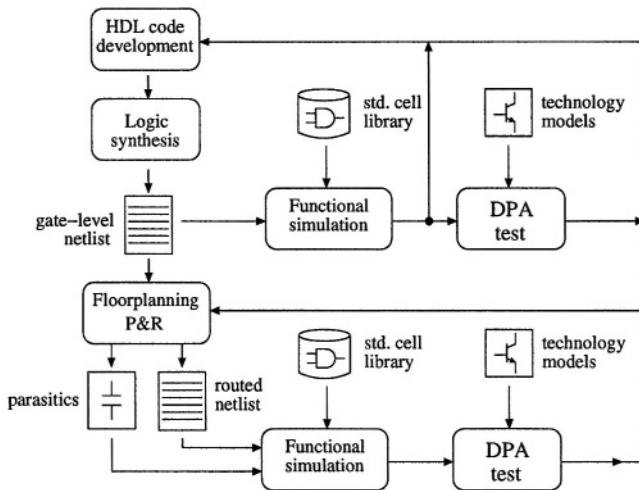
In this paper, a reliable simulation methodology is reported which exploits an accelerated transistor-level simulator to collect a database of current consumption waveforms. It has been successfully used to implement a simulated DPA attack thus allowing to evaluate the resistance of a generic cryptographic processor during its design flow. With the proposed technique, different trade-offs between DPA resistance and cost can be easily tested in simulation to drive the choice of a proper countermeasure. Although the developed methodology is generic, it

is particularly useful to evaluate the effectiveness of gate-level countermeasures by adding a check for DPA resistance inside a standard digital design flow.

In Section 2, a DPA-aware digital design flow is described with special emphasis on the implementation details for the simulated DPA attack. Using the developed methodology, a gate-level countermeasure based on a random pre-charging of combinatorial networks and registers has been studied. The proposed countermeasure is discussed in Section 3 while, in Section 4, simulation results using an AES co-processor [12] as a case study are reported.

## 2 A DPA-Aware Design Flow

Usually, when designing a cryptographic co-processor, a standard digital design flow is followed, where only functional and timing requirements are verified in simulation. The resistance to side-channel attacks is experimented once a prototype has been manufactured and, if the security requirements are not matched, a redesign is needed or software countermeasures are added to improve the device behavior. As a results, the development time gets longer and it is likely to obtain a design less optimized than the result which could be carried out with an approach where security requirements are taken into account from the very beginning.



**Fig. 1.** A DPA-aware digital design flow.

In Figure 1 a digital design flow is depicted where a check for DPA resistance has been added. The DPA characteristics are tested both during the HDL code development, where different countermeasures can be experimented, and after the layout design, once the parasitic capacitances have been extracted. In fact, in

a semi-custom layout, due to the asymmetry in the interconnection lengths, the differences in power consumption increase and, as a result, the effectiveness of a DPA countermeasure decreases if the parasitic effects are taken into account. An unsatisfactory result for the second test can trigger modifications to the layout structure (bus length, full-custom layout of critical blocks, etc...) or changes to the HDL code.

To implement a DPA check, a simulation of the current consumption for the device under test is required. Commercial transistor-level simulators based on look-up table models are enough fast and accurate to obtain a set of current consumption waveforms for a cryptographic co-processor. The proposed methodology has been implemented using Nanosim from Synopsys [13].

The simulated curve database  $\{IDD_i(t), i = 1, \dots, N\}$  is processed using a proprietary software which has been developed by the authors to implement a DPA attack. As shown in Figure 2, the software features four main modules: a pre-processing tool which can perform some re-sampling, low pass filtering and re-synchronization of the input curves; a module for the  $\{IDD(t)\}$  set partitioning, an analysis software to generate the DPA curves and a C/C++ implementation for the cryptographic algorithm under attack. The software has been developed for the AES algorithm but, thanks to its modular architecture, it can be easily adapted to other algorithms.

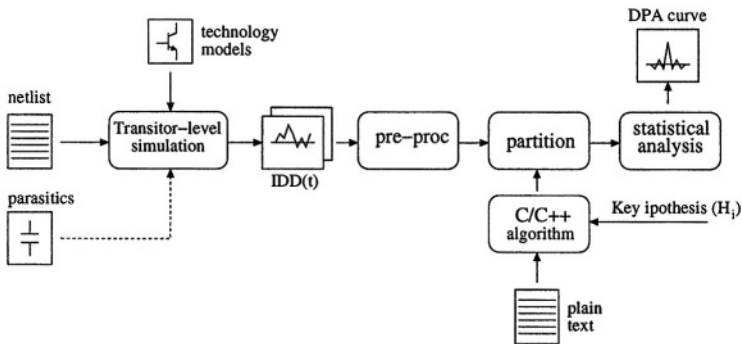


Fig. 2. Test for DPA resistance.

### 3 A Random Pre-charging Countermeasure

Among the different hardware techniques to prevent DPA attacks, gate-level countermeasures feature some important advantages: they can be implemented in a semi-custom digital design flow without any full-custom design, are easily ported on different technologies and standard-cell libraries and, moreover, they can be applied to existing cryptographic IP cores with modifications to their RTL code.

By using the presented design flow for DPA-secure cryptographic processors, a countermeasure based on a random pre-charging of combinatorial networks and registers has been tested. When designing a DPA countermeasure, two different approaches can be followed: we can increase the noise on the current consumption waveforms or decrease the useful signal, i.e. the correlation between the current signal and the processed data.

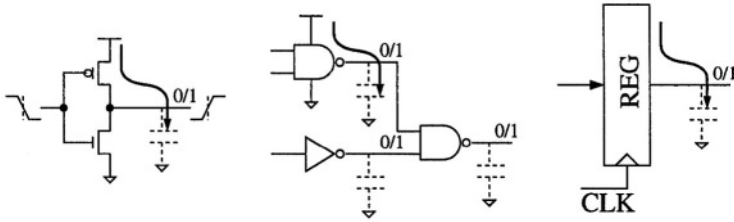


Fig. 3. Power consumption in a static CMOS circuit.

A gate in a static CMOS logic family only consumes energy from the power supply when its output has a 0-to-1 transition (Figure 3). Naturally, each transition propagates through the circuit thus producing both 0-to-1 and 1-to-0 consequent transitions. Therefore, the global current consumption  $IDD(t)$  that can be observed is, directly or indirectly, correlated with both the transitions and can be seen as a function of the transition from an initial state  $S_k$  to a final state  $S_{k+1}$  of the circuit:

$$IDD(t) \propto \sum_i i_{DD}(S_k(i) \rightarrow S_{k+1}(i), t) \tag{1}$$

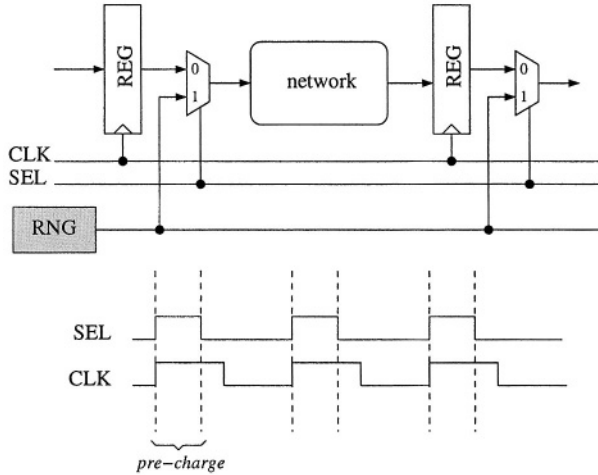
where  $i_{DD}(S_k(i) \rightarrow S_{k+1}(i))$  is the current contribution due to the  $i$ -th node.

If an intermediate random state  $S_r$  is inserted, it follows:

$$IDD(t) \propto \sum_i \{i_{DD}(S_k(i) \rightarrow S_r(i), t) + i_{DD}(S_r(i) \rightarrow S_{k+1}(i), t)\}. \tag{2}$$

Since the state  $S_r$  is random, the two contributions in (2) are expected to be less correlated with the secret data than the current contribution in (1) where the transition involves two states both correlated with the secret data. Such hypothesis is confirmed by the simulation results reported in Section 4 for the presented case study.

The dummy state  $S_r$  can be inserted using a random pre-charging technique as shown in Figure 4 for a pipeline stage. At the beginning of each clock period, a random input is fed to the combinatorial network and, once the network state is stable, the real data is processed. Observe that even if the network input is random, the logic state for the internal nodes is not perfectly random due to correlations introduced by the network itself. However, the diffusion property of



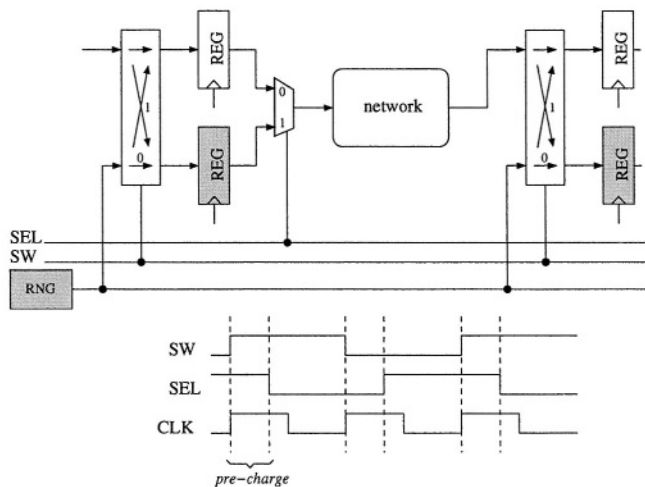
**Fig. 4.** Random pre-charging of combinatorial networks.

logic functions used in cryptographic algorithms allows to use the pre-charging with a random input to obtain a near-random state  $S_r$ . The more the dummy state  $S_r$  is random the more the countermeasure is effective.

A random source is needed but this is not an issue since a truly random number generator (RNG) is always available in a chipcard. A linear feedback shift register (LFSR) can be employed to expand the seed from the RNG thus obtaining the required amount of uniformly distributed bits. Moreover, the pre-charge time does not affect the algorithm data rate since typical clock frequencies in chipcard applications are much slower than maximum clock frequencies allowed by sub-micron CMOS processes.

The circuit shown in Figure 4 has a main drawback: registers are not pre-charged with a random data. Since a significant contribution to the overall current consumption is due to the switching of the internal registers, a pre-charging mechanism has been developed as depicted in Figure 5. Each register is doubled and a commutator is used to store the real data or a random quantity in a register pair. When a random data is stored in an upper register, the real data is store in the lower one and vice versa. Combinatorial networks are pre-charged as in the previous architecture.

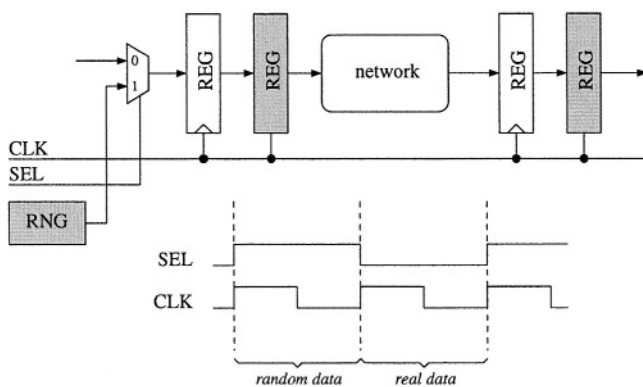
Each stage of the pipeline in Figure 5 is fed with a new random data from the RNG. However, exploiting the diffusion property of cryptographic algorithms, which has been already used to justify the pre-charging technique in Figure 4, the architecture shown in Figure 6 has been proposed. The doubled registers are cascaded, a single random input is considered at the pipeline beginning and the commutators are removed. The result is an interleaved pipeline where the real and a random data flows are processed simultaneously. Of course, a double clock frequency is required to maintain the same data rate as the original



**Fig. 5.** Random pre-charging of combinatorial networks and registers.

pipeline. Power consumption is also doubled but, with respect to other available countermeasures, this is an acceptable overhead.

Observe that other random inputs can be added in intermediate nodes of the pipeline to obtain different trade-offs between chip area and resistance to power analysis attacks. Moreover, if the DPA countermeasure can be disabled, the architecture in Figure 6 can be used to process two data flows in parallel thus doubling the algorithm throughput.



**Fig. 6.** Random interleaved pipeline.



### 4 Case Study: AES

The discussed countermeasure has been applied to an AES co-processor as shown in Figure 7. A minimum area architecture has been adopted where the internal registers are doubled according to the interleaved structure depicted in Figure 6 and two random inputs (RNG) have been inserted.

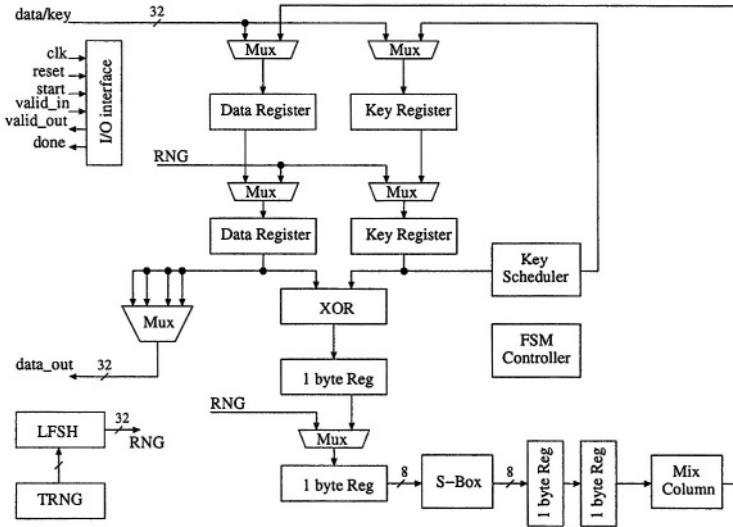


Fig. 7. AES architecture with the proposed countermeasure.

The AES core has been synthesized using a standard-cell library for a 0.18 $\mu$ m CMOS process manufactured by TSMC and the countermeasure effectiveness has been tested using the proposed simulation methodology.

The target quantity for the DPA attack is an output bit from the first S-box execution during the first AES round. In the input plain texts, the byte in input to the considered S-box is changed from 00 to FF while keeping the other bytes fixed so that the algorithmic noise is minimized [2]. Therefore, the current waveform database includes 256 curves and, using an entry-level workstation, a 60 hours simulation time is needed to collect the whole database. Of course, this kind of attack allows to extract only one byte of the encryption key, but the procedure can be easily repeated for the other S-boxes. Anyway, the extraction of a single key byte is sufficient to test the effectiveness of the adopted countermeasure.

In order to obtain an estimate for the DPA resistance improvement, the AES core without the random pre-charging has been also simulated. In Figure 8 the maximum of the DPA curves as a function of the key hypothesis  $H_i$  is shown: for the correct key ( $H_i = 0$ ), a 30dB attenuation of the DPA peak is achieved using

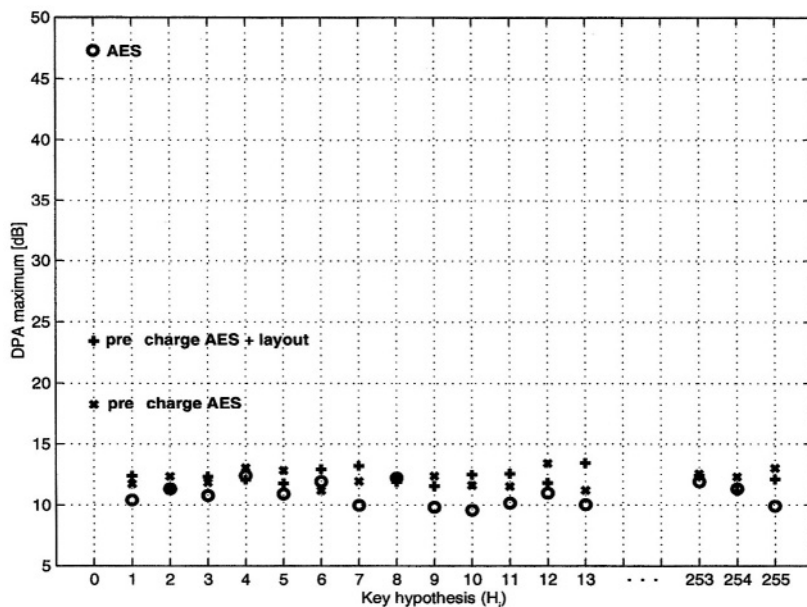


Fig. 8. Maximum of the DPA curves.

the proposed countermeasure. As expected, if the layout parasitic capacitances are taken into account, the peak attenuation is lowered to 25dB.

## 5 Conclusions

A simulation methodology has been proposed which allows to estimate the effectiveness of DPA countermeasures. Using the discussed methodology, a check for DPA resistance can be introduced in a digital design flow, gate-level countermeasures can be easily tested and different trade-offs between cost and security can be experimented.

A countermeasure based on a random pre-charging of combinatorial networks and registers has been evaluated using an AES co-processor as a case study. A 30dB attenuation for the DPA characteristics has been achieved with respect to the AES core without any countermeasure, with a negligible penalty in terms of chip area. The proposed countermeasure has been proved to be reliable and suitable to be implemented in existing cryptographic cores with minor changes to their RTL descriptions.

## References

1. P. Kocher, J. Jaffe and B. Jun, *Differential power analysis*, Proc. Advances in Cryptology (CRYPTO '99), Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, pp. 388-397, 1999.

2. T. S. Messerges, E. A. Dabbish and R. H. Sloan, *Examining Smart-Card Security under the Threat of Power Analysis Attacks*, IEEE Trans. Computers, vol. 51, no. 5, pp. 541-552, May 2002.
3. J. Coron, *Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems*, Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES '99), Lecture Notes in Computer Science, vol. 1717, Springer-Verlag, pp. 292-302, 1999.
4. C. Clavier, J. Coron and N. Dabbous, *Differential Power Analysis in the Presence of Hardware Countermeasures*, Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES '00), Lecture Notes in Computer Science, vol. 1965, Springer-Verlag, pp. 252-263, 2000.
5. J. Daemen and V. Rijmen, *Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals*, Proc. Second Advanced Encryption Standard Candidate Conf., available at <http://csrc.nist.gov/encryption/aes/round1/conf2/aes2conf.htm>, 1999.
6. M. A. Hasan, *Power Analysis Attacks and Algorithmic Approaches to Their Countermeasures for Koblitz Curve Cryptosystems*, IEEE Trans. Computers, vol. 50, no. 10, pp. 1071-1083, Oct. 2001.
7. L. Benini, E. Omerbegovic, A. Macii, M. Poncino, E. Macii, F. Pro, *Energy-aware design techniques for differential power analysis protection*, Proc. Design Automation Conf. (DAT '03), pp. 36-41, 2003.
8. H. Saputra, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, R. Brooks, S. Kim, and W. Zhang, *Masking the energy behavior of DES encryption*, Proc. Design, Automation and Test in Europe Conf. (DAT '03), pp. 84-89, 2003.
9. A. Shamir, *Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies*, Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES '00), Lecture Notes in Computer Science, vol. 1965, Springer-Verlag, pp. 71-77, 2000.
10. N. Weste and K. Eshraghian, *Principle of CMOS VLSI design*, 2nd ed., Addison-Wesley, 1994.
11. K. Tiri, M. Akmal and I. Verbauwhede, *A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards*, Proc. IEEE 28th European Solid-State Circuit Conf. (ESSCIRC '02), 2002.
12. National Institute of Standards and Technology, *FIPS PUB 197: Advanced Encryption Standard (AES)*, Nov. 2001.
13. Synopsys, *Nanosim User Guide*, Release 2002.03, March 2002.

# A Flexible and Accurate Energy Model of an Instruction-Set Simulator for Secure Smart Card Software Design\*

Ulrich Neffe<sup>1</sup>, Klaus Rothbart<sup>1</sup>, Christian Steger<sup>1</sup>, Reinhold Weiss<sup>1</sup>,  
Edgar Rieger<sup>2</sup>, and Andreas Muehlberger<sup>2</sup>

<sup>1</sup> Institute for Technical Informatics, TU Graz  
Inffeldgasse 16/1, 8010 Graz, Austria  
{neffe,rothbart,stege,rweiss}@iti.tugraz.at

<sup>2</sup> Philips Austria GmbH Styria  
Business Line Identification  
Mikronweg 1, 8101 Gratkorn, Austria  
andreas.muehlberger@philips.com

**Abstract.** Smart cards are mostly used to process and store confidential and secure information. Several attacks based on power consumption analysis are known to retrieve this information. This paper presents the energy model of a cycle-accurate instruction-set simulator for secure software development. The model is designed to decouple instruction and data dependent energy dissipation, which leads to an independent characterization process and allows stepwise model refinement to increase estimation accuracy. The model has been evaluated for a high-performance smart card CPU and an use-case for secure software is given.

## 1 Introduction

Smart cards are small computing platforms with limited resources. They are mainly limited in chip size and power consumption. Important reasons for chip size limitations are the required low costs but also the mechanical stability to avoid a break of the chip when the card is transported in the purse. Power consumption is limited by international standards dependent on the application area. For instance, the standard GSM 11.11 limits the current to 10 mAmps at 5 V supply voltage. Due to chip size and power constraints smart cards are devices operating with low performance which limits the complexity of typical smart card applications.

Power considerations can be divided into two groups. The first group is with a view to power supply. Smart cards are often used in mobile devices like mobile phones which are supplied by a battery. Due to this reason they should be optimized for low power. But smart cards are also used with contact-less communication interfaces like in facility access cards, transport applications or

---

\* This work has been supported by the Austrian government under grant number 806014.

e-passports. They are supplied by a radio frequency (RF) field which provides a strictly limited amount of power. If the power consumed by such a smart card exceeds this limit a reset can be triggered by the power control unit or otherwise the chip maybe stays in an unpredictable state. Therefore the smart card has to be optimized for low power with the constraint to avoid peaks in power consumption.

The second group is with a view to system security. Smart cards are often used to process and store confidential information. Secure data storage is used in bank cards, in the *subscriber identification module (SIM)* used in GSM mobile phones, and ticketing applications. Different non-invasive attacks based on power analysis are best known to retrieve this information from a smart card. *Simple power analysis (SPA)* and *differential power analysis (DPA)* are attacks based on the analysis of the power consumption profile of a smart card. A lot of effort is necessary to protect cryptographic algorithms against these attacks. Protection often results in a loss of performance and increases data memory usage and code size.

A large part of a smart card software architecture does not deal directly with confidential information and cryptographic algorithms, for instance the memory management unit. But the power profile of these software parts gives detailed timing information to the attacker and allows him to trigger attacks to disturb the system always at the same point of time. For instance, each method has its own power profile and an attack can be triggered after a particular method has been invoked. Memory, performance and time-to-market constraints do not allow the protection of the whole software. But an extended software design flow can be used to perform power-aware software development to increase system security.

The proposed design flow is presented in Fig. 1. State-of-the-art smart card software development is done on a smart card evaluation board. In the presented approach an instruction-level power simulator is used to estimate the power consumption profile in parallel to the evaluation board. A power analyzer processes the results to support the engineer with more comfortable information. Power

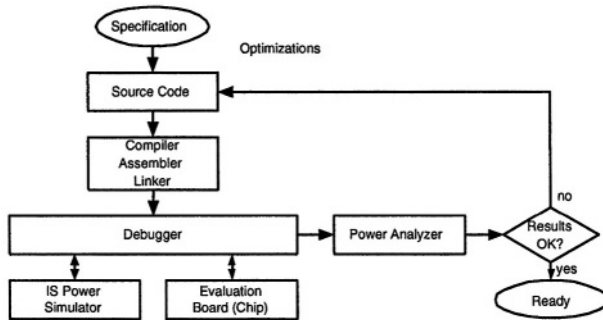


Fig. 1. Design flow for secure software development

analysis can also be performed by the use of physical measurement, but the reasons for peaks and gaps in the power profile are difficult to identify because of smart card protection mechanisms and the microchips capacitance (Sect. 4).

This paper presents the energy model of an instruction-level simulator for a high-performance smart card CPU [1], which decomposes the energy consumption into an instruction and data dependent part to get maximal flexibility and to allow step-wise model refinement. The remainder of this paper is organized as follows. Section 2 surveys related work for software power estimation. Section 3 presents the instruction level power model of a high performance smart card. Results are presented in Sect. 4 and Sect. 5 concludes.

## 2 Related Work

One of the first instruction-level power models has been introduced by Tiwari *et al.* [2] ten years ago. This model defines *base costs* to characterize a single instruction. *Circuit state overhead* takes into considerations circuit switching activity between consecutive instructions. *Inter-instruction costs* model inter-instruction locks, pipeline stalls and cache misses. Several authors presented techniques to minimize required measurements and evaluated effects on total energy consumption.

Sami *et al.* presented in [4] a pipeline aware power model for VLIW architectures. The work presented there describes an energy model which combines the flexibility of instruction-level models with the accuracy of microarchitectural-level models. The model is based on the analytical decomposition of the energy consumed by a software instruction into the contributions of the active processor functional units and the pipeline stages.

A data dependent approach for instruction-level power estimation was proposed in [6]. They analyzed a commercial DSP core from STMicroelectronics and demonstrated the big influence of operand values on power consumption. The analytical approach developed for this core considered changing instruction costs for switching between consecutive instructions, minimum costs for a particular instruction and activity elements. Activity elements are defined as elements with a big influence on power consumption like internal busses. Characterization has been done with a VHDL simulation of the whole CPU in combination with a transistor level simulator. Results have shown an accurate cycle by cycle estimation with an error less than 8%.

Another approach also considering data dependencies has been published in [5]. The energy model is based on the processor block structure containing functional units, internal and external busses and memories. It consists of four parts: instruction dependent costs inside the CPU, data dependent costs inside the CPU, instruction dependent costs in the instruction memory and data dependent costs in the data memory. Instruction loops were used to characterize all necessary parameters of the model. The measured precision was 1.7% for this model.

### 3 Energy Model

The intention of this model is a flexible and accurate combination of the instruction-level energy model and data dependent models. The proposed model decouples instruction dependencies and data dependencies as presented in (1). The total energy  $E_{\text{total}}$  consumed by a program is the sum of the energy consumption per cycle  $E_{\text{cycle}}$  of all clock cycles  $n_{\text{total}}$ . The energy per clock cycle can be decomposed into four parts: instruction dependent energy dissipation  $E_i$ , data dependent energy dissipation  $E_d$ , energy dissipation of the cache system  $E_c$ , and finally the dissipation of all external components  $E_e$  containing the bus system, memories and peripherals. This also leads to an independent instruction and data characterization process and enables a model refinement based on the requirements for accuracy.

$$E_{\text{total}} = \sum_{n=0}^{n_{\text{total}}} E_{\text{cycle}}(n) = \sum_{n=0}^{n_{\text{total}}} [E_i(n) + E_d(n) + E_c(n) + E_e(n)] . \quad (1)$$

#### 3.1 Instruction Dependency Model

The instruction dependent part of this model is based on the instruction-level energy model defined by Tiwari *et al.* [2]. It defines *base costs* (BC) for each instruction and considers switching activity between different consecutive instructions by the use of a *circuit state overhead* (CSO). The instruction dependent energy dissipation per cycle  $E_i(n)$  is the sum of the base costs  $E_{\text{BC}}$  and the circuit state overhead  $E_{\text{CSO}}$ . Base costs and circuit state overhead are calculated based on (2) and (3). Where  $n_s$  is the number of pipeline stages and  $i[j]$  is the instruction executed in stage  $j$ .

$$E_{\text{BC}} = \sum_{j=1}^{n_s} BC_j, \text{ with } BC_j = \begin{cases} BC(i[j], j), & \text{if } j \text{ is active.} \\ BC_{\text{stall},j}, & \text{if } j \text{ is stalling.} \end{cases} . \quad (2)$$

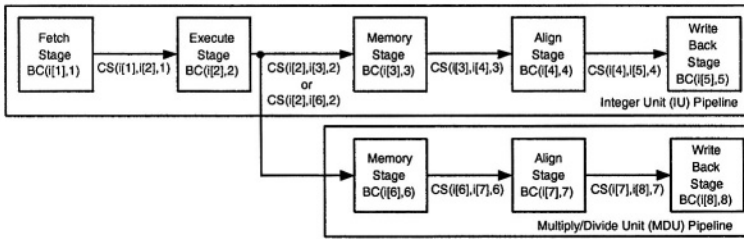
$$E_{\text{CSO}} = \sum_{j=1}^{n_s-1} CS(i[j], i[j+1], j) . \quad (3)$$

Base costs are characterized in the same way as proposed in [2]. To characterize a particular instruction a loop is used which is small enough to fit in the instruction cache and large enough to minimize the effect of the needed branch instruction. But the difference to the original approach is the selection of operands. The same operands are used for the entire loop and source values are mostly set to zero. This choice minimizes switching activity due to operand and data dependencies like source register switches or switching activity in the ALU. The pipeline aware model [4] is used for cycle by cycle estimation. The main idea of this approach is to distribute the measured base costs among all pipeline stages. A uniform distribution is used for this model, the general context between the measured base costs and the base costs for one stage describes (4):

$$BC(i, j) = \frac{T_i \times \widetilde{BC}_i - N_{i,nop} \times BC_{nop,j} - N_{i,stall} \times BC_{stall} - \alpha_i}{N_i}. \quad (4)$$

$BC(i, j)$  is the power consumed by stage  $j$  when executing instruction  $i$ ,  $\widetilde{BC}_i$  is the base cost value measured for instruction  $i$ ,  $BC_{stall}$  is the base cost value for stalling,  $T_i$  is the number of clock cycles,  $N_{i,nop}$  is the number of stages executing  $nop$ ,  $N_{i,stall}$  is the number of stages in *stall* state,  $N_i$  is the number of active stages for instruction  $i$  and  $\alpha_i$  is a constant factor caused by stalling of instruction  $i$ .

The use of this equation can be illustrated with the target smart card processor. An abstract view of the pipeline architecture gives Fig. 2. The architecture comprises an integer unit (IU) and a separate multiply-/divide unit (MDU) pipeline. The first and second pipeline stage are shared between both pipelines. If an instruction stays in a pipeline stage longer than one cycle, the control unit stalls only the previous pipeline stages and does not affect subsequent stages.



**Fig. 2.** Instruction dependent energy model for the pipeline architecture

The first example for using (4) is a loop of instructions without stallings. One instruction can be fetched per clock cycle ( $T_i = 1$ ), five pipeline stages IU pipeline) are active per cycle ( $N_i = 5$ ), three pipeline stages (MDU pipeline) are executing  $nop$  ( $N_{i,nop} = 3$ ), no stalling occurs ( $N_{i,stall} = 0$ ) and no additional factor appears ( $\alpha_i = 0$ ). This results in (5) which is used for all instructions executing without stallings.

$$BC(i, j) = \frac{\widetilde{BC}_i - 3 \times BC_{nop,j}}{5}. \quad (5)$$

The second example is a loop of *div* instructions (division). The considered time is twelve clock cycles ( $T_i = 12$ ), multiplied by eight pipeline stages results in 96 different pipeline states. 16 of them are active ( $N_i = 16$ ), 22 of them are stalled ( $N_{i,stall} = 22$ ), 58 are executing a  $nop$  instruction ( $N_{i,nop} = 58$ ) and  $\alpha_i$  takes into consideration register file activation.

A loop with two alternating instructions is used to measure the CSO  $\widetilde{CS}$  between subsequent instructions. The CS between two consecutive pipeline stages  $j$  and  $j+1$  executing instructions  $i_1$  and  $i_2$  can be calculated with (6).  $n_{MDU}$  and



$n_{IU}$  are the numbers of pipeline stages for MDU and IU pipeline, respectively.  $\gamma_j$  is the specific weight for each pipeline transition. Equation (7) specifies  $\gamma_j$  equal for all transitions, but this can be changed to get more realistic behavior.

$$CS(i_1, i_2, j) = \gamma_j \left\{ \widetilde{CS} - \sum_{k=1}^{n_{MDU}} BC_{nop,k} - \frac{1}{2} \sum_{k=1}^{n_{IU}} BC(i_1, k) + BC(i_2, k) \right\}. \quad (6)$$

$$\gamma_j = \frac{1}{n_{IU} - 1}. \quad (7)$$

All other inter instruction costs as described above are resolved by the simulator. The instruction dependent model considers all effects with regard to instruction executions but is independent of actual data values or registers used by an instruction. The next subsection presents how this model can be extended with a data dependent model very flexibly.

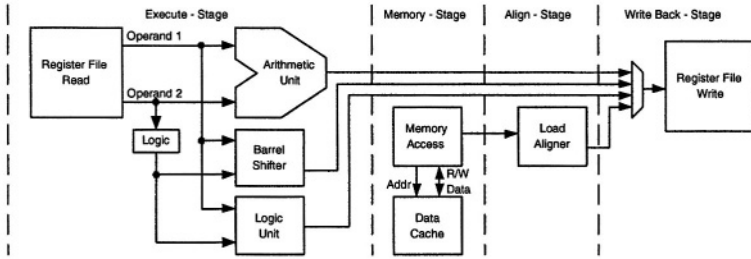
### 3.2 Data Dependency Model

The data dependent energy consumption per cycle  $E_d(n)$  is the sum of the energy consumption of all pipeline stages. The energy consumption of a particular pipeline stage  $E(i[j], j)$  is determined by the instruction  $i[j]$  executed in this stage  $j$  as presented in (8).

$$E_d(n) = \sum_{j=1}^{n_s} E(i[j], j). \quad (8)$$

Each instruction is responsible for the calculation of its energy consumption for the pipeline stages. A micro-architectural energy model shared between all instructions is used to consider data dependent switching activity of busses and functional units. An instruction can select shared busses and functional units out of this model, but bit patterns and specific properties are calculated with its own instruction energy model.

The characterization of the basic behavior of each instruction is the first step to get a data dependent model. A loop containing pairs of the same instruction with different source and target registers is used for this. The basic power consumption of this loop is determined by using the same source operand values as for base cost characterization. An additional power consumption compared to the base cost loop can be measured due to source and target register switching. This power value  $P_{base}$  is used as reference for all following measurements  $\widetilde{P}$ . During the next step the source operand values of the first instruction are the same as in step one, but the source operands of the second instruction are changed. Thus the data path is brought back to an initial state every second instruction. The data dependent power consumption  $P_d$  is the difference between the measured power value  $\widetilde{P}$  and  $P_{base}$ . A footprint of each instruction can be drawn up by using this methodology. The footprints can be used to identify instructions sharing the same functional units. For example, the instructions *add*, *add immediate* and



**Fig. 3.** Data dependent energy model for smart card CPU created based on characterization loops

*sub* have the same power behavior due to sharing the same adder. Of course, the mathematical functionality has to be taken into consideration. All instructions with the same behavior are arranged in groups. The next step is the identification of internal busses shared among different groups of instructions. Again a loop of alternating pairs of instructions is used. First, the first source operands of the two instructions are the same (e.g. 0xaa...) and second, they differ with maximum hamming distance (e.g. 0xaa... and 0x55...). The difference between the power values are used for bus switching characterization.

This methodology applied to the considered smart card CPU results in an architectural model for the IU-pipeline as presented in Fig. 3. This model was integrated into the pipeline model presented in Fig. 2. Each instruction has its own data dependent energy model, which contains references to the used shared energy models for each pipeline stage. This is a very flexible approach because data dependence power models can be extended or refined by referencing additional or other shared models.

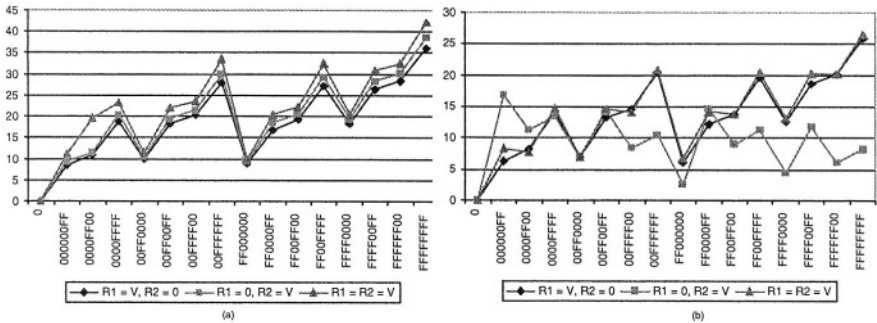
## 4 Evaluation and Results

A cycle-accurate instruction-set simulator for the considered CPU has been developed and a coarse grain energy model was integrated. The first section discusses the accuracy of the power simulator and the second section the usage for secure software development. The evaluation of the energy model was performed in several steps. The first step was the evaluation of the base cost loops. These are based on (4) and (5). Estimation errors can occur because of the parallel parts of the IU and MDU pipeline and stalling conditions for multi-cycle instructions. The second evaluation step was performed in un-cached mode. This results in a stalling of the fetch stage and insertion of *nop* instructions due to the slip mechanism. The third step compared the simulation of the CSO loops to the real power values. Table 1 surveys the results of the instruction dependent part of the energy model.

Figure 4 presents parts of the data dependent footprints of the *add* and *and* instructions. The maximum deviation of both is around 40% of the base costs.

**Table 1.** Evaluation of the instruction dependent part. Simulation results are compared to measurements on the chip

Program	Mean error [%]	Min error [%]	Max error [%]	Standarddeviation [%]
BC	0.09	-0.01	0.17	0.08
BC uncached	0.23	-0.98	0.90	0.38
CSO	-0.16	-4.88	2.94	1.52



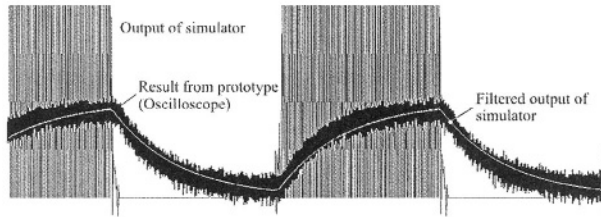
**Fig. 4.** Footprints for: (a)ADD; (b)AND

The power behavior of the *add* instruction depends on the input and output switching activity and is symmetric with regard to the source operands. The *and* instruction has a asymmetric behavior when the value of source operand two is higher than operand one. This leads to a power consumption dependent on the position of the least significant “one” in the operand value. Figure 4(b) compares the power values for source register one and two with the corresponding other register set to zero. This leads to a difference of 17% of  $P_{base}$  between the calculation of *and 1,0* and *and 0,1*.

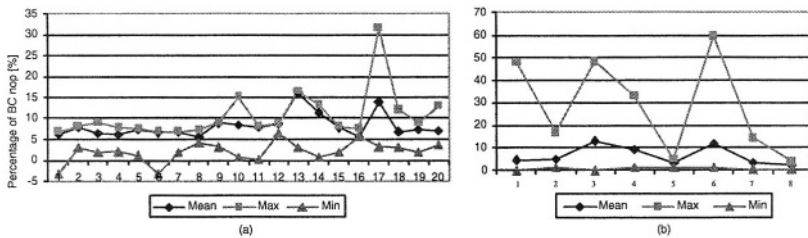
A benchmark suite comprising sorting algorithms, checksum calculation (CRC-16), the *data encryption standard* (DES) and a prime number generation was used to evaluate the accuracy of the entire energy model. The first column of Tab. 2 shows the differences between the measured average power consumption and the energy model only considering BC and IIC. Column two considers BC and CSO and column three takes into account all integrated energy models. The last column presents the timing error of the instruction-set simulator, which is important for total energy dissipation. The systematic underestimation of column one and two depict that the instruction dependent part consumes around 93% of total energy dissipation, whereas only three to five percentage are due to CSO. The data dependent part has also a small impact on total energy dissipation, but it is responsible for peaks in the power profile. The used coarse grain energy model does not consider all data dependent effects, which leads to the errors reported in column four of Tab. 2. The CRC16 and DES algorithm use extensively logic and shift instructions. A more accurate energy model for these

**Table 2.** Several benchmarks for simulator accuracy evaluation. All values are compared to measurement results on the chip

Program	Base costs only	Base costs with CSO	Error with all models	Standard-deviation	Timing-error
	$P_{total}$ [%]	$P_{total}$ [%]	$P_{total}$ [%]	$P_{total}$ [%]	$n_{total}$ [%]
Bubblesort	-12.1	-7.4	0.31	12.3	0.003
Selectionsort	-11.9	-7.5	-2.33	11.26	0.006
Quicksort	-11.3	-7.1	-3.18	11.27	0.099
Prime	-4.5	-3.1	-1.65	9.5	0.0001
CRC16	-11.8	-8.1	-4.39	11.4	0.007
DES	-16.9	-11.9	-3.09	15.9	0.45



**Fig. 5.** Power profile of the real chip compared with the simulator output



**Fig. 6.** Security evaluation of software: (a)unoptimized code; (b)optimized code

instructions and the corresponding data dependent inter instruction effects is necessary due to the high estimation error of three and four percent.

The microchips capacitance has to be considered for evaluation of the cycle accuracy. A low-pass filter function is used to consider all capacities. The cycle-accurate power profile produced by the power simulator is filtered by this digital low-pass filter. Figure 5 compares the output of the simulator and the filtered output with the power consumption measured with an oscilloscope. A loop containing *and* instructions with different data values was used to produce this “step function”.

As mentioned before, this simulator is designed to provide information for software design with regard to the power profile to increase software security.

An example for the usage of the simulator presents Fig. 6. Figure 6(a) shows the minimum, maximum and mean power profile values of a small instruction block of a loop in the DES key generation algorithm. Figure 6(b) shows the same loop with performance optimized code. The average power consumption of the optimized code is 1.6% lower compared to un-optimized code. But the standard deviation is 6.9% compared to 1.4% in the un-optimized case. Therefore un-optimized seems to be “more secure” than performance optimized code. A higher utilization of parallel hardware units in optimized code also causes peaks up to 60% in the power consumption profile. Peaks exceeding power limits can be easily identified and often removed by instruction reordering.

## 5 Conclusions

This paper has presented an instruction-set power simulator for secure software development. The energy model decomposed software energy dissipation into an instruction and a data dependent part. This allows control- and data-path energy characterization and modelling independent of each other. A coarse-grain energy model of a functional unit can be replaced with a fine-grain model transparently to the remaining model. The results for estimation accuracy were presented, but accuracy can be increased with a higher effort for modelling. Further work comprises software optimization for low power with regard to security.

## References

1. MIPS Technologies, Inc.: MIPS32 4KS<sup>TM</sup> Processor Core Family Software User's Manual. Revision 1.04, 2001.
2. Tiwari, V., Malik, S., Wolfe, A.: Power Analysis of Embedded Software: A First Step towards Software Power Minimization. *IEEE Trans. VLSI Systems*, Vol. 2, No. 4, pp.437-445, 1994.
3. Russel, J.T., Jacome, M.F.: Software Power Estimation for high-performance 32-bit embedded processors. in *Proc. Int. Conf. Computer Design: VLSI in Computers and Processors*, pp. 328-333, 1998.
4. Sami, M., Sciuto, D., Silvano, C., Zaccaria, V.: An Instruction-Level Energy Model for Embedded VLIW Architectures. in *IEEE TRans. on Computer-aided DEsign of Integrated Circuits and Systems*, Vol. 21, No. 9, 2002.
5. Steinke, S., Knauer, M., Wehmayer, L., Marwedel, P.: An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimization. in *Proc. of the PATMOS'01*, Yverdon-les-bains, Switzerland, 2001.
6. Sarta, D., Trifone, D., Ascia, G.: A Data Dependent Approach to Instruction Level Power Estimation. in *Low Power Design, 1999 Proc. IEEE Alessandro Volta Memorial Workshop*, pp. 182-190, Italy, 1999.

# The Impact of Low-Power Techniques on the Design of Portable Safety-Critical Systems\*

Athanasios P. Kakarountas<sup>1</sup>, Vassilis Spiliotopoulos<sup>1</sup>,  
Spiros Nikolaidis<sup>2</sup>, and Costas E. Goutis<sup>1</sup>

<sup>1</sup> VLSI Design Laboratory, Dpt. of Electrical & Computer Eng., University of Patras,  
Patras, GR – 26500, Greece  
{kakaruda, bspili, goutis}@ee.upatras.gr  
<http://www.vlsi.ee.upatras.gr>

<sup>2</sup> Electronics Dvn., Dpt. of Physics, Aristotle University of Thessaloniki,  
Thessaloniki, GR – 54006, Greece  
snikolaid@physics.auth.gr

**Abstract.** The application of typical low-power techniques on safety-critical systems may result in degradation of crucial safety properties of the system. However, existing techniques to address this impact, considered that low-power mode is applied for short time intervals. Thus, addressing degradation of the safety properties only on checkers was considered sufficient. A novel approach is introduced that considers both requirements for safe operation and low-power dissipation. The benefits of this approach are exhibited and a comparison to alternative approaches is offered.

## 1 Introduction

The tendency of international market to minimize size of the commonly used electronic devices is expected to move on in the field of safety-critical systems. A characteristic example of this category of systems is the field of portable medical devices. The motivation to minimize portable medical devices size can easily be understood thinking of the benefits offered to a lot of chronically disease patients. Our fellows having health problems are pushed aside without the chance to participate to common life's activities, due to the size of the portable devices and their small autonomy. Thus, most of the medical devices manufacturers aiming at this market group are developing new, elegant, small-sized wearable devices to assist them in day-to-day life. However, most of the currently available devices are based on safety techniques not suitable for the targeted degree of safety, as it is imposed by the international standards [1],[2]. Furthermore, not only the followed safety techniques are not considered suitable for portable safety-critical devices but also the low-power approaches for the implementation of these systems may result in poor safe operation levels [3].

In this paper, the impact of low-power techniques application on safety-critical systems is exhibited. A probabilistic evaluation measure, which was presented in [4] by

---

\* This work was supported by the COSAFE 28593 ESPRIT IV project of E.U. and MICREL Medical Devices S.A.

Lo and Fujiwara, is used to analyze the impact of low activity, on a circuit's network, on the safety properties of the circuit. The results are then extended to include the whole system. Novel structures are introduced to address this impact, meeting in parallel low-power dissipation and safe operation. The presented design approach is not limited to portable medical devices but also to any safety-critical application requiring low-power dissipation.

The rest of this paper is organized as follows. In section 2, typical low-power techniques are presented and the main philosophy to minimize low-power dissipation on higher levels of abstraction of a system is reported. Then, issues regarding testing during normal operation are presented, covering testing approaches that meet the standards' requirements [1],[2]. Additionally, a probabilistic measure, which is used to evaluate the designed system in terms of reliability and fault coverage in the field of time, is presented in section 3. In section 4, two design approaches are presented that address the impact of low activity of the system's bus on the checking mechanisms. An alternative approach, which addresses this impact to the whole system when in low-power mode, is introduced. In section 6, this work is concluded exploiting the experimental results of section 5.

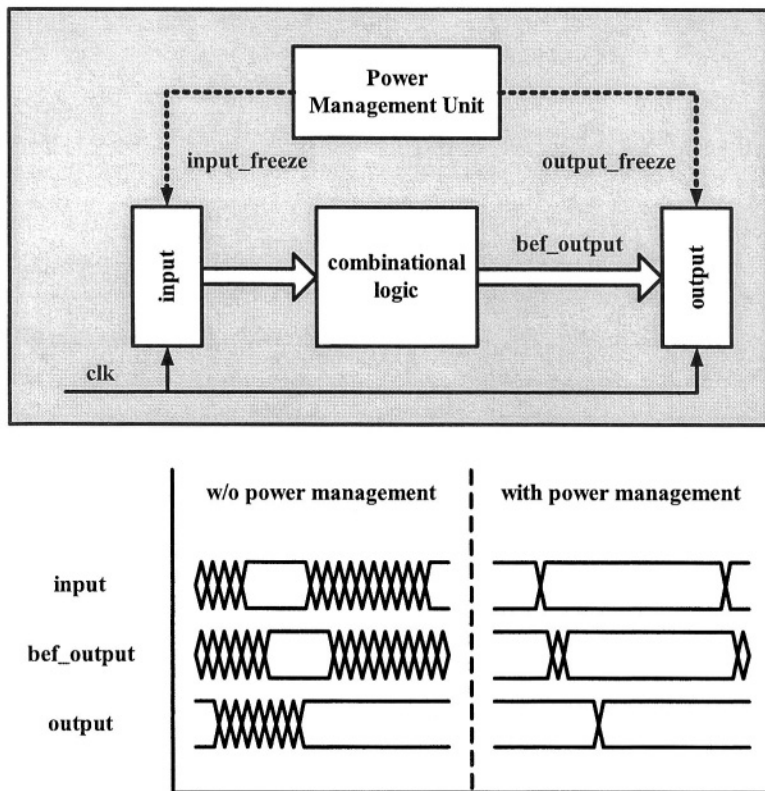
## 2 Typical Design Techniques for Portable Safety-Critical Systems

Several techniques are commonly used to develop portable safety-critical systems. They can be discriminated in two categories; techniques aiming to reduce power dissipation and techniques aiming to achieve a higher degree of safe operation. These two categories are presented in the rest of this section, mainly focusing on details that exhibit the philosophy that lies beneath.

### 2.1 Low-Power Techniques

The most common and broadly met technique used to dynamically manage power is the reduction of the activity on the units' inputs. A plethora of methods have been proposed on this technique but the main discipline is the introduction of local control signals to manipulate the functionality of the circuit. Global signals like clock or reset can be gated by a local control signal and their propagation in the rest of the system can be dynamically controlled. Alternatively local enable signals can be produced to activate and deactivate a circuit of the system, when it is needed. In Fig. 1 these approaches are illustrated as well as the results of power management on the lines.

Another method is the re-scheduling of the tasks to be executed (e.g. algorithm transformation) in order to reduce the activity on large global buses. This technique is similar to the latter mentioned, due to its aim to control the activity on lines. Although it is applied on an alternative level of abstraction the targeted effect is almost the same. Similar results can also be obtained if alternatively to the typical binary system a more appropriate for low-power dissipation is preferred. For example, a logarithmic number system is appropriate for the arithmetic power function, resulting in low design complexity and simultaneously reduced power dissipation compared to the binary system.



**Fig. 1.** Application of power management control on a circuit’s input and output. The activity of the circuit’s lines with and without the power management unit is illustrated in the waveforms

Alternative approaches that aim to control power dissipation by manipulating characteristics of the integration process are not considered. However, these techniques also aim to minimize line activity. More information about the latter techniques can be found in [5],[6],[7],[8]. Concluding this section, it can be said that the common approach to minimize power dissipation of a system is the control of its lines’ activity.

## 2.2 Online Testing Techniques

On-line testing procedures are widely common in nowadays systems to enable testing during system operation. The majority of on-line testing techniques is focused on the so called non-concurrent on-line testing. During this mode of operation either the system is halted to perform a testing procedure, or non operable subsystems at a time instance, are subject to a testing procedure. Special circuits are embedded to the system for testing purposes, commonly known as the Built-In Self-Testing (BIST) units [9]. Research concerning optimization of on-line testing in terms of silicon, power dissipation and performance has advanced system design resulting in near automation of system analysis and BIST unit synthesis.



However, non concurrent on-line testing is not suitable for safety critical systems that require safe and uninterrupted operation. According to the international standards [1],[2] this category of applications may be tested using concurrent on-line testing. This special type of on-line testing is based on information and/or hardware redundancy applied on the Circuit under Test (CuT), resulting in structures known as self-checking circuits. The most common technique is the use of multi-channeled structures [10] resulting in a proportional increase of power dissipation and silicon. Thus cost is significantly increased while autonomy of safety critical devices is reduced. However in [11] it was presented that the use of information redundancy is ideal to meet safe operation specifications while low silicon and power dissipation requirements can be achieved.

An alternative design approach of concurrent on-line testing is based on a special type of circuits, the Totally Self-Checking circuits (TSC hereinafter) [12]. This type of circuit has the property for self- test and check; detecting error occurrence even in itself. In contrast to non concurrent on-line testing techniques, concurrent on-line testing doesn't require a special module to generate the test vectors; the system's inputs and buses serve as the required test vectors. The previous design approach and the one based on the TSC circuits are the ones that achieve the highest degree of safe operation, detecting errors in near real-time. However, there is the need to develop design approaches and techniques so that safety-critical systems embedding concurrent on-line testing can also meet other design constraints and requirements of the application.

### 3 A Probabilistic Measure for Totally Self-Checking Circuits

The typical fault model widely used is based on the single fault occurrence. According to this fault model, faults occur one at a time, and the time interval between the occurrences of any two faults is sufficiently long so that all input code words are applied to the circuit. The information bits derived from the data bits and the encoding bits are referred hereinafter as code word. In [4] a complete probabilistic measure is proposed for the SC circuits, which is analogous to the reliability of fault-tolerant systems. The probability to achieve TSC Goal in a circuit is defined as follows:

$$TSCG(t) = \text{Prob}\{\text{TSC goal is guaranteed at cycle } t\} = R(t)+S(t) . \quad (1)$$

where  $R(t)$  represents the conditional probability that the circuit guarantees the TSC goal when the circuit is fault-free at cycle  $t$ , and  $S(t)$  is the conditional probability that a circuit guarantees the TSC goal when faults occur before or on cycle  $t$ . The term  $S(t)$  is calculated by adding all the probabilities that the circuit is fault-secure and/or self-testing with respect to the first fault and the probabilities that a fault is detected before a second one occurs. The term  $R(t)$  is the qualitative representation of the reliability of a circuit and it is relative to the constant failure rates. Due to the limitations of the paper size, the reader is advised to further read [4], where the mathematical analysis of the model is found, as well as to read [3], where the controversial nature of the concurrent on-line testing and low-power dissipation requirements of an application are showed off.

In [3] it is stated that to keep the  $TSCG(t)$  of a circuit to a high level, its input bits must present a certain activity. In addition, a system that includes TSC checkers must

maintain the  $TSCG(t)$  to the higher levels. The FSES tool, which is presented in [13], was used to generate the  $TSCG(t)$  of every design approach. The innovation, which is introduced in this paper, is a novel design technique for the co-existence of power management units and Totally Self-Checking circuits, of the whole system and not only of the TSC checkers, as it is proposed in [3].

## 4 Design Approaches to Address the Impact of Low-Power Techniques Application on Portable Safety-Critical Systems

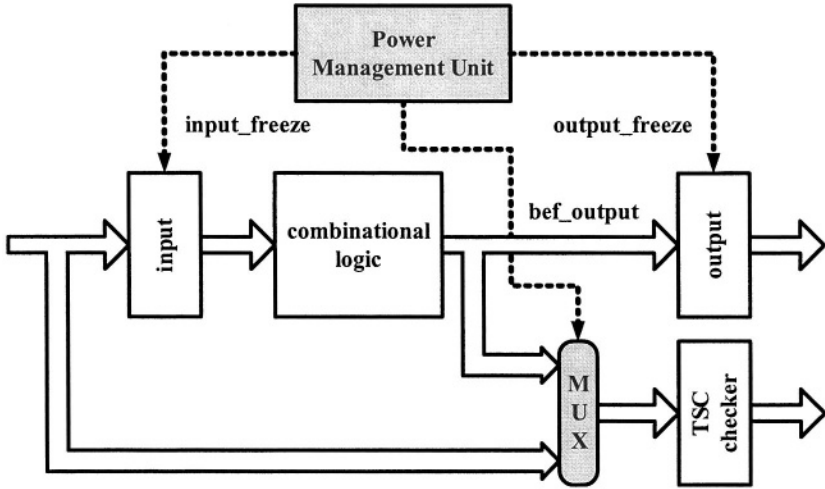
The application of low-power techniques, aiming to control activity on the lines of a circuit, using structures as the one shown in Fig. 1, results in degradation of the TSC circuit's reliable and safe operation. The techniques, presented in [3], were targeting the embedded TSC checkers. Although this approach offers continuous monitoring of the TSC checkers operation, it didn't guarantee error detection on the rest of the circuit, during a low-power mode. This turned most faults to be masked, without having the appropriate time interval to detect it before a new error occurred. The latter violates the hypothesis of the single stack-at fault model as it was mentioned in the previous section. Thus, although the approaches, presented in [3], were addressing degradation of the TSC property of a circuit during low-power mode, in a sufficient degree, a novel design approach is required to cover faults occurring to the whole circuit. In this section the design approaches of [3] are presented in short and then a new design approach is introduced that can result in the optimum solution.

### 4.1 Previous Work

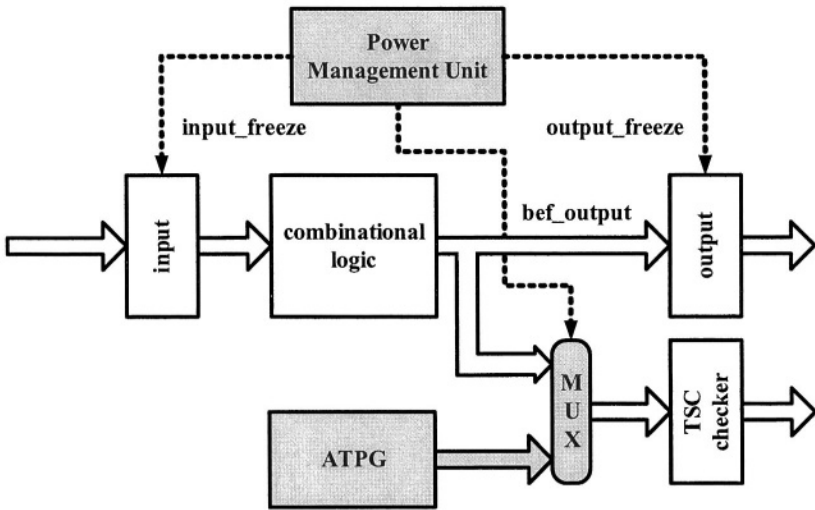
The design approaches that have been proposed and/or being used until nowadays are based on the works presented in [10] and [3]. The multiplication of the circuit under test was proposed by von Neumann in [10]. Thus, multi-channeled structures are generated, with outputs continuously monitored and compared. During a low-power mode, when the output is rarely changed due to a fault, fault occurrence is almost not detectable. Additionally, the power and area penalties introduced by these structures are high. This approach is quite common to safety critical systems requiring low design complexity and high performance.

In [3], two design approaches were presented to address degradation of the TSC property of the TSC checkers during a low-power mode of operation. One technique is based on input re-usability, where input test vectors were acquired from an alternative input source than the initial. This technique exploits the possibility that a similar encoding scheme is used, either as an input or an output, by another circuit. Thus, it can serve as the input required by the TSC checkers, as shown in Fig. 2. This technique presents the lowest design complexity and introduces the lowest area overhead.

On the other hand, when input re-usability cannot be applied, the by-pass technique is proposed to fill the requirement for continuous testing. A structure based on this technique is illustrated in Fig. 3. This technique requires that the input vectors are generated by an Automatic Test Pattern Generator and feed to the TSC checker through a multiplexer. Thus, area overhead is increased compared to the previously mentioned input re-use technique.



**Fig. 2.** The input re-use technique to address low activity of the TSC checker lines. This technique requires only the use of a multiplexer to feed the TSC checker with an alternative input, while in low power mode



**Fig. 3.** The input by-pass technique to address low activity of the TSC checker lines. This technique requires a Built-In Automatic Test Pattern Generation Unit to feed the TSC checker, while in low power mode

### 4.2 Proposed Design Approach

An alternative Automatic Test Pattern Generator (ATPG hereinafter) to that required by the by-pass technique can be used to include the rest of the circuit. Additionally, a special circuit is introduced to control the rate of test vector generation. In general the topology is similar to the. The main difference lies in the test vector generation rate.

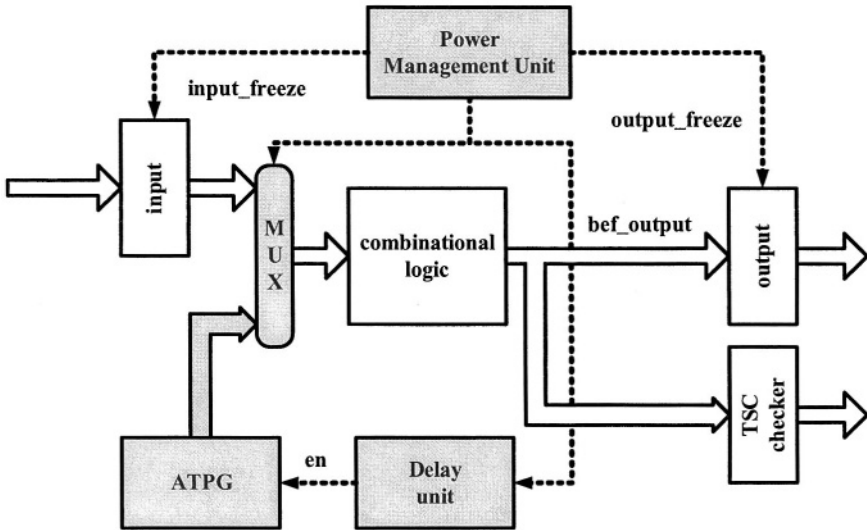


Fig. 4. The proposed technique that includes the whole circuit, guaranteeing safe operation and low-power dissipation during low-power mode

In the case of an on-line BIST, test vectors are required to be generated with the highest rate. This results in high speed testing and fast error detection. However, speed is significant due to the requirement to fulfill testing procedure before the system resumes to normal operation. In contrast, in the case of concurrent on-line testing normal operation is simultaneously the testing operation mode. Thus, continuous feed of inputs is required, as proved in [3] and [4].

The arrival rate of the input test vectors can vary, resulting in a wide range of  $TSCG(t)$ , as presented in [4]. Exploiting the evaluation results derived from the FSSES tool, presented in [13], a targeted TSC circuit can be characterized regarding a rate threshold for which the circuit meets the requirements for safe operation. The output of the Automatic Test Pattern Generation is then delayed, by a factor derived from a selected test vector generation rate. If this rate is not sufficient to keep the circuit to high safety levels, another one is selected until the requirements regarding safe operation are met.

The topology of the proposed design approach is illustrated in Fig. 4. As it can be seen, the delay unit is responsible to generate the enable control signal for the ATPG to generate the next test vector. The proposed design approach can be expanded to the whole safety-critical system, offering low power dissipation at the selected high safety level.

## 5 Experimental Results

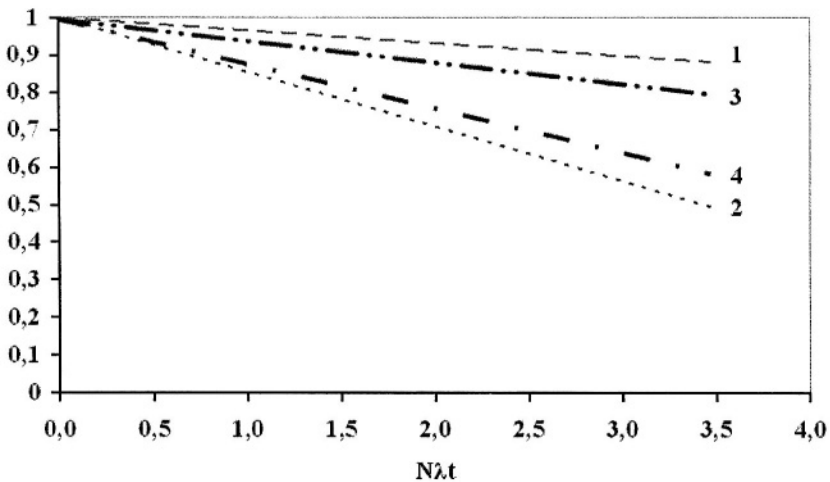
A TSC carry-ripple adder [13] can serve as an example, in order to evaluate the benefits offered by the proposed design approach. The FSSES tool, reported a  $TSCG(t)$  as illustrated in Fig. 5 (curve 1). In this case, the input test vector arrival rate is ideal, assuming that at each clock cycle the input changes its value. However, if a low-

power technique is applied, which reduces significantly line activity (less than 10% of the initial activity), then the  $TSCG(t)$  that is reported is showed in Fig.5 (curve 2).

Trying to address the degradation of the  $TSCG(t)$ , the TSC adder was re-designed using the proposed design approach and the by-pass technique, as it was presented in [3]. The resulted  $TSCG(t)$  is illustrated in Fig. 5 (curves 3 and 4 respectively). From the illustrated curves, it can be derived that although the by-pass technique addresses the  $TSCG(t)$  degradation, the benefits are negligible compared to the proposed design approach that elevates the  $TSCG(t)$  levels to an acceptable level (almost to 90% of the initial).

The COSAFE microcontroller [13], can serve as a second example to show off the power dissipation penalty introduced by the proposed design approach. Implementing the microcontroller using the by-pass technique, a 7.5mW maximum power dissipation (upper bound) was observed. Also, the fault coverage was reduced to 98% after special design efforts, while the  $TSCG(t)$  was degraded significantly in the field of time (56% of the initial).

However, when the proposed technique is applied, the fault coverage is not reduced, while  $TSCG(t)$  tends to 89% of the initial one. The test input arrival rate that is used as a delay, when the system enters a low-power mode, is equal to 20% of that in normal operation. As a result, the maximum power dissipation is increased, reaching a value of 8.1 mW. However, this trade off meets the international standards requirements, compared to the latter COSAFE implementation, which was based on the by-pass technique.



**Fig. 5.** The  $TSCG(t)$  of: (1) a TSC adder with an ideal arrival rate of the input test vectors, (2) the same TSC adder under low-power mode (without altering the arrival rate of the input test vectors), (3) the same TSC adder under low-power mode (with a 23% of the ideal arrival rate of the input test vectors) and (4) the same TSC adder re-designed using the by-pass technique

## 6 Conclusions

In contrast to the design techniques, presented in [3], to address degradation of a circuit's TSC checkers during low-power mode, a novel technique is proposed that addresses degradation of the whole circuit. Exploiting the results offered by the tool FSES, presented in [13], and the probabilistic evaluation model, presented in [4], the trade-off between low line activity and high level of safe operation can be optimized, at the desired degree.

This work, introduces for the first time a realistic design approach to achieve low-power dissipation in a TSC circuit, aiming solely to the circuit's power characteristics. The presented approach offers low-power dissipation, meeting the targeted high levels of safe operation. Additionally, the whole circuit is subject to concurrent on-line testing, in contrast to previous approaches, which targeted to guarantee the circuit's TSC checker's safe operation.

## References

1. DIN V VDE 0801: Principles for computers in safety-related systems. 1990
2. DIN V 19250: Measuring - Controlling - Regulating; basic safety considerations for the MCR protection appliances. 1990
3. Kakarountas, A.P., Papadomanolakis, K., Nikolaidis, S., Soudris, D., Goutis, C.E.: Confronting Violations of the TSCG(t) in Low-Power Design. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'02). (2002) 2606–2609
4. Lo, J.-C., Fujiwara, E.: Probability to Achieve TSC Goal. In: IEEE Transactions on Computers, Vol. 45. (1996) 450–460
5. Raghunathan, A., Jha, N.K., Dey, S.: High-Level Power Analysis and Optimization. Kluwer Academic Publishers (1998)
6. Chandrakasan, A., Brodersen, R.W.: Low Power Digital CMOS Design. Kluwer Academic Publishers (1995)
7. Pedram, M.: Power Minimization in IC Design: Principles and Applications. In: ACM Transactions on Design Automation of Electronic Systems. (1996) 3–56
8. Rabaey, J.M., Pedram, M.: Low Power Design Methodologies. Kluwer Academic Publishers (1995)
9. Abramovici, M., Breuer, M.A, Friedman, A.D.: Digital systems testing and testable design. IEEE Press, New York (1992)
10. von Neumann, J.: Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components. Princeton University Press (1956)
11. Kakarountas, A.P., Papadomanolakis, K. S., Spiliotopoulos, V., Nikolaidis, S., Goutis, C. E.: Designing a Low-Power Fault-Tolerant Microcontroller for Medicine Infusion Devices. In: Proceedings of Design, Automation & Test in Europe (DATE'02). (2002)
12. Lala, P.K.: Self-Checking and Fault-Tolerant Digital Design. Morgan Kaufmann Publishers (2001)
13. Papadomanolakis, K.S., Kakarountas, T., Tsoukalis, A., Nikolaidis, S., Goutis, C.E.: ASIP Specification and Functionality Description. In: CoSafe project public deliverable CO-SAFE D2IC&D. (1999)

# Modular Construction and Power Modelling of Dynamic Memory Managers for Embedded Systems\*

David Atienza<sup>1</sup>, Stylianos Mamagkakis<sup>3</sup>, Francky Catthoor<sup>2\*\*</sup>,  
J.M. Mendias<sup>1</sup>, and D. Soudris<sup>3</sup>

<sup>1</sup> DACYA/U.C.M, Avda. Complutense s/n, 28040 - Madrid, Spain

<sup>2</sup> IMEC vzw, Kapeldreef 75, 3000 Leuven, Belgium

<sup>3</sup> VLSI Design Center-Democritus Univ., Thrace, 67100 Xanthi, Greece

**Abstract.** Portable embedded devices must presently run multimedia and wireless network applications with enormous computational performance requirements at a low energy consumption. In these applications, the dynamic memory subsystem is one of the main sources of power consumption and its inappropriate management can severely affect the performance of the system. In this paper, we present a new system-level approach to cover the large space of dynamic memory management implementations without a time-consuming programming effort and to obtain power consumption estimates that can be used to refine the dynamic memory management subsystem in an early stage of the design flow.

## 1 Introduction

Recently, with the emerging market of new portable devices that integrate multiple services such as multimedia and wireless network communications, the need to efficiently use Dynamic Memory (DM from now on) in embedded low-power systems has arisen. New consumer applications (e.g. 3D video applications) are now mixed signal and control dominated. They must rely on DM for a very significant part of their functionality due to the inherent unpredictability of the input data, which heavily influences global performance and memory footprint of the system. Designing them using static worst case memory footprint solutions would lead to a too high overhead in memory footprint and power consumption for these systems [5]. Also, power consumption has become a real issue in overall system design (both embedded and general-purpose) due to circuit reliability and packaging costs [14]. Thus, optimization in general (and especially for embedded systems) has three goals that cannot be seen independently: memory footprint, power consumptions and performance.

Since the DM subsystem heavily influences performance and is a very important source of power consumption and memory footprint, flexible system-level implementation and evaluation mechanisms for these three factors must be available at an early stage of the design flow for embedded systems. Unfortunately, general approaches that integrate all of them do not exist presently at this level of abstraction for the DM managers implementations involved.

\* This work is supported by the Spanish Government Research Grant TIC2002/0750 and the European founded program AMDREL IST-2001-34379.

\*\* Also professor at ESAT/KUL.

Current implementations of DM managers can provide a reasonable level of performance for general-purpose systems [15]. However, these implementations do not consider power consumption or other limitations of target embedded platforms where these DM managers must run on. Thus, these general-purpose DM managers implementations are never optimal for the final target platform and produce large power and performance penalties. Consequently, system designers must face the need to manually optimize the implementations of the initial DM managers in a case per case basis and without detailed profiling of which parts within the DM managers implementations (e.g. internal data structures or links between the memory blocks) are the most critical parts (e.g. in power consumption) for the system. Moreover, adding new implementations of (complex) custom DM managers often proves to be a very programming intensive and error prone task that consumes a very significant part of the time spent in system integration of DM management mechanisms. In this paper, we present a new high-level programming and profiling approach (based on abstract derived classes or mixins [11] in C++) to create complex custom DM managers and to evaluate their power consumption at system-level. This approach can be used to effectively obtain early design flow estimates and implementation trade-offs for system developers.

The remainder of the paper is organized as follows. In Section 2, we describe some related work. In Section 3 we present the proposed construction method for DM managers and the necessary profile framework to obtain detailed power consumption estimations. In Section 4, we shortly introduce our drivers and present the experimental results obtained. Finally, in Section 5 we draw our conclusions.

## 2 Related Work

In the software community much literature is available about DM management implementations and policies to be used in general-purpose systems [15]. In memory management for embedded systems [8], the DM is usually partitioned into fixed blocks to store the dynamic data. Then, the free blocks are placed in a single linked list [8] due to performance constraints with a simple (but fast) fit strategy, e.g. first fit or next fit [15]. Also, in recent real-time operating system synthesis approach for embedded systems [9], dynamic allocation is supported with custom DM managers based on region allocators [15] for the specific platform features.

Another recent method to improve performance of the DM subsystem is to simulate the system with partially customizable DM management frameworks. In [1], a C++ framework where you can partially redefine some functionality (e.g. `malloc()` function) of the DM subsystem has been proposed, but it does not consider changes in the implementation structure of DM managers. Also, [2] outlines an infrastructure to improve performance of general-purpose managers. However, its definition for performance exploration of general-purpose DM managers restricts its flexibility to isolate and explore the influence of basic implementation parts of custom DM managers (e.g. fit algorithms [15]) for other metrics (e.g. power consumption).

Regarding profiling of the DM subsystem, recent work has been done to obtain profiling from assembly code and even a higher abstraction level [14]. Nevertheless, current methods do not yet include detailed enough run-time profiling analysis to evaluate the



influence on power consumption of the basic implementation components of DM managers (e.g. fit algorithms or maintenance data structures). Hence, they are not sufficient for modern dynamic applications. In addition, several analytical and abstract power estimation models at the architecture-level have received more attention lately [3]. However, they do not focus on the DM hierarchy of the system and the power consumed by DM managers at the software level.

### 3 Construction and Profiling of Layered DM Managers

The implementation space of DM managers is very broad and we need to cover it in a flexible and extensible way. Therefore, we use a C++ approach which combines abstract derived classes or mixins [11] with template C++ classes [13]. In the remainder of the text, we use the definition of mixins as used in [11]: a method of specifying extensions of a class without defining up-front which class exactly it extends.

In Figure 1 we show the basic concepts used in this approach. In the first example of Figure 1, a subclass of `SuperClass` is declared with `SuperClass` itself being a template argument and consequently also the subclass is defined. Then, `MyMixin` class is reusable for one or more parent classes that will be specified in the different instantiations of `MyMixin` class. In the second example of Figure 1, another class is defined (i.e. `MyClass`), where the template argument is not used as a parent class, but instead as internal private data members. In our approach, as we show in the following sections, the first concept is used to refine the functionality of the custom DM managers and the second one is used to specify its main components, e.g. heaps, data structures, etc. As a result of this very modular approach, we can combine both concepts to build very customized DM managers starting from their basic structures (e.g. data structures, fit algorithms, etc.) and later on add detailed profiling for each of these basic structures. In conventional approaches [1,2,15] this kind of modeling and detailed profiling of basic structures of DM managers is not possible. The main reason is that in such approaches the DM managers are built as complex software engineering modules where all the different components (e.g. fit algorithms, data structures) are combined and deeply embedded in their implementations. Thus, only a limited number of variations in the final

```

// Example 1: Basic MyMixin Class
template <class SuperClass>
class MyMixin : public SuperClass{
    // MyMixin class definitions };

// Example 2: Abstract parent class inside MyClass
template <class SuperClass>
class MyClass{
    SuperClass* data;
    // template class definitions };

```

**Fig. 1.** Parametrized Inheritance used with mixins in C++

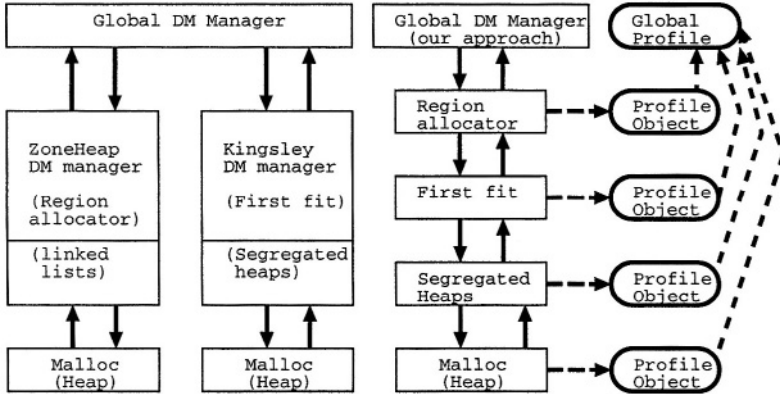
implementation can be explored by the designer due to the time-consuming effort of reprogramming their global structures.

### 3.1 Construction of Layered DM Managers with Profile Support

Using the previously explained concepts of abstract derived classes and template C++ classes, we have redefined the library proposed in [2] and integrated our own profile framework (see Subsection 3.2 for a detailed description of this framework) to be able to explore and profile power consumption, memory footprint and memory accesses in the basic construction categories we distinguish for DM managers. These categories are the following: creating block structures, pool division based on different criterion, fit algorithms, order of the blocks within the pools (address, size, etc.), coalescing (or merging blocks) and splitting blocks [16]. In Figure 2 we show an example of the construction of a DM manager with basic blocks and how our profile framework can be added to any part of it with a fine granularity. First, the basic heaps of the manager and the basic allocation blocks requested to the system are defined (class `BasicHeap` in Figure 2). Second, the two basic data structures to test within the manager, i.e. double linked lists (`DLLList`) and binary trees (`BTree`) are implemented. Third, they are instantiated for the basic sizes to use in the DM manager. Then, the profile objects (see Subsection 3.2 for

```
// Basic blocks for heap requests to the system
template<typename MyT>
    class BasicHeap: public TypeClass<MyT,mheap>;
// Data types of the dynamic memory manager
template<typename MyT, class SuperClass>
    class DLLList { // Implementation of a double link list
        // list with generic data size MyT };
template<typename MyType, class SuperClass>
    class BTree { //Implementation of binary tree
        // with generic data size MyT };
// Two basic data types instantiated for the memory manager,
class I_DLLList : public DLLList<int, BasicHeap<int> >{};
class D_BTree : public BTree<double, BasicHeap<double> >{};
// Declaration of profile objects to profile the manager
_profile *prof1, *prof2, *profileGlobal;
// Memory manager with 2 segregated-fit lists of different data types,
// best or fit policy and profile objects
class DMMHeap: public
    SegLists<profileGlobal, // Global profile object
        list_Sizes, // List of sizes for the segList
        numElemFirstList, // Number of lists with type 1st segList
        BestFit<I_DLLList<prof1> >, // 1st segList
        FirstFit<D_BTree<prof2> > // 2nd segList
    > {};
```

Fig.2. Example of custom DM manager with profiling objects



**Fig. 3.** Example of the structure of a custom DM manager. On the left, built with the approach proposed in [2] where the layers are really interdependent. On the right, our own approach

more details about their use) to obtain the necessary information about power consumption, memory footprint and memory accesses are created. Finally, the DM manager is created as a combined structure of two different segregated lists [15], which are formed by different dynamic data structures inside (DLList or BTree) and different allocation policies (best fit or first fit) [15].

As Figure 3 shows, we can build custom DM managers from its basic blocks and obtain power estimations from them in a much more flexible way than the structure proposed in [2]. For example, if due to the characteristics of the final system it is necessary to combine two different allocation strategies from two different general-purpose managers in the same global manager, using [2] we would need to create both DM managers and combine them later as independent heaps because a great part of the structure of each DM manager is fixed. On the contrary, our approach allows to create a global DM manager using just a single heap. This global manager would be composed by several intermediate layers that define a very customized and flexible implementation structure including the two different allocation strategies in the same heap. This example is depicted in Figure 3. Thus, we can merge the two allocation heaps saving memory footprint because the memory can be reused for both. Also, our final structure is simpler to compose because parts of the maintenance data structures can be shared and accessed simultaneously (e.g. pointers of the memory blocks). Hence, the number of memory accesses and eventual power consumption of the DM manager are reduced (as shown in Section 4, Table 6 with `ObstLea`). Finally, note that any modification in the implementation structure of the heap only requires to substitute a very limited number of layers. Therefore, the programming effort to do it is reduced heavily.

### 3.2 Structured Profile Framework and Power Model

Apart from simplifying the effort of exhaustively covering the implementation space of DM management, the presence of multiple layers in the DM managers also gives a lot

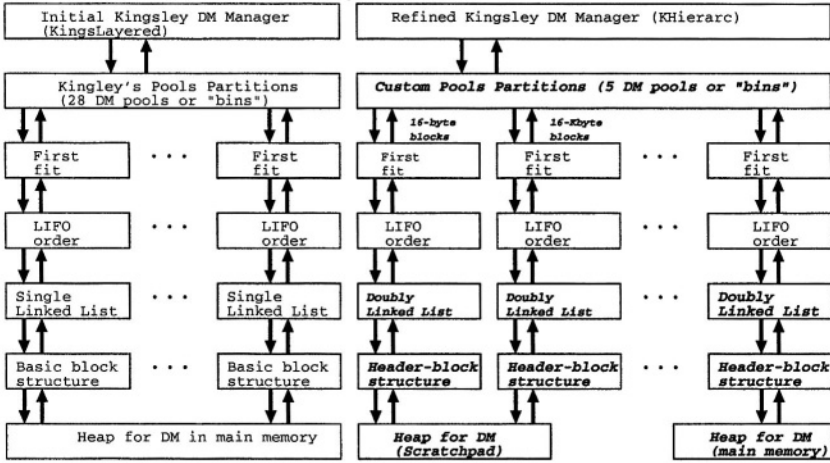
of flexibility to profile their characteristics at different levels, e.g. memory accesses of each implementation layer in the internal structure, as Figure 3 indicates. This detailed profiling is required for a suitable optimization (e.g. for power consumption) of the DM manager since small changes in the implementation of some layers can completely change the global results of the DM manager in the system, even if most of the implementation structure remains the same. For example, as we explain in our case studies in Section 4, a LIFO reuse strategy of the blocks can produce completely different results compared to a FIFO reuse strategy. However, this detailed profiling at the level of the individual layers in the DM manager requires a new system-level profiling framework that is flexible enough to handle all kinds of combinations between the layers. Since more than one layer can constitute the part of the manager to measure, the profiling information must be grouped and cannot be collected at one layer only. Therefore, we have integrated a similar approach to the one proposed in [5] for complex dynamic data types. As Figure 3 depicts, it consists of an object-oriented profiling framework that decouples this information from the class hierarchy of the DM managers, providing accurate run time information on memory accesses, memory footprint, timing information and method calls. Then, we can use this information to obtain power model estimates for the DM managers using a realistic model of the underlying memory hierarchy in a post-execution phase. Thus, the application runs at its normal speed and the total evaluation time for one DM manager is reduced from several hours of simulation in typical cycle-accurate simulations to few minutes including the post-execution phase.

For this post-execution phase, we use an updated version of the CACTI model [4], which is a complete energy/delay/area model for embedded SRAMs that depends on memory footprint factors (e.g. size or leaks) and factors originated by memory accesses (e.g. number of accesses or technology node). The main advantage of CACTI is that it is scalable to different technology nodes. For the results shown in Figure 5, Figure 6 and Table 1, we use the  $.13\mu\text{m}$  technology node. Note that any other model for a specific memory hierarchy can be used just by replacing this power module in the tools.

## 4 Case Studies and Experimental Results

We have applied the proposed method to three case studies that represent different modern multimedia and network application domains: the first case study is part of a new 3D image reconstruction system, the second one is a 3D rendering system based on scalable meshes and the third one is a scheduling algorithm from the network domain. All the results shown are average values after a set of 10 simulations for each application and DM manager implementation. The obtained results (e.g. execution time, power consumption estimations) were all very similar (variations of less than 2%).

The first case study is a 3D vision reconstruction application [10] (see [12] for the full code of the algorithm with more than 1 million lines of high level C++). It heavily uses DM due to the variable features of input images. This implementation reconstructs 3D images by matching corners [10] detected in 2 subsequent frames. The operations done on the images are particularly memory intensive, e.g. each matching process between two frames with a resolution of  $640 \times 480$  uses over 1Mb, and the accesses of the algorithm (in the order of millions of accesses) to the images are randomized. Thus, classic image

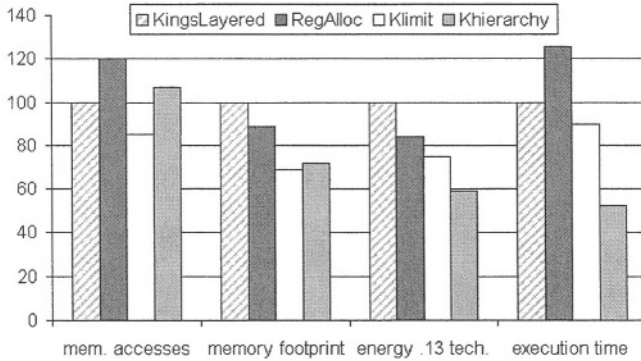


**Fig. 4.** On the left, initial implementation structure of Kingsley DM manager with our approach. On the right, our final refined version of it, i.e. KHierarc, main changes in bold

access optimizations as row-dominated accesses versus column-wise accesses cannot be applied to reduce the memory accesses and power consumption values.

For this case study, we have implemented and profiled several DM managers starting from a general-purpose one and refining its implementation using our approach. First of all, we have implemented one of the fastest general-purpose managers, i.e. Kingsley DM manager [15] (KingsLayered in Figure 5). But it has a considerable fragmentation due to its use of power-of-two segregated-fit lists [15]. A graphical representation of its implementation structure with our layered-approach is shown in Figure 4. As Figure 5 shows, its memory footprint is larger than any other DM manager in our experiments, but its total execution time is faster than the new region-semantic managers [15] frequently found in current embedded systems, i.e. RegAlloc in Figure 5.

After implementing and profiling these two generic DM managers, we have observed that most of the accesses in Kingsley occur in just few of the “bins” (or memory pools of the heap) [15], due to the limited range of data type sizes used in the application [5]. Therefore, we try to reduce its memory waste by modifying its design with our layers and by limiting the number of bins to the actual sizes used in the application (5 main sizes), as Figure 4 shows at the top in its right graph. This variation is the most significant change in its internal structure and allows to define the custom manager marked as KLimit in Figure 5. We can see that its improvement is already significant in energy dissipated per matching process of two frames. Then, we try to improve its structure even further with our layered approach. Thus, the bins that produce most of the accesses (the bins for allocation sizes of 16 bytes with the maintenance information of the manager and the data types of blocks of 16 Kbytes) are easily separated using our infrastructure of layers from the global heap used in the manager. They are now handled in a different and small heap (57 Kbytes) that is placed permanently in the scratchpad, as Figure 4 indicates at



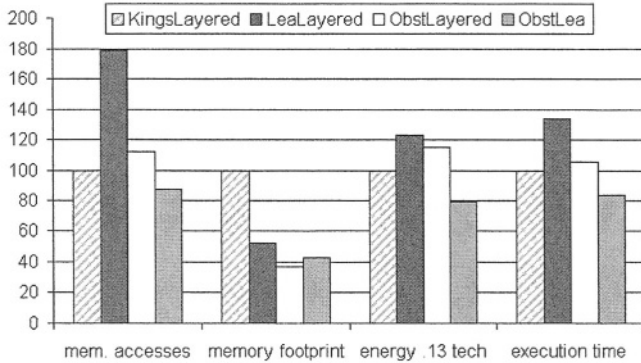
**Fig. 5.** Profiling results of different DM managers (normalized to Kingsley, i.e. KingsLayered) in the 3D Image Reconstruction System per each matching process of two frames

the bottom in its right graph. This custom DM manager, which is optimized according to the final memory hierarchy, is depicted on the right side of Figure 4 and marked as KHierarchy in Figure 5.

The latter figure shows that KHierarchy DM manager has increased its total amount of memory accesses and total memory footprint compared to KLimit, but most of the accesses of the manager are now in the on-chip scratchpad memory (i.e. 95%). Also note that the increase in memory footprint is mainly due to data copied between the different levels of the memory hierarchy and this increase is not really significant comparing it with the accesses saved to the off-chip memory. Hence, we can observe that the total energy dissipation and execution time of this custom memory manager have decreased enormously compared to the other ones in Figure 5.

The second case study is a realistic example of new 3D rendering applications where scalable meshes [6] are used to adapt the quality of each object displayed on the screen according to the position of the user watching at them at each moment. In this case we have implemented with our approach one of the best general-purpose DM managers (in terms of the combination of speed and memory footprint) [15,2], i.e. LeaAllocator v2.7.2 [15]. Apart from it, we have used Kingsley [15] to compare both in memory footprint, memory accesses and total energy consumption figures. Also, we have tested a well-known custom DM manager optimized for a stack-like DM behavior, i.e. Obstacks [15]. As Figure 6 shows, the Lea Allocator (LeaLeayered) obtains average values for a certain trade-off in performance and memory footprint. However, its energy dissipation is very high due to the additional accesses for its complex maintenance structure. Also, Figure 6 indicates that Kingsley suffers from high fragmentation, but produces a lot less accesses. Thus, with completely different characteristics, both managers are close in their final figures for power consumption. Also, Obstacks has few accesses during the first 3 phases of the rendering process due to their partial stack-like behavior, but suffers from high penalty in memory accesses and energy dissipation per frame in these last three phases. Hence, its final values are not as good as expected.

These results suggest the convenience of a custom DM manager that combines the behaviour of Obstacks with Lea in the last three phases. We have built it with our



**Fig. 6.** Profiling results of different DM managers (normalized to Kingsley, i.e. KingsLayered) in the 3D Rendering System for one object in one frame

approach (3 weeks) and it is marked as *ObstLea* in Figure 6. This figure shows that this new manager accomplishes very good overall results. Also, our DM manager designs have a similar execution time (differences of less than 6% in execution time) compared to the original (manually-designed) versions of *Obstacks* and *Lea*, but with a clear improvement in design complexity on our side. Our version of the *Lea* Allocator has around 700 lines of C++ code instead of more than 20000 lines of C code as in the original *Lea* implementation, and 400 lines of C++ code for our version of *Obstacks* compared to 2500 lines approximately of its state-of-the-art implementation.

The third case study presented is the Deficit Round Robin (DRR) scheduling application taken from the NetBench benchmarking suite [7]. It is a fair scheduling algorithm implemented in many routers today where the scheduler visits each internal queue and forwards the corresponding packets according to their size and priority. The DRR application was profiled in our results for realistic input traces of 100000 packets.

For this case study, we have implemented and profiled using our approach different versions of the same basic structure in the DM manager, i.e. Power-of-two segregated-fit lists [15], without coalescing or splitting services. For speed requirements, we have started from the structure of the general-purpose Kingsley manager and have implemented two variations of it. One uses a LIFO single linked freelist (*Kings+LIFOSLL* in Table 1) and the other one a FIFO double linked freelist (*Kings+LIFODLL* in Table 1). Finally, we have also designed a custom DM manager with FIFO single linked list structure using a segregated fit algorithm (*SegFitSSL FIFO* in Table 1). Our results show that not only the global policy of the manager is important, but also a careful study of the ideal structure of reuse, data types, etc. inside the managers. The results obtained are shown in Table 1. Note that Table 1 is divided in the energy contribution of the off-chip memories and on-chip memories (i.e. lines labelled as *on-chip* values) for each DM manager to the total. We consider in this case that an on-chip scratchpad memory of 16 KBytes is available for all the DM managers.

As Table 1 indicates, the memory footprint is the same for the managers because they are all power-of-two segregated-fit lists with the same internal data organization.

**Table 1.** Profiling values of DM managers in the DRR application for streams of 100000 packets

memory manager	memory accesses	memory usage (B)	memory power ( $\mu\text{nW}$ )	execution time (secs)
SegFitSLL FIFO	$2.00 \times 10^6$	$2.09 \times 10^6$	$13.28 \times 10^6$	115.04
(on-chip values)	$0.25 \times 10^6$	$16.38 \times 10^3$	$49.60 \times 10^3$	—
Total:	$2.25 \times 10^6$	$2.09 \times 10^6$	$13.33 \times 10^6$	115,04
Kings+LIFOSLL	$1.25 \times 10^6$	$2.09 \times 10^6$	$83.01 \times 10^6$	64.25
(on-chip values)	$0.15 \times 10^6$	$16.38 \times 10^3$	$31.00 \times 10^3$	—
Total:	$1.40 \times 10^6$	$2.09 \times 10^6$	$8.33 \times 10^6$	64,25
Kings+FIFODLL	$1.75 \times 10^6$	$2.09 \times 10^6$	$11.62 \times 10^6$	135.63
(on-chip values)	$0.22 \times 10^6$	$16.38 \times 10^3$	$43.40 \times 10^3$	—
Total:	$1.97 \times 10^6$	$2.09 \times 10^6$	$11.66 \times 10^6$	135,63

However, as we have previously mentioned, it can be seen that by implementing a different allocation reuse scheme (e.g. FIFO and LIFO) we can gain considerably on performance and memory power consumption. The best performance and lowest power consumption figures are achieved by the manager with a Kingsley basis and a LIFO single linked list structure. This is due to the fact that its data structures can be updated using less memory accesses than the others considering the run-time access pattern observed with our profiling. In fact, when one packet has arrived to a certain queue, more packets are likely to arrive in a short period of time to the same queue and with the same size. Thus, the FIFO implementation achieves the best results in power consumption by increasing locality in memory references more than the any other solution.

Finally, to evaluate the speed up of the refinement process, remark that the DM managers for this application constitute around 400 lines of C++ code each and took us one week to build them, profile their components and refine their implementation. Each allocator has 5 layers and since all are variations of segregated fit algorithms, 2 layers were reused in each implementation. Also, remark that our approach is not limited to any specific memory hierarchy, in each case the final DM manager is optimized for the specific memory hierarchy of the system.

## 5 Conclusions

Consumer applications (e.g. multimedia) have grown lately in complexity and demand intensive DM requirements that must be heavily optimized (e.g. power, memory footprint) for an efficient mapping on current low-power embedded devices. System-level exploration methodologies have started to be proposed to consistently perform that refinement. Within them, the manual exploration and optimization of the DM manager implementation is one of the most time-consuming and programming intensive parts. In this paper we have presented and shown in realistic examples the applicability of a new system-level approach that allows developers to implement DM managers with high maintainability. At the same, it allows to acquire detailed profiling information (e.g. power consumption) of the basic implementation structures of DM managers that can be used to refine their initial implementations.



## References

1. G. Attardi, et al. A customiz. mem. manag. framework for C++. *Sw. Pract. and Exp.*, 1998.
2. E. D. Berger, et al. Composing high-performance mem. allocators. In *Proc. of PLDI.*, 2001.
3. R. Y. Chen, et al. Speed and Power Scaling of SRAM's. *ACM TODAES*, 6(1), 2001.
4. V. Agarwal, et al. Effect of technology scaling on microarchitectural structures. TR2000-02, USA(2002).
5. M. Leeman, et al. Power estimation approach of dyn. data storage on a HW SW boundary level. In *Proc. of PATMOS*, Italy(2003).
6. D. Luebke, et al. *Level of Detail for 3D Graphics*. Morgan-Kaufman (2002).
7. G. Memik, et al. Netbench: A benchmarking suite for network processors, 2001.
8. N. Murphy. Safe mem. use with dynamic alloc. *Embedded Systems* (2000).
9. Rtems, Open-Source Real-Time OS. (2002) <http://www.rtems.com/>.
10. M. Pollefeys, et al. Metric 3D surface reconst. from uncalibrated images. In *Lecture Notes in Computer Science*, Springer-Verlag (1998).
11. Y. Smaragdakis, et al. Mixin layers: Object-Oriented implementation techn. for refinement and collaboration-based designs. *Trans. on SW Engineering and Methodology* (2002).
12. Target jr. <http://computing.ee.ethz.ch/sepp/targetjr-5.0b-mo.html>.
13. D. Vandevoorde, et al. *C++ Templates, The Complete Guide*. Addison Wesley, UK(2003).
14. N. Vijaykrishnan, et al. Evaluating integrated HW-SW optimizations using a unified energy estimation framework. *IEEE Trans. on Computers* (2003).
15. P. R. Wilson, et al. Dynamic storage allocation. In *Worksh. on Mem. Manag.*, UK(1995).
16. D. Atienza, et al., DM manag. design method. for reduced mem. footprint in multim. and network apps., in *Proc. of DATE* (2004).

# PIRATE: A Framework for Power/Performance Exploration of Network-on-Chip Architectures

Gianluca Palermo and Cristina Silvano

Dipartimento di Elettronica e Informazione  
Politecnico di Milano, Milano, Italy  
{gpalermo,silvano}@elet.polimi.it

**Abstract.** In this paper, we address the problem of high-level exploration of Network-on-Chip (NoC) architectures to early evaluate power/performance trade-offs. The main goal of this work is to propose a methodology supported by a design framework (namely, *PIRATE*) to generate and to simulate a configurable NoC-IP core for the power/performance exploration of the on-chip interconnection network. The NoC-IP core is composed of a set of parameterized modules, such as interconnection elements and switches, to form different on-chip micro-network topologies. The proposed framework has been applied to explore several network topologies by varying the workload and to analyze a case study designed for cryptographic hardware acceleration in high performance web server systems.

## 1 Introduction

Designing complex System-On-Chip (SoC) solutions, such as multi-processors (MPs) or network processors, requires a flexible platform-based approach for both hardware and software sides of embedded architectures. The growing diffusion of MP-SoC embedded applications based on the platform-based design approach requires a flexible tuning framework to assist the phase of *Design Space Exploration (DSE)*. The overall goal of the DSE phase is to optimally configure the parameterized MP-SoC platform in terms of both energy and performance requirements depending on the target application. MP-SoC hardware platforms are usually built around a *network centric* architecture interconnecting a set of processors and IP (Intellectual Property) cores to the memory subsystem and I/O interfaces. The communication infrastructure can connect up to tens of master and slave IP nodes through the network architecture. In this scenario, the target architecture is based on the *Network-on-Chip (NoC)* approach [1], where on-chip communication represents the core of the overall system design methodology and it can be specified somewhat independently of the modules composing the platform. The design of high-performance, reliable, and energy efficient interconnect-oriented MP-SoCs raises new challenges in terms of design methodologies. A simulator to estimate performance, while varying network parameters, faces only one aspect of the architectural exploration at the system-level. To efficiently support system-level architectural exploration, we need a

high-level simulator and a power estimator to dynamically evaluate accurate power/performance trade-offs. This is especially necessary because the interconnection network accounts for a significant fraction of the whole energy consumed by the SoC, and this fraction is expected to grow in the next future [2] due to the growing complexity of packet routing and transaction management policies.

In this paper, we address the problem of the power/performance exploration of the NoC architectural design space at the system-level. The main goal is to support the exploration and optimization of network-centric on-chip architectures from the early stages of the design flow, when only the high-level description of the system can be used to simulate the timing and power behavior, and any other detailed information of the system will be known after the synthesis and optimization phases.

In particular, we propose a methodology to generate and to simulate a configurable NoC-IP architecture to support the efficient yet accurate exploration of the interconnection system of a SoC. Starting from the system-level specification of the interconnection network, the proposed framework supports the generation of a configurable system-level model described in SystemC to dynamically profile the given application. A set of power models are dynamically plugged-in in the simulator to provide accurate and efficient power figures for the different architectures.

The proposed framework has been used to explore two different design scenarios. First, we discuss the results obtained by the exploration of different NoC topologies by varying the workload. Second, we present the results obtained by applying the proposed methodology to a case study: An on-chip cryptographic accelerator for high performance web server systems.

The rest of the paper is organized as follows. A review of the most significant works appeared in literature to support the design of NoC architectures is reported in Section 2. The proposed design exploration framework is described in Section 3. Section 4 discusses the experimental results carried out to evaluate the accuracy and efficiency of the proposed framework, while some concluding remarks and future directions of the work have been outlined in Section 5.

## 2 Background

Many on-chip micro-network architectures and protocols have been recently proposed in literature, in some cases focusing on specific system configurations and application classes. The *SPIN Micronetwork* [3] represents a on-chip micro-network based on deterministic routing. The *Silicon Backplane Micronetworks* [4] based on a shared-medium bus and time-division multiplexing offers an example of transport layer issues in micro-network design. The *Octagon* on-chip communication architecture for OC-768 Network Processors has been proposed in [5].

The design of high-performance, reliable, and energy efficient interconnect-oriented MP-SoCs raises new challenges in terms of design methodologies ([1], [6]). Low-level power estimation tools (such as *Synopsys Power Compiler*) re-

quire synthesized RTL descriptions. Although these tools provide good levels of accuracy, they require long simulation time, becoming an unfeasible alternative for complex MP-SoCs. The exploration of MP-SoCs requires architectural-level power and performance simulators such as *SimpleScalar* [7], *Wattch* [8], and *Platone* [9]. A set of system level power optimization tools have been presented in [10].

In this direction, the *Orion* approach [11] consists of a power-performance interconnection network simulator to explore power-performance trade-offs at the architectural level. The framework can generate a simulator starting from a micro-architectural specification of the interconnection network. In the *Orion* approach, a set of architectural-level parameterized power equations have been defined to model the main modules of the interconnection network (such as FIFO buffers, crossbars, and arbiters) in terms of estimated switch capacitances to support dynamic simulation.

Power models for on-chip interconnection architectures have been recently proposed in literature. A survey of energy efficient on-chip communication techniques has been recently presented in [2]. Techniques operating at different levels of the communication design hierarchy have been surveyed, including circuit-level, architecture-level, system-level, and network-level. Patel *et al.* [12] focused on the need to model power and performance in interconnection networks for multiprocessor design, and they proposed a power model of routers and links. An estimation framework to model the power consumption of switch fabrics in network routers has been proposed in [13]. The work introduces different modelling methodologies for node switches, internal buffers and interconnect wires for different switch fabric architectures under different values of traffic throughput. The proposed modelling and simulation framework is limited only to switch fabrics. More recently, the same authors introduced in [14] a packetized on-chip communication power model for multiprocessors network design.

### 3 PIRATE NoC-IP Architecture

*PIRATE* is a Network-on-Chip IP core composed of a set of parameterizable interconnection elements and switches connected by different topologies. The switch architecture is based on a *crossbar topology*, where the ( $N \times N$ ) network connects the  $N$  input FIFO queues with  $N$  output FIFO queues. The number of each input (output) FIFO queue is configurable as well as queue length. The incoming packets are stored in the corresponding input queue, then, the packets are forwarded to the destination output queue. The I/O queues and the crossbar are controlled by the Switch Controller, that includes a static routing table and the arbitration logic. The *PIRATE* switch is a synchronous element requiring a one-cycle delay for each hop. The switch is based on wormhole routing and static routing defined by a routing table customized by the designer.

The design of *PIRATE* NoC-IP is supported by an automatic framework.

### 3.1 PIRATE Design Framework

The main goal of the *PIRATE* framework is the exploration of the design space to define the optimal configuration of the interconnection system for the target application. The *PIRATE* framework is mainly composed of the following modules:

- Generator of synthesizable Verilog RTL model for the configurable NoC-IP core;
- Automatic power characterization flow;
- Cycle-based SystemC simulation model for dynamic profiling of power and performance.

**Synthesizable RTL Generator.** This module consists of an automatic generator of the Verilog RTL description of the generic *PIRATE* NoC configuration. The module receives as input a configuration file describing the micro-architectural parameters and the structure of the network. In the configuration file the designer can specify the following set of parameters:

- *Switch Architecture*: The switch architecture is parametric in terms of number and length of I/O queues and interconnection width;
- *Network Topology*: A set of parameters defines the interconnection topology of the switches in the network. The explored standard network topologies are: Ring, Double Ring, Mesh, Cube, Binary-Tree, and Octagon. Other network topologies can be generated *ad hoc* to optimize the target architecture;
- *Connections Encoding*: The information flowing through the network can be encoded. Using this set of parameters the designer can insert standard encoding/decoding techniques;
- *Information Flow in the NoC*: The routing mechanism in the switches is based on a static routing table. The designer can customize the routing table for each switch to balance and to optimize the traffic in the network.

**Power Characterization Flow.** A methodology to automatically create the power models of the *PIRATE* NoC-IP has been defined and implemented to dynamically profile at system-level the power behavior of each module of the interconnection system. The methodology is based on the power characterization of each parameterized module of the interconnection system (i.e. switch, encoder/decoder, etc.). For each module, at the end of the characterization flow, we automatically generate the corresponding analytical model based on its design space parameters and the traffic information derived from simulation.

We define the *design space* as the set of all the feasible architectural implementations of a module. A possible configuration of the target module is mapped to a generic point in the design space as the vector  $a \in \mathcal{A}$ , where  $\mathcal{A}$  is the architectural space defined as  $\mathcal{A} = S_{p_1} \times \dots \times S_{p_l} \times \dots \times S_{p_n}$  where  $S_{p_l}$  is the ordered set of possible configurations for parameter  $p_l$  and “ $\times$ ” is the cartesian product. To exemplify our method, let us consider the power model of the switch. According

to [13], our power model of the switch is dependent on traffic and design space parameters:

$$P_{SW}(\mathcal{A}, TR) = B_0(\mathcal{A}) + B_1(\mathcal{A}) \times TR \quad (1)$$

where  $TR$  is the traffic factor,  $\mathcal{A}$  is the architectural space defined above and  $B_0, B_1$  are the model coefficients.

Due to its configurability, the NoC IP core spans a very large design space  $\mathcal{A}$  and this causes the impossibility to characterize each configuration, since each point to be characterized requires an RTL synthesis and a variable number of gate-level simulations and power estimations. To reduce the number of configurations to be characterized, our approach is based on the fundamental theories of the statistical design of experiments (*DoE*) [15] [16].

Figure 1 shows the proposed automatic power characterization flow. It is based on the standard *Synopsys* [17] gate-level power estimation flow (dark grey) and it is composed of three main phases:

- *Design of Experiments*: This phase concerns the planning of the experiments before synthesis and simulation of each module. It consists of the sampling of

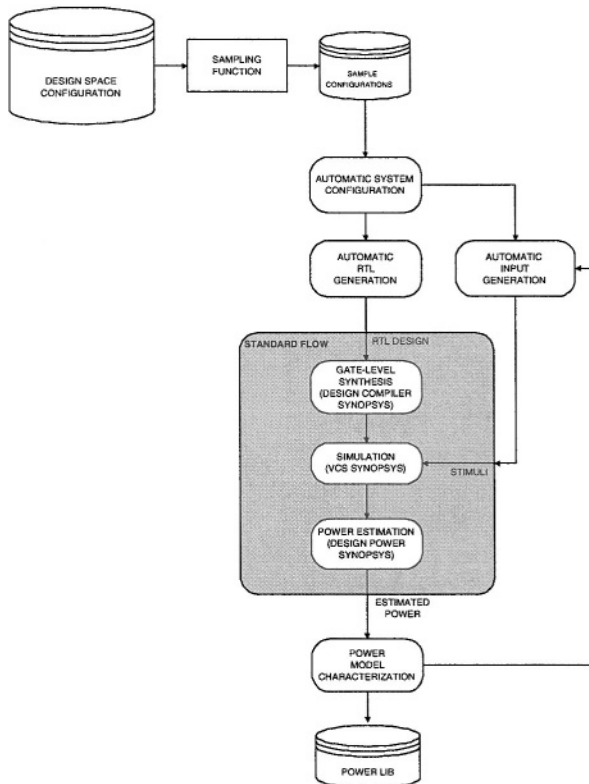


Fig. 1. The proposed power characterization flow

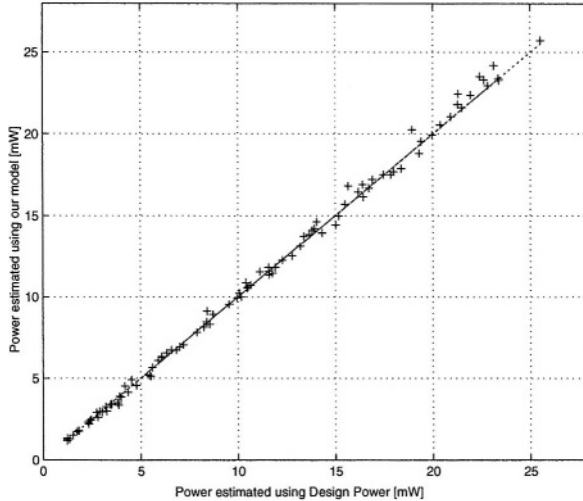
the design space and the automatic generation of the RTL description of the target module and the corresponding testbench for the following simulation phase.

- *Standard Power Estimation Flow*: This phase represents the core of the characterization methodology and it is based on the *Synopsys* tool chain. The standard flow receives as input the RTL description of the elements generated and, by using the *Synopsys Design Compiler* tool and the target technology library, the flow performs the synthesis of the corresponding gate-level description. The value for the power dissipation of the element under analysis can be obtained after the gate-level *VCS* simulation (by using the input patterns automatically generated in the previous phase of the flow) and the power estimation by using *Design Power*.
- *Empirical Model Building*: This phase generates the power model of the interconnection elements. In the power model characterization phase, the values of the coefficients of the high-level model are computed by linear regression over the set of experiments given during the *DoE*. The phase receives as input the power estimation values of a point in the design space for the given stimuli. By using this information, the flow can choose to re-simulate and to re-estimate the power of the element with other input stimuli to obtain a more accurate modelling [16]. The derived power model is then inserted in the power library.

The proposed methodology is technology-dependent, thus providing very accurate models. Currently, the STMicroelectronics 0.18  $\mu\text{m}$  HCMOS8 technology is used. The validation results are shown in Figure 2 to evaluate the accuracy of the power model of the switch given in Equation (1). The scatter plot compares the power estimates obtained by our method and those obtained by using *Synopsys Design Power*. In the scatter plot, the points are very close to the diagonal, with a standard deviation within 5%.

**Cycle-based Simulation Model.** The *PIRATE* NoC-IP simulation model has been developed at system-level by using SystemC 2.0 [18]. The model has been developed at Bus Cycle Accurate (BCA) level, therefore providing cycle-based accurate timing information and power estimates. The model is completely configurable in terms of NoC micro-architectural parameters, such as network topology, number of masters/slaves for each switch, length of input/output queues for each switch. Basically, the simulator receives as input the same configuration file used to create the synthesizable RTL description of the network and it generates the simulatable interconnection module described in SystemC.

The power models obtained by the automatic characterization flow presented above have been plugged in the system-level simulation model to perform a fast power-performance exploration of the NoC system based on dynamic profiling. From one side, the system-level simulation can guarantee the *efficiency* during the exploration phase, from the other side, the plug-in of power models estimated at gate-level can guarantee the *accuracy*. The network simulator can support two types of power analysis: *Static Analysis*, providing the total average power for



**Fig. 2.** Scatter plot of power estimated by our model with respect to power estimated by Synopsys Design Power for the switch design space.

the target application, and *Dynamic Analysis*, providing a cycle-accurate power and performance profiling.

## 4 Experimental Results

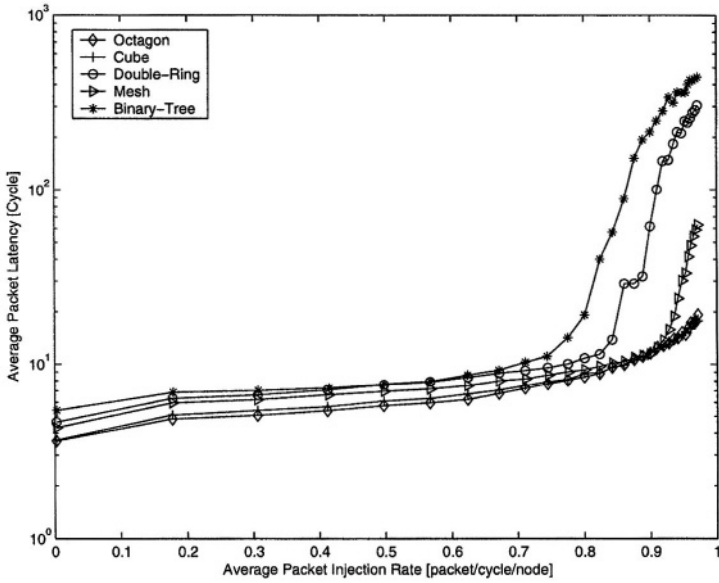
In this section, we show how the *PIRATE* framework can be used for a fast system-level exploration of power performance trade-offs of on-chip micro networks. Two different exploration scenarios have been analyzed.

First, we discuss the experimental results obtained by the exploration of the power/performance effects of different traffic patterns to the parameterizable NoC-IP. The exploration compares different network topologies. Second, we present the results obtained by applying the proposed methodology to the exploration of the CryptoSoC case study: An on-chip cryptographic accelerator for high performance web server systems based on SSL/TLS protocol. CryptoSoC has been developed in the MEDEAplus A304 project. In this case, the exploration analyzes power/performance trade-offs for different architectural configurations of the modules composing the NoC-IP, given the representative workload of the target network application.

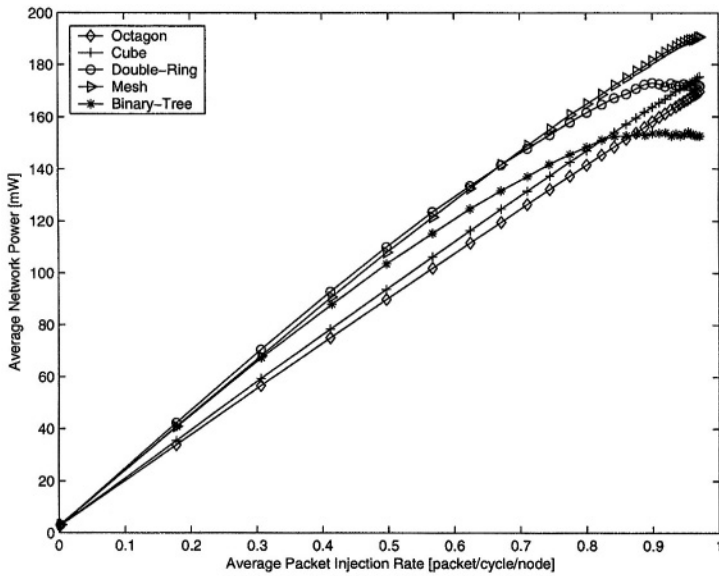
### 4.1 Exploration of NoC Topologies

The average power and latency associated with the parameterizable NoC-IP have been analyzed by varying the packet injection rates. We simulated and compared five different network topologies connecting 8 nodes throughout the

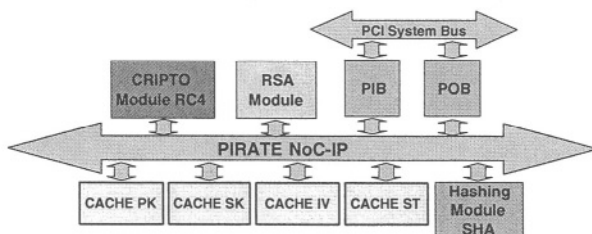




**Fig. 3.** Average packet latency with respect to average packet injection rate for different network topologies of the *PIRATE* NoC-IP



**Fig. 4.** Average network power with respect to average packet injection rate for different network topologies of the *PIRATE* NoC-IP



**Fig. 5.** The CryptoSoC Architecture

**Table 1.** Comparison of different network topologies for the CryptoSoC architectures in terms of average packet latency and average network power.

Topology	Avg. Packet Latency [Cycle]	Avg. Network Power[mW]
Octagon	2.80	16.19
Cube	3.22	16.30
Double-Ring	2.87	14.92
Mesh	2.92	16.18
Ad-Hoc	2.10	14.05

NoC-IP: Octagon, Cube, Double-Ring, Mesh, and Binary-Tree. For all topologies, the simulator generates uniformly distributed random traffic: each node injects packets in the network uniformly to random node destinations following the Poisson distribution for the injection time.

For different network topologies, Figure 3 and Figure 4 respectively show the average packet latency and the average network power with respect to the average packet injection rate. In both figures, a higher number of sample points for high values of average packet injection rates have been evaluated to find the saturation points with more accuracy. A similar trend can be noted for all topologies, but with a different behavior *before* with respect to *after* the corresponding saturation point. The behavior *before* saturation shows the average packet latency slightly increasing as the workload varies from 0 to 0.2, and almost constant for workload from 0.2 to 0.75. The average network power increases proportionally with the workload in the range from 0 to 0.75. *After* the saturation point, the average packet latency grows rapidly, while the average power remains constant or slightly decreases, because the network cannot handle a higher packet injection rate. Octagon and Cube topologies outperform the other topologies in terms of both latency and power for all workload values, and they tend to saturate for workload values close to 1. The Binary-Tree presents the early saturation point for latency (around 0.75), and Double-Ring and Mesh topologies also early saturate (around 0.85 and 0.9 respectively). A similar behavior can be noted for the average power.

## 4.2 Exploration of CryptoSoC Architecture

The CryptoSoC architecture (shown in Figure 5) is composed of the following modules: Public Keys (PK) Cache, Sessions Keys (SK) Cache, Initialization Vectors (IV) Cache, Hashing Function State (ST) Cache, Hashing Module SHA, Cryptographic Module RC4, RSA Module, and the Packet In/Out Buffers to interface the PCI System Bus. In this case, the exploration analyzes power/performance trade-offs for different micro-architectural network topologies given a representative workload of the target network application. Furthermore, we explored a custom network topology designed for the CryptoSoC architecture based on a dynamic profiling of the application information flow to eliminate switch congestions.

Table 1 reports the average packet latency and the average network power for different network topologies, given the workload for the target application. The analyzed standard topologies are Octagon, Cube, Double-Ring, and Mesh, while the last row reports the *ad hoc* optimized topology. The *ad hoc* topology slightly outperforms the other standard topologies for both average network power and packet latency. Among the standard topologies, the Double-Ring topology presents the lower average network power due to its simple structure. Due to the low workload of the target application, the average packet latency of the Double-Ring is similar to the more complex Octagon topology.

## 5 Conclusions

In this paper, we propose a design framework to generate and to simulate a configurable NoC-IP core to support the fast exploration of the on-chip interconnection network for MP-SoC applications. The NoC-IP is composed of a set of parameterized switches to form different network topologies. The proposed framework has been applied to explore the power/performance effects of different different network topologies at varying traffic patterns. Furthermore, we applied the proposed framework to a case study, an on-chip cryptographic hardware accelerator for high performance web server systems. Future work is directed towards the plug-in of the *PIRATE* NoC-IP into a high-level domain-specific flexible MP-SoC platform oriented towards networking applications and supporting quality of service.

## References

1. L. Benini and G. De Micheli. Networks on chips: A new *soc* paradigm. *Computer*, pages 70–78, January 2002.
2. M.B. Srivastava V. Raghunathan and R.K. Gupta. A survey of techniques for energy efficient on-chip communication. In *Proc. of DAC-40: ACM/IEEE Design Automation Conference*, pages 900–905, June, 2003.
3. A. Greiner L. Mortiez C. A. Zeferino A. Adriahantenaina, H. Charlery. Spin: A scalable, packet switched, on-chip micro-network. In *Proc. of DATE-2003: Design, Automation and Test in Europe Conference and Exhibition*, March, 2003.

4. [www.sonicsinc.com](http://www.sonicsinc.com).
5. S. Dey F. Karim, A. Nguyen and R. Rao. On-chip communication architecture for oc-768 network processors. In *Proc. of DAC-38: ACM/IEEE Design Automation Conference*, June, 2001.
6. A. Mihal K. Keutzer S. Malik J. Rabaey A. Sangiovanni-Vincentelli M. Sgroi, M. Sheets. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Proceedings of the 38th Design Automation Conference*, June 2001.
7. D. Burger, T. M. Austin, and S. Bennett. Evaluating future microprocessors: The simplescalar tool set. Technical Report CS-TR-1996-1308, University of Wisconsin, 1996.
8. D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. of ISCA-00*, pages 83–94, 2000.
9. Tony D. Givargis and Frank Vahid. Platune: A tuning framework for system-on-a-chip platforms. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 21(11):1317–1327, November 2002.
10. L. Benini and G. De Micheli. System-level power optimization: techniques and tools. *ACM Transactions on Design Automation of Electronic Systems.*, 5(2):115–192, January 2000.
11. L.S. Peh H.S. Wang, X. Zhu and S. Malik. Orion: A power-performance simulator for interconnection networks. In *MICRO-35: Int. Symposium on Microarchitecture, Instambul, Turkey*, Nov., 2002.
12. S. Yalamanchili C. Patel, S. Chai and D. Schimmel. Power constrained design of multiprocessor interconnection networks. In *Proc. of ICCD*, pages 408–416, Oct., 1997.
13. T. T. Ye, L. Benini, and G. De Micheli. Analysis of power consumption on switch fabrics in network routers. In *Proc. of DAC-39: ACM/IEEE Design Automation Conference*, June, 2002.
14. L. Benini T. T. Ye and G. De Micheli. Packetized on-chip interconnect communication analysis for *mpsoc*. In *Proc. of DATE-2003: Design, Automation and Test in Europe Conference and Exhibition*, March, 2003.
15. Douglas C. Montgomery. *Design and Analysis of Experiments, 5th Edition*. John Wiley & Sons, New York, 2001.
16. G. E. Box and N. R Draper. *Empirical Model-Building and Response Surfaces*. John Wiley & Sons, New York, 1987.
17. [www.synopsys.com](http://www.synopsys.com).
18. *SystemC 2.0 User's Guide*. 2002.

# Power Consumption of Performance-Scaled SIMD Processors

Anteneh A. Abbo, Richard P. Kleihorst, Vishal Choudhary, and Leo Sevat

Philips Research Laboratories, Prof. Holstlaan 4, NL-5656 AA,  
Eindhoven, The Netherlands  
Anteneh.A.Abbo@philips.com

**Abstract.** Single-Instruction Multiple-Data (SIMD) processing is valuable in numerous compute intensive application areas, especially in pixel processing on programmable processors. With SIMD, the nice thing is that the actual hardware can be scaled to the performance and the power consumption demands of the application domain, while the software suite remains equal. In this paper, we discuss the effect of scaling on power consumption, cost and performance which is related to the characteristics of applications ranging from mobile to medical video processing. Our analysis is based on experience obtained with the Philips Xetal SIMD processor [1,2].

## 1 Introduction

Real-time video processing on cost and power consumption restricted platforms is becoming relevant in a number of applications such as mobile communications, home robotics and even medical image processing. Part of the video processing chain is dedicated to (early vision) pixel processing where huge amounts of pixels have to be processed but the algorithms are similar for each pixel.

If we want to implement programmable cores for this pixel processing domain, a wise choice are SIMD processors which prove very fruitful in this area [3,4]. Their parallel architecture strongly reduces the number of memory accesses, the clock speed, and instruction decoding overhead, thereby enabling higher performance and lower power consumption [1,2].

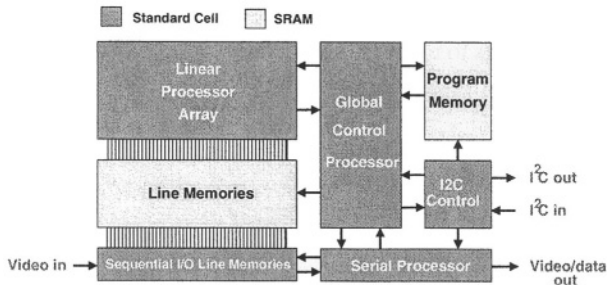
A very nice feature of SIMD processors is the regularity of the design. This enables design-cost effective scaling of the platform for different performance regions by simply increasing or reducing the number of parallel processors. All of the hardware design can be reused and what is even more important is that inherently the software suite remains the same. So using SIMD for the pixel processing pipeline lowers the design cost and time-to-market of rapidly changing electronics.

In this article we will investigate what happens to the *power consumption* and *area* if we would scale a known single chip SIMD pixel processor, (Philips' Xetal) for a set of applications demanding different performances.

The remainder of the paper is organized as follows: in Section 2 we will shortly introduce the Xetal architecture, in Section 3 we will show some application areas and their performance demands. In Section 4 we analyse the actual scaling and in Section 5 conclusions are drawn.

## 2 The SIMD Platform

For our platform we choose the Xetal SIMD processor which has proven itself for the pixel processing tasks of all application areas in this paper. The original IC measures  $22 \text{ mm}^2$  in  $0.18\mu$  CMOS technology and is meant for VGA ( $640 \times 480$  pixels) at 30 fps. The top-level architecture is shown in Figure 1. Different digital functional blocks and memory modules in the IC with some of the main communication lines are shown. There are two programmable processors: a Global Control Processor for control-based algorithms and a Linear Processor Array (LPA), with 320 identical processing elements, for pixel-based algorithms. The LPA uses 16 line memories for temporary storage of data. Additionally 4 sequential line memories are used for serial-to-parallel conversion of the video input and parallel-to-serial conversion of the LPA output. The program memory is shared by both processors and can store up to 1024 instructions.



**Fig. 1.** Architecture of the Xetal SIMD processor

Xetal can achieve a maximum performance of 1560 instructions per pixel @ 24 MHz. Which, at VGA 30 frames per second, translates to 9 GOPS. One operation can be an arithmetic instruction such as multiply-add or a compare instruction. The design is clock-gated, so when fewer instructions per pixel are needed, the processors are switched off for the remaining cycles. Consequently, the power consumption ranges from a few mW upto 2W, depending on the load. For more info, please see [1].

## 3 Description of Applications

The nature of the application determines the scaling that needs to be applied to the SIMD processor in order to meet the performance. Consequently, the SIMD architecture needs to be scaled up or down. The trend in all application areas is high performance, yet low power consumption and low cost.

### Mobile-Multimedia Processing

This class of applications is characterised by low-cost, moderate computational complexity and low-power. The latter objective can sometimes be compromised for the

former since mobile devices are active relatively for short period of time compared to the standby duration and the battery energy is wasted mainly in the standby phase.

The computational complexity for this class of applications varies from 300 MOPs for basic camera pre-processing (VGA @ 30 fps:  $640 \times 480 \times 30 \text{ pix/sec} \times 30 \text{ OPs/pix} = 277 \text{ MOPs}$ ) to 1.5 GOPs for more complex pre-processing including auto white-balance, exposure time control (about 150 operations per pixel).

### **Intelligent Home Interfaces**

This class of applications corresponds to emerging house robots with vision features. Typical examples include intelligent devices with gesture and face recognition [5], autonomous video guidance for robots, and smart home surveillance cameras. These devices cover the medium-cost range. They need to operate in uncontrolled environments (lighting conditions, etc.), and smarter (complex) algorithms are needed to achieve the desired performance. The power aspect remains an issue especially in standalone modules such as battery powered surveillance cameras. Because of the extra intelligence needed in this class of applications, the number of operations per pixel is in the order of 300 or more. This translates to more than 3 GOPs for a 30 frames-per-second VGA size video stream.

### **Industrial Vision and Medical Imaging**

Unlike the previous two cases, applications in this segment are cost-tolerant and more emphasis is given to achieving high-performance at a given power budget. The scaling here is mainly in accordance with the incoming video format, one can expect a corresponding increase in the SIMD array with increase in the resolution of image sensors. In this class of applications a number of basic pixel-level operations such as edge detection, enhancement, morphology, etc. need to be performed. The video rate exceeds 30 fps and the computational complexity is often more than 4 GOPs.

## **4 Performance-Driven SIMD Scaling**

The discussion on application profiling indicates that the performance requirements vary by orders of magnitude from 300 MOPs to more than 4 GOPs. Usually cost-sensitive applications allow for quality degradation and are satisfied with lower performance figures. However, performance comes at a lower price with advances in CMOS technology which keeps on pushing up the lower limit and allowing the use of more complex algorithms.

In this section, we address the scaling of a massively parallel SIMD architecture to match the computational complexity of a given application. The impact of scaling is studied with respect to the different SIMD building blocks and quantified in terms of silicon area and power dissipation. The silicon area directly relates to the cost while the power dissipation dictates the applicability of the device in a system with maximum power constraint.

### **4.1 Scaling SIMD Machines and Power Consumption**

For a given performance requirement, scaling the number of processors in the SIMD machine has direct impact on the power consumption. While the power consumption

directly relates to the rate the battery is discharged, dissipation determines the complexity and cost of packaging and cooling of the devices. In most mobile applications, both battery life and packaging are important issues. The analysis in this section is based on the well known CMOS dynamic power dissipation formula:  $Power \propto CV^2f$ , where  $C$  is the switched capacitance,  $V$  is the supply voltage and  $f$  the switching frequency [6].

Energy consumption of an SIMD machine consists of the following components: computation modules ( $E_{comp}$ ), neighbour communication network ( $E_{comm}$ ), memory blocks ( $E_{mem}$ ) and control and address generation units ( $E_{caddr}$ ). Equations [1 ... 5] give the intrinsic energy model of the components as a function of the convolution filter width ( $W$ ), number of processing elements ( $P$ ), number of pixels per image line ( $N$ ) and the size of the working line memory ( $N(W-1)$ ). The model parameters have been derived based on a high-level power estimation Petrol [7,8] and later calibrated with measurement results of the Xetal chip. The bases for choosing a convolution algorithm in our investigation is the fact that convolution involves all the four components (computation, memory, communication and control) that contribute to energy consumption. In the formulae, it is assumed that the different SIMD configurations operate at the same supply voltage.

$$E_{comp} = \alpha_1 W^2 \quad (1)$$

$$E_{comm} = \alpha_2 W \left( \frac{N}{P} + 2 \right) \times \\ 1 + 2 \frac{P}{N} \sum_{k=1}^{\lfloor \frac{W}{2} \rfloor} \left( \frac{N}{P} \lfloor \frac{k-1}{N/P} \rfloor + k \bmod \frac{N}{P} + \frac{N}{P} \right) \quad (2)$$

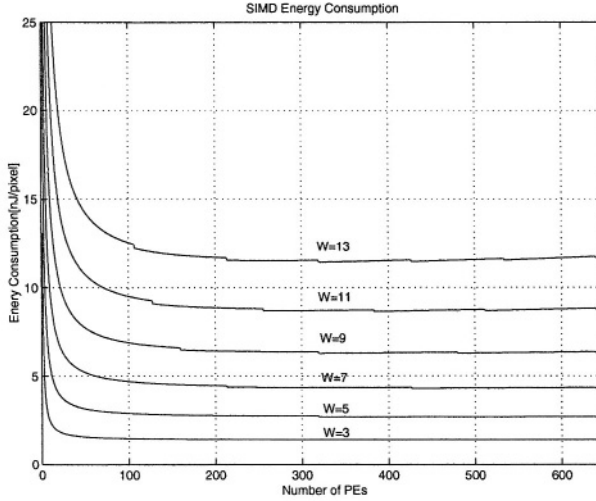
$$E_{mem} = \left( \frac{W^2}{P} + W \right) \times \\ (\beta_1 + \beta_2 N + \beta_3 (W-1) + \beta_4 N (W-1)) \\ + \left( \frac{2W^2}{P} \right) (\beta_1 + \beta_2 P + \beta_3 \frac{N}{P} + \beta_4 N) \quad (3)$$

$$E_{caddr} = \left( \frac{W^2}{P} + W \right) (\beta_5 + \beta_6 (W-1)) \\ + \frac{2W^2}{P} (\beta_5 + \beta_6 \frac{N}{P}) \quad (4)$$

$$E_{tot} = E_{comp} + E_{comm} + E_{mem} + E_{caddr} \quad (5)$$

The computation energy ( $E_{comp}$ ) is a quadratic function of the filter width ( $W$ ) and does not depend on the number of processing elements as the same number of arithmetic operations needs to be done for all configurations. On the other hand, the other energy components depend on all three dimensions ( $W, P, N$ ) and have been modelled by a first order approximation. In essence, scaling the SIMD architecture affects the number of accesses to the working line memories. With each access, a certain amount of energy is consumed by the communication channel, the control and address and generator, and the memory block. As the number of processors increases, the number of accesses to memory decreases thereby reducing the total energy dissipation. In general, the following relationship holds between the energy components:  $E_{mem} > E_{comp} > E_{caddr} > E_{comm}$ .





**Fig. 2.** SIMD energy consumption per filter kernel for  $N = 640$  and constant supply voltage

Figure 2 shows curves of the total energy for  $N = 640$  (the number of pixels in a VGA image line) with filter width as parameter. The curves in Figure 2 show that beyond a certain degree of parallelism the saving in energy is very marginal. While the trend is the same, larger filter kernels benefit from increased number of PEs.

It should be noted that configurations with more PEs can handle increased throughput (pixels per second) for the same algorithmic complexity. The increase is proportional to  $P$  since  $P \leq N$  and the filter kernels can be fully parallelised over the pixels in an image line. The minimum number of processing elements needed to meet the real-time constraint is given by  $P_{min} = \lceil C_{alg} \times f_{pixel} / f_{max} \rceil$ , where  $C_{alg}$  is the algorithmic complexity in number of instructions,  $f_{pixel}$  the pixel rate and  $f_{max}$  the maximum clock frequency of the processing elements (PEs). The clock frequency of the PEs can be increased further by optimisation and pipelining. While optimisation for speed leads to larger PE sizes and increases computation energy dissipation, the impact of pipelining on the SIMD scaling needs to be investigated further.

When throughput comes to the picture, power dissipation becomes a more convenient metric for comparison. Figure 3 shows the power dissipation versus the number of PEs with performance ( $Perf = C_{alg} \times f_{pixel}$ ) as parameter. The starting point of the curves corresponds to the minimum number of PEs ( $P_{min}$ ) that can provide the indicated performance for a clock frequency of  $f_{max} = 50$  MHz. Increasing parallelism beyond  $P_{min}$  increases the chip cost (area) which has been traded for lower power dissipation through supply voltage and frequency scaling [9].

In Figure 4, the energy scaling factor is shown which has been used to generate the power dissipation curves of Figure 3. The scaling factor starts with unity at the smallest number of PEs ( $P_{min}$ ) that corresponds to the maximum achievable performance when the PEs operate at maximum speed ( $f_{max}$ ) and highest operating voltage  $V_{dd,max}$ . As the number of PEs increases, the same performance can be achieved at lower operating

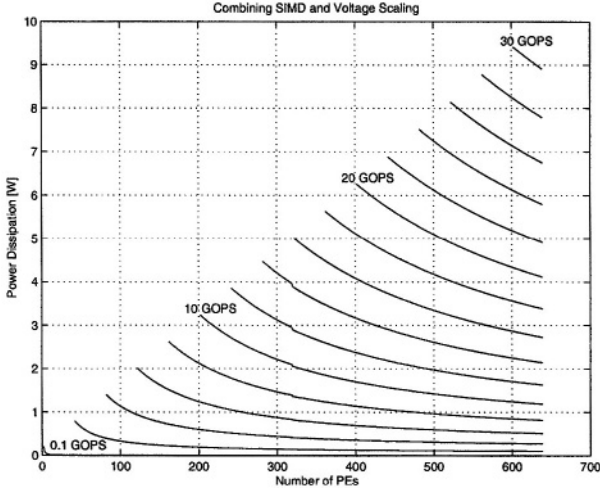


Fig. 3. Impact of combining SIMD scaling with voltage scaling on power dissipation

voltage and frequency. Consequently, the consumed energy decreases by a factor given on the scaling axis. The energy scaling factor (Eqn. 12) has been derived from the CMOS propagation delay model (Eqn. 6) given in [10]. A threshold voltage of  $V_{th} = 0.5V$  and maximum supply voltage of  $V_{dd\_max} = 1.8V$  have been assumed in the plotted curves (in accordance with the nominal voltages of a  $0.18 \mu m$  CMOS technology). To allow for noise margin, the lowest operating voltage has been set to  $V_{dd\_min} = V_{th} + 0.2V$ .

$$T_{pd} \propto 1/(V_{dd} - V_{th})^{3/2} \tag{6}$$

$$\Rightarrow f \propto (V_{dd} - V_{th})^{3/2} \tag{7}$$

$$f/f_{max} = [(V_{dd} - V_{th})/(V_{dd\_max} - V_{th})]^{3/2} \tag{8}$$

$$\begin{aligned} \Rightarrow V_{dd} &= (V_{dd\_max} - V_{th})(f/f_{max})^{2/3} + V_{th} \\ &= (V_{dd\_max} - V_{th})(P_{min}/P)^{2/3} + V_{th} \end{aligned} \tag{9}$$

$$energy \propto V_{dd}^2 \tag{10}$$

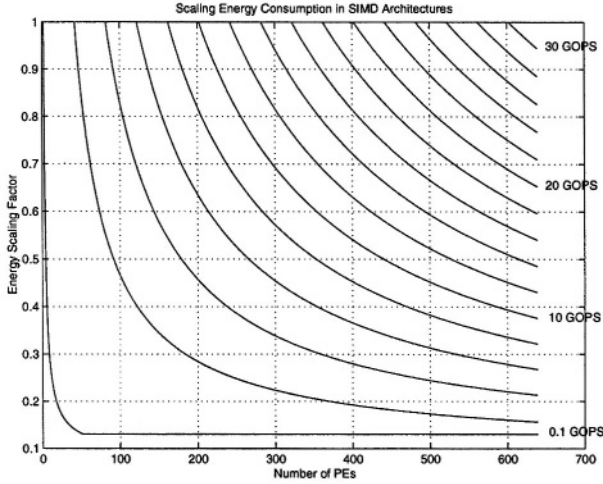
$$eratio = \frac{Energy(Increased Parallelism)}{Energy(Lowest Parallelism)} \tag{11}$$

$$\begin{aligned} eratio &= (V_{dd}/V_{dd\_max})^2 \\ &= \left[ \frac{(V_{dd\_max} - V_{th})(P_{min}/P)^{2/3} + V_{th}}{V_{dd\_max}} \right]^2 \end{aligned} \tag{12}$$

## 4.2 Impact of SIMD Scaling on Silicon Area

### Scaling the Linear Processor Array

The LPA, being the actual work-horse of the SIMD machine, can easily be scaled according to the desired performance, cost and power figures. The basic investigation in the



**Fig. 4.** Energy scaling factor with computation demand as parameter

previous section is directly applicable for scaling the LPA. Thus, area of the processor array can be given by  $A_{parray} = P \times A_{PE}$ .

**Scaling the Line Memories**

The size of on-chip line-memories is dictated by the algorithm to be executed and is independent of the number of PEs used in the SIMD configuration. From area point of view, the line-memories do not scale; they only change in layout shape since more pixels would be allocated per PE as the number of PEs decreases. The silicon area contribution of the line-memories becomes  $A_{lmem} = N_{lines}A_{line}$

**Scaling the Global Controller**

Like the line-memories, the global controller also doesn't scale with change in the number of PEs. This is to be expected since in the SIMD principle, the global controller is already a shared resource by all PEs. The global controller area is simply  $A_{gcon}$ .

**Scaling the Program Memory**

The program memory scaling depends on the degree of loop unrolling. In order to avoid the overhead of loop control, one might think of repeating the same code as many times as the number of pixels per PE. Assuming an algorithm contains  $C_{alg}$  operations per pixel and loop unrolling is done  $N_{unrl}$  times, the area contribution of the program memory can be described in terms of the area of a single instruction as  $A_{pmem} = C_{alg}N_{unrl}(1 + \gamma)A_{instr}$ . When fewer PEs are used, more addressing bits are needed in the instruction word since more pixels need to be multiplexed per PE. The factor  $\gamma = \lceil \log_2(P_{max}/P) / \log_2 N_{lines} \rceil$  models the increase in  $A_{instr}$  due to address space expansion.

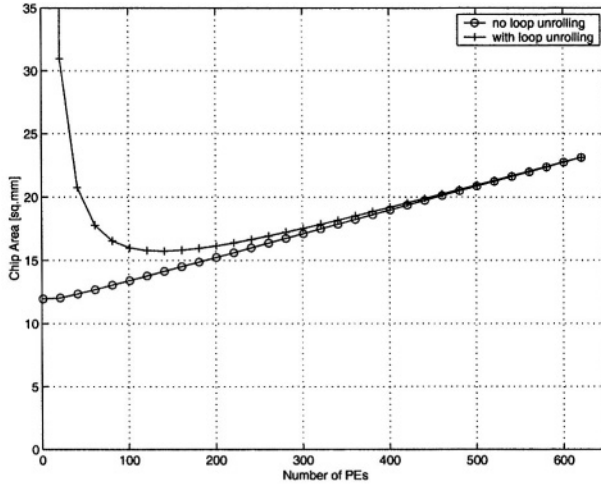


Fig. 5. Area impact of SIMD scaling based on a  $0.18 \mu\text{m}$  design data

### 4.3 Summary

To summarise the SIMD scaling issue, we collect the components into one cost function described in terms of silicon area:  $A_{SIMD} = A_{parray} + A_{lmem} + A_{gcon} + A_{pmem}$ . Figure 5 shows how the SIMD area scales with the scaling in the number of PEs. The curves are offset by an amount equivalent to the line-memory and global controller areas which do not scale. Since the size of the program memory is small relative to the other components, the relative impact of loop unrolling is minimal for large array sizes. For lower number of processing elements ( $P < 200$ ), the area of the non-scaling components dominates. Under this condition, it is sensible not to scale down the number of PEs so that the performance loss in the case of no loop-unrolling can be compensated for. However, when fewer number of PEs have to be used for cost reason, maximum loop unrolling is not sensible to do and other measures need (such as increasing the PE speed) to be taken to combat the impact of loop overhead on the real-time performance. When combined with the power scaling curves, the area scaling curve provides a quantitative means for performance-driven SIMD scaling.

## 5 Conclusions

In this paper, we have shown the impact of SIMD scaling with regard to power consumption and silicon area (cost). We showed that more parallelism saves energy, but depending on the algorithm complexity, the gain starts to stabilise when a certain number of processors is reached. The area increase depends on whether loop unrolling is performed or not, but is linear with an offset. For low-cost applications, a design at maximum silicon speed is desired with a sufficient level of parallelism to obtain some power savings.

## References

1. R.P. Kleihorst, A.A. Abbo, A. van derAvoird, M.J.R. Op de Beeck, L. Sevat, P. Wielage, R. van Veen, and H. van Hertten, "Xetal: A low-power high-performance smart camera processor," in *ISCAS 2001*, Sydney, Australia, may 2001.
2. A.A. Abbo and R.P. Kleihorst, "A programmable smart-camera architecture," in *ACIVS2002*, Gent, Belgium, Sept 2002.
3. P. Jonker, "Why linear arrays are better image processors," in *Proc. 12th IAPR Conf. on Pattern Recognition*, Jerusalem, Israel, 1994, pp. 334–338.
4. R. Kleihorst et al., "An SIMD smart camera architecture for real-time face recognition," in *Abstracts of the SAFE & ProRISC/IEEE Workshops on Semiconductors, Circuits and Systems and Signal Processing*, Veldhoven, The Netherlands, Nov 26–27, 2003.
5. R.P. Kleihorst H. Fatemi and H. Corporaal, "Real-time face recongition on a smart camera," in *ACIVS2003*, Gent, Belgium, Sept 2003.
6. H. Veendrick, *Deep-Submicron CMOS ICs*, Kluwer, 2000.
7. R. Manniesing, R. Kleihorst, A. van der Avoird, and E. Hendriks, "Power analysis of a general convolution algorithm mapped on a linear processor array," *Kluwer Journal of VLSI Signal Processing*, vol. 37, pp. 5–19, Apr 2004.
8. R.P. Llopis and K. Goosens, "The petrol approach to high-level power estimation," in *International Symposium on Low Power Electronics and Design*, Monterey, CA, 1998, pp. 130–132.
9. A. Chandrakasan R. Brodersen and S. Sheng, "Design techniques for portable systems," *IEEE Int. Solid State Circuits Conf.*, 1993.
10. S. Sun and G.Y. Tsui, "Limitations of cmos supply volatage scaling by mosfet threshold voltage variation," *IEEE Journal of Solid State Circuits*, vol. 30, no. 8, Aug 1995.

# Low Effort, High Accuracy Network-on-Chip Power Macro Modeling

Andrea Bona, Vittorio Zaccaria, and Roberto Zafalon

STMicroelectronics, Advanced System Technology - R&I  
Via Olivetti 2, 20041 Agrate Brianza, Italy

**Abstract.** This paper deals with a new and effective methodology to minimize the Design of Experiments (DoE) needed to characterize a set of energy macro models of innovative Network-on-Chip architectures. By properly combining regression (i.e. polynomial) and interpolation (i.e. table-based) techniques, this methodology aims to reduce the overwhelming complexity of sampling the huge, non-linear design space involved with the system operations of a high-end parametric on-chip-network module. Eventually, the outcoming power models are linked to a standard SystemC simulator by means of a specific class library extension, running at Bus Cycle Accurate (BCA) and Transaction Modeling Level (TLM). In this context, the power model is accurate and highly correlated, with an average error of 2% and a RMS of 0.015 mW vs. the reference power figures, measured on the gate level mapped design. Without affecting the general applicability of our approach, the proposed methodology is exploited through an automatic model characterization and building flow, targeting an industrial on-chip communication IP (STBus). The experimental figures show that our DoE optimization techniques are able to trade-off power accuracy with model building cost, leading up to 90% reduction of the sampling space.

**Keywords:** Network-on-Chip power analysis, communication based low power design, system-level energy optimization.

## 1 Introduction

Embedded systems are today able to provide a number of new services that will arguably become common application in the next few years. Multimedia capabilities (audio/video streaming) in personal communicators, huge computing power and storage size (especially from clusters of processors) and high rate accessibility from mobile terminals are just an example of the most important.

The IC design bottleneck, in today's complex and high performance System on Chip (SoC), is moving from computation capacity to communication bandwidth and flexibility. That's why system designers need to leverage on pre-validated components and IPs (processor cores, controllers and memory arrays) to be able to deal with such a difficult design problem. Design methodology will have to support IP re-use in a plug-and-play style and this methodology will have to be extended to system buses and hierarchical interconnection infrastructures.

These complex SoCs have to be able to manage all the problems related to the signal integrity in order to provide a functionally correct and reliable operation under

data uncertainty and noisy signaling. Moreover the on-chip physical interconnection system will become a limiting factor for both performance and energy consumption, also because the demand for component interfaces will steadily scale-up in size and complexity.

In this paper, starting from the results achieved in [21], we present a new and effective methodology to minimize the Design of Experiments (DoE), needed to characterize a set of innovative energy macro models of Network-on-Chip architectures. Such a smart DoE methodology is seamlessly integrated into an automatic model building environment, enabling the high-level (BCA/ Transaction level) energy modelling for a Network-on-Chip (NoC). This model will allow the system designer to profile the entire platform under test since the very early stages of the design; when usually only a software model of the whole system exists.

The characterization campaign of a high configurable IP, such as a NoC, may often become unfeasible. The computational effort to characterize the power model for a NoC is even larger than the characterization task for an industrial fully featured ASIC library.

The methodology is applied on a versatile, high performances and configurable communication infrastructure: STBus. STBus is an interconnect IP largely configurable in terms of communication protocols, data bus width, datapath topology, arbitration scheme and routing resources [14][15]. Although a reasonable subset of the total STBus design-space is chosen (i.e. 8 initiators, 8 targets, 8 request and 8 response resources, 7 arbitration policies, 2 load capacitance range, 3 data path width range and 2 types of protocols), the gate-level power characterization is still a very hard task. The size of the STBus's design space configurations is in excess of  $3.4 \cdot 10^5$ . This would lead to more than 52 CPU years to fully characterize the power model on a Pentium4 platform running Linux at 2.2GHz clock freq. Each configuration usually needs 1 high effort synthesis +150 gate-level simulations (~2000 cycles each) + power measure. It is quite clear that running an exhaustive characterization for such a complex IP is far to be feasible in a reasonable time, even by leveraging on distributed computers. In this paper a smart *response surface method* approach is adopted to deal with this problem and shrinking the characterization effort. This approach is based on the following assumption: only a selected sub-set of all the possible configurations is synthesized and characterized. The remaining set of coefficients are estimated by accessing an appropriate set of models. Each model is based on an analytic equation or on a look-up table, built up through a proper response surface method. The power characterization process becomes much easier to be managed. Although this approach may lead to some loss in accuracy, it will be shown in the following sections that the global model efficiency can be taken well under control.

The paper is organized as follows: Section 2 introduces a short background on Network-on-Chip. Section 3 illustrates the STBus versatile interconnect IP as an industrial example of NoC infrastructure and the current power modeling methodology. Section 4 introduces the theory background of the proposed methodology while Section 5 summarizes the experimental results.

## 2 Background

Although the main concepts and the terminology of **Network-on-Chip** design has been introduced quite recently [1][2][3], both the industrial and research communities have been starting to realize the strategic importance of shifting the design paradigm of high-end digital IC from a deterministic, wire-based interconnection of individual blocks and IPs, to a thorough communication-based design methodology [4][7][9], aiming to face with data packetization and non-deterministic communication protocols in next generation's SoCs.

With the advent of 90nm and 75nm CMOS technology, the challenges to fix the Network-on-Chip (NoC) issue "by design" will include:

- To provide a functionally-correct, reliable operation of the interconnected components by exploiting appropriate network infrastructure and protocols, i.e. interconnections to be intended as "on chip micro-network" [5][6][7], which is an adaptation of the OSI protocol stack [18].
- To achieve a fluid "flexibility vs. energy-efficiency" system exploration, allowing an effective network centric power management [8][11][12]. Unlike computation energy in fact, the energy for global communication does not scale down with technology shrinking. This makes energy more and more dominant in communications.

Reaching those goals will be crucial to the whole semiconductor industry in the next future, in order to face with the escalating range of signal integrity and physical wiring issues, who are making the target IC reliability harder and exponentially expensive to achieve. As of today, there is a limited availability of tools able to consistently support this emerging design methodology. Indeed, some high level models (i.e. Transaction Level and Bus Cycle Accurate) for functional/performance system simulations are smoothly coming up [13]. However, power predictability of NoCs still remains an open issue.

## 3 Current STBus Power Modeling Methodology

STBus is versatile, high performances interconnect IP allowing to specify the communication infrastructure in terms of protocol, interface and parametric architectures [14][15]. It comes with an automated environment (*STBus generation kit*) suitable to support the whole design flow, starting from the system-level parametric network specification, all the way down to the mapped design and global interconnect floor-plan [16]. The protocol modes supported by STBus are compliant with VSIA standard [19]. In fact, they can scale up from *Peripheral*, to *Basic* and to *Advanced* mode, conventionally named Type-1, Type-2 and Type-3, respectively.

The STBus architecture builds upon the *node* module, configurable switch fabrics who can be instantiated multiple times to create a hierarchical interconnect structure. The topology of the switch fabric can be selected by choosing the number of resources dedicated to the request and the response packets; for example a shared bus interconnect has only 1 request and 1 response resources at a time, while a full crossbar has as many request and response resources as the number of initiators and targets connected to the node. Eventually, *type converter* and *size converter* modules can be



adopted to interface heterogeneous network domains working under different protocols (i.e. Type-1, 2 and 3) and/or different data-path widths.

### 3.1 Energy Characterization Flow

The energy macro-model of the whole STBus interconnection is partitioned into sub-components, corresponding to each micro-architectural block of the interconnection fabrics that are *node*, *type-converter* and *size-converter*. For sake of simplicity, in this paper we will show the results of the *node* component. However, the same automatic flow is currently applied to the whole STBus architecture. The proposed model relies on the bus utilization rate, i.e. the number of cells traveling through the bus, as well as on the interconnection topology (i.e. the number of masters/targets), which need to be pre-characterized, once and for all, through an accurate gate-level simulation for each target technology. The power characterization flow consists of 4 major steps depicted in Fig. 1.

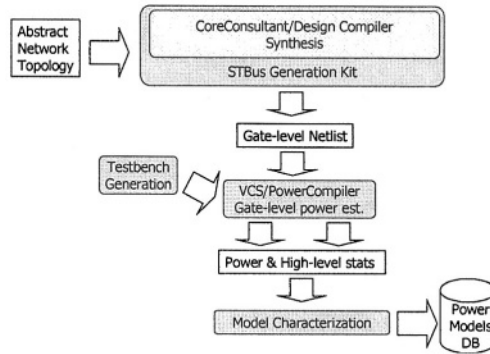


Fig. 1. STBus power characterization flow.

As already mentioned in section 3, the *STBus generation kit* allows the designer to automatically synthesize a gate-level netlist starting from a system-level parametric network specification. This is done by inferring the corresponding RTL code and, then, synthesizing all the way down to the mapped design [16]. Thus, an extensive set of gate-level power simulations (VCS/PowerCompiler) is launched within a Testbench Generation framework, specifically tuned to fulfill the many requirements imposed by the STBus protocols and, at the same time, to sensitize the node under a wide range of traffic workloads. Specifically, the test-benches can be configured in terms of average latency per master request and slave response and type of operations to be performed on the bus. The operations can be splitted in two categories (load and store) as they can play with different operand sizes (from 1 to 32 bytes).

### 3.2 STBus Energy Model

In this section, we introduce the power model for a generic configuration  $n$  of a node. The configuration of an STBus node identifies a specific instance out from the design space  $S$ :

$$S = \{ n \mid n = \langle i, t, rqr, rpr, p, C_L, dps, Type \rangle \} \quad (1)$$

where  $i$  is the number of initiators,  $t$  is the number of targets,  $rqr$  is the number of request resources,  $rpr$  is the number of response resources,  $p$  is the type of arbitration policy (STBus has 7 arbitration policies),  $C_L$  is the output pin capacitance (range:  $C_{Lmin} = 4$  Standard Loads ;  $C_{Lmax} = 1$  pF),  $dps$  is the data-path size (range: 32, 64 and 128 bit) and  $Type$  is the protocol mode (Type-2 and 3, in this case).

Based on an extensive experimental background, we recognize a fairly linear relationship between node energy and the rate of sent and received packet cells across all of the interconnection node's ports. Such a behavior matches with a set of random configuration samples across the entire design space and it has been confirmed during the model validation phase (see section 0).

The energy model for a generic configuration  $n$  of the STBus node is the following:

$$E(n) = P(n) \cdot C \cdot T_{clk} \quad (2)$$

where  $P(n)$  is the average power consumption of the node during a simulation of  $C$  clock cycles, with a clock period of  $T_{clk}$ . The power consumption  $P(n)$  is a linear combination of three contributions, according to the following equation:

$$P(n) = B(n) + P_{sent}(n) \cdot \frac{r_s}{C} + P_{rec}(n) \cdot \frac{r_r}{C} \quad (3)$$

where  $B(n)$  is the average base cost depending on the specific configuration  $n$  of the node,  $P_{sent}(n)$  is the additive power cost due to cell sent from the masters to the slaves and  $r_s$  is the total number of cells sent,  $P_{rec}(n)$  is the power cost due to each packet cells received by the masters,  $r_r$  is the total number of cells received by the masters and  $C$  is the number of clock cycles. In essence, the power model characterization consists in determining the value of the coefficients  $B(n)$ ,  $P_{sent}(n)$  and  $P_{rec}(n)$  for each specific configuration  $n$  of the node. The total avg. switching activity coming out from the Test-benches is kept at 0.5. As far as the interconnection capacitive load " $C_L$ " is concerned, our model supports a linear interpolation between  $C_{Lmin}$  and  $C_{Lmax}$  in order to provide a quite accurate estimation of the switching power under the specific load of the current instance.

This model has shown an average error of 1% on synthetic benchmarks and an error of 9% on realistic benchmarks such as a multiprocessor platform running scientific applications on top of an operating system.

## 4 Smart DoE to Boost STBus Power Modeling

Due to the huge domain size, the optimization of the Design of Experiments (DoE) to be adopted when characterizing the power macro-model is a key methodology issue to ensure the actual task feasibility. We decided to adopt a *response surface method* approach to allow this problem to be manageable and actually solvable through an automatic tool flow. Indeed, although this methodology has been developed to cope with STBus, the described solution can be easily applied to a number of generic parametric IP as well as to third party's on chip Bus architectures.

### 4.1 Surface Reconstruction Methods

Let us consider Equation 1 and Equation 3. Their coefficients are functions over the dominium  $S$ , which can be considered the cartesian product of two subspaces:

$$S = G \times X \tag{4}$$

$$G = \{ g \mid g = \langle i, t \rangle \} \tag{5}$$

$$X = \{ x \mid x = \langle rqr, rpr, p, C_v, dps, Type \rangle \} \tag{6}$$

The coefficients  $B$ ,  $P_{sent}$  and  $P_{rec}$  can be seen as a function of two variables:

$$f(n) = f(g, x) \tag{7}$$

The variable  $g$  belongs to a discretized space called ‘grid’, over the set of possible pairs  $\langle initiator, target \rangle$ , while variable  $x$  represents the remaining parameters, according to equation (6).

By considering  $x$  fixed, each coefficient becomes a surface over the set  $G$ . Experimentally, the surface shows to have a very smooth behavior; as an example, Figure 2 shows a representative behavior of the coefficient  $B$ .

The problem of fast characterization of the entire design space can be thought as reconstructing a surface by directly characterizing only a small subset of points  $G_s \subset G$ . The given methodology must assure that for an increasing ratio  $z = |G_s|/|G|$ , (i.e., the characterization effort of the power model library), the approximation error decreases. Ideally, for  $z=1$ , the error should be 0.

Our approach, can be decomposed in three fundamental steps:

1. Choice of a representative set  $X_s \subset X$  to be used as a training set for the evaluation of the different reconstruction methods.
2. Automatic optimization of the surface reconstruction methods over the space  $G \times X_s$ . The output of this stage will be a combination of algorithm and sub-grid  $G_s$  suitable to minimize fitting error and characterization effort.
3. Perform the actual characterization of  $G_s \times X$ .

The considered surface reconstruction methods are twofold.

- Regression-based algorithms: analytic approximation of a surface, usually a polynomial expression, in which each of the coefficients is fitted to minimize an overall square error. The procedure to fit these parameters is often called Least Square Approximation. Three regression methods have been considered: *linear*, *quadratic* and *cubic*.
- Interpolation-based algorithms: surface approximation by using multiple piecewise polynomials or more generic functions. Being based on look-up tables, the response surface always fits with all of the training set points. The following interpolation algorithms have been analyzed: *cubic*, *bicubic*, *bilinear* and *linear grid-data*. Algorithms belonging to the *spline* interpolation class have been experimentally rejected due to a worse accuracy.

As far as the sub-grid constraints is concerned, the regression based methods do not enforce any limitation on the sub-grid topology  $G_s$ , while interpolation methods

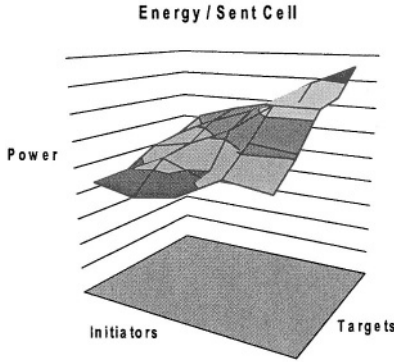


Fig. 2. An example surface for the base cost.

have several topology constraints, for which the reader is referred to the specific literature [20].

### 4.2 Choice of a Reference Training Set $X_s \subset X$

The training set  $X_s$  is composed by a random selection within  $X$  such that each value of the parameters  $rqr, rpr, p, C_L, dps, Type$  (see eq. 4) is evaluated at least once. In this way it is possible to uniformly visit the characterization space assuring that the probability to have a particular configuration with a different behavior with respect to the training-set is minimized. More specifically in our design case, considering 8 request and 8 response resources, 7 arbitration policies, 2 load capacitances, 3 data path sizes and 2 protocols, our method allows to reduce the training set from  $3.4 \cdot 10^5$  down to 30 configurations, without any significative degradation in accuracy.

### 4.3 Algorithm Benchmarking

Each of the afore mentioned algorithms has been benchmarked by varying the ratio  $z=|G_s|/|G|$  and by sampling the design space  $G$  in order to find an optimal solution  $G_s$ . The search for the optimal solution is driven by the following cost function:

$$C(G_s) = \sigma(G_s)\eta(G_s) \tag{8}$$

Where  $\sigma(G_s)$  is a guess of the characterization effort and  $\eta(G_s)$  is the maximum error between the predicted coefficients and the actual coefficients.

Then, for each value of  $z=|G_s|/|G|$ , we find out the optimal reconstruction algorithm and the corresponding grid  $G_s$  to be adopted for the final characterization campaign.

## 5 Experimental Results

In this paragraph we show the experimental figures supporting the proposed methodology. The experimental flow aims at selecting the optimal grid  $G_x$  together with the best reconstruction algorithm for a given grid size  $|G_x|$ . The analysis is performed on a subset of the entire design space configurations ( $X_s \subset X$ ), as explained in section 4.2.

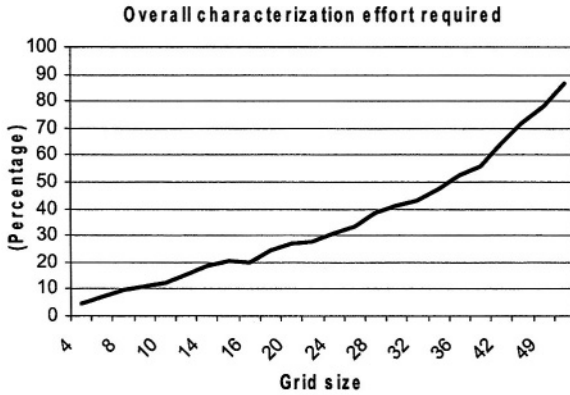


Fig. 3. Characterization effort for each grid-size

For each algorithm and grid size  $|G_x|$ , an iterative random search is applied to find the best grid, by optimizing the cost function given in Equation 8. The cost function is evaluated on the set of configurations that have been chosen as training set. As a first step, a preliminary analysis on the characterization effort has been performed. The characterization cost of the optimal grid ( $\sigma(G_x)$ ) has a strong correlation with respect to the grid size. Figure 3 shows the behavior of the characterization effort vs. the grid-size (100% is the maximum complexity space exploration, i.e., exhaustive exploration). Based on experimental evidences, it comes evident that the characterization effort ( $\sigma(G_x)$ ) has a loose dependency on the reconstruction algorithm unlike the maximum error, this last being worth to track in the cost function.

### 5.1 Regression Analysis

In this paragraph we show the results of the analysis of the maximum error given by the linear, quadratic, and cubic regression. As can be depicted from Figure 4, linear regression has a very poor behavior in terms of maximum error. Furthermore, the error diverges as more points are added to the training set. This is due to the fact that the cost function to be minimized has a tight relation with the synthesis effort. This may lead the search process to a set of solutions where low characterization effort is granted at the expense of a larger maximum error. Quadratic regression works like the linear regression, although the error is much smaller. This is due to the fact that the reconstruction method, for the same grid size, has a better modeling efficiency. Cubic regression has the same behavior of the quadratic except poor performance when

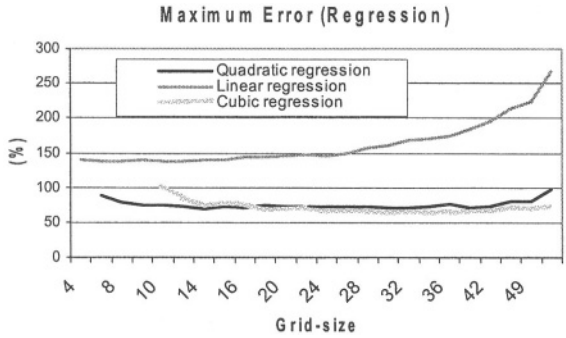


Fig. 4. Regression maximum error

dealing with small grid-size. Note that, for each curve, there is a variable minimum grid size necessary for applying the reconstruction algorithm (e.g. 6 points for two dimensional quadratic). All these motivations justify the choice of the quadratic regression as representative algorithm for the regression family. Although the max error for quadratic regression is about 70%, the average error for all the regression methods is under 15% and the standard deviation is bound to 10%.

### 5.2 Interpolation Analysis

In this paragraph we show the analysis of maximum error given by the following interpolation methods: Bi-Linear, Bi-Cubic, Linear Grid Data and Cubic Grid Data.

Bilinear interpolation, also known as *first-order interpolation*, computes the approximation of the target function on a given point  $p$  by using a bilinear function computed on the four points in the training set nearest to  $p$ . Bicubic interpolation reduces re-sampling artifacts even further by using the 16 nearest points and by using bicubic waveforms. It preserves the fine granularity available in the original surface, at the expense of additional computational time.

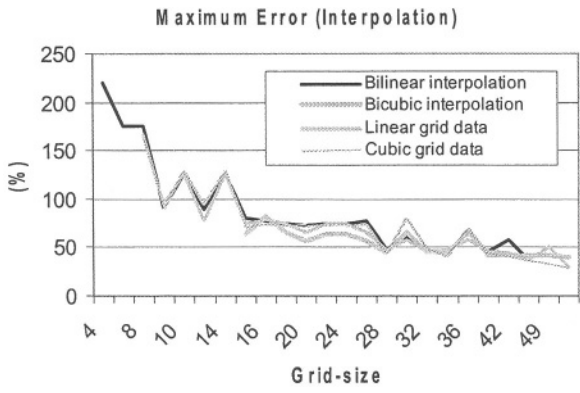


Fig. 5. Interpolation maximum error

Linear and cubic grid data methods are based on bilinear and bicubic interpolation respectively, where a Delaunay triangulation is applied on both of them, before the actual interpolation. The Delaunay triangulation is a transformation allowing to extract more information contents from the training set and, in some cases, lowering the maximum error.

Figure 5 shows the behavior of the maximum error by increasing the number of points in the grid. The cubic grid data technique shows the best performance. In general, for all these methods, the max error can be controlled by increasing the number of points and, in theory, it reaches 0% for the maximum grid size. Also in this case, the reported average error is very low (less than 15%) with a standard deviation under 6.5%.

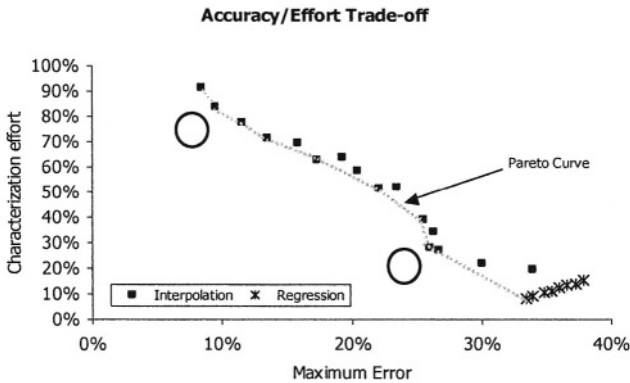


Fig 6. Characterization effort vs. Maximum error Trade-off

### 5.3 Combined Analysis

To compare the various methods we are going to focus our attention on the best surface reconstruction method for each category (regression and interpolation). As far as the regression is concerned, the quadratic algorithm shows to be the most promising for small grid-sizes, even better than the interpolation-based methods. On the other side, regarding interpolation, the cubic grid data technique is the most accurate in terms of maximum error. Accordingly, Figure 6 shows the best Accuracy/Effort trade-off for each grid size  $|Gs|$ , with respect to *Quadratic Regression* (shortened as ‘Regression’) and *Cubic Grid Data Interpolation* (shortened as ‘Interpolation’). Moreover, the figure shows the Pareto curve highlighting points that are not dominated in both directions by any other point. This curve is useful to screen out the solutions that optimize both the merit figures and, at the same time, meet specific constraints.

For example, a quadratic regression method should be used if the characterization effort is a dominating constraint. In fact, in this case we have only 10% of the original characterization effort at the expense of a 33% maximum error. For medium-high characterization effort, the Cubic Grid Data interpolation has shown to reduce the

maximum error more than the other methods. It leads to a maximum error of about 18% at the expense of 62% of the original characterization effort.

## 6 Conclusion

A new and effective methodology to minimize the Design of Experiments (DoE) of NoC power models has been presented. The DoE minimization is crucial to characterize a set of innovative energy macro models of Network-on-Chip architectures. By properly combining regression (polynomial) and interpolation (table-based) techniques, this methodology aims to reduce the overwhelming complexity of sampling the huge, non-linear design space involved with the system operations of a high-end parametric on-chip-network module. The experimental figures show that our DoE optimization techniques are able to trade off power modeling approximation with model building cost, leading up to 90% reduction of the sampling space.

## References

- [1] J. Duato, S. Yalamanchili, L. Ni, "*Interconnection Networks: an Engineering Approach*", IEEE Computer Society Press, 1997.
- [2] K. Lahiri, S. Dey et al., "*Efficient Exploration of the SOC Communication Architecture Design Space*", Proc. of ICCAD-2000, Nov. 2000, S. Jose, USA.
- [3] W. Dally, B. Toles, "*Route Packets, not Wires: On-Chip Interconnection Network*", Proceedings of 38<sup>th</sup> DAC 2001, June 2001, Las Vegas, USA.
- [4] Sangiovanni Vincentelli, J. Rabaey, K. Keutzer et al., "*Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design*", Proceedings of 38<sup>th</sup> DAC 2001, June 2001, Las Vegas, USA.
- [5] F. Karim, A. Nguyen et al., "*On Chip Communication Architecture for OC-768 Network Processors*", Proceedings of 38<sup>th</sup> DAC 2001, June 2001, Las Vegas, USA.
- [6] K. Lahiri, S. Dey et al., "Evaluation of the Traffic Performance Characteristics of System-on-Chip Communication Architectures", Proc. 14<sup>th</sup> Int'l Conference on VLSI Design 2001, Los Alamitos, USA.
- [7] L. Benini, G. De Micheli, "*Network on Chip: A New SoC Paradigm*", IEEE Computer, January 2002.
- [8] T. Ye, L. Benini, G. De Micheli, "*Analysis of power consumption on switch fabrics in network routers*", Proceedings of 39<sup>th</sup> DAC 2002, June 2002, New Orleans, USA.
- [9] S. Kumar et al., "*A network on chip architecture and design methodology*", International Symposium on VLSI 2002.
- [10] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "*Orion: A Power-Performance Simulator for Interconnection Networks*", International Symposium on Microarchitecture, MICRO-35, November 2002, Istanbul, Turkey.
- [11] T. Ye, G. De Micheli and L. Benini, "*Packetized On-Chip Interconnect Communication Analysis for MPSoC*", Proceedings of DATE-03, March 2003, Munich, Germany, pp. 344-349.
- [12] J. Hu and R. Marculescu, "*Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures*", Proceedings of DATE-03, March 2003, Munich, Germany, pp. 688-693.
- [13] T. Grotker, S. Liao, G. Martin and S. Swan, "*System Design with SystemC*", Kluwer Academic Publishers, 2002.



- [14] “*STBus Communication System: Concepts and Definitions*”, Reference Guide, STMicroelectronics, October 2002.
- [15] “*STBus Functional Specs*”, STMicroelectronics, public web support site, [http://www.stmcu.com/inchtml-pages-STBus\\_intro.html](http://www.stmcu.com/inchtml-pages-STBus_intro.html), STMicroelectronics, April 2003.
- [16] Synopsys Inc., “*Core Consultant Reference Manual*”, “*Power Compiler Reference Manual*” and “*VCS: Verilog Compiled Simulator Reference Manual*”, v2003.06, June 2003.
- [17] C. Patel, S. Chai, S. Yalamanchili, and D. Schimmel, “*Power-constrained design of multiprocessor interconnection networks*,” in Proc. Int. Conf. Computer Design, pp. 408-416, Oct. 1997.
- [18] H.Zimmermann, “*OSI Reference Model – The ISO model of architecture for Open System Interconnection*”, IEEE Trans. on Communication, n 4, April 1980.
- [19] VSI Alliance Standard, “*System-Level Interface Behavioral Documentation Standard Version 1*”, Released March 2000.
- [20] Box, George E. P. and Draper Norman Richard. *Empirical model-building and response surfaces*, John Wiley & Sons New York, 1987
- [21] Bona, V. Zaccaria and R. Zafalon, “*System Level Power Modeling and Simulation of High-End Industrial Network-on-Chip*”, IEEE DATE-04, February 2004, Paris, France

# Exploiting Dynamic Workload Variation in Offline Low Energy Voltage Scheduling\*

Lap-Fai Leung<sup>1</sup>, Chi-Ying Tsui<sup>1</sup>, and Xiaobo Sharon Hu<sup>2\*\*</sup>

<sup>1</sup> Department of Electrical and Electronic Engineering  
Hong Kong University of Science and Technology  
Clear Water Bay, Hong Kong SAR, China  
{eefai, eetsui}@ee.ust.hk

<sup>2</sup> Department of Computer Science and Engineering  
University of Notre Dame  
Notre Dame, IN 46556, USA  
shu@cse.nd.edu

**Abstract.** In this paper, a novel off-line voltage scheduling algorithm, which exploit the dynamic workload variation is proposed. During the construction of the voltage schedule, instead of optimizing the energy consumption assuming all the tasks are running in the worst case workload, we derive a schedule that results in low energy consumption when the tasks are running at a given workload distribution while at the same time can guarantee no deadline violation when the worst-case scenario really happens. By doing so, more slacks are generated and lower voltages can be used when the tasks are really running at workloads that are less than the worst case values. This work can be viewed as an interaction between the off-line voltage scheduling and on-line dynamic voltage scaling. The problem is formulated as a constrained optimization problem and optimal solution is obtained. Simulation and trace-based results show that, by using the proposed scheme, significant energy reduction is obtained for both randomly generated task sets and real-life applications when comparing with the existing best off-line voltage scheduling approach.

## 1 Introduction

Real-time embedded systems (RTES) are prevalent in many applications such as automobiles, consumer electronics, etc. With the advances of technology and the increasing demand of functionality on such systems, energy consumption is becoming a critical concern in RTES design, especially for mobile applications. RTES are generally composed of a number of tasks to be executed on one or more embedded processors and the tasks have to be finished with a hard deadline. To reduce energy consumption, many modern embedded processors support both variable supply voltage and the controlled shutdown mode [1,2,3]. How to maximally exploit the benefit

---

\* This work was supported in part by Hong Kong RGC CERG under Grant HKUST6214/03E, HKUST HIA02/03.EG03 and HKUST DAG 02/03.EG26.

\*\* Hu's work is supported in part by U.S. National Science Foundation under grant number CCR02-08992.

provided by such hardware has been an active research topic during the past several years.

Dynamic voltage scaling (DVS), i.e., changing the supply voltage and the clock frequency of a processor at runtime according to the specific performance constraints and workload, is proven to be very effective for reducing energy consumption [4,5]. Having an effective voltage schedule, i.e., the voltage to be used at any given time, is critical to harvest the DVS benefit. There are two main approaches to find a voltage schedule. One approach is to determine the schedule during runtime only. A number of research results on this approach have been published, e.g. [6,7,8,9]. These results can work with either real-time or non-real-time tasks. The basic principle is to use the runtime workload information and the slack generated to determine the voltage being used for the following tasks. Though such approaches have been shown to result in energy saving, they do not exploit the fact that much information about tasks in an RTEs, such as task periods, deadlines, and worst-case execution cycles (WCEC), are available offline. It is not difficult to see that not using such information may lose opportunities to further reduce the energy consumption.

To complement the above runtime approaches, the other category of voltage scheduling work finds the desired voltage schedules offline based on the available task information, e.g., [4,5,10,11,12,13]. These techniques are generally applicable to real-time tasks with hard deadlines. To ensure that the schedules obtained in offline do not violate any timing constraint, the worst-case execution cycles (WCEC) of each task is always used in the offline analysis. Such offline voltage schedules can be altered to some extent at runtime by using the slacks resulted from the tasks not executing at the WCEC to lower the voltage computed offline [5,10]. The new voltage of a task is calculated by extending the reserved execution time by the slack distributed for the task.

The effectiveness of the offline approach combined with the runtime DVS is very much dependent on how the slacks are generated and distributed during run-time, which in turn depends on the off-line static schedule. Therefore, it is important to schedule real-time tasks in such a way that maximum slack can be generated and exploited and this depends on the actual workload of the tasks. For many real-time systems, tasks are often having a workload that is close to the average case execution cycle (ACEC) and only occasionally execute at WCEC. For these systems, WCEC and the workload distribution can be obtained at design time through static analysis, profiling, or direct measurements [14]. For example, in an MPEG4 video encoding system, the processing of most of the macro-blocks does not need WCEC. As we pointed out above, all the existing offline voltage scheduling techniques use WCEC to obtain the static schedules. Hence, the schedules are only effective when the actual execution cycles are close to WCEC. In general, the slack distributions based on these schedules greatly limit the flexibility and effectiveness of utilizing the slacks generated from the actual smaller execution cycles during runtime.

In [15], an Uncertainty-based Scheduling was proposed which used ACEC for the task re-ordering during off-line scheduling. However this approach works only for tasks that are independent. Therefore only ordering of the tasks is generated and there is no special static voltage schedule of the tasks.

In this paper, we present a novel offline scheduling approach which works for both dependent and independent task-sets. The schedule results in the best slack distribution in terms of energy saving if the tasks are executing with a workload close to the ACEC or according to a given workload distribution, while still guarantees no dead-

line violation if the tasks need WCEC to finish. Actually we can view this as an interaction between the off-line voltage scheduling and on-line voltage scaling through the use of ACEC. The interaction depends on the end-time of a task obtained in the off-line scheduling. When slack ( $S_i$ ) is available for a task  $T_i$  during runtime, the new voltage assigned to  $T_i$  depends on the new available time which is equal to the sum of  $S_i$  and  $(te_i - te_{i-1})$ , where  $te_i$  and  $te_{i-1}$  are the scheduled end-time of  $T_i$  and  $T_{i-1}$ , respectively. We can see that the run-time voltage depends on the scheduled end-time of a task. The existing static voltage scheduling algorithm calculates the end-time of the tasks based on the WCEC. This will not give an optimal slack utilization for the run-time dynamic voltage scaling. Here we use the ACEC or the workload distribution to obtain the optimal end-time instead. In order to meet the deadline requirement during runtime, the WCEC is also considered judiciously in the problem formulation. This makes sure that sufficient amount of time is reserved for each task such that no deadline violation occurs even under the worst case situation. To the best of our knowledge, this is the first work that incorporates the ACEC, the workload distribution, and the WCEC together during the offline variable voltage scheduling. Our focus is to reduce the overall energy consumption for the entire time duration when an RTES is deployed for an application under normal situation, e.g. the energy consumed by an MPEG encoder in encoding a long video clip. We have applied our technique to a number of real-time tasks sets and the results are very encouraging. For example, comparing with the existing best static scheduling approach [11], the energy consumption is reduced by as high as 41% for randomly generated task sets and 11.3% for an MPEG encoding system.

## 2 Preliminaries and Motivation

In this paper we assume a frame-based real time system in which a frame of length  $L$  is executed repeatedly [9]. A set of  $N$  tasks  $T = (T_1, \dots, T_N)$  is to execute within each frame and is to complete before the end of the frame. The task execution is assumed to be non-preemptive. (Note that our technique works for both independent tasks and dependent tasks as well as for multiple processors by applying the same modeling approach as that in [8]. For simplicity, we only consider the single processor case.) The precedence constraints among the tasks are represented by an acyclic data and control flow graph  $G$ . For tasks that have different periods, each instance of a task within the hyper-period is considered to be a task. Each task  $T_i$  is associated with the following parameters. (We use upper case letters for constants and lower case for variables.)

$C_i$	Effective switching capacitance of task $T_i$	$v_i$	Supply voltage of task $T_i$
$W_i$	Execution cycles of task $T_i$	$d_i$	Execution time of $T_i$
$R_i$	Release time of task $T_i$	$e_i$	Energy consumption for executing $T_i$
$D_i$	Deadline of task $T_i$		

The clock cycle time,  $CT$ , and task  $T_i$ 's execution time  $d_i$  can be computed as

$$CT = \frac{\lambda \cdot v_i}{(v_i - V_{th})^\alpha} \quad (1)$$

$$d_i = W_i \cdot CT = W_i \cdot \frac{\lambda \cdot v_i}{(v_i - V_{th})^\alpha} \tag{2}$$

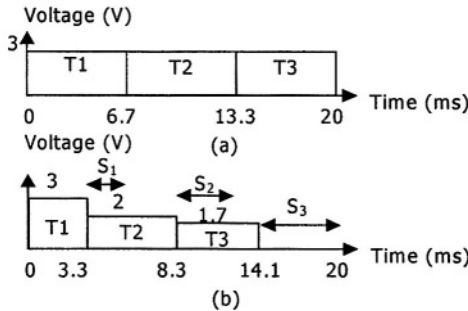
where  $V_{th}$  is the threshold voltage of the processor,  $\lambda$  is a circuit related parameter and  $\alpha$  is the process constant between 1 and 2. The total energy consumption of executing task  $T_i$  is given by

$$e_i = C_i W_i v_i^2 \tag{3}$$

We use a simple example to illustrate the effect of a static schedule on the energy saving when dynamic slack is exploited in the runtime DVS. Suppose an RTES contains three tasks with the parameters of each task specified in Table 1 (assuming the release time of each task is 0sec.).

**Table 1.** Task parameters

Task	$C_i$ ( $\mu$ F)	WCEC	ACEC	Actual execution cycles	$D_i$ (ms)
$T_1$	1	20	10	10	10
$T_2$	1	20	10	10	15
$T_3$	1	20	10	10	20



**Fig. 1.** (a) Static voltage schedule (b) Actual voltage

Figure 1 (a) gives the optimal static schedule if WCEC are taken by all tasks. For simplicity, we assume the clock cycle time is inversely proportional to the supply voltage and the minimum and maximum supply voltages are 0.7V and 5V, respectively. Figure 1(b) gives the actual dynamic run-time schedule when dynamic slack redistribution is carried out. During runtime, tasks finish earlier since their actual execution cycles are smaller than the WCEC. Here we use a greedy slack distribution to redistribute the slack to the successive tasks. All the slack obtained from the finished tasks is given to the next task for execution. For example, slack  $S_1$  obtained from task  $T_1$  is 3.3ms as shown in Figure1(b) and is utilized fully by the task  $T_2$ . The supply voltage of  $T_2$  is re-calculated based on the WCEC of  $T_2$  and the scheduled end-time and is given by  $v_2=20/(13.3-3.3)=2$ . Similarly, slack  $S_2$  generated by task  $T_2$  is 5ms and  $T_3$  can use an even lower voltage. By using (3), the overall energy consumption for executing the tasks based on the schedule given in Figure 1 (a) is 158.9 $\mu$ J. It is clear that the dynamic slack redistribution indeed leads to more energy saving com-

paring with just using the static voltage scheduling only. However, if we know that the tasks most probably have workload close to the ACEC value during actual execution, can we do better?

Let's examine the static schedule in Figure 1 more closely. In this schedule, each task is associated with a predetermined end time,  $te$ , e.g.,  $T_1$ 's end time is 6.7ms,  $T_2$ 's is 13.3ms. These end times are then used in the dynamic slack distribution process to compute a new voltage schedule. Therefore, the static schedule essentially determines the latest end time for each task. (Note that this *predetermined* end time can be different from the *actual* end time when a task does not assume the WCEC. Since this predetermined end time is used frequently in our discussion, we simply call it the end time). During static voltage scheduling, these end times are obtained so that the tasks will complete by their deadlines and the overall energy consumption is minimized if all tasks take on the WCEC values. Now, let us consider another different schedule where the end time of each task is given as follows:  $te_1$  is 10ms,  $te_2$  is 15ms and  $te_3$  is 20ms. Using this schedule and the same greedy slack distribution as above, we obtain the runtime schedule as shown in Figure 2(a). At time zero, the WCEC of  $T_1$  and its end time 10ms is used to compute the voltage needed. Since the actual execution cycles of  $T_1$  is 10, a slack of 5ms is generated by  $T_1$ .  $T_2$  fully utilizes this slack to compute its voltage by using the end time of 15ms and its WCEC. Now  $T_2$  uses the actual execution cycles and finishes at 10ms. The new voltage for  $T_3$  can be obtained in the same manner. The overall energy consumption for the schedule in Figure 2(a) is  $120\mu\text{J}$ , a 24% improvement comparing with that of the schedule in Figure 1(b).

Though the schedule used by Figure 2(a) leads to a bigger energy saving, it is important that the schedule can still meet the deadline requirement when tasks assume the WCEC. The schedule dictates that the end time of each task is no later than its deadline. However, if the schedule is not carefully chosen, the tasks may not be able to finish by their deadlines during runtime. Figure 2(b) shows the situation if the tasks do need the WCEC during runtime. At time zero, a 2V voltage is adopted for  $T_1$ . Since  $T_1$  takes the WCEC, it will not finish until 10ms. The voltages for  $T_2$  and  $T_3$  can be computed accordingly. Note that a 4V voltage is needed to execute both  $T_2$  and  $T_3$  in order to meet the timing constraints. If the maximum voltage level for the processor is 3.3V, the schedule would not be feasible. Therefore, simply using the task deadlines as the desired end times does not always give a feasible schedule.

We would like to point out that for the schedule in Figure 2(b), when tasks really take the WCEC, it consumes  $720\mu\text{J}$  energy, which is a 33% increase comparing with that of the schedule in Figure 1(a). However since in normal running condition, the actual execution cycles of a task are close to the average value, the overall gain is higher. Based on this observation, we would like to find a static schedule that result in better energy saving on average but still satisfy the timing requirements for the worst case.

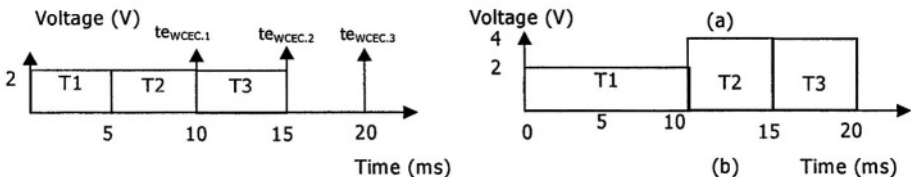


Fig. 2. Actual voltage schedule if the tasks assume the (a) ACEC. (b) WCEC.

### 3 Our Approach

From the discussion above, we can see that the greedy slack distribution (or any other slack distribution) relies heavily on the task end time calculated in the static schedule. Existing static voltage scheduling techniques all employ the WCEC in order to guarantee that no deadline violation occurs during runtime. Because of the use of the WCEC, the end time of each task is more restricted. If we could extend the end time of each task to as long as that allowed by the worst-case execution scenario, it can achieve much more energy saving on average. Assume that the effective switching capacitance, the workload distribution, WCEC, release time and deadline of each task are all given, we want to find a desired schedule, i.e., the desired end time of each task, which strives to maximize the potential energy saving when the tasks are executing based on the workload distribution.

In this section, we show that this scheduling problem can be formulated as a mathematical programming problem. We ignore the voltage transition overhead in our formulation due to the following reasons: In most RTES applications, the task execution time is much longer than the voltage transition time. For example, the execution time of the tasks in the MPEG encoder is in the order of milliseconds that are larger than the voltage transition time  $\Delta t$  which is in the order of microseconds. As stated in [16], the increase of energy consumption is negligible when the transition time is small comparing with the task execution time. In the rest of this section, we adopt the following convention:  $x$  and  $\bar{x}$  indicate the average case  $x$  and the worst case  $x$ , respectively. For example,  $\bar{W}_i$  and  $\overline{W}_i$  are the average execution cycles and the worst case execution cycles of task  $T_i$ , respectively.

#### 3.1 Problem Formulation

We consider the case where the execution order of a given task set is already available. That is,  $T_i$  is executed before  $T_j$  if  $i < j$ . The task execution order can be determined based on existing algorithms, such as the earliest deadline first algorithm, or the heuristic algorithm proposed in [11]. We first assume the processor can use any voltage value within a specified range. This assumption will be relaxed in the next sub-section. The continuous voltage assumption is acceptable for processors with more than 10 voltage levels, such as the Crusoe processor [2].

Determining the schedule that optimizes the energy consumption based on the tasks workload distribution while satisfying the timing requirement in the worst case can be formulated as a Non-Linear Programming (NLP) problem. The model consists of an objective function that minimizes the expected energy of the system when tasks take on the workload distribution subject to a set of resource and timing constraints. The interesting part of this formulation is the way we relate the expected execution cycle based on the probability density function of the workload and the worst case execution cycle. If the probability density function is not known, we can use the ACEC in the formulation as an approximation. In [15], it is shown that this is a good enough approximation of the average energy consumption. In the following, we will first describe the formulation based on the ACEC. Later we will extend the formulation assuming the probability density function is known.

The objective function of the NLP formulation is

$$\text{Min. } \sum_{i=1}^N (C_i W_i v_i^2) \quad (4)$$

Let  $te_j$  and  $ts_j$  be the end time and the start time of  $T_j$ , respectively. To meet the release time and deadline requirements as well as the voltage range requirement, the following constraints are obtained:

$$R_i \leq ts_i \quad \forall 1 \leq i \leq N \quad (5)$$

$$te_i \leq D_i \quad \forall 1 \leq i \leq N \quad (6)$$

$$V_{\min} \leq v_i, \bar{v}_i \leq V_{\max} \quad \forall 1 \leq i \leq N \quad (7)$$

$$te_i = ts_i + \bar{W}_i \cdot \frac{\lambda \cdot v_i}{(v_i - V_{th})^\alpha} \quad \forall 1 \leq i \leq N \quad (8)$$

Also, we need to make sure that there has enough allowable working time between the end time of  $T_i$  and  $T_{i+1}$  for  $T_{i+1}$  to finish if all the tasks use WCEC. We express this by the following constraint:

$$te_{i+1} - te_i \geq \bar{W}_{i+1} \cdot \frac{\lambda \cdot \bar{v}_{i+1}}{(v_{i+1} - V_{th})^\alpha} \quad \forall 1 \leq i \leq N - 1 \quad (9)$$

If we do not need to consider dynamic slack distribution, we would have  $te_i \leq ts_{i+1}$  in order to ensure that no task executions are overlapped. Allowing the slacks of the finished tasks to be utilized by the subsequent tasks can be thought as that the start time of  $T_{i+1}$  becomes earlier than the end time of  $T_i$  as  $T_i$  uses ACEC instead of WCEC. Here we assume that the greedy slack distribution is used, i.e., the slack of the current finishing task will be used fully by the next task. Also to reduce the complexity of the problem formulation, we assume that the slack available for  $T_{i+1}$  is mainly from the slack generated by the immediate preceding task  $T_i$ . Then, the difference between  $te_i$  and  $ts_{i+1}$  is bounded by the difference of the worst case execution time and the average case execution time, i.e., the slack of  $T_i$ . Therefore, we have the following constraint:

$$ts_{i+1} \geq te_i - \frac{\lambda \cdot (\bar{W}_i - W_i) \cdot \bar{v}_i}{(\bar{v}_i - V_{th})^\alpha} \quad \forall 1 \leq i \leq N - 1 \quad (10)$$

Now we consider the case that the probability density function of the workload is known in advance through static analysis, profiling, or direct measurements [14]. Here a more accurate objective function can be obtained. It is difficult and almost impossible to assign a variable for every possible execution cycle value. Here we use a binning method instead. We divide the execution cycles of the task into  $K$  different ranges (bins). The average workload of each bin  $k$  of  $T_i$  is  $W_{i,k}$  and the associated occurrence probability is  $P_{i,k}$ . From eqns (8) and (10), we know that the voltage used for task  $T_i$  depends on the start-time  $ts_i$  which in turn depends on the slack obtained from the previous  $T_{i-1}$ . Now the slack generated by  $T_{i-1}$  follows the probability density function of  $T_{i-1}$ 's workload. We denote the start-time of  $T_i$  when the workload of the previous task  $T_{i-1}$  is  $W_{i-1,k}$  by  $ts_{i,k}$ , and the corresponding supply voltage by  $v_{i,k}$ . We also



set the initial condition for the start-time and voltage of the first task  $T_i$  such that:  $ts_{i,k} = 0$  and  $v_{i,k} = \bar{v}_i$ . Furthermore we assume the workload of  $T_i$  is independent of the workload of  $T_{i-1}$ . Using the above notations, the objective function can be rewritten as following:

$$\text{Min. } \sum_{i=1}^N \sum_{k=1}^K [C_i \cdot p_{i,k} \cdot W_{i,k} \cdot \sum_{m=1}^K p_{i-1,m} v_{i,m}^2] \tag{11}$$

The other constraints (5), (7), (8) and (10) also need modifications accordingly. We group the modified constraints (5) and (7) into (12)

$$R_i \leq ts_{i,k}, V_{\min} \leq v_{i,k}, \bar{v}_i \leq V_{\max} \quad \forall 1 \leq i \leq N, \forall 1 \leq k \leq K \tag{12}$$

For the constraints of (8) and (10), we get the following constraints accordingly.

$$te_i = ts_{i,k} + \bar{W}_i \cdot \frac{\lambda \cdot v_{i,k}}{(v_{i,k} - V_{th})^\alpha} \quad \forall 1 \leq i \leq N, \forall 1 \leq k \leq K \tag{13}$$

$$ts_{i+1,k} \geq te_i - \frac{\lambda \cdot (\bar{W}_i - W_{i,k}) \cdot \bar{v}_i}{(\bar{v}_i - V_{th})^\alpha} \quad \forall 1 \leq i \leq N - 1, \forall 1 \leq k \leq K \tag{14}$$

From the above formulation, we determine the optimal value of  $te_i$  of each task  $T_i$  such that the overall objective function is minimized with the corresponding workload distribution functions. The end time values ( $te_i$ ) are then used in the runtime DVS algorithm to determine the actual voltage to be used. The nonlinear functions in the above NLP are separable convex, and separable convex nonlinear programming is proven to be polynomial time solvable [17]. Therefore, the NLP problem can be solved quite efficiently. For example, solving the NLP of 100 tasks takes less than 1 second using the setting described in Section 4.

### 3.2 Discrete Voltage Levels

In this sub-section, we tackle a more general case than that in Section 3.1. Specifically, we consider the case that the processor only has discrete voltage levels. Most processors available in the market can support only a certain number of voltage levels. For example, the Athlon 4 Processor from AMD [3] can only operate at five voltage levels. To cater for this, we extend the formulation in Section 3.1 by using the integer linear programming (ILP) to solve our problem. To simplify the discussion, we show the case that only ACEC and WCEC are available.

Suppose we have  $H$  discrete voltage levels, ranging from  $V_{\min}$  to  $V_{\max}$ . We introduce  $2 * N * H$  binary variables. Specifically, we have  $z_{ih}$ , and  $\bar{z}_{ih}$ , where  $z_{ih}=1$  (or  $\bar{z}_{ih}=1$ ) if  $T_j$  uses  $V_h$  in ACEC (or WCEC) and  $z_{ih}=0$  (or  $\bar{z}_{ih}=0$ ) otherwise. For a given  $V_h$ , cycle time  $CT_h$ , delay  $d_h$  and energy consumption per cycle  $e_{j,h}$  are computed by (1)-(3). Using the similar approach as in Section 3.1, we can re-write the objective function as:

$$\text{Min. } \sum_{i=1}^N [C_i (W_i \cdot \sum_{h=1}^H z_{ih} V_h^2)] \tag{15}$$

For the constraints (8) and (9), we get the following accordingly.

$$te_i = ts_i + \sum_{h=1}^H W_i \cdot z_{ih} \cdot \frac{\lambda \cdot V_h}{(V_h - V_{th})^\alpha} \quad \forall 1 \leq i \leq N \quad (16)$$

$$te_{i+1} - te_i \geq \sum_{h=1}^H \bar{W}_{i+1} \cdot \bar{z}_{(i+1)h} \cdot \frac{\lambda \cdot V_h}{(V_h - V_{th})^\alpha} \quad \forall 1 \leq i \leq N-1 \quad (17)$$

Regarding the effect of dynamic slack distribution, constraint (10) is rewritten as

$$ts_{i+1} \geq te_i - \sum_{h=1}^H \bar{z}_{ih} \cdot \frac{\lambda(\bar{W}_i - W_i) \cdot V_h}{(V_h - V_{th})^\alpha} \quad \forall 1 \leq i \leq N-1 \quad (18)$$

To ensure the validity of the ILP solution, the binary variables must satisfy the following constraints:

$$\sum_{h=1}^H z_{ih} = 1, \sum_{h=1}^H \bar{z}_{ih} = 1 \quad \forall 1 \leq i \leq N \quad (19)$$

Note that the ILP problem size can grow rather large for a large task set. Solving the ILP problem exactly can be difficult. A series of experiments have been done on the ILP solving time. For a system containing 10 tasks and a processor operates at 15 discrete voltage levels, it takes about 500 seconds to find the optimal selection of the supply voltage level. For large task-sets, we adopt the following approximation method based on the fact that the NLP problem in Section 3.1 can be solved efficiently. The integer constraints in the ILP are relaxed to the NLP formulation in Section 3.1 and the NLP formulation is solved first. The continuous voltage variable for each task will be rounded up to the next higher discrete voltage level. Using a higher supply voltage guarantees that the deadline requirements are met.

## 4 Experimental Results

To demonstrate the effectiveness of the proposed technique, several experiments using both randomly-generated examples and real-life applications were carried out. We first constructed 100 random task sets, which have 50 tasks each. Each task set was repeatedly simulated for 100 times. Similar to the experimental settings in [10], we consider the number of execution cycles of each task varying between the best case (BCEC) and worst case (WCEC). The ACEC/WCEC ratio is ranging from highly flexible execution (=0.1) to almost fixed (=0.9). The WCEC of each task was chosen from a uniform distribution between 100 and 1000, and the deadline  $D_j$  of a particular task  $T_j$  was adjusted such that the processor utilization is about 2/3 when all the tasks are running at the maximum speed [10]. Also, we employed the bin method mentioned in Section 3.1 with 10 bins for a more accurate result. Similar to the work in [11], the processor is assumed to have 15 evenly distributed discrete voltage levels.

We compare the improvement of energy reduction with the algorithm proposed in [11] (we denote it as BFP) and the static voltage scaling algorithm (SVS) proposed in [7] with different ACEC/WCEC ratio on a single-processor system. Figure 3 shows the improvement on energy reduction using the proposed method over BFP and SVS. It is important to mention that the existing approach in [11] using the worst case tim-

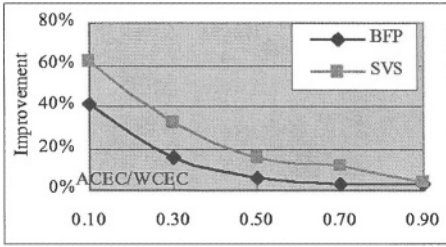


Fig. 3. Artificial tasksets

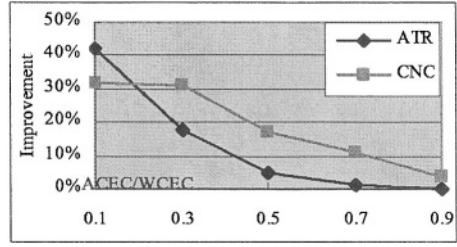


Fig. 4. ATR and CNC

ing information in the static scheduling phase already provides a large amount of energy saving when DVS is applied at runtime. But our approach contributes even more energy saving after applying the DVS at runtime as shown in the results.

In Figure 3, we can see that comparing with BFP, the improvement reaches the highest value, 41%, when the ACEC/WCEC value is 0.1. This is because there are a lot of slacks available and our approach provides a much better slack utilization and minimizes the overall energy consumption. However, when there is little slack available, i.e., when the normalized mean value of the execution cycle is high, there is not much improvement because the processor utilization rate is so high and there is little room for both methods to reduce the energy consumption. Also, we can see that the gain in energy reduction over the SVS approach is even larger. The improvement is over 60% if ACEC/WCEC ratio is equal 0.1.

To further validate the proposed algorithm, we applied our algorithm to three real-life applications, namely, a computer numerical control CNC [18], an automatic target recognition (ATR) system [9] and a MPEG4 video encoding system (MPEG). We compared the energy reduction with BFP. Since the periods of the tasks in CNC are different, we computed the hyper-period of the tasks by taking the least common multiple of the period of each task. Figure 4 summarizes the improvement in energy reduction for CNC and ATR. It can be seen that the improvement over BFP are about 32% for CNC and 42% for ATR when the ACEC/WCEC ratio is 0.1. For the MPEG4 encoder system, we obtain the workload using the dynamic for several movie clips and the simulation was done on 100 frames for each clips. The energy reduction improvement is about 11.3% for slow motion movie and 9.4% for fast motion movie on average. Slow motion movie outperforms fast motion movie because the workload can be more accurately estimated using the ACEC.

For all the experiments that we did, the solving time is very reasonable. For smaller task-sets, the solving time is less than 1 second while for the larger task-sets such as CNC which has about 300 tasks, the solving time is about 4 seconds.

## 5 Conclusions

A novel energy reduction strategy in static scheduling phase was introduced. The workload variation of each task is taken into account during the offline static voltage scheduling. By doing so, the runtime DVS is given more flexibility and potential to generate more slack under the given workload distribution. Experimental results showed that significant energy reduction was achieved.

## References

- [1] T. Burd, T. Pering, A. Stratakos and R. Brodersen, "A dynamic voltage scaled micro-processor system," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 1571-1580, 2000.
- [2] <http://www.crusoe.com/>
- [3] AMD Athlon 4 Processor, Data Sheet Reference #24319, AMD, Inc., 2001.
- [4] Hong, D. Kirovski, G. Qu, M. Potkonjak and M. Srivastava, "Power optimization of variable voltage core-based systems," *DAC*, pp. 176-181, 1998.
- [5] T. Ishihara and H. Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *Proceedings ISLPED*, pp. 197-202, 1998.
- [6] W. Kim and J. Kim and S. L. Min, "A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis," *DATE*, 2002.
- [7] P. Pillai and K. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *ACM Symposium on Operating Systems and Principles*, 2001.
- [8] A. Sinha and A. Chandrakasan, "Dynamic Voltage Scheduling Using Adaptive Filtering of Workload Traces", 14th Intl. conference on VLSI Design, Bangalore, India, Jan. 2001.
- [9] Dakai Zhu, Rami Melhem, and Bruce R. Childers, "Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems," *IEEE Trans. On Parallel and Distributed Systems*, Vol. 14, No.7, July 2003.
- [10] F. Gruian, "Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors," in the *Proceedings of ISLPED*, pp. 46-51, August, 2001.
- [11] Y. Zhang , X. Hu and D.Z Chen, "Task Scheduling and Voltage Selection for Energy Minimization," *DAC*, pp 183-188, 2002.
- [12] S. Saewong and R. Rajkumar, "Practical voltage-scaling for fixed-priority rt-systems," In *Proceedings of the 9th RTAS*, pp. 106-114, 2003.
- [13] Y. L. A. K. Mok, "An integrated approach for applying dynamic voltage scaling to hard real-time systems," in *Proceedings of the 9th RTAS*, pp. 116-123, 2003.
- [14] Y.-T. S. Li, S. Malik, and A. Wolfe, "Performance estimation of embedded software with instruction cache modeling," *Design Automation Electron. Syst.*, vol. 4, no. 3, 1999.
- [15] F. Gruian and K. Kuchcinski, "Uncertainty-Based Scheduling: Energy-Efficient Ordering for Tasks with Variable Execution Time," *ISLPED*, pp. 465-468, 2003.
- [16] Bren Mochocki, Xiaobo Sharon Hu and Gang Quan, "A realistic variable voltage scheduling model for real-time applications," *ICCAD*, pp. 726-731, 2002.
- [17] D. Hochbaum and J. Shanthikumar, "Convex separable optimization is not much harder than linear optimization," *Journal of ACM*, vol. 37, No. 4, pp. 843-862, 1990.
- [18] Kim, N., Ryu, M., Hong, S., Saksena, M., Choi, C.-H., and Shin, H, " Visual assessment of a real-time system design: a case study on a CNC controller," *The 17th IEEE Real-Time Systems Symposium*, pp. 300-310, 1996.

# Design of a Power/Performance Efficient Single-Loop Sigma-Delta Modulator for Wireless Receivers

Ana Rusu<sup>1,2</sup>, Alexei Borodenkov<sup>1</sup>, Mohammed Ismail<sup>1,3</sup>, and Hannu Tenhunen<sup>1</sup>

<sup>1</sup> Royal Institute of Technology (KTH) Stockholm, IT-University/ IMIT/ LECS/ESDlab  
Isafjordsgatan 39, SE-16440 Kista, Sweden

{ana, ismail, hannu}@imit.kth.se, boa@kth.se

<sup>2</sup> Technical University of Cluj-Napoca, G. Baritiu str, 400027 Cluj-Napoca, Romania

<sup>3</sup> Ohio State University, 2015 Neil Avenue, Columbus, OH 43210 USA

**Abstract.** In order to design high performance sigma-delta A/D converters, it is essential to estimate the Figure-Of-Merit in the design process. This paper describes the design of a power/performance efficient single-loop multibit sigma-delta modulator for wireless applications. Power dissipation is minimized by optimizing the architecture and by a careful design of analog circuitry. A 3<sup>rd</sup> order 4-bit  $\Sigma$ - $\Delta$  modulator with feedforward path is designed in 0.18 $\mu$ m CMOS process operating from 1.8V supply voltage. The modulator dissipates 8.6 mW and achieves a dynamic range of 84/95 dB over a bandwidth of 2000/100 kHz.

## 1 Introduction

Transceivers for future digital communications applications need to be portable, battery-powered, wireless and multistandard. The performance of a multi-standard receiver is directly influenced by the performance of the A/D converter. This leads to more and more demanding specifications for the required A/D converters. Sigma-delta modulators have been receiving much attention in mixed-signal ICs. The intensive ongoing research on  $\Sigma$ - $\Delta$  modulators shows the potential of  $\Sigma$ - $\Delta$  converters as the most promising candidate for high-speed, high-resolution and low-power mixed-signal interfaces.

This paper presents a systematic design strategy of a single-loop multibit  $\Sigma$ - $\Delta$  modulator for a zero-IF/low-IF receiver. The strategy is based on the Figure-Of-Merit, which takes into account the power consumption, the dynamic range and the signal bandwidth to find the most power/performance efficient  $\Sigma$ - $\Delta$  modulator architecture with respect to these design parameters. A behavioral design approach based on Matlab/Simulink environment has been used to choose the appropriate architecture according to the WCDMA receiver specifications. The paper is organized as follows: Section 1 is the introduction. Section 2 addresses the  $\Sigma$ - $\Delta$  modulator architecture selection. Section 3 discusses the proposed  $\Sigma$ - $\Delta$  modulator and the circuit nonidealities effects. The circuit design considerations and the simulation results are presented in Section 4. Finally, in Section 5, conclusions are given.

## 2 Architecture Selection

This section explores tradeoffs among the wide variety of architectures that can be used to implement a  $\Sigma\text{-}\Delta$  modulator for wireless communications. The focus is on selecting the appropriate architecture given the wireless system specifications. It is necessary to estimate the modulator performance at behavioral level before starting the circuit-level design. Two main issues should be concerned: which is the best architecture to fulfill application requirements and what are the requirements of modulator's building blocks for a given architecture.

In order to achieve an optimum trade-off between bandwidth, resolution and power for  $\Sigma\text{-}\Delta$  A/D converters the design performance requirements are used in conjunction with technology-related constraints, like sampling frequency. A best-case estimation of this trade-off can be derived from the Figure-Of-Merit, [1]:

$$\text{FOM} = \frac{P_d}{\text{DR} \cdot (2\text{BW})} \quad (1)$$

where DR is the dynamic range, BW is the signal bandwidth and  $P_d$  is the power dissipation. To achieve a given dynamic range

$$\text{DR} = \frac{3}{2} \cdot \left( \frac{2L+1}{\pi^{2L}} \right) \cdot (\text{OSR})^{2L+1} \cdot (2^B - 1)^2 \quad (2)$$

the designer has three degrees of freedom: the oversampling ratio (OSR), the order of noise shaping (L) and the internal quantizer resolution (B). In order to maximize the signal bandwidth of a  $\Sigma\text{-}\Delta$  A/D converter, it is necessary to increase the sampling frequency as well as to reduce the oversampling ratio (OSR). Since the maximum sampling frequency of CMOS SC circuits is limited by the speed of the technology, it is necessary to choose a low oversampling ratio. However, to maintain performance at a reduced OSR, it is necessary to reduce in-band quantization noise power, [2].

Consider implementing a  $\Sigma\text{-}\Delta$  modulator for WCDMA standard, which should be reconfigured for GSM standard. The WCDMA standard has the highest bandwidth (1.92 MHz), which implies high bandwidth circuits, while GSM standard requires the highest dynamic range (80-90 dB) and determines the power consumption. The combination of these contradictions is used to select the modulator architecture for both standards. Eq. (2) suggests that many possible combinations can meet the specifications, but we need to achieve high performance and to minimize FOM. A low FOM can be achieved through a proper system design which requires a suited architecture and a proper choice of the implementation method, optimizing the architecture parameters, scaling properly the overall modulator, and optimizing for speed and power all building blocks.

In order to achieve a wideband and high resolution A/D conversion, two different approaches have been used for wireless applications. The high-order single-loop multibit  $\Sigma\text{-}\Delta$  modulator is a very attractive approach, [3]-[6]. By using an aggressive high-order noise transfer function combined with multibit feedback it is possible to achieve a high SNR at a low OSR. Since the internal signal swing is reduced with the increase in number of feedback signal bits, the multibit  $\Sigma\text{-}\Delta$  modulator requires a lower slew-rate and thus less power for analogue circuits than the single-bit implementation.

Also, the increase in signal-to-noise ratio (SNR) due to the use of multibit internal DAC can be used in order to reduce the oversampling ratio and hence reduce power dissipation of the analogue part. Besides these improvements, the single-loop multibit  $\Sigma$ - $\Delta$  modulators offer an improved stability. This allows a more aggressive noise shaping which results in an additional accuracy improvement. The main drawback of this approach is that high linearity of the internal D/A converter in the feedback path of the modulator is required. Therefore, the overall sigma-delta converter linearity and resolution are limited by the precision of the multibit DAC. Many dynamic element-matching (DEM) techniques have been proposed to improve the accuracy of the internal D/A converter, [2], [5], [7]. Another approach to realize a wideband and high dynamic range A/D converter is to replace the high-order single-loop modulator with a cascade of sigma-delta modulators, [8], and [9]. Cascaded sigma-delta structures realize high-order noise shaping by cascading sigma-delta stages of second-order or first-order to avoid instability. Reducing the quantizer's resolution to 1 bit may eliminate the dependence on feedback D/A converter linearity. One way to achieve further reduction of quantization noise is to use a multibit quantizer in the final stage. The main drawback of the cascaded approach is the sensitivity to analog-digital mismatch.

The implementation is an important issue in  $\Sigma\Delta$  modulators design. Probably, the most robust implementation of a sigma-delta A/D converter is using the switched-capacitor technique. Their robustness and inherent linearity combined with implementation efficiency are the reasons why switched-capacitor technique has been chosen in our design. In this way we take the advantages of precise loop coefficients and integrator gains by using sampling and feedback capacitors ratio, higher tolerance to clock jitter and opamp settling errors. Using high enough slew-rate and gain-bandwidth product for opamps allows operating at a necessary sampling frequency and oversampling ratio.

The Figure-Of-Merit estimations have been done as in [1] to guaranty a dynamic range of 70 dB over a signal bandwidth of 1.92MHz, at the lowest power dissipation. We used the 0.18 $\mu$ m CMOS technology, a supply voltage of 1.8 V and we assumed that the opamp is a folded-cascode OTA with a total current of  $4I_b$ . Considering the nonidealities of practical SC implementation and the power dissipation-performance tradeoff, the best FOM has been obtained with a third-order single-loop 4-bit modulator with an OSR of 16. To reach the required specifications, a 3<sup>rd</sup> order single-loop 4-bit modulator with feedforward signal path was chosen, [10].

### 3 Power/Performance Efficient Sigma-Delta Modulator

#### 3.1 System Level Considerations

A 3<sup>rd</sup> order single-loop 4-bit  $\Sigma$ - $\Delta$  modulator architecture which has a large input range and low integrator output swings is used in our design, [10]. The proposed 3<sup>rd</sup> order single-loop multibit  $\Sigma$ - $\Delta$  modulator with feedforward signal path is shown in Fig 1.a. Fig. 1.b shown the traditional 3<sup>rd</sup> order single-loop multibit  $\Sigma$ - $\Delta$  modulator. By adding a feedforward signal path in a single-loop multibit  $\Sigma$ - $\Delta$  modulator, [11], [12] the input signal will largely be cancelled at the integrator's outputs. This results in a unity signal transfer function over the total frequency range, without changing the noise trans-

fer function. The modulator is designed with a 3rd-order loop filter and symmetric 17-level quantization to avoid the modulator mismatch. Alternating delayed and no delayed integrators are used to reduce peaking of the noise transfer function, thus to improve the stability of the loop.

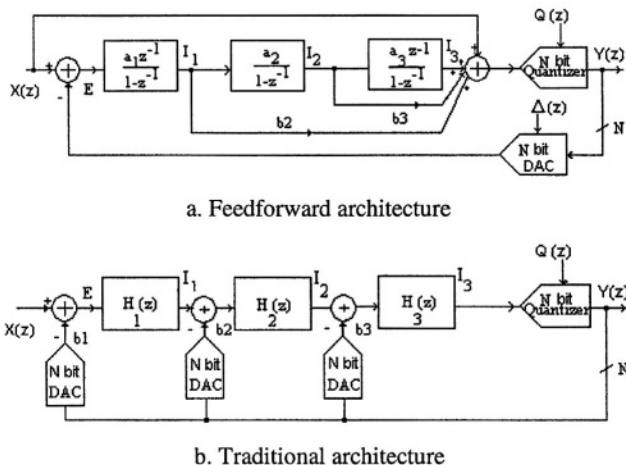


Fig. 1. 3<sup>rd</sup> order multi-bit sigma-delta modulators

By applying linear analysis to the architecture shown in Fig. 1.a, it can be seen that the modulator output is given by

$$Y(z) = STF(z) \cdot X(z) + NTF(z) \cdot Q(z) \tag{3}$$

The signal transfer function is a unity-gain,  $STF(z)=1$  and the noise transfer function,  $NTF(z)$  is

$$NTF(z) = \frac{(1-z^{-1})^3}{1+(a_1 b_3 - 3) \cdot z^{-1} + [3 - 2a_1 b_3 + a_1 a_3 (a_2 + b_2)] z^{-2} + (a_1 b_3 - a_1 a_3 b_2 - 1) \cdot z^{-3}} \tag{4}$$

where  $a_i$  are the integrator gain coefficients and  $b_i$  are the feedforward coefficients. The error signal  $E(z)$  is given by

$$E(z) = [1 - STF(z)]X(z) - NTF(z) \cdot Q(z) = -NTF(z) \cdot Q(z) \tag{5}$$

Therefore, the integrators will process only quantization noise and the modulator sensitivity to nonlinear dc gain, finite bandwidth and slew rate of the opamps is significantly reduced. The integrator output swings are smaller than that of the traditional one. The behavioral simulations show that the first integrator maximum output voltage swing is reduced by 30% compared with the traditional 3<sup>rd</sup> order multibit sigma-delta modulator. The integrator output swings are shown in Fig 2.

The noise transfer function is designed with a 3<sup>rd</sup> order high-pass filter by choosing the following coefficient values:  $a_1=1, a_2=1, a_3=1, b_2=2, b_3=3$ . However, this is not an optimal coefficient selection and noise transfer function can be further improved without loop stability degradation. For an optimal coefficients selection CLANS [13]



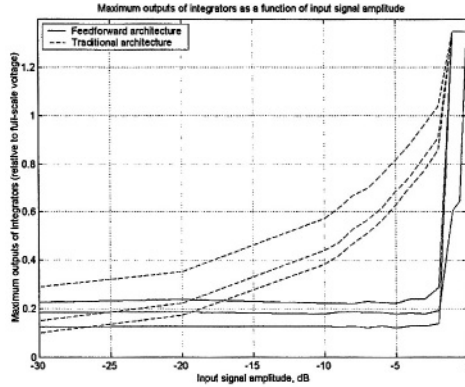


Fig. 2. The integrator outputs as a function of input signal amplitude

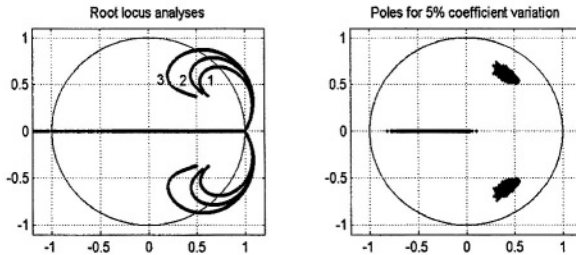


Fig. 3. Root locus analyses and poles location for coefficients variation: CLANS (1)  $a_1=1, a_2=0.6, a_3=1, b_1=1, b_2=1.3, b_3=2.1$ ; Behavioral simulations (2)  $a_1=1, a_2=0.9, a_3=1, b_1=1, b_2=1.3, b_3=2.5$ ; (3)  $a_1=1, a_2=1.0, a_3=1, b_1=1, b_2=1.6, b_3=2.6$ .

and several behavioral simulations have been used. The root locus analysis [14] has been used for stability analysis. Fig 3 shows the root locus with selected noise transfer function and the poles location variation when all modulator coefficients are varied within 5% range. Maximum pole distance is 0.82, which guarantees that modulator will not become unstable. The selected coefficients provide a stable modulator with a high overload level and high resolution. The final selected coefficients to generate the peak signal to noise and distortion ratio (SNDR) and to guarantee the stability are:  $a_1=1, a_2=0.9, a_3=1, b_2=1.3, b_3=2.5$ . An improved signal-to-noise ratio can be achieved by adding a local feedback loop around the last two integrators, which it requires a little more analog circuitry, [5]. The behavioral simulations show that the feedforward architecture can, in terms of stability, tolerate a higher variation in the modulator coefficients than the traditional one.

As in all multibit sigma-delta modulators the main drawback is the requirement of high internal D/A converter accuracy. To improve the accuracy of the internal D/A converter the Pseudo-Data-Weighted-Averaging (P-DWA) technique, proposed in [5] has been used in our design. The P-DWA technique eliminates the tonal behavior with insignificant SNDR degradation and additional circuit complexity.

### 3.2 Analysis of Circuit Nonidealities

The behavioral design approach has been used to investigate the overall circuit nonidealities effects, to optimize the system parameters and to establish the specifications for the analog cells. Simulation has been performed using Matlab/Simulink.

The first integrator is the critical block because its nonidealities affect the overall performance of  $\Sigma\Delta$  modulator. Several nonidealities of the SC integrator have been included in the behavioral model: finite and nonlinear opamp DC gain, slew-rate and gain-bandwidth limitations, capacitor mismatch, opamp noise and thermal noise. The behavioral models have been implemented in Matlab/Simulink similar to [15], [16].

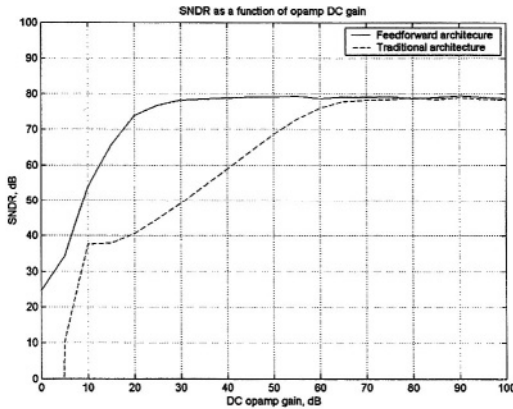


Fig. 4. Signal-to-Noise+Distortion Ratio vs. opamp dc gain

The SC integrators with a finite opamp gain have been used in our behavioral model to estimate the required opamp dc gain. Fig 4 shows the SNDR degradation due to finite opamp dc gain. Simulation results show that the proposed modulator can tolerate a relaxed opamp dc gain in the first integrator, without performance degradation. The minimum required opamp gain to avoid the performance degradation is 40dB. However, in order to provide some margin for degradation due to other nonidealities, a minimum opamp gain of 60dB is selected for our design. The effect of nonlinear dc transfer function of opamp is also considered in our behavioral simulations. In the behavioral model we consider that the opamp presents an open-loop gain whose dependency on the output voltage can be approximated by a polynomial function. Fig 5 shows the output spectrum of the traditional and feedforward architectures in the presence of opamp nonlinearity. As we expect, the power spectrum of the feedforward architecture shows that there is no visible harmonic distortion even for severe dc gain nonlinearity (7%). In the traditional architecture, the harmonic tones can obviously be seen at -83dB, severely degrading the SNDR of the modulator.

The nonlinearity error generated by the unit elements mismatch in the multibit D/A converter appears as an extra noise  $\Delta(z)$  at the D/A converter output. In the behavioral model of the multibit D/A converter we assumed that the mismatch error of the unit-elements has Gaussian distribution with standard deviation  $\sigma$ . For the behavioral simulation results in Fig. 5 the pseudo-DWA algorithm has been used in the feedback

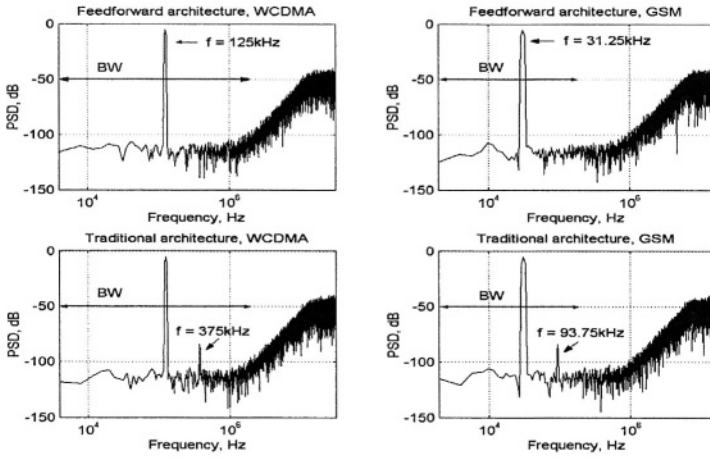


Fig. 5. Power spectral densities in the presence of opamp nonlinearity

D/A converter. For opamp slew-rate and gain-bandwidth limitations we used the modeling approach interpreting these effects as a nonlinear gain, [16]. Using the behavioral simulations we have estimated the opamp requirements for the specified dynamic range. Simulation results show that the opamp bandwidth needs to be at least 180 MHz and the slew rate at least 70V/us. The switches thermal noise and the opamps noise have been included in the behavioral model by using a noisy integrator model as in [15]. The clock jitter has also been considered in the behavioral model.

The behavioral design approach shows that the proposed architecture is suitable for WCDMA receiver application with input bandwidth of 2 MHz and dynamic range of at least 70 dB. In the next section, we will mainly focus on the building blocks design and power dissipation reduction.

## 4 Circuit Design Considerations

### 4.1 Design of the Building Blocks

Power optimization is a task for every design level: architecture, building block and transistor circuit. For the SC  $\Sigma$ - $\Delta$  modulator, the key design aspects for low power at circuit level are to dimension the capacitors and the OTA's transistors.

The proposed modulator is implemented with fully differential SC circuits in 0.18 $\mu$ m CMOS technology, operating from 1.8V supply voltage. The block diagram of the fully differential  $\Sigma$ - $\Delta$  modulator is shown in Fig 6. The design of the building blocks has been done based on the behavioral simulation results. The sampling capacitors of the first integrator have been determined by the noise requirements due to the  $kT/C$  noise of the switches. Bottom-plate sampling and delayed-switch drive techniques have been used in the integrator to minimize signal-dependent charge-injection. All the timing phases are generated by the nonoverlapped clock drive block and all switches are implemented as transmission gates. The integrator specifications

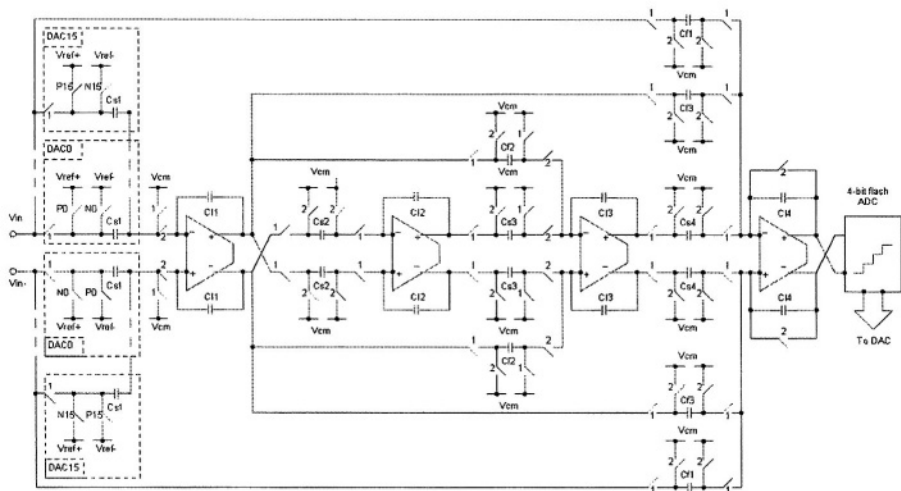


Fig. 6. The SC 3<sup>rd</sup> order 4 bit sigma-delta modulator with feedforward signal path

determined in previous section have been used to select an appropriate circuit topology for the OTAs. The goal is to select a topology, which can meet the integrator performance requirements at minimum power dissipation. The summation junction is implemented with a SC unity gain amplifier. To optimize the OTA design, the fully differential folded cascode OTA with a SC common-mode feedback has been chosen. The dc gain of the first OTA is 63.7 dB, the unity-gain bandwidth is 354 MHz with a 2pF load, the phase margin is 72° and the slew-rate is 300V/us. The other OTAs have been scaled down to minimize  $kT/C$  noise, power and die area, without SNDR degradation. The static power consumption in the OTAs is 2.6mW for the first integrator and the active summation circuit and 0.84mW for the second and third integrators. The 4-bit quantizer is implemented with a CMOS flash A/D converter. In our design, the comparators are implemented as low-power dynamic latch followed by a simple static SR latch. The feedback D/A converter is implemented in a fully differential switched-capacitor configuration. The sampling capacitors in the first integrator act as unit elements in the D/A converter. The modulator dissipates 8.6mW, when the pseudo-DWA algorithm has not been applied in the feedback D/A converter.

## 4.2 Simulation Results

Simulations for the designed third order 4-bit sigma-delta modulator with feedforward signal path were performed using a sampling frequency of 64 MHz, OSR of 16 for the WCDMA standard and a sampling frequency of 32 MHz, OSR of 160 for the GSM standard.

Fig. 7 presents the SNDR, for both WCDMA and GSM standards. It can be seen that the feedforward architecture has higher peak SNDR and can process higher input amplitude without being overloaded.

The overall performance of the proposed sigma-delta modulator in WCDMA/GSM modes is summarized in Table 1.

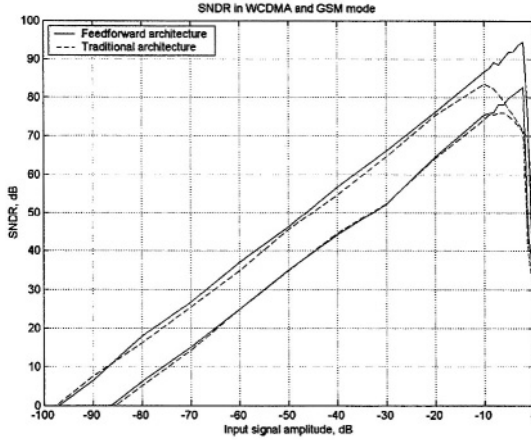


Fig. 7. SNDR versus input signal amplitude

Table 1. Performance Summary of the Modulator

Architecture	3 <sup>rd</sup> order SC feedforward with 4 bit SC DAC	
Process	TSMC 0.18um CMOS process	
Supply voltage	1.8V	
Mode	WCDMA	GSM
Sampling Frequency	64 MHz	32 MHz
Signal Bandwidth	2 MHz	100 kHz
OSR	16	160
Peak SNDR	83dB @ -2dBFS	94dB @ -2dBFS
Dynamic Range	84 dB	95 dB
Peak SFDR	104dB @ -6dBFS	
Power consumption Analog part	8.6 mW	

## 5 Conclusions

A systematic design strategy for a power/performance efficient sigma-delta modulator for wireless receivers is presented. To reduce the power consumption and the nonlinearity effects of the analog circuits, a 3<sup>rd</sup> order 4-bit sigma-delta modulator with feedforward path was employed in our proposed approach. The building blocks have been designed to optimize their power consumption. The simulation results indicate that the proposed architecture has good tolerance to circuit imperfections, dissipates 8.6 mW at 1.8V supply and achieves a peak SNDR of 83/94 dB in WCDMA/GSM mode.

## References

1. Gerfers, F., Min Soh, K., Ortmanns, M., Manoli, Y.: Figure of Merit Based Design Strategy for Low-Power Continuous-Time  $\Sigma\Delta$  Modulators. In: IEEE ISCAS, vol. 4 (2002) 233-236
2. Norsworthy, S. R., Schreier, R. and Temes, G.C.: Delta-Sigma Converters-Theory, Design and Simulation. In: New York: IEEE Press (1997)
3. Burger, T. and Qiuting Hueng: A 13.5mW, 185 MSample/s  $\Sigma\Delta$  modulator for UMTS/GSM dual-standard IF reception. In: IEEE ISSCC Conf. Digest of Tech. Papers (2002) 44 -45
4. Gomez, G. and Haroun, B.: A 1.5 V 2.4/2.9 mW 79/50 dB DR Sigma-Delta modulator for GSM/WCDMA in a 0.13 um digital process. In: IEEE ISSCC Conference Digest of Technical Papers (2002) 242-244
5. Hamoui, A.A., Martin, K.: High-Order Multibit Modulators and Pseudo Data-Weighted-Averaging in Low-Oversampling  $\Sigma\Delta$  ADCs for Broad-Band Applications. In: IEEE Trans. Circuits Syst. I:Regular Papers, vol. 51, no. 1 (2004) 72-85
6. Miller, M. R. and Perie C. S.: A Multi-Bit Sigma-Delta ADC for Multi-Mode Receivers. In: IEEE Journal of Solid-State Circuits, Vol. 38, no. 3 (2003) 475 -482
7. Baird, R.T. and Fiez, T.S.: Linearity enhancement of multibit  $\Sigma\Delta$  A/D and D/A converters using data weighted averaging. In: IEEE Trans. Cir. Syst. II, vol. 42, no. 12 (1995) 753-762
8. Dezzani, A. and Andre, E.: A dual-mode WCDMA/GPRS Sigma-Delta Modulator. In: IEEE ISSCC Conference Digest of Technical Papers (2003) 58-59
9. Oliyai, O., Clement, P., and Gorisse, P.: A 5-mW  $\Sigma\Delta$  Modulator with 84 dB Dynamic Range for GSM/EDGE. In: IEEE Journal of Solid-State Circuits, vol. 37, no. 1 (2002) 1-10
10. Rusu, A. and Tenhunen, H.: A third-order multibit  $\Sigma\Delta$  modulator with feedforward signal path. In: IEEE NEWCAS (2003) 145-148
11. Silva, J., Moon, U., Steensgaard, J. and Temes, G. C.: Wideband low-distortion delta-sigma ADC topology. In: IEE Electronics Letters, vol.37, no. 12 (2001) 737-738
12. Vink J. and Rens J. van: A CMOS Multi-bit Sigma-Delta Modulator for Video Applications. In: IEEE ESSCIRC (1998)
13. Kenney, J.G. and Carley, L.R.: CLANS: A high-level synthesis tool for high resolution data converters. In: IEEE ICCAD-88. Digest of Technical Papers (1988) 496 - 499
14. Baird, R.T. and Fiez, T.S.: Stability Analysis of High-Order  $\Delta-\Sigma$  Modulation for ADC's. In: IEEE-Transactions on Circuits and Systems-II, vol. 41 (1994) 59-62
15. Malcovati, P. et al: Behavioral modeling of switched-capacitor sigma-delta modulators. In: IEEE Trans. Circuits Syst. II, vol.50, no. 3 (2003) 352-364
16. Medeiro, F., Perez-Verdu, P., Rodriguez-Vazquez, A., Huertas, J. L.: Modeling OpAmp-Induced Harmonic Distortion for SC Sigma-Delta Modulator Design. In: IEEE ISCAS, vol. 5 (1994) 445-448.

# Power Aware Dividers in FPGA

Gustavo Sutter<sup>1</sup>, Jean-Pierre Deschamps<sup>2</sup>, Gery Bioul<sup>3</sup>, and Eduardo Boemo<sup>1</sup>

<sup>1</sup> School of Engineering, Universidad Autónoma de Madrid, Spain  
{gustavo.sutter, eduardo.boemo}@uam.es

<sup>2</sup>Dept. Eng. Electrònica, Elèctrica i Automàtica, Universitat Rovira i Virgili,  
Tarragona, Spain

<sup>3</sup> Universidad Nacional del Centro, Tandil, Argentina,  
gbioul@exa.unicen.edu.ar

**Abstract.** This paper surveys different implementations of dividers on FPGA technology. A special attention is paid on ATP (area-time-power) trade-offs between restoring, non-restoring, and SRT dividers algorithms for different operand widths, remainder representations, and radices. Main results show that SRT radix-2 present the best ATP figure. In combinational implementation, an important power improvement, - up to 51% - with respect to traditional non-restoring implementations is obtained. Moreover, up to 93% power improvement can be achieved if pipelining is implemented. Finally, the sequential implementation is another important way to reduce the consumption in more than 89 %.

## 1 Introduction

A widely implemented class of division algorithm is based on digit recurrence. The most common implementation in modern microprocessors is *SRT* (Sweeney, Robertson, and Tocher) division. Digit recurrence, specifically SRT, and other division algorithm surveys can be found in [1-5]. This paper is focused on floating-point dividers, namely with operands in range [1,2).

Many implementations of SRT dividers on FPGA were recently presented. In [6], dedicated Virtex II multipliers are used to implement radix 2, 4, and 8 SRT dividers. Paper [7] presents a minimally redundant radix-8 SRT division scheme, and previous results of a radix-4 SRT are pointed out. In [8], radix 2, 4, and 8 SRT division schemes are iteratively implemented as fully combinational and pipelined circuits. In contrast this paper is based on the implementation reported in [9], where low-level component instantiations in parameterized VHDL code are used in order to keep control over implementation details.

Power consumption, has not been taken into account in these previous works. Thus, this paper tries to contribute to this research area by studying - and optimizing- the power dimension. Section 2 presents the division algorithms; and section 3 briefly discusses the details of Virtex implementation. Finally, section 4 presents some experimental results.

## 2 Algorithms

Given two non-negative real numbers  $X$  (the dividend) and  $D$  (the divisor), the quotient  $q$  and the remainder  $r$  are non-negative real numbers defined by the following expression:  $X = q.D + r$  with  $r < D.ulp$ , where  $ulp$  is the unit in the least significant position. If  $X < D$  and  $D$  are the (unsigned) significant of two IEEE-754 floating-point numbers, then they belong to the range  $[1,2)$ , and  $q$  lies in the range  $[0.5, 1)$ . This result can be normalized by shifting the quotient by one bit to the left, and adjusting the exponent accordingly.

Division generally does not provide finite length result. The accuracy must be defined beforehand by setting the allowed maximum length of the result ( $p$ ). The number of algorithmic cycles will therefore depend upon the aimed accuracy, not upon the operand length ( $n$ ).

**Restoring and non-restoring algorithm:** To divide two integers, the most well known procedures are restoring and non-restoring digit-recurrence algorithms [3],[4]. The corresponding FPGA implementations are straightforward, and the area/time figure is always better for non-restoring. Figure 1 depicts restoring and non-restoring division algorithm. In the latter one, a correction step must be added in order to modify the last remainder whenever negative.

<i>Restoring division algorithm</i>		<i>Non-restoring division algorithm</i>	
$r(0) := X;$ <i>for i in 1 .. p loop</i> rest_step( $r(i-1)$ , $D, q(i), r(i)$ ); <i>end loop;</i>	rest_step ( $a, b, q, r$ ) $z := 2*a - b;$ <i>if z &lt; 0 then</i> $q := 0; r := 2*a;$ <i>else</i> $q := 1; r := z;$ <i>end if;</i>	$r(0) := X;$ <i>for i in 0 .. p-1 loop</i> nonr_step( $r(i-1)$ , $D, q(i), r(i)$ ); <i>end loop;</i> $q(p) := 1; q(0) := 1 - q(0);$	nonr_step ( $a, b, q, r$ ) <i>if a &lt; 0 then</i> $q := 0; r := 2*a + b;$ <i>else</i> $q := 1; r := 2*a - b;$ <i>end if;</i>

Fig. 1. Restoring and non-restoring division algorithms

**SRT division:** As others digit-recurrence algorithms, SRT generates a fixed number of quotient bits at every iteration. The algorithm can be implemented with the standard radix- $r$  ( $r = 2^k$ ) SRT iteration architecture presented at figure 2.a. The  $n$ -bit integer division requires  $t = n/k$  iterations. An additional step is required in order to convert the signed-digit quotient representation into a standard radix-2 notation. The division  $x/d$  produces  $k$  bits of the quotient  $q$  per iteration.

The quotient digit  $q_j$  is represented using a radix- $r$  notation (radix complement or sign-magnitude). The first remainder  $w_0$  is initialized to  $X$ . At iteration  $j$ , the residual  $w_j$  is multiplied by the radix  $r$  (shifted by  $k$  bits on the left, producing  $r.w_j$ ). Based on a few most significant bits of  $r.w_j$  and  $d$  ( $n_r$  and  $n_d$  bits respectively), the next quotient digit  $q_{j+1}$  can be inferred using a quotient digit selection table ( $Qsel$ ). Finally, the product  $q_{j+1} \times d$  is subtracted to  $r.w_j$  to form the next residual  $w_{j+1}$ .

Figure 2.b exhibits the general architecture; the last block *cond\_adder* is only necessary if a positive remainder has to be calculated (not essential in floating-point implementations). For the hardware implementation of an SRT divider, some important parameters have to be traded-off:

**Radix  $r = 2^k$ :** For large values of  $k$ , the iteration number  $t$  decreases but each step is more complex (larger  $Qsel$  tables, complex products  $q_{j+1} \times d$ ).



**Residual representation  $w_j$ :** Traditionally in ASIC implementations, a redundant number system such as carry-save is used for  $w_j$  to accelerate the operation  $q_{j+1} \times d - r.w_j$ . It is not necessary in current FPGAs: The dedicated carry logic circuitry makes ripple-carry faster than carry-save additions or subtractions for small bit widths.

**Quotient representation:** To speed up the subtractive division, redundant digit sets of the form of  $\{-\alpha, -\alpha+1, \dots, 0, \dots, \alpha-1, \alpha\}$  are used. The radix- $2^\alpha$  quotient is represented by a signed-digit redundant number system. It ensures that the next quotient digit determination is possible only based on a few most significant bits of the remainder and divisor ( $nr$  and  $nd$  respectively). Higher values of  $\alpha$  lead to simpler quotient digit selection (smaller values of  $nr + nd$  for the address of the  $Qsel$  table) but also to more complex products  $q_{j+1} \times d$ .

### 3 FPGA Implementations

Complete description of the algorithms and area-time tradeoffs in Virtex and Virtex 2 implementations are summarized in [9]. The architectures analyzed are: traditional restoring and non-restoring algorithms, then SRT radix 2, 4, 8, and 16 with 2's complement remainder, and finally, a novel implementation of SRT radix-2 with carry-save remainder representation. These division algorithms are implemented in different versions: fully combinational (minimum latency), pipelined with different logic depth (maximum throughput), and finally, sequential implementation with a choice of granularities (minimum area).

#### 3.1 Radix-2 Restoring and Non-restoring

Integer division is traditionally done with restoring or non-restoring algorithms. The adjustment to fractional operands is trivial. The restoring division algorithm, implemented with the algorithm depicted in figure 1, needs  $p$  *rest\_step* cells. Each cell uses an  $(n+1)$ -bit subtractor and an  $n$ -bit 2-1 multiplexer. This leads to  $(n+1)$  slices. Finally, an  $n$ -bit divider with  $p$ -bit accuracy use  $p \cdot (C_{rest\_step}(n)) = p \cdot (n+1) = p \cdot n + p$  slices.

Non-restoring division is more efficient in Virtex FPGA. The *nonr\_step* is implemented with  $p$   $(n+1)$ -bit adder-subtractor. Each cell needs  $n/2+1$  slices. If the final remainder is required, an additional conditional adder ( $n/2$  slices) is necessary to adjust a possibly negative remainder.

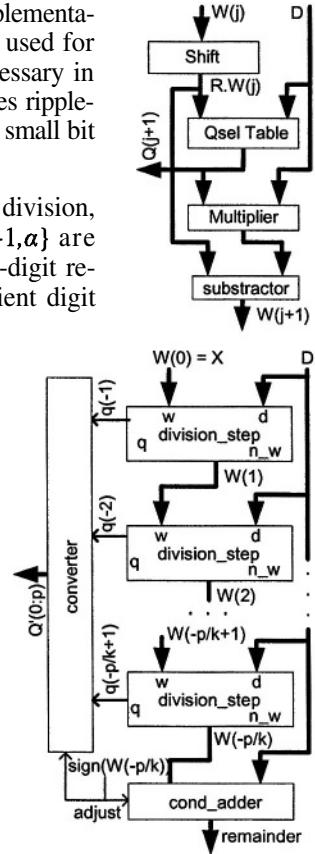
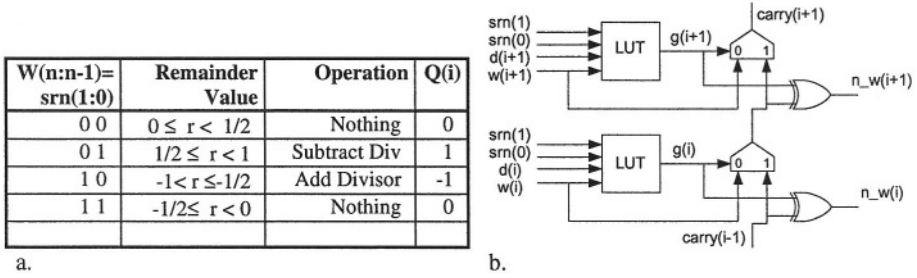


Fig. 2. a. SRT division step  
b. General SRT array

### 3.2 SRTs Algorithms with 2's Complement Remainder

SRT dividers for radix-2, 4, 8 and 16 were implemented. In radix-2 the  $Q_{sel}$  table is trivial (actually, it doesn't exist). The most significant two bits of the remainder are used to settle on the operation to be executed in the next division step (figure 3.a).

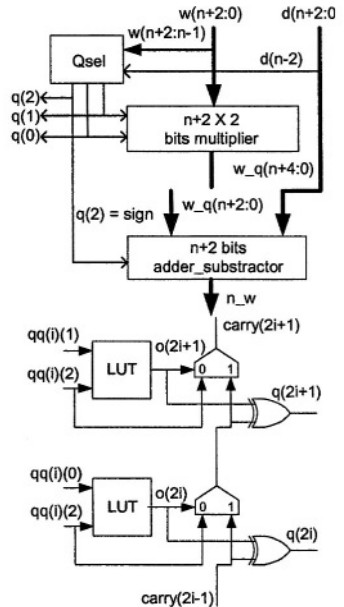


**Fig. 3.** SRT radix-2, with 2's complement remainder. a. Operations to be done in a radix-2 SRT divider cell. b. Configuration of a Virtex slice for  $srt\_step\_r2$  cell.

Therefore,  $Q_{sel}$ , the multiplier (multiplying by -1, 0 or 1 only) and the subtractor can be integrated in a single cell ( $srt\_step\_r2$ ) that uses  $(n+1)/2$  slices only. The slice detail of  $srt\_step\_r2$  is shown at figure 3.b. The  $carry$  (-1) is filled with  $srn(0)$  (the second most significant bit of the previous remainder). The logic function  $g(i)$  implemented in each LUT is  $s_0 \overline{d}w + s_0 d\overline{w} + s_1 \overline{d}w + s_1 d\overline{w}$ .

The total area of the SRT radix-2 corresponds to a  $p$   $srt\_step\_r2$  cell in  $p.(n+1)/2$  slices, a conditional adder ( $n/2$  slices) if positive remainder is necessary, and a converter cell ( $p/2+1$  slices), which is similar to the area of the non-restoring divider. That is,  $p.(n+1)/2 + n/2 + p/2+1 = (pn+n+2p)/2+1$  slices. It can be observed in figure 3.a that the operation carried out by  $srt\_step\_r2$  is, for half of the time, “doing nothing”: this characteristic of algorithm leads to a lower activity and consequently, lower power consumption.

For the higher radices, the  $Q_{sel}$  table, the multiplier, and the adder-subtractor are necessary. Figure 4.a exhibits a SRT radix-4 division step, and table 1 display the parameters for the different radices. An additional *converter* translates signed representation of radix-n digits to 2's complement, as is shown in figure 3.b for radix-4.

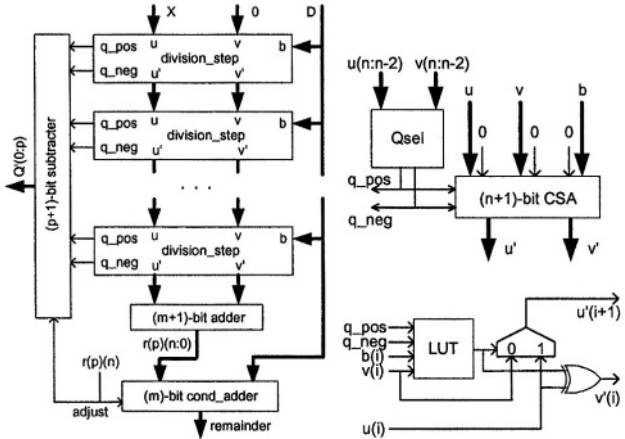


**Fig. 4.** SRT radix-4 divider details. a.  $srt\_step\_r4$  cell. b. Slice detail for SRT radix-4 converter.

### 3.3 Radix-2 SRT with Carry-Save Remainder

The block diagram of SRT radix-2 carry-save remainder is shown at figure 5.a. The division\_step cell is depicted in figure 5.b. The first (leftmost) 3 bits of  $u$  and  $v$  are required to address the  $Q_{sel}$  table where  $q_{pos}$  and  $q_{neg}$  are extracted. It has been empirically established that 3+3 bits from the carry-save representation are good enough to make a proper selection of the quotient digit [13], although 4+4 bits are suggested in [3] and [4].

The  $Q_{sel}$  cell is implemented using 8 LUTs (4 slices) together with F5mux, F6mux multiplexers. The carry-save adder (CSA) of figure 5.b can be implemented within  $(n+1)$  slices using the cell of figure 5.c. Each CSA digit is calculated with one LUT only (together with a *muxcy* and a *xorcy*), but, due to routing limitations, only one CSA digit can be calculated per slice. For that reason, the division cell area is  $C_{cell_{est}} = n + 4$  slices.



**Fig. 5.** SRT radix-2 with remainder in carry-save format. a Array implementation. b *division\_step*. c. Slice contents in implementation of carry-save adder.

**Table 1.** Parameters for SRT radix-2, 4, 8, and 16.  $Q_{sel}$  table size, remainder and divisor bits utilized, quotient range and bits utilized, remainder width, and number of stages.

Radix	Qsel Table					Quotient		Rema inder width	Stages
	Total Bits	Remainder bits	Divisor bits	Sign Calculus	slices	Range	bits		
2	2	$W(n-1)$	-	$W(n)$	-	$\{-1, 0, 1\}$	2	$N+1$	P
4	5	$W(n+1:n-1)$	$d(n-2)$	$W(n+2)$	2	$\{-3, \dots, 0, \dots, 3\}$	3	$N+3$	$P/2$
8	9	$W(n+2:n-2)$	$d(n-2:n-4)$	$W(n+3)$	17	$\{-7, \dots, 0, \dots, 7\}$	4	$N+4$	$P/3$
16	12	$W(n+4:n-2)$	$d(n-2:n-6)$	$W(n+5)$	141	$\{-15, \dots, 0, \dots, 15\}$	5	$N+5$	$P/4$

## 4 Implementation Results

The circuits are implemented in a Virtex XCV800hq240. They are described in VHDL instantiating low level primitives such as LUTs, *muxcy*, *xorcy* [10] when necessary. Xilinx ISE 6.1 tool [11] and XST [12] for synthesis were utilized. A common pin assignment, the preservation of the hierarchy, speed optimization, and timing

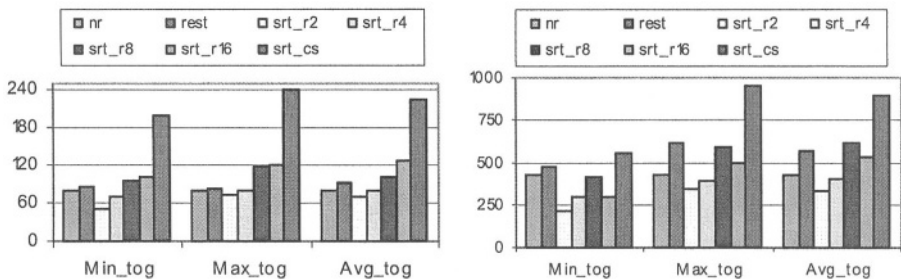
constraints were part of the experimental setup. Area and delay figures were extracted from Xilinx tools. On another hand, power consumption was measured using a Xilinx prototype board AFX PQ240-100. It was separated in static, dynamic (data-path and synchronization), and off-chip power as described in [14]. Chip measurements were done using three different sequences: a) random vectors (*avg\_tog*); b) a sequence with a high transition probability (*max\_tog*) and finally, c) a sequence with low activity (*min\_tog*). The test vectors were entered with a pattern generator [15]. All the circuits were implemented and measured under identical conditions. Off-chip power was determined by measuring the average input current corresponding to the pad ring. This component was almost constant for all divider. At the output, each pad supported the load of the logic analyzer, lower than 3 pF [16].

#### 4.1 Results in Array Implementations

Table 2 shows, for Virtex devices, area in slices, and delay expressed in ns. Up to 24 bits, non-restoring and SRT radix-2 (*srt\_r2*) shows best results in delays. For greater operand sizes, SRT carry-save remainder (*srt\_cs*), SRT radix 16 (*srt\_r16*), and 4 (*srt\_r4*) are the best options. In terms of area, SRT radix-2 and non-restoring (*nr*) are always the best. On the opposite side, restoring (*rest*) and SRT radix-16 consume more area. Best results in area  $\times$  delay figure are provided by SRT radix-2 up to 24 bits, and SRT radix-4 for bigger divider sizes.

**Table 2.** Results for array implementations in Virtex.

N P	nonRest		rest		srt_r2		srt_r4		srt_r8		srt_r16		srt_cs	
	Slices	Delay	Slices	Delay	Slices	Delay	Slices	Delay	Slices	Delay	Slices	Delay	Slices	Delay
40	880	251.7	1640	329.1	861	293.2	940	243.7	1112	277.3	2258	245.7	1779	238.6
32	576	180.6	1056	238.3	561	198.1	624	187.8	804	224.6	1666	191.1	1183	176.2
24	336	118.7	600	158.4	325	125.5	372	125.7	487	154.6	1137	138.4	695	141.4
16	160	68.8	272	91.5	153	69.2	184	82.4	243	83.9	676	81.8	335	87.9



**Fig. 6.** Dynamic power consumption in mW/MHz. a. 16 bits dividers; b. 32 bits dividers.

Figure 6 depicts power consumption expressed in mW/MHz for the different vector sequences within 16 and 32 bits width. For 16 bits, SRT radix-2 presents the best results, improved in an average of 18.5% with respect to non-restoring divider, and

71% with respect to the SRT with carry-save remainder. For 32-bit representation SRT radix-2 and radix-4 are the best options. SRT radix-2 improves by up to 51,2 % the results of non-restoring division, and up to 78 % the results of SRT with carry-save remainder (the fastest one). Finally, area-time-power (ATP) for the different 32-bit dividers is presented in figure 7. The *srt\_r2*, *srt\_r4*, and *nr\_f* offer best ATP figure. The analysis of area  $\times$  delay  $\times$  power relations of merit points to SRT radix-2 as the best choice, followed by SRT radix-4.

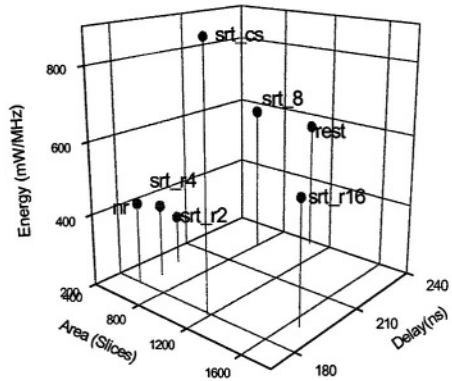


Fig. 7. Area-Time-Power for 32-bit dividers for the *avg\_tog* sequence.

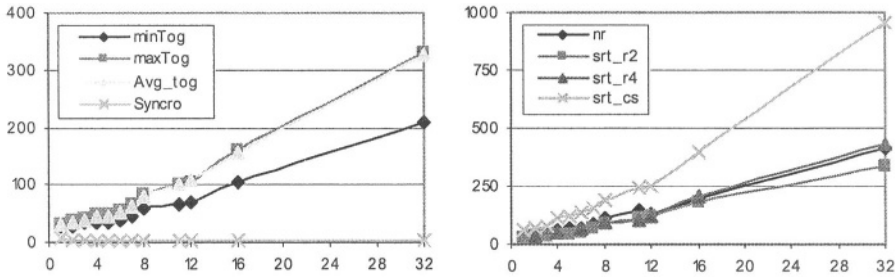
Table 3. Area in slices, register utilization, and maximum frequency, for different logic depths and architectures.

LD	C	Non-Restoring				SRT radix 2 SRL				SRT radix 4				SRT carry save			
		slices	FF	MHz		slices	FF	sr1	MHz	slices	FF	sr1	MHz	slices	FF	sr1	MHz
1	33	1968	2705	101.7	1747	2274	88	111.7	-	-	-	-	3356	3298	90	79.1	
2	16	1256	1328	55.9	1169	1152	52	53.8	1288	1265	39	62.6	2257	1647	52	48.3	
3	11	1066	933	49.5	1012	835	48	49.6	-	-	-	-	1939	1164	48	43.1	
4	8	943	688	34.5	915	641	40	34.3	967	632	30	39.4	1745	871	40	33.7	
5	7	905	617	33.4	888	583	40	32.5	-	-	-	-	1689	780	40	30.7	
6	6	866	538	25.3	858	521	36	26.4	907	508	36	31.3	1629	685	36	28.4	
7	5	822	454	22.5	825	455	28	23.9	-	-	-	-	1566	586	28	24.9	
8	4	779	368	20.0	786	385	32	20.2	835	372	12	21.7	1508	483	16	20.0	
11	3	738	289	15.6	725	321	-	14.7	-	-	-	-	1456	386	-	16.8	
12	3	740	292	14.4	728	327	-	13.6	782	315	-	16.2	1462	392	-	15.4	
16	2	697	208	10.8	675	224	-	10.2	736	222	-	10.6	1355	256	-	11.2	
32	1	656	128	5.6	625	128	-	5.2	752	226	-	5.3	1251	147	-	5.9	

## 4.2 Results for Pipeline Implementations

The effective frequency of each node of a digital circuit can be significantly incremented by the occurrence of glitches. Although glitches do not produce errors in well-designed synchronous systems, they can be responsible for up to 70 % of the circuit activity [17]. The useless consumption associated to glitches can be decreased in two ways: equalizing all circuit paths [18] (almost impossible in FPGA), or inserting intermediate registers or latches to reduce the logic depth [19-23].

In order to reduce the power consumption, pipeline versions of non-restoring algorithm, SRT radix-2, SRT radix-4 and SRT radix 2 carry-save remainder representation were constructed. The circuits utilize SRL (LUTs configured as shift-register) whenever possible. It allows the designer to condense up to 16-bit shift-register (SR) in a single LUT. Table 3 shows area, expressed in slices, register count, and maximum



**Fig. 8.** Power consumption (mW/MHz) vs. logic depth. a. SRT radix-2 pipelining implementation. b. Different pipeline implementations using *avg\_tog* sequence.

bandwidth in MHz, for 32-bits divider implementations and different logic depth (LD), defined as the maximum number of division steps between successive register banks. Pipelining in FPGA shows a low impact in area due to the embedded registers distributed into the slices and the SRL characteristics of LUT.

Figure 8.a presents dynamic power consumption versus logic depth for non-restoring implementations. The three different patterns have similar shape: it decreases practically linearly with the reduction of LD. It stands out the low influence of the synchronization power.

As more pipeline stages are added, fewer glitches are produced, and the power is lowered. This reduction in the activity makes less important the architecture selected. Thus, the different dividers have similar consumption. Figure 8.b shows, for the three different architectures, the dynamic power consumption as a function of the logic depth. A maximally pipelined architecture (LD = 1) saves up to 93 % of the dynamic power consumption with respect to the fully combinational architecture (LD=32). That is, combinational architectures consume more than twelve times more than the fully pipelined version.

**Table 4.** Results for iterative implementations in Virtex.

**4.3 Results for Iterative Implementations**

To reduce area, using iterative architecture is a common technique. The general architecture adds a state machine that controls the data-path. It is constituted by *g* consecutive *division\_step*'s and the corresponding registers to store intermediate values. The circuit calculates at each clock *g.r* bits, and an extra cycle is necessary for remainder calculation. Then, a total of *p/g.r* cycles are used to complete the operation plus an extra cycle if the remainder is needed.

Table 4 shows the results for iterative implementations in Virtex. The amount of bits calculated at a time (G), and the total clock cycles necessary (C), slices and register

	G	C	slices	FF	P(ns)	F(MHz)	L(ns)
non_rest	1	32	113	203	8,9	112,0	285,7
	2	16	124	202	13,7	72,8	219,8
	4	8	155	200	23,8	42,1	190,2
	8	4	219	196	44,9	22,3	179,6
srt_r2	1	32	135	240	8,0	124,6	256,9
	2	16	139	236	13,4	74,5	214,8
	4	8	169	236	24,1	41,5	193,0
srt_r4	8	4	229	236	47,9	20,9	191,7
	2	16	134	221	12,7	78,8	202,9
	4	8	169	221	21,9	45,7	175,2
srt_r16	8	4	240	222	41,2	24,3	164,6
	4	8	336	255	23,0	43,5	183,9
	8	4	603	294	41,9	23,9	167,5
srt_cs	1	32	179	269	11,9	83,8	382,0
	2	16	210	267	17,7	56,6	282,6
	4	8	282	267	29,9	33,4	239,3
	8	4	426	267	52,5	19,0	210,2

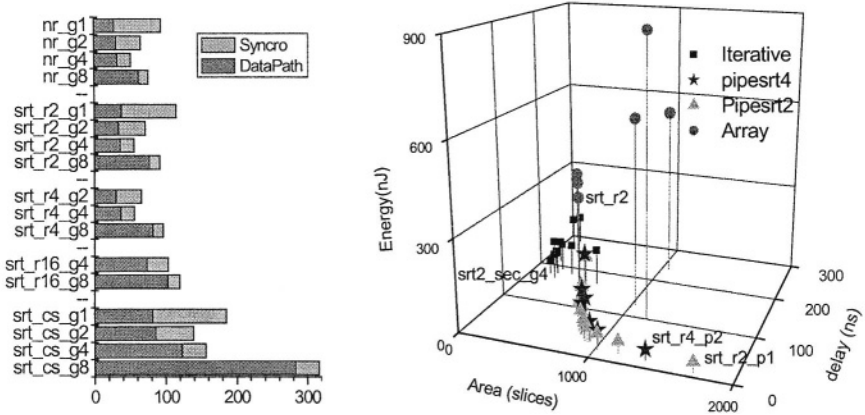
utilization. Finally, minimum period, maximum frequency and latency are presented. As  $G$  grows, the latency decreases at the cost of extra area. Best results in terms of latency are provided by SRT radix-4. Minimum value for area  $\times$  latency figure is obtained for  $G=2$ .

Figure 9.a shows the average energy for an operation in  $n$  Joules for 32-bit width divider. The synchronization and data-path components are also displayed. The synchronization power decreases as  $G$  grows, mainly because of smaller cycles. In the opposite, the data-path consumption grows with  $G$ , mainly because the glitches increases. Optimum  $G$  value seems to be 4, apart from SRT carry-save remainder (*srt\_cs*).

Non-restoring division with  $G=4$  (*nr\_g4*) shows lowest power consumption, while SRT radix-2 with  $G=4$  (*srt\_r2\_g8*) and SRT radix-4 with  $G=4$  (*srt\_r4\_g4*), have similar energy consumption. An important point is that the value of  $G$ , better that a particular algorithms, defines the power figure. In SRT radix-2,  $G=4$  save 51% energy with respect to  $G=1$ . The energy savings with respect to the fully combinational implementations are: 85 % as regards SRT radix-2, 89 % as regards non-restoring division, and 94% as regards the SRT carry-save remainder.

### 4.4 Architectural Comparisons

Figure 9.b shows ATP figure for some 32-bit circuits. The array implementations have the lowest latency, but as the cost of a great area and excessive power dissipation. Pipeline offers the best throughput, with a relatively low increment in area with respect to array implementations and a good power figure, but the initial latency could be prohibitive for some applications. Finally, sequential implementations have the smaller area, a delay less than twice the one of arrays, but have a good power figure.



**Fig. 9.** a. Dynamic power consumption for different sequential divider implementations. b. Area-Time-Power for sequential, array, and pipeline implementations.

## 5 Conclusions

This paper has presented a power analysis for improved architectures and implementations of SRT division on mantissa operations (floating-point numbers).

The circuits were implemented in VHDL, instantiating low-level primitives when necessary. Array implementations, pipelined with different logic depths, and sequential implementations were constructed and the power was measured.

For array implementations SRT radix-2 has the best ATP figure, reducing power consumption up to 51 % with respect to traditional non-restoring division and 93 % with respect to SRT radix-2 with carry-save remainder representation.

Pipeline architectures offer an important way to reduce the power consumption. The measurements show reductions of up to 93 % of dynamic power in a fully pipelined divider with respect to an entirely combinational architecture. Such improvement is obtained with a relative low impact in area.

The sequential implementations use lower area resources, with a relative low impact in delay, but with an important power reduction, -up to 89 %- with respect to fully combinational implementation. An important criterion in sequential implementations is the amount bits calculated at a time (G). G=4 show the best power consumption and ATP figure, while G = 2 has the best Area  $\times$  Latency figure. Non-restoring and SRT radix-2 exhibit the best results.

More researches are needed to explore further alternatives such as larger operand sizes, or ad-hoc disabling architectures to reduce glitch propagations. The dividers features are currently analyzed for Virtex 2. An optimized in ATP fully IEEE compliant floating-point unit is another key research interest.

**Acknowledgement.** This work is supported in part by Project TIC2001-2688-C03-03 of the Spanish Ministry of Education and Science, and in part by Project 07T/0052/2003-3 of the *Consejería de Educación de la Comunidad de Madrid*, Spain

## References

1. S.F. Oberman and M.J. Flynn. "Division algorithms and implementations". *IEEE Transactions on Computers*, 46(8):833–854, August 1997.
2. M.D. Ercegovic and T. Lang. *Division and Square-Root Algorithms: Digit-Recurrence Algorithms and Implementations*. Kluwer Academic, 1994.
3. B. Parhami. *Computer Arithmetic: Algorithms and Hardware Design*. Oxford University Press, 2000.
4. M.Ercegovic and T.Lang. *Digital arithmetic* San Francisco: Morgan Kaufmann, cop. 2004
5. P.Soderquist and M.Leeser. *Area and Performance Tradeoffs in Floating-Point Divide and Square-Root Implementations*, ACM Computing Surveys, Vol. 28, No. 3, September 1996.
6. J-L.Beauchat and A.Tisserand, "Small Multiplier-Based Multiplication and Division Operators", *12th Conference on Field Programmable Logic and Applications*, pp. 513-522. 2002.
7. B.R. Lee and N. Burgess, "Improved Small Multiplier Based Multiplication, Squaring and Division" *11<sup>th</sup> IEEE symposium on Field-Programmable Custom Computing Machines.2003*.
8. X. Wang and B.E. Nelson, "Tradeoffs of Designing Floating Point Division and Square Root on Virtex FPGAs" *11<sup>th</sup> IEEE symposium on FCCM*. 2003.



9. G.Sutter, G.Bioul, and J-P.Deschamps, "Comparative Study of SRT-Dividers in FPGA", Conf.on Field Programmable Logic and Applic. (FPL'04), Antwerp, Belgium, Sept 2004.
10. Xilinx Inc, Libraries Guide for ISE 6.1 available at [www.xilinx.com](http://www.xilinx.com), 2003.
11. Xilinx Inc, Xilinx ISE 6 Software Manuals, available at [www.xilinx.com](http://www.xilinx.com), 2003.
12. Xilinx Inc, XST User Guide 4.0, available at [www.xilinx.com](http://www.xilinx.com), June 2003.
13. G.Sutter, "FPGA implementation and comparison of SRT dividers", UAM, Technical Report, December 2003.
14. E. Todorovich, G. Sutter, N. Acosta, E. Boemo and S. López-Buedo, "End-user low-power alternatives at topological and physical levels. Some examples on FPGAs", Proc. DCIS'2000, Montpellier, France, Nov. 2000.
15. Tektronix inc, TLA7PG2 Pattern Generator Module User Manual. [www.tektronix.com](http://www.tektronix.com).
16. Tektronix inc, TLA 700 Series Logic Analyzer User Manual. [www.tektronix.com](http://www.tektronix.com).
17. A. Shen, A. Gosh, S. Devadas and K. Keutzer, "On average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks", *Proc. ICCAD-92 Conf*, pp.402-407, IEEE Press, 1992.
18. M. Pedram, "Power Minimization in IC Design: Principles and Applications", *ACM Trans. On Design Automation of Electronic Systems* ", vol. 1, n°1, pp.3-56, January 1996.
19. A. Chandrakasan, S. Sheng and R. Brodersen, "Low-Power CMOS Digital Design", *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 4, pp. 473-484. April 1992
20. E. Mussol and J. Cortadella, "Low-Power Array Multiplier with Transition-Retaining Barriers", *Proc. PATMOS'95, Fifth Int. Workshop*, pp. 227-235, Oldenburg, October 1995.
21. J. Leiten, J. van Meerbergen and J. Jess, Analysis and Reduction of Glitches in Synchronous Networks", *Proc. 1995 ED&TC*, pp.1461-1464. New York: IEEE Press, 1995
22. E. Boemo, G. Gonzalez de Rivera, S.Lopez-Buedo and J. Meneses, "Some Notes on Power Management on FPGAs", *LNCS*, No.975, pp.149-157. Berlin: Springer-Verlag 1995.
23. E. Boemo, S. Lopez-Buedo, C. Santos, J. Jauregui and J. Meneses, "Logic Depth and Power Consumption: A Comparative Study Between Standard Cells and FPGAs", *Proc. XIII DCIS Conference (Design of Circuit and Integrated Systems)*, Madrid, Univ. Carlos III: Nov 1998.
24. Sutter G, Todorovich E, Lopez-Buedo S, and Boemo E. "Logic Depth, Power, and Pipeline Granularity: Updated Results on XC4K and Virtex FPGAs" III Workshop on Reconfigurable Computing and Applications JCRA03, Madrid Spain, Sept2003.

# A Dual Low Power and Crosstalk Immune Encoding Scheme for System-on-Chip Buses

Zahid Khan<sup>1</sup>, Tughrul Arslan<sup>1,2</sup>, and Ahmet T. Erdogan<sup>1</sup>

<sup>1</sup> University of Edinburgh, Scotland, UK  
Z.Khan@ed.ac.uk

<sup>2</sup> Institute for System Level Integration livingston, Scotland

**Abstract.** Crosstalk causes logical errors due to data dependent delay degradation as well as energy consumption and is considered the biggest signal integrity challenge for long on-chip buses implemented in Ultra Deep Submicron CMOS technology. Elimination or minimization of crosstalk is crucial to the performance and reliability of SoC designs. This paper presents a novel on-chip bus encoding scheme targeting high performance generic SoC systems. In addition to its efficiency in terms of power, the scheme eliminates three types of crosstalk that cause miller-like transition on two or three adjacent wires simultaneously. The paper describes the technique, its implementation (using the widely adopted AMBA-AHB SoC bus standard) and provides experimental results indicating upto 38% energy saving for systems implemented in 0.18 $\mu\text{m}$  CMOS technology.

## 1 Introduction

The scaling of CMOS technology to ultra deep sub micron has increased the sensitivity of CMOS technology to various noise mechanisms such as crosstalk noise (due to capacitive and inductive coupling), power supply noise, leakage noise etc. Of all these, the crosstalk noise due to capacitive coupling is dominant as it causes delay faults, logical malfunctions and energy consumption on long on-chip buses. The coupled capacitance ( $C_c$ ) between long parallel wires is of magnitude several times larger than the wire-to-substrate capacitance ( $C_s$ ) [1]. In addition to its dependence upon technology as well as structural factors such as wire spacing [2], wire width, wire length [2], wire material, coupling length, driver strength [3], signal transition time etc, the coupled capacitance also depends upon the data dependent transitions and will increase or decrease depending upon the relative switching activity between adjacent bus wires [1]. For the case in which three adjacent wires undergo opposite state transition, the coupled capacitance on the center wire becomes 4 times the coupled capacitance in case only one wire changes state while all others remain silent. This increase in  $C_c$  causes 4 times increase in delay and energy consumption (due to four times increase in crosstalk noise) compared to single wire change [4].

Previous low power coding schemes aimed at reducing the node switching activity for low power [5] and [6]. This is efficient for off-chip buses where node capacitance is several times larger than the coupled capacitance and where impedances are properly adjusted to reduce crosstalk noise. However, for on-chip buses major source of

energy consumption is the inter-wire coupled capacitance and its minimization is necessary for saving energy consumption. Reducing the inter-wire coupling capacitance without eliminating any type of worst case crosstalk (*type-4*, *type-3* and *type-2* discussed in section 2) will result in low power but will not reduce the maximum bound on delay penalty that limits the performance and reliability of high speed on-chip buses. The CBI Scheme in [7] reduces the net coupled switched capacitance but does not eliminate any type of worst case crosstalk. This implies that from delay perspective, the method is not much advantageous over the raw data. The work presented in [8] is well suited for both power efficiency and elimination of all types of worst. However, since the method exploits the probabilistic information of the data stream, it cannot be applied to a data the statistical properties of which can not be known a priori and, therefore, cannot be applied to generic SoC systems. The work in [9] exploits the locality and temporal correlation that exist in address buses for both low power and reducing worst crosstalk coupling. However, the method cannot be applied to data buses which are neither local with regard to data nor temporally correlated. The encoding scheme presented in this paper targets the crosstalk problem from both power and delay perspectives. It transforms the incoming data in such a way as to eliminate worst crosstalk types (*type-4*, *type-3* and *type-2* discussed in section 2). By doing so, the errors associated with crosstalk induced delay and glitches can be eliminated. At the same time, the scheme provides power reduction by minimizing self and eliminating worst coupled switched capacitance. The method is well suited to generic data buses and does not require prior probabilistic information of the input data stream. The remaining sections of the paper are organized such that section 2 provides an expression for energy consumption as function of self and coupled switching activity, section 3 explains the method and its implementation using a generic SoC platform based on AMBA-AHB bus protocol, section 4 and 5 provide results and conclusion.

## 2 Expression for Net Switching Activity

This section presents four types of crosstalk by taking three adjacent wires into consideration. The classification of the crosstalk into types is done to emphasize two aspects of the encoding scheme. The first is elimination of worst crosstalk and second the energy efficiency.

For a 3-bit bus, a *type-1* crosstalk occurs if one of the three wires changes state e.g. a transition from 110 to 111 will cause a *type-1* crosstalk. For *type-1* crosstalk, the coupled capacitance is  $C_r$ . A *type-2* crosstalk occurs if center wire is in opposite state transition with one of its adjacent wires while the other wire undergoes the same state transition as the center wire, e.g. a transition from 001 to 110 will cause a *type-2* crosstalk. For this type of crosstalk, the coupled capacitance will be  $2 \cdot C_r$ . A *type-3* crosstalk occurs if the center wire undergoes opposite state transition with one of the two wires while the other is quiet. A transition from 101 to 110 will cause a *type-3* crosstalk and the coupled capacitance of the center wire in this crosstalk will be  $3 \cdot C_r$ . For the case of *type-4* crosstalk all three wires transition to opposite state with respect to each other and their previous bus state. A transition, for example, from 101 to 010 will cause a *type-4* crosstalk and the coupled capacitance of the center wire rises to  $4 \cdot C_r$ . *Type-4*, *type-3* and *type-2* are classified as the worst crosstalk coupling [4].

The authors in [10] have given approximate energy function for the self and coupled switching activity considering lumped model of the bus. The same lumped model is considered here to formulate energy expression in terms of supply voltage, node capacitance, coupling capacitance, self, type-4, type-3, type-2 and type-1 switching activity for a generic n-bit bus in the time interval  $[0, N]$ . The expression is given in 1.

$$E = E_{0 \rightarrow 1} \cdot \{N_x + \lambda \cdot (4 \cdot N_4 + 3 \cdot N_3 + 2 \cdot N_2 + 1 \cdot N_1)\} \tag{1}$$

Here  $N_x, N_4, N_3, N_2$  and  $N_1$  are the self, type-4, type-3, type-2 and type-1 switching activity in time interval  $[0, N]$ .  $E_{0 \rightarrow 1} = C_L V_{dd}^2$  and  $\lambda = C_i / C_L =$  ratio of coupling capacitance to wire-to-substrate capacitance. For  $0.18\mu\text{m}$  and minimum distance between wires,  $\lambda = 3.2$ [1]. The proof of equation 1 is not presented here. The net switching activity is given by 2:

$$E/E_{0 \rightarrow 1} = N_x + \lambda \cdot (4 \cdot N_4 + 3 \cdot N_3 + 2 \cdot N_2 + 1 \cdot N_1) \tag{2}$$

Equation 2 provides only approximate results for the total energy consumption on the bus for the crosstalk on wire  $i$  due to wire  $i+1$  and  $i-1$  (for  $i > 1$ ) is considered. The assumption is valid as coupling effect varies inversely with the spacing between wires. The results presented in this paper for energy estimation are based on calculating  $N_x, N_4, N_3, N_2, N_1$  and then finding out the net switching activity as given by equation 2. Equation 3 then gives the energy saving:

$$\text{Energy Saving} = (1 - N_c / N_u) \cdot 100 \tag{3}$$

Where  $N_u$  and  $N_c$  are the net switching activity (as given by equation 2) in the unencoded and corresponding encoded data.

### 3 Proposed On-Chip Bus Encoding Scheme

Spatially redundant Limited Weight Code has hamming weight less than the data it represents and is used in off-chip communication for energy efficiency [2] [11]. The concept of the limited weight code can be exploited for the case of on-chip communication to achieve two performance goals. The first is worst crosstalk coupling elimination and the second is the energy efficiency.

For a codeword of width  $n$ , a  $k$ -LWC is a code whose code words have hamming weight  $\leq k$ . For a vector  $m$  of space  $2^m$ ,  $k$ -LWC code must satisfy the following inequality [11].

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \dots + \binom{n}{k} \geq 2^m \tag{4}$$

A semi perfect code uses all  $n$ -tuples with  $weight < k$  and only some of the  $n$ -tuples with  $weight = k$  as code words [11].

Based on the inequality a semi perfect  $m/2$ -LWC codebook (where  $m$  is the width of the raw input data) has been developed in such a manner that if two code words of hamming weight greater than zero are summed using a modulo-2 adder, the resulting

pattern will have no worst crosstalk coupling. The resulting pattern will also reduce self switched capacitance to a considerable extent. A sample 4-to-6-bit spatially redundant semi perfect  $m/2$ -LWC codebook is presented below in Table 1 for reference. The bit pattern in the codebook is chosen such that the code words have only *type-3* and *type-1* crosstalk couplings and when the code to be transmitted on the bus is modulo-2 added with the present bus state, the resulting pattern will have only *type-1* coupling. If at any instant, a sample is taken of the pattern present on the bus, the pattern will be completely different from the codeword. The codeword is absorbed in the bit stream and is unrecognizable as the stream contains information from the previous code words. The codeword is recovered at the receiving end by performing modulo-2 summation of the present and the previous bus states.

**Table 1. Semi-Perfect 2-LWC Codebook**

4-bit data	6-bit code	4-bit data	6-bit code	4-bit data	6-bit code	4-bit data	6-bit code
0000b	000000b	0100b	000101b	1000b	100100b	1100b	100000b
0001b	000001b	0101b	001000b	1001b	010001b	1101b	100001b
0010b	000010b	0110b	001001b	1010b	010010b	1110b	100010b
0011b	000100b	0111b	001010b	1011b	010100b	1111b	010000b

The encoding algorithm consists of the following three steps:

*Step 1:*

$$y^m(n) = x^m(n) \oplus x^m(n-1) \quad (5a)$$

The first step consists of modulo-2 adding the state (binary sequence) already transmitted  $x^m(n-1)$  and the state to be transmitted  $x^m(n)$ .

*Step 2:*

$$L^p(n) = m/2 - LWCCodebook(y^m(n)) \quad (5b)$$

The  $m$ -bit binary pattern  $y^m(n)$  is input to the semi perfect  $m/2$ -LWC codebook that is specifically developed to eliminate worst crosstalk coupling and reduce self switching, the output is the corresponding  $p$ -bit semi-perfect  $m/2$ -LWC codeword.

*Step 3:*

$$D^p(n) = L^p(n) \oplus D^p(n-1) \quad (5c)$$

The codeword  $L^p(n)$  to be transmitted on the bus is modulo-2 added with the previous bus state  $D^p(n-1)$ . The resulting pattern is immune to worst crosstalk coupling. This step is the most crucial and important part of the Encoder architecture as it performs two functions. The first is to eliminate the worst crosstalk coupling and second to reduce self switching activity at the expense of a modest increase in type-1 coupling.

**Decoding** is also a three step process and every step is the exact reverse of its corresponding encoding step.

*Step 1:*

From 5c, modulo-2 adding both sides with  $D^p(n-1)$

$$\begin{aligned} D^p(n) \oplus D^p(n-1) &= L^p(n) \oplus D^p(n-1) \oplus D^p(n-1) \\ \Rightarrow D^p(n) \oplus D^p(n-1) &= L^p(n) \end{aligned} \quad (6a)$$

In this step, present and previous bus states are modulo-2 added to recover the limited weight codeword.

*Step2:*

$$y^m(n) = m - LWCodebook(L^p(n)) \quad (6b)$$

In this step, the limited weight codeword recovered from the first step is used to regenerate the m-bit binary pattern.

*Step 3:*

The third step recovers the original binary sequence.

$$x^m(n) = y^m(n) \oplus x^m(n-1) \quad (6c)$$

Initially the first step of modulo-2 adding the present and previous states of the raw input data (5a) was not included in the encoding algorithm. The method proved to be more power consuming on the data that has repeated same non-zero sequences. The inclusion of the first step is proved below:

Let  $x^n(n)$ ,  $x^n(n+1)$  and  $x^n(n+2)$  are three consecutive raw input data values to be transmitted on the bus. The corresponding code words are  $L^p(n)$ ,  $L^p(n+1)$  and  $L^p(n+2)$  and the bit pattern on the channel is  $D^p(n)$ ,  $D^p(n+1)$  and  $D^p(n+2)$  where

$$D^p(n) = D^p(n-1) \oplus L^p(n) \Rightarrow D^p(n) = L^p(n) \text{ as } D^p(n-1) = 0$$

(initial condition)

$$D^p(n+1) = D^p(n) \oplus L^p(n+1) = L^p(n) \oplus L^p(n+1)$$

$$D^p(n+2) = D^p(n+1) \oplus L^p(n+2) = L^p(n) \oplus L^p(n+1) \oplus L^p(n+2)$$

Therefore, if  $x^n(n) = x^n(n+1) = x^n(n+2)$  then  $L^p(n) = L^p(n+1) = L^p(n+2)$  and  $D^p(n) = L^p(n)$ ,  $D^p(n+1) = 0$  and  $D^p(n+2) = L^p(n)$

The data pattern at the bus should remain constant as the input data pattern remains the same. However, transition increases due to inversion on every second non-zero same data input values. This can be avoided if step (5a) is inserted in encoding the data on the bus.

## 4 Encoder and Decoder Implementation

The proposed low power and crosstalk immune encoder and decoder architecture has been implemented on AMBA-AHB data bus. AMBA-AHB is the abbreviation for Advanced Microcontroller Bus Architecture- Advanced High Performance Bus. AHB is the standard of ARM and is widely used in today's high performance SoC systems. AHB data bus has been divided into different byte lanes so as to enable 8, 16, 32-bit or

higher order transactions. The proposed Codec Architecture (encoder and decoder) has been implemented to comply with all transfer protocols of the AMBA AHB bus. However, the research work presented in this paper is limited to 8, 16 and 32 bit transfer sizes.

For 8-bit transfer of data on AHB bus, the 8-bit lane is partitioned into two 4-bit clusters. The partitioning is done for area efficiency of the Codec Architecture. For 4-bit cluster the memory required for the codebook of table 1 is only  $16 \times 6 = 96$  bits. For two clusters, that would be  $96 \times 2 = 192$  bits. For 8-bit lane that has 256 binary sequences and to encode all  $2^8$  binary sequences will need a codebook that will occupy  $2^8 \times 12 = 3072$  bits of memory. This implies that partitioning provides a saving of 93.75% memory at the expense of one extra shield wire between the two partitions. Therefore,  $m=4$  and  $p=6$  are chosen for the Codec Architecture in figures 1 and 2. The 2-LWC codebook presented in Table 1 is used to encode the clustered data. A shield wire is inserted between the code words of the two clusters. This shield wire will eliminate any possibility of worst crosstalk coupling between code words of the two clusters. The insertion of the shield wire will cause some loss of energy due to *type-1* coupling. An 8-bit bus is therefore extended to 6-bit codeword + shield wire + 6-bit codeword = 13-bits. The Encoder and Decoder cause some delay in signal propagation. However, the delay is well within tolerable limits of the pipelined architecture and does not degrade the performance of the AHB data bus. The area overhead of 32-bit Codec Architecture is calculated to be only 7.012% of the overall architecture of the AMBA-AHB bus.

## 5 Simulation Results

The proposed method was implemented using  $0.18\mu\text{m}$  CMOS technology and energy consumption was measured based on self and coupled switching activity. Applied data streams consist of zero mean uniformly distributed random data (r), and application specific biomedical (b) data with transfer sizes of 8, 16 and 32 bits. The total number of self ( $N_s$ ) and coupled ( $N_c, N_p, N_r, N_i$ ) switching transitions are calculated (all switching activity discussed in the result are calculated at the out data bus from the encoder) for the three types of data using three transfer protocols (8, 16 and 32-bit) of the AMBA-AHB data bus for the case of unencoded, Bus Invert (BI), coupling driven bus invert (CBI) and the proposed low power method. The energy saving is calculated using the relation given in equation 3. Figure 1 provides percentage energy saving for BI, CBI and the proposed encoding scheme while Figure 2 provides percentage presence of the switching activity for a sample 8-bit random (r-8) data set. The BI and CBI are used for comparison as both do not require in advance the probabilistic information of the data. The CBI scheme proved to be more power consuming. The reason for this increase is that CBI depends on the total coupled switched capacitance in a particular data transfer and if an 8-bit data sample has less than 8 total coupled switched transitions in a particular transfer, the CBI coder will not invert the data. The biomedical data is characterized by a high degree of correlation with minimum crosstalk coupling probability per data transfer. Out of the 14642 samples, none of the data samples got inverted by the CBI coder. The de-correlation at the output of the CBI encoder caused increase in total switched capacitance for the biomedical data set. The CBI Scheme is power efficient in the case in which the data has more crosstalk

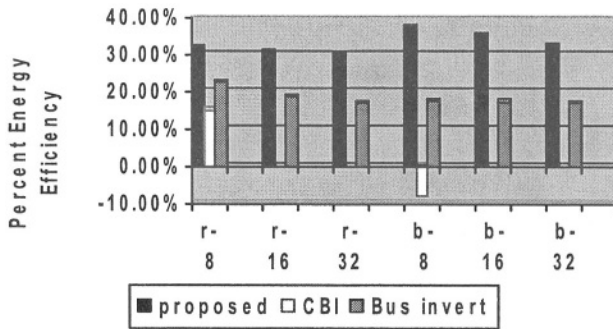


Fig. 1. Percent Energy Efficiency

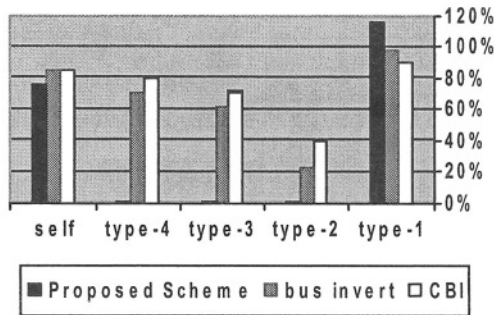


Fig. 2. % switching activity in 8-bit random data set (Percent switching activity presence= switching activity in the encoded data / switching activity in the unencoded data)

coupling so that inversion of the data occurs more frequently. The proposed scheme can also be applied to high capacitance off-chip buses as it reduces the self switching activity from 25% for uniformly distributed random data to 49% for highly correlated data such as those produced in biomedical applications. The power consumption of the internal combinational and sequential elements of the Codec Architecture has been measured using synopsis design power for a sample 8-bit random data set example. The power consumption is found to be only 2.01% of the overall power consumption of the SoC system on which the Codec Architecture has been implemented. This implies that the internal power consumption of the codec is negligible as compared to saving on the on-chip data bus. The method proposed proved to be more power as well as crosstalk noise efficient than BI, and CBI.

## 6 Conclusion

The authors have presented a technique that addresses energy loss and delay problems (due to crosstalk noise) faced by today's tightly coupled on-chip buses implemented in



ultra deep submicron SoC systems. The technique provides energy saving, for 0.18 $\mu\text{m}$  CMOS technology, ranging from 32% for highly decorrelated uniformly distributed random data to 38% for highly correlated application specific data such as those produced in biomedical applications. From delay perspective, the technique eliminates  $N_4$ ,  $N_3$  and  $N_2$  crosstalk completely thereby reducing the bit error probability and improving the reliability and robustness of the on-chip communication to a considerable extent.

## References

- [1] P.P. Sotirsadis, A. Chandrakasan. Bus energy minimization by transition pattern coding (TPC) in deep sub-micron technologies. Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference, 2000 Page(s):322-327
- [2] C. Guardiani, C. Forzan, B. Franzini, D. Pandini. Modeling the Effect of Wire Resistance in Deep Submicron Coupled Interconnects for Accurate Crosstalk Based Net Sorting. Patmos International Workshop on Power and Timing Modeling, Optimization and Simulation. Oct 7-9, 1998
- [3] Payam Heydari, Massoud Pedram. Analysis and Reduction of Capacitive Coupling Noise in High Speed VLSI Circuits IEEE International Conference on Computer Design (ICCD), Austin, Texas, Sept. 2001. pp.104-109
- [4] Chunjie Duan, Anup Tirumala, S.P. Khatri. Analysis and avoidance of cross-talk in on-chip buses. Hot Interconnects 9, 2001. 22-24 Aug. 2001. Page(s): 133-138
- [5] M.R. Stan, W.P. Burleson. Bus-invert coding for low-power I/O. VLSI Systems, IEEE Transactions, Volume:3 Issue:1, Mar. 1995, Page(s): 49 -58
- [6] L. Benini, G.D. Micheli, E. Macii, D. Sciuto, C. Silvano. Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems. VLSI, 1997. Proceedings. Seventh Great Lakes Symposium on, 13-15 Mar. 1997 Page(s): 77 - 82
- [7] Ki-Wook Kim, Kwang-Hyun-Baek, N. Shanbhag, C.L. Liu, Sung-Mo Kang. Coupling-driven signal encoding scheme for low-power interface design. Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on , 5-9 Nov. 2000 Page(s): 318-321
- [8] C.G. Luyh, Taewhan Kim. Low power bus encoding with crosstalk delay elimination. ASIC/SOC Conference, 2002. 15th Annual IEEE International, 25-28 Sept. 2002 Page(s): 389 -393
- [9] Lu. Tiehuan, J.Henkel, H. Lekatsas, W. Wolf. Enhancing signal integrity through a low-overhead encoding scheme on address buses. Design, Automation and Test in Europe Conference and Exhibition, 2003, Mar. 3-7, 2003
- [10] P.P. Sotiriadis, A. Chandrakasan. Low power bus coding techniques considering inter-wire capacitances. Custom Integrated Circuits Conference, 2000. CICC. Proceedings of the IEEE 2000, 21-24 May 2000 Page(s): 507 -510
- [11] M.R. Stan, W.P. Burleson. Coding a terminated bus for low power. VLSI, 1995. Proceedings., Fifth Great Lakes Symposium on , 16-18 Mar. 1995, Page(s): 70 -73

# The Effect of Data-Reuse Transformations on Multimedia Applications for Different Processing Platforms

N. Vassiliadis, A. Chormoviti, N. Kavvadias, and S. Nikolaidis

Section of Electronics and Computers, Department of Physics,  
Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece  
{nivas, ahorm, nkavv}@skiathos.physics.auth.gr

**Abstract.** Multimedia applications are characterized by an increased number of data transfers and storage operations. Appropriate transformations can be applied at the algorithmic level to improve crucial implementation characteristics. In this paper, the effect of data-reuse transformations on power consumption and performance of multimedia applications, realized on General Purpose (GPP) and Application Specific Instruction set Processors (ASIP), is examined. An ASIP for multimedia applications, designed based on a complete methodology, and the ARM9TDMI are used to evaluate this effect. Results prove the efficiency of the ASIP solution and indicate that both GPP and ASIP approaches can benefit, from such transformations, in terms of energy consumption and performance.

## 1 Introduction

The popularity of multimedia systems used for computing and exchanging information is rapidly increasing. With the emergence of portable multimedia applications (mobile phones, laptop computers, video cameras, etc) the power consumption has been promoted to a major design consideration due to the requirements for long battery life, large integration scale and the related cooling and reliability issues [1][2]. Consequently, there is great need for power optimization strategies, especially in higher design levels, where the most significant savings are achieved.

A number of code transformations can be applied to any algorithm aiming at a memory hierarchy where copies of data from larger memories that exhibit high data-reuse are stored to additional layers of smaller memories. In this way, exploiting the temporal locality of data memory references [1], the greater part of the accesses is performed on smaller memories. Since accesses to smaller levels of the memory hierarchy are less power costly, significant power savings can be obtained [3][4].

Different hardware architectures can be used for the implementation of target applications. The use of application specific integrated circuits (ASICs) leads to high performance, small area and power consumption. However they completely lack flexibility since only a specific algorithm can be mapped on the system. A flexible solution, with a negligible time-to-market (TTM), is the use of an existing GPP. Such a solution is rather unlikely to be viable, due to the fact that conventional RISC/DSP ap-

proaches pose limitations in tuning the architecture towards narrow application domains or they may be prohibitively expensive in respect to energy consumption[5]. Thus, the embedded systems industry has shown an increasing interest in ASIPs, which are processors tailored to the needs of the targeted applications [6], providing the right balance between flexibility, performance and power consumption.

In this paper a methodology for the implementation of an ASIP, from a hardware-software perspective, is followed. Based on this methodology an Application Specific Instruction set Processor for multimedia applications is designed. The effect of data-reuse transformations on an application, which is executed on this ASIP is examined. The Two Dimensional Three-Step Motion Estimation video coding algorithm is used as benchmark. A comparative study indicates that known benefits of data reuse transformations in power and performance on GPP are still valid for an ASIP approach.

In section 2, a brief description of the data-reuse methodology and the used benchmark is given. The followed ASIP design flow is presented in section 3. Results are discussed and analyzed in section 4 while we conclude in section 5.

## 2 Data Reuse Transformations

In data-dominated applications such as multimedia algorithms, significant power savings can be achieved by developing a custom memory organization that exploits the temporal locality in memory accesses[1]. According to the proposed methodology, data sets that are often being accessed in a short period of time are identified and placed into smaller memories leading to a new memory hierarchy. Hence, power savings can be obtained by accessing heavily used data from smaller foreground memories instead of large background memories. Such an optimization requires architectural modifications that consist of adding layers of smaller memories to which frequently used data can be copied. Consequently, there is a trade off here; on the one hand, power consumption is decreased because data is now read mostly from smaller memories, while on the other hand, power consumption is increased because extra memory transfers are introduced.

An exploration of all architectural alternatives is required for finding the optimum solution. This data-reuse exploration is performed by applying a number of code transformations to the original code, which are determined by the group of data sets that are being used in the algorithm. These transformations are extracted according to the methodology described in [3] [4].

As a benchmark, for the application of data-reuse transformations, the popular motion estimation, two dimensional Three-Step Search(TSS) algorithm is used. Motion estimation algorithms are used in MPEG video compression systems [7,8] to remove the temporal redundancy in video sequences which is determined by the similarities amongst consecutive pictures. Instead of transmitting the whole picture, only the displacements of pixel blocks (motion vectors) between neighboring pictures (frames) and the difference values for these blocks have to be encoded. The calculation of the motion vector is performed by means of the matching criterion, a cost function to be minimized [8].

TSS consists basically of four nested loops, one for each block in the frame, one for each step, one for each candidate block and one for every pixel in the block. The number of the corresponding transformation, produced by the application of the data

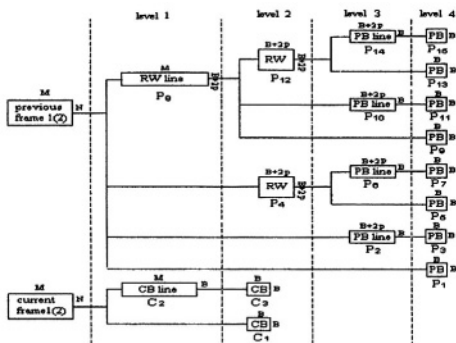


Fig. 1. Custom Memory Hierarchy

reuse transformations on the TSS, and the size of the introduced memory, given parametrically, annotate each rectangle in the Fig.1.

### 3 ASIP Design Flow

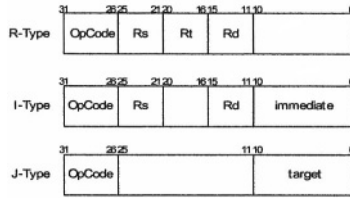
An important issue in ASIP design is the identification of the tradeoffs involved in instruction set and micro-architecture design, which requires efficient architecture design space exploration. In this paper an existing processor architecture, is used at the initial step of the exploration. The goal of the methodology is to extend the existing instruction set of the selected processor by identifying and incorporate new instructions from which the target application can benefit in terms of performance and power consumption. At the same time these newly defined instruction should not introduce significant hardware overhead and mostly not to increase the critical path of the processor.

The different steps of the followed design flow are presented below. The TSS algorithm and its data reuse transformations is used as a representative benchmark. This algorithm is popular in MPEG video compression systems. In addition, since the main objective of this work is to study the effect of data reuse transformations on alternative hardware implementations, the selection of such an algorithm, suitable for the applications of such transformations, is straightforward.

#### 3.1 Establishment of an Architecture Template

In our work, the architecture model assumed is a single-issue machine with a five-stage pipeline, with separate instruction and data buses (Harvard architecture), consistent to the RISC (Reduced Instruction Set Computer) paradigm. The control model is data stationary and execution is performed in-order. The MIPS processor architecture [9] was selected to initialize the design flow.

The initial 32-bit Instructions of the MIPS processor can be grouped in three basic formats, illustrated in Fig.2. R-type instructions use two source registers and one destination register. I-type instructions feature one source register, one destination



**Fig. 2.** Basic Instruction Formats

register and an immediate field containing a constant value. Load and Store instructions and conditional branches, for example, belong to this category. J-type format is typically used by jump instructions and contains only a target field.

### 3.2 Front-End Compilation

At first, the application described in a high level language, in particular ANSI C, must be compiled for the target architecture. The GNU-GCC [10] for embedded architectures, configured as a cross-compiler for the MIPS architecture, is used for this reason. The TSS algorithms with the different data reuse transformations are compiled and machine code of each transformation is generated in this stage.

### 3.3 Dynamic Profiling

The produced machine code is dynamically profiled with the GNU [10] tools (gcc, binutils, gdb) configured for the MIPS processor. Profiling information at the levels of C and assembly code, generated by the execution of the application on the target machine is collected. In this way heavily executed portions of the code can be identified. New candidate instructions from which the application can benefit in terms of performance (at first) can be revealed. Dynamic profiling was performed on all, modified by the data reuse transformations, versions of the application code.

Average profiling values from all the different transformations, at the level of the C code, indicate that the control flow of the application, namely the loop and case statements (“for”, “while”, “loop”, “if”) in the C code consists 24% of the total execution cycles. The bigger portion of the execution time is consumed on the addressing equations and on access to the different memory layers. That is the 62% of the total execution cycles. Only 14% of the execution time is consumed on pure computational micro-operations.

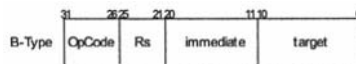
In addition profiling information on the assembly code indicates that there are patterns of instructions that appear with high frequency on the above identified heavily executed portions of the code. In particular the loop statements are executed with two instructions overhead, one instruction for the iteration of the loop and one for the conditional branch of the program. Clearly a newly defined complex instruction that can control the flow of the program with one cycle overhead can boost the performance of the processor. Also various implementations of the branch instruction could have the same results.

As it was observed, a large portion of the execution time is consumed in the addressing of and access on the different memory layers. Load and Store instructions with opcodes refereeing directly to a specific memory layer should be added. Also an arithmetic operation, in particular an addition, always exists before a memory access due to the addressing calculation. Combining the addition with Load Store instructions, resulting in the support of the appropriate addressing modes, significant speed up of the processor's performance can be achieved. It must pointed out that the requirement for such addressing modes imposes the use of different pipeline stages for the EX and MEM stage.

### 3.4 Instruction Set Extension

The previously identified instructions from which the processor can benefit, must be incorporated in the existing instruction set. Additional hardware resources and changes on the processor architecture necessary for the execution of the new instructions must be taken into account.

A new type of instruction is added to support an alternative implementation of the conditional branch instruction. Fig.3, illustrates the new B-Type format of those instructions. The difference is that the I-Type instruction format presented in Fig.1 can support a conditional branch in which registers, Rs and Rd, are compared and if the branch is taken the program branches to the address indicated by the immediate value. The B-Type format can support comparison between an immediate constant value and the Rs register and branch to the address indicated by the target field, in the case that the branch must be taken. The B-Type format does not require any additional computational hardware components. Slight changes on the Instruction Decoding Unit, the Hazard detection Unit and the overall control logic must be made, to support the new format. These changes slightly increase the area requirements with no penalty to the critical path (performance).



**Fig. 3.** The B-Type Instruction Format

A second extension to the instruction set aims to the reduction of the loop overhead from two instructions to one. For this reason the branch instructions of the I-Type and B-Type are combined with an increment operation. The Rs register is incremented, the result is compared with the Rd register, for the I-Type format, or the immediate value for the B-Type format, and based on the result a branch is performed or not. The incremented value of the Rs register is written back in the same register. An increment unit must be included in the Arithmetic/Logical Unit to support the increment/branch MOP. The delay of the new instruction is not in the critical path length therefore no reduction on the performance is occurred. Also minor changes in the control logic of the processor must be performed.

As have been observed the largest portion of the execution cycles is consumed on the addressing equations and memory access. Clearly new defined addressing modes must be incorporated. Firstly separate Load and Store operations for each memory

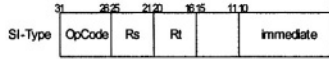


Fig. 4. The SI-Type Instruction Format

Table 1. Instruction Set Extensions

Description	Instruction Format Type	Additional Hardware Requirements	Penalty
Branch_Rs_Immed_Target	B-Type	Control Logic	Area
Inc+Branch_Rs_Immed_Target	B-Type	Control Logic + Incrementer Unit	Area+Delay
Inc+Branch_Rs_Rd_Target	I-Type	Control Logic + Incrementer Unit	Area+Delay
Add+SW_L#_Rs_Rt_Rd	R-Type	Control Logic	Area
Add+LW_L#_Rs_Rt_Rd	R-Type	Control Logic	Area
Add+SW_L#_Rs_Rt_Immed	SI-Type	Control Logic	Area

L# is the desired level of the custom memory hierarchy

layer are included in the instruction set. Therefore, overhead due to the partition of the memory addresses, is removed. Secondly the new Load/Store operations are combined with an addition operation, to produce a new addressing mode, since such MOPs are always executed sequentially. The new complex instruction has a format identical with the R-Type. The Rs and Rt are added, result is used as a memory address in which the content of the Rd register will be stored, or a memory address, the contents of which are going to be loaded in the Rd register. In addition a complex add+Store instruction with immediate value of the data to be stored is included in the instruction set. The SI-Type instruction format is illustrated in Fig.4. The Rs register is added to the Rt register. The produced result is used as the address of the memory layer, provided by the opcode, in which the immediate value is going to be stored. In order for the new instructions to be supported by the architecture, appropriate decoding of the opcodes must be added to give direct access to the desired memory layer. Then, the pipeline stages of the Execution and Memory, must be controlled for the new addressing mode. In order for the new defined instructions to be supported by the processor architecture, additional decode logic and control signals, that add a slight hardware overhead, must be included. Also, no increase on the critical path has occurred due to the incorporation of these instructions, resulting in no degradation on performance. The instruction set extensions are summarized in Table 1.

### 3.5 Code Re-generation

Code Re-Generation is performed by taking into account the new defined instructions. First the original code is parsed and the MOPs are reordered in order to construct the previously identified instruction patterns. The patterns are then substituted by the new defined instructions. Finally, due to the fact that there is no flush unit in the processor architecture, MOPs are reordered to keep the pipeline as full as possibly[11]. Fig.5 illustrates an example.

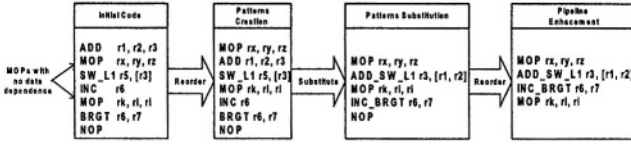


Fig. 5. Code Re-Generation

### 3.6 Cycle Accurate Simulation

For the evaluation of the candidate new defined instructions a cycle accurate simulation model, in the SystemC language, is constructed. The application code with the instruction set extensions are executed on the simulator. Execution cycles for each transformation are derived. In addition information about the execution of particular instructions and access to crucial hardware components like memories, are collected.

### 3.7 Hardware Model

A model in a hardware description language (VHDL) is designed. All the necessary micro-architectural and hardware modifications in order for the new defined instructions to be supported by the designed processor are incorporated. The design is synthesized on a popular standard cell technology and information for the performance (critical path), power consumptions and the area requirement of the processor are collected. Information from the previous executed steps are collected. Results are evaluated and they are presented in the next section.

## 4 Experimental Results

The different versions of the TSS application code, resulting by the application of the data-reuse transformations, were compiled for the AMR9TDMI and the ASIP cores. Cycle accurate simulations were performed on the ARMulator and a SystemC simulator for the ASIP that was designed for this reason, respectively. The ARM9TDMI is a five stage pipelined processor and was selected as the GPP core based on the architectural similarities with the ASIP in order to achieve fair comparisons. Due to these architectural similarities, the ARM core is producing very close results with the original MIPS architecture and it is used in order to obtain comparisons with a popular commercial GPP. Information on the total executed cycles and accesses to different memory layers were collected from the simulators.

The TSS was executed on digital pictures of  $M \times N = 144 \times 176$  pixels. The block size  $B$  was set to 16 while the search window size  $[-p, p]$  was set to  $[-7, 7]$ .



### 4.1 Performance Results

Total execution cycles for ARM9TDMI and ASIP are presented in Fig.6 and 7 respectively, for each transformation. As it can be observed data-reuse transformations can be used not only for energy savings, on data intensive applications, but also to boost the performance of the processor. Specifically, the P4 transformation provides the highest performance for ARM9TDMI and ASIP and can be used to achieve performance gain of 29% and 54% for the two processors respectively, compared with the original TSS.

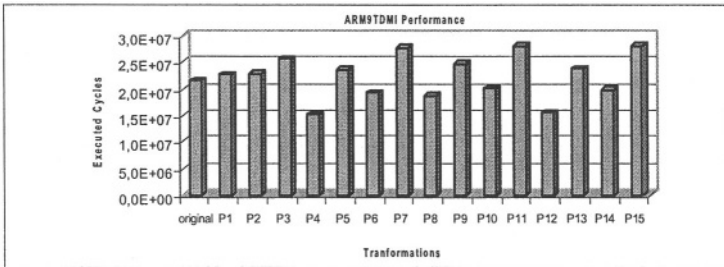


Fig. 6. Executed Cycles for ARM9TDMI

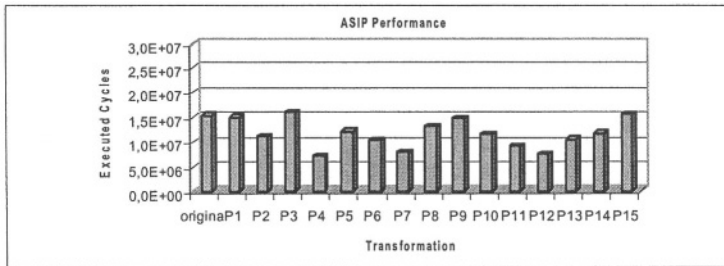


Fig. 7. Executed Cycles for ASIP

In addition, the well known efficiency of the ASIP solution towards the GPP (ARM) for the target application is proved. In particular for the most efficient P4 transformation, in terms of performance, ASIP is capable to deliver 54% performance gain compared with the ARM9TDMI core.

The designed VHDL model of the ASIP was synthesized on the STM 0.1 8um technology and a critical path of 4ns (250 MHz) was obtained. Based on the ARM data-sheet [12] the ARM9TDMI implemented in the same technology can work with 200MHz clock frequency (worst case). Due to the fact that low power memories with small performance were used for both cases, the clock period (frequency) was determined from the largest memory access time and it is 5.53ns (180Mhz) [13] for all transformations.

## 4.2 Energy Results

As already mentioned, data-intensive applications are dominated by power consumption due to data and instruction memories accesses [3]. Based on this assumption, energy consumption estimations for the different transformations on the ARM9TDMI and ASIP are presented in Fig.8 and 9 respectively.

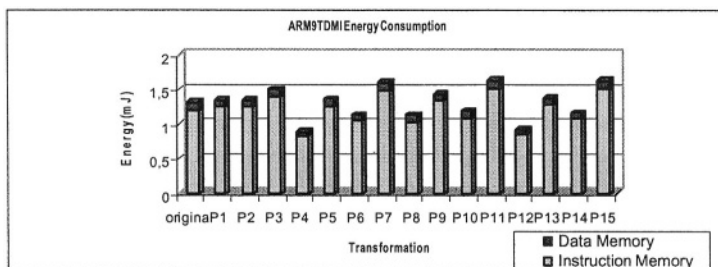


Fig. 8. Energy Consumption for the ARM9TDMI

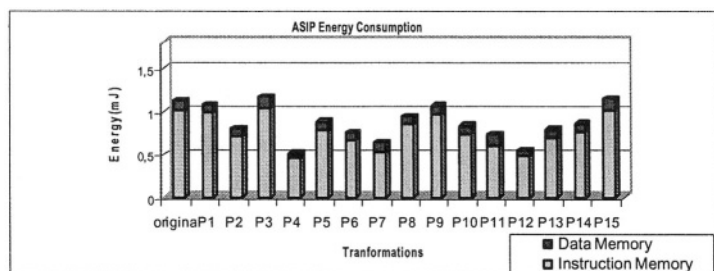


Fig. 9. Energy Consumption for the ASIP

For the calculation of the energy consumptions, access to the instruction memory and the used data memory layers for each transformation and each processor were obtained from the cycle accurate simulations. SRAM memories with appropriate size were used for each layer of the data memory. ROM memory was used for the instruction memory. For the ARM9TDMI 4KB instruction memory was used. Because of the 50% smaller code size, compared to ARM, that ASIP provides, a 2KB ROM instruction memory was used in this case. Power consumption for each memory were obtained from the Embedded Memory Generator of [13].

Results indicates that energy consumption is dominated by the energy consumption due to access on the instruction memory. Such a comparison is not straightforward, since it is dependent by algorithmic parameters, and it is not in the intentions of this paper.

Furthermore, results clearly indicate that energy savings can be obtained through data-reuse transformations based on the reduced number of accesses on the instruction memory. P4 is also the best transformation, in terms of energy savings, for both ARM and ASIP processors achieving 32% and 54% energy savings respectively, compared with the original TSS. In addition 42% energy savings can be achieve by using the P4

transformation on the ASIP compared with ARM9TDMI. These energy savings, of the ASIP towards ARM, result from the smaller number of access to the Instruction Memory but also due to the smaller Instruction Memory size of ASIP.

## 5 Conclusions

In this paper, the effect of data-reuse transformations on multimedia applications implemented on different processing platforms, namely a GPP and an ASIP has been presented. An ASIP for multimedia applications has been designed for this reason. The popular, on video compression systems, TSS algorithm and its modifications imposed by data-reuse transformations were used as benchmarks. Performance and energy consumption of these benchmarks, were estimated, on the ASIP and the ARM9TDMI for comparisons reasons.

Results indicate that both solutions, namely ASIP and GPP, can benefit in terms of performance and energy consumption by selecting the appropriate custom data memory hierarchy. In addition, ASIP can achieve highest performance and energy reduction through these transformations, compared to a GPP.

## References

1. F. Catthoor, S. Wuytack et al: Custom Memory Management Methodology, Kluwer Academic Publishers, Boston, 1998.
2. A. Chandrakasan and R. Brodersen: Low Power Digital CMOS Design, Kluwer Academic Publishers, Boston, 1995.
3. S. Wuytack, J-P. Diguët, F. Catthoor, H. De Man : Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Mappings, special issue of IEEE Transactions on VLSI Systems on low power electronics and Design, Vol. 6, No. 4, pp. 529-537, December 1998.
4. S. Kougia, A. Chatzigeorgiou, N. Zervas, S. Nikolaidis : Analytical Exploration of Power Efficient Data-reuse Transformations on Multimedia Applications. International Conference on Acoustics, Speech and Signal Processing, Utah, May 2001
5. M.F. Jacome and G. de Veciana,: Design Challenges for New Application-Specific Processors, IEEE Design and Test of Computers, Vol. 17, No. 2, pp. 40-50, 2000.
6. MK. Jain, M. Balakrishnan, A. Kumar : ASIP Design Methodologies : Survey and Issues. Proceedings of 14<sup>th</sup> International Conference on VLSI Design, pp. 76-81, January 2001
7. International Organization of Standardization, Working Group on Coding of Moving Pictures and Audio, MPEG-4 Video Verification Model Version 18.0, Pisa, January 2001.
8. Vasudev Bhaskaran and Konstantinos Konstantinides, Image and Video Compression Standards: Algorithms and Architectures, Second Edition, Kluwer Academic Publishers, Boston, 1999.
9. J.Hennessy and D.Patterson,: Computer Architecture: A quantitative approach, Morgan Kaufmann,1991.
10. GNU Website : <http://www.gnu.org>
11. I.-J. Huang and A. Despain, : Synthesis of application specific instruction sets, IEEE Trans. on CAD,pp. 663-675, 1995.
12. ARM Ltd. ARM9 Datasheet, 2003
13. Dolphin Website:<http://www.dolphin.fr/flip/ragtime>

# Low Power Co-design Tool and Power Optimization of Schedules and Memory System

Patricia Guitton-Ouhamou, Hanene Ben Fradj, Cécile Belleudy, and Michel Auguin

Laboratoire d'Informatique, signaux et Systèmes de Sophia-Antipolis, Les Algorithmes-bat.  
Euclide 2000, route des Lucioles-BP 121, 06903 Sophia-Antipolis Cedex. FRANCE  
{guitton,benfradj,belleudy,auguin}@i3s.unice.fr

**Abstract.** Telecommunication applications integrate more and more functionalities, and a huge capacity of memory is required; as a result power consumption increases and battery lifetime becomes a serious limitation. An other consequence is that the time to realize such systems becomes too long! This paper presents methods to optimize consumption/time for HW/SW codesign. Then, a refinement step is executed to optimize the schedulings by applying the Dynamic Voltage Scaling/ Dynamic Frequency Scaling technique (DVS/DFS), and founding the best distribution of data between internal memory and external memory in order to decrease the global consumption. The novelty of our work consists in taking into account the consumption parameter during the allocation and scheduling steps; refining the scheduling by exploiting the DVS technique; optimizing the system memory consumption by improving the data storage in internal memory of the processor.

## 1 Introduction

Telecommunication applications integrate more and more functionalities, so power consumption increases and battery life time becomes a serious limitation. For example, the 4<sup>th</sup> generation of wireless telephone has to provide services of the 3<sup>rd</sup> generation of wireless telephone, bluetooth or wi-fi services, some local networks, and in the same time, internet... Many and various services are required. Furthermore frequency of microprocessors is multiplied by two at each new generation (2 or 3 years), area increases about 25 % each generation, and the software size is multiplied by 2 every year! The consequence is that energy and power consumption dramatically increase. However system designer productivity increases only about 21 % each year against transistors per chip growth rate of 58 %. To improve system's productivity, it is needed to develop tools to help designers. As multimedia applications are greedy in terms of memory storage, it is necessary also to improve the memory management strategy.

Our work is composed of different steps: synthesis of architecture (allocation/scheduling), refinement of schedules and memory optimizations. The synthesis step based on list-scheduling algorithm optimizing the consumption. As applications become increasingly sophisticated and processing power increases, several commercial variable voltage microprocessors were developed (StrongARM, XScale). Dynamic Voltage Scaling/ Dynamic Frequency Scaling (DVS/DFS) becomes a key

technique to reduce energy dissipation by lowering the supply voltage and operating frequency. In this way, refinements of schedules exploiting the DVS/DFS technique are proposed for the components of the system architecture with DVS/DFS capabilities. Finally, the energy of the memory system is reduced by optimizing the distribution of data so that the internal (on chip) memories are used efficiently. This paper is organized as follows: section 2 introduces the energy/power consumption problem and lists some previous work. Section 3 discusses the synthesis of system architectures and some optimization (in area, time and consumption) methods for HW/SW co-design are presented. Section 4 gives results for this step. Section 5 presents some schedule refinements exploiting the DVS/DFS technique and memory optimizations. Finally, some results are presented in section 6, followed by conclusion and future work.

## 2 Consumption in Co-design

Some tools for co-design take into account energy and power consumption. Our goal is to design an architecture that corresponds to an accurate compromise between run time, area and energy. Most of works in this domain are made at logic and RTL levels. As numerical systems are more and more complex, our work addresses the system level. MOCSYN [1] is a system level approach that takes into account the energy due to task execution and communication. After allocation of the tasks on cores, it synthesizes the interconnect, that means it chooses the best types of buses to support communication. The allocation is optimized to reduce the energy dissipation due to communication. However MOCSYN does not consider the DVS technique, a main technique that allows to decrease the global consumption. In addition, it doesn't optimize the memory architecture, whereas a great contribution in energy and power comes from the memory. COSYN-LP [2] is based on a clustering strategy that optimizes power by modifying priorities in allocation according to an average power dissipation. As MOCSYN, COSYN-LP does not apply the DVS technique, and does not optimize the memory structure. IMPACCT [3] the firstly builds architecture with a schedule of tasks. Then, the method checks that peak power is met. If not, some tasks are delayed. The goal is to limit parallel execution of tasks. Then, the algorithm optimizes energy by modifying the scheduling. In addition, the idle mode of some units is managed during a final step. This method integrates more techniques to reduce the consumption than the two previous tools. More generally, their work focuses on the management of the energy and power of battery: the scheduling is adapted according to the period of charge and discharge of the battery in an efficient way [4]. Their goal is to integrate algorithms in a real time OS to get an important power savings by exploiting the DVS/DFS technique. These techniques concern the EDF (Earliest First Deadline) and the RM (Rate monotonic) scheduling strategies, that means the tasks must be periodic. This way is very efficient, if there is an optimal number of changes in frequency/voltage supply. Indeed, when frequency/voltage supply is modified, there is a consumption/time penalty due to the stabilization period (stabilization of the PLL, new clock signal propagation...). Some works as [5, 6] show that it is possible to determine an optimal number of variations (frequency/voltage supply) to have a gain in consumption. Unoptimized variations lead to penalties that may be more important than benefits. With regard to memories, some consumption models have been devel

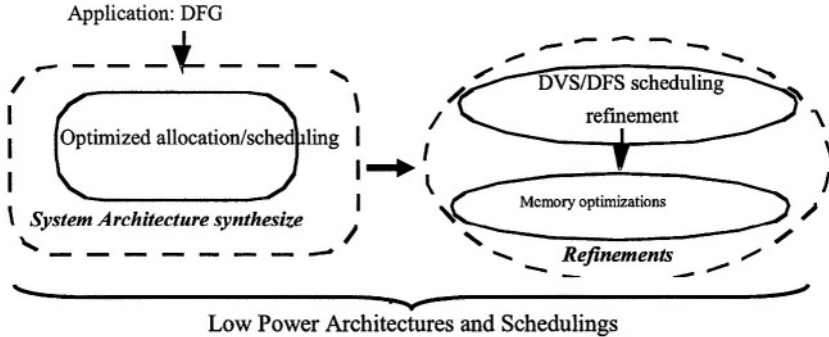


Fig. 1. Overview of the overall strategy

oped, but they are very complex [7]. Models in [8] are very sophisticated and not applicable at system level. [9] concerns architecture synthesis. The contribution of our work consists in taking into account the consumption during allocation and scheduling, managing low power modes of the processors (RISC, DSP...) and hardware units (ASICs, FPGA...), refining the schedule by exploiting the DVS/DFS technique, and optimizing the localization of data in memories. All these steps are coupled in our method. Let us see an overview of this work in figure 1.

### 3 System Architecture Synthesis

The co-design tool CODEF [11] automatically explores architectures that satisfy time and area constraints. The CODEF-LP tool is extended with an optimized allocation/scheduling step and some refinements for low power. To take into account the consumption, it is necessary to estimate or to measure the consumption of processors and hardware units [10]. These values are collected in libraries of CODEF-LP.

#### 3.1 Libraries/Consumption Estimation

To synthesize architectures, CODEF-LP works with two libraries.

**Unit library.** This library contains parameters needed to represent the operating modes (for example, processor in sleep mode, FPGA in slow mode...). Power values are given for the average supply voltage and the average clock cycle (frequency). This model is also used for ASIC and FPGA.

**Implementation library.** For each task  $\tau_i$  and for each unit  $u_j$  that may execute  $\tau_i$ , we associate an implementation entry in the library that is described by the task  $\tau_i$ , the unit  $u_j$  (processor, accelerator...), the number of cycles to execute  $\tau_i$  on  $u_j$ , a ratio of memory accesses corresponding to the load/store equation and the power consumption. This consumption is an average value. The energy consumption is computed as follows:

$$E_{total} = \sum_{i=1}^U E_{idle_i} + \sum_{j=1}^{N_i} E(\tau_{a(j,i)})$$

$$E_{idle_i} = \left( T_{max} - \sum_{j=1}^{N_i} (t_{\tau_{a(j,i)}}^e - t_{\tau_{a(j,i)}}^b) \right) \times P_{idle}^i \text{ and}$$

$$E_{\tau_{a(j,i)}} = (t_{\tau_{a(j,i)}}^e - t_{\tau_{a(j,i)}}^b) \times P_{\tau_{a(j,i)}}^i + E_{access}^i(\tau_{a(j,i)})$$

where U is the number of units in the architecture,  $N_i$  the number of allocated tasks to the unit i,  $T_{max}$  is the total execution time of the application on the architecture,  $a(j,i)$  represents the  $j^{th}$  allocated task to the unit i,  $t^e$  (and  $t^b$ ), the end time (and beginning time) of the task execution on the unit. The beginning and end times are determined by CODEF, after the allocation/scheduling step. The value  $P^i(\tau_{a(j,i)})$  is the power due to the execution of the task. The energy  $E_{access}^i(\tau_{a(j,i)})$  results from external memory accesses when CODEF adds an external memory because the available space in the internal memory is too limited. This energy is detailed in section 3.3.

### 3.2 Synthesis Algorithm

We now describe CODEF-LP, and particularly the algorithms for allocation and scheduling to optimize the energy. CODEF is based on a List-scheduling allocation technique. Tasks in the DFG which have their predecessors already allocated and scheduled are analyzed. Two sets of lists are constructed with available implementations of tasks ... from library. Tasks with low urgency for which units in the architecture can be reused are assigned to list  $L_{reuse}$ . Tasks whose implementations require to add a new unit to the architecture are assigned to list  $L_{new}$ . Implementations that lead to a high urgency are assigned in list  $L_{others}$ . A task on a unit with DVS/DFS discrete capabilities is considered as different implementations, are associated with each <frequency, power supply> of the unit.

In CODEF-LP each list of implementations are sorted in increasing order of the parameter

$$\zeta_E(i, j) = \frac{E(\tau_i, u_j)}{\left( \sum_k E(\tau_i, u_k) \right) / n}, \text{ with } E(\tau_i, u_j) \text{ energy of } \langle \tau_i, u_j \rangle, \text{ and } \left( \sum_k E(\tau_i, u_k) \right)$$

the sum of energies for all n possible implementations of  $\tau_i$ .

The goal of this parameter is to counterbalance a local choice of an implementation with its effect on the entire system.

At each iteration of the list scheduling process, CODEF-LP constructs these lists and select for the most time critical task  $\tau_i$  the first implementation encountered in lists  $L_{reuse}, L_{new}, L_{others}$ .

### 3.3 Memory Synthesis

If the on chip memory of a unit is unable to store the data of a task, an external memory is added. This synthesis technique is trivial and not optimal. The power dissipated by system memory is evaluated as follows:

$$E_{access}^i(\tau_{a(j,i)}) = T_{access}(\tau_{a(j,i)}) \times P_{access}^m + (t_{\tau_{a(j,i)}}^e - t_{\tau_{a(j,i)}}^b - T_{access}(\tau_{a(j,i)})) \times P_{idle}^m$$

$$\text{With } T_{access}(\tau_{a(j,i)}) = \rho(\tau_{a(j,i)}) \times C(\tau_{a(j,i)}) \times t_m$$

where  $C(\tau_{a(j,i)})$  is the number of cycles to execute  $\tau_k$  on unit  $i$ ,  $\rho(\tau_{a(j,i)})$  is the ratio of  $C(\tau_{a(j,i)})$  corresponding to cycles where the task accesses to the memory, and  $t_m$  access time.  $P_{access}^m$  is the power for a single memory access, and  $P_{idle}^m$  is the power of the memory when it is in an idle state.

## 4 Results of Optimized System Architecture Synthesize

We present here some results of this first allocation with low power consideration. We have considered an embedded video application. Compared with CODEF, solutions provided with this new allocation technique are about 50% less energy consuming for an increased area of only 14 %. In both case time constraints are met.

## 5 Refinement on Architecture and Scheduling

In this part, our goal is to refine scheduling by exploiting the DVS and DFS capabilities. Indeed, this technique is one of the main techniques that allow to decrease the consumption. Reducing the frequency allows to decrease the power supply, and so to decrease the consumption. The reduction is possible according to:

$$T_{\min} = \frac{1}{f_{\max}} = \frac{k V_{dd}}{(V_{dd} - V_{th})^2}$$

$V_{dd}$  is the supply voltage,  $V_{th}$  is the threshold voltage, and  $k$  is a technological term. If in the schedule there are idle periods of some units in the architecture it is possible to slow down some executions of tasks. Slowing down some tasks allows to decrease the supply voltage and so the power and the energy. In this section, it is considered that the units in the architecture can have DVS capabilities with discrete frequencies, sleep and idle modes. Penalties due to frequency changings are also considered.

Tow refinement methods are applied on the schedule provided in the previous step. they consist in local and global scaling of the clock frequencies.



### 5.1 Refinement of Scheduling

**Local scaling of scheduling.** After scheduling, mobilities of tasks can be defined according to the deadlines. Therefore, a scaling factor can be defined for each task. The processor is scaled to a lower frequency defined by the scaling factor of the task in order to meet its deadline. The deadline of a task can be set up manually or automatically determined. It is the maximum end time of the task which respects the dependencies with its successors and deadlines of the application. The frequency is the

nearest available discrete frequency scaled by  $\frac{n}{\text{availabletime}}$ , where n is the number of cycles to execute the task. This local strategy introduces a set of (frequency, voltage) changes which lead consumption/time penalties. So a global strategy is considered.

**Global scaling of scheduling.** The goal is to determine the frequency and the supply voltage that reduce the consumption. If the execution time of the whole application is smaller than the deadline, the algorithm analyzes the units with DVS capabilities.

Consider the schedule provided before refinement in figure 2. Only one unit in the architecture has DVS/DFS capabilities. The unit with DVS can be scaled down since the execution time of the whole application is lower than the deadlines of the application.

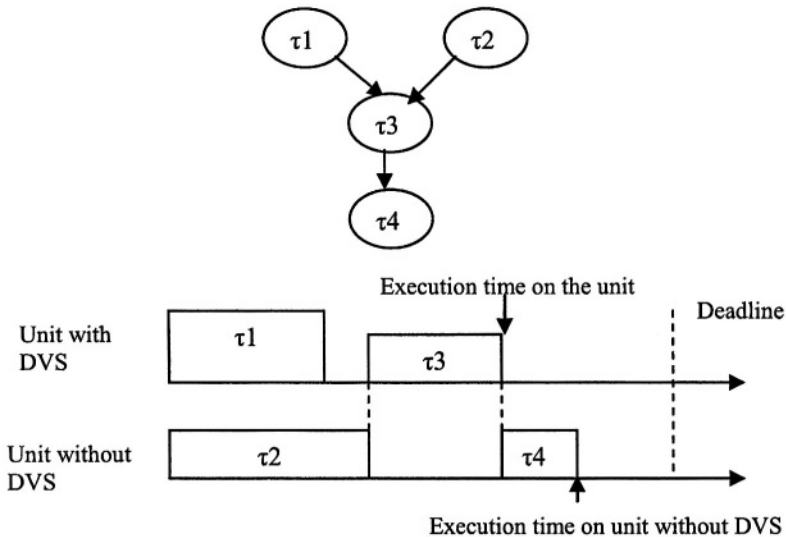


Fig. 2. Example of schedule

The scaling factor applied to this unit is computed according equation:

$$\text{scaling factor} = \frac{\text{deadline} - \text{execution time of unit without DVS}}{\text{execution time on the unit}} \tag{1}$$

In order to check a priori the feasibility of this scaling, a condition is introduced in order to deal with data dependencies between tasks. This condition is  $\eta \leq 1$  with

$$\eta = \frac{f_s \times t_i}{T_{di}}$$

,  $f_s$  is the scaling factor,  $t_i$  the execution time of the task executed by this unit,  $T_{di}$ , the available time for this task. This condition is computed for each task, on the updated scheduling. If the condition is not respected with this factor, a new factor is deduced from  $\eta = \frac{f_s \times t_i}{T_{di}} \leq 1 \Rightarrow f_s$ . If task1 is executed by unit u1 at frequency f1

with an execution time t1, when frequency is decreased, the execution time is  $t_1 \times \text{scaling factor}$  : the frequency is deduced by  $\frac{\text{number of cycles of the task}}{t_1 \times \text{scaling factor}}$ . If

the architecture is composed of more than one unit with DVS, the strategy is applied successively to each unit. However the order in which the units are analyzed may give different results. The unit scaled first is the unit with the greatest benefit in consumption. If there are two or more units (e.g. u1 and u2) with initial global execution energy  $E_1$  (resp.  $E_2$ ), with  $E_1^*$  ( $E_2^*$ ) the energy after scaling, we calculate  $\Delta 1 = E_1 - E_1^*$  and  $\Delta 2 = E_2 - E_2^*$ . If  $\Delta 1 > \Delta 2$ , the scaled unit is u1, otherwise u2. The advantage of this algorithm is to allow a reduction of frequency and supply voltage without penalties.

**Mixed scaling of scheduling.** In order to benefit from the advantages of the local and global scaling, the both approaches have been combined. The global strategy is applied first and if there are some tasks with mobilities in the scaled schedule, the local scaling is applied.

**Mixed scaling of scheduling.** In order to benefit from the advantages of the local and global scaling, the both approaches have been combined. The global strategy is applied first and if there are some tasks with mobilities in the scaled schedule, the local scaling is applied.

## 5.2 Memory Optimization

Memory energy contributes significantly to the overall energy dissipation in SoC-based embedded systems. Thus memory is an important candidate for energy optimization. In this way and to achieve our target, another optimization is realized in this work to reduce consumption of memories. Our co-design tool CODEF-LP adds external memory to the system on chip if internal memory of unit is unable to store the data of the tasks allocated on it. As internal memories are more interesting in term of consumed energy and access time than the external ones, we propose to use as well as possible the internal memory of the processor. This is achieved by releasing the internal memory of the processor from the data of the task which has just completed its execution in order to use it to store the data of the next task on the processor. Before any data transfer, it is necessary to check two conditions in order to be sure to have an energy benefit:

1. The transfer time from and towards the external memory ( $T_{tr}$ ) is lower than the time between the execution of two successive tasks on the processor ( $T_{idle}$ ).
2. The energy of transfer ( $E_{Transfer}$ ) + energy dissipated if the data are stored in the internal memory ( $E_{Tl\_interne}$ ) < energy dissipated if the data are stored in the external memory ( $E_{Tl\_externe}$ ).

## 6 Results

To compare our approach, we consider the example introduced by COSYN-LP [2] and presented in figure 3.

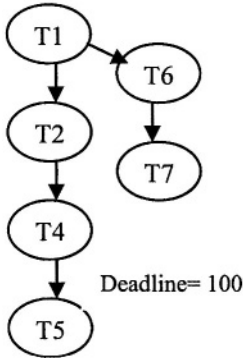


Fig. 3. Example of COSYN-LP

Table1. Power consumption of tasks

Task	Power
T <sub>1</sub>	0.2
T <sub>2</sub>	0.2
T <sub>3</sub>	0.2
T <sub>4</sub>	0.2
T <sub>5</sub>	0.2
T <sub>6</sub>	0.8
T <sub>7</sub>	0.8

The schedule given in [2] is illustrated in figure 4.

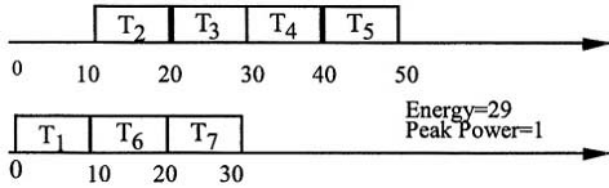


Fig. 4. Schedule a

As deadline=100, it is possible to slow down the processors by a global scaling factor computed using (1)

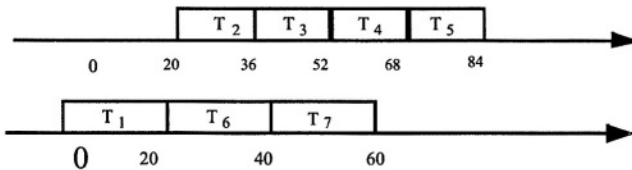


Fig. 5. Schedule b

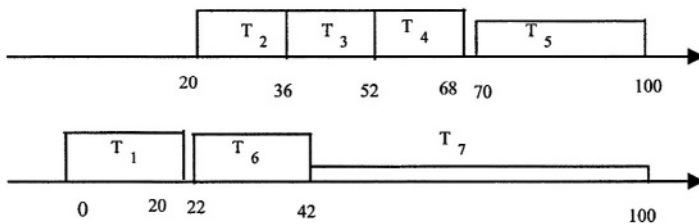
After applying the global strategy for both units, the energy is reduced to 13, so an energy saving about 55% is obtained compared with the non optimized solution. Memory optimization strategy can be applied to this schedule with a transfer data time equal to 2. We notice that T6 data are transferred towards the internal memory by realizing this one of T<sub>1</sub> data. The internal memory is unable to store all the data of

$T_7$ , those ones are remained in external memory. Another transfer is done for  $T_4$  data of unit 2. The time constraint prohibits the transfer of  $T_5$  data towards the internal memory. Thus we obtain an energy saving about 43% compared to the schedule b. This strategy was applied for both units as it is shown in Table 2, the internal memory size is 12 kbytes.

**Table 2.** Data size and location in memory

	Internal Memory size (kbytes)	Task	data size of tasks (kbytes)	Data localization (before optimization)	Data localization (after optimization)
Unit 1	12	T1	8	intern	intern
		T6	6	extern	intern
		T7	16	extern	extern
Unit 2	12	T2	8	intern	intern
		T3	2	intern	intern
		T4	6	extern	intern
		T5	11	extern	extern

Finally, the local strategy is applied (schedule c): each task is slowed down according to their mobilities. The execution time is 100, corresponding to the deadline and the energy saving is about 49 % compared to the non optimized solution (schedule a provided in [2]), and the peak power saving is about 25% compared to the non optimized solution. The final schedule is illustrated in figure 6



**Fig. 6.** Schedule c

## 7 Conclusion

In this paper, design space exploration for low power system design is presented. Energy consumption is considered in successive steps of a system level design flow: low power allocation/scheduling in the design space exploration, data distribution that promotes the utilization of internal memories and refinement steps to optimize the scheduling of tasks. These refinements consist in local and global adjustments of the frequencies of the processors with DVS/ DFS capabilities. Energy and power savings are obtained with these techniques. As future work, we would like to particularly study the consumption of the interconnect and the design of memory hierarchy during the system design flow. It would be also interesting to study the on line management of low power modes under operating system scheduling.

## References

1. R.P. Dick, N. K. Jha.: MOCSYN: Multiobjective core-based single-chip system synthesis, in Proc. Design Automation & Test in Europe Conf, Mars 1999, pp. 263-27.
2. Bharat P. Dave, Ganesh Lakshminarayana, Niraj K. Jha: COSYN: Hardware-Software Co-Synthesis of Heterogeneous Distributed Embedded Systems in proceedings of Design Automation Conference, California (1997), pp 703-708.
3. Pai H. Chou, Jinfeng Liu, Dexin Li, Nader Bagherzadeh, : IMPACCT: Methodology and Tools for Power-Aware Embedded Systems, Design Automation for Embedded Systems, 7-205-232, 2002, Kluwer Academic Publishers.
4. J. Liu, P. H. Chou, N. Aboughazaleh, F. Kurdahi: A Constraint-based Application Model and Scheduling Techniques for Power-aware Systems , Proceedings of the Ninth International Symposium on Hardware/Software Codesign CODES 2001, p. 153, Copenhagen, Denmark.
5. N. AbouGhazaleh, D. Mossé, B. Childers, R. Melhem: Toward The Placement of Power Management Points in Real Time Applications, COLP'01 (Workshop on Compilers and Operating Systems for Low Power), Barcelona, Spain, 2001.
6. D. Mossé, H. Aydin, B. Childers, R. Melhem, Compiler-Assisted Dynamic Power-Aware Scheduling for Real-Time Applications, Workshop on Compilers and Operating Systems for Low Power, Oct. 2000.
7. M. Kamble, K. Ghose: Modeling energy dissipation in low power caches. Technical Report, CS-TR-98-02, Departement of computer science, SUNY6Binghamton, 1998.
8. F. Catthoor, E. de Greef, S. Suytack: Custom memory management methodology: Exploration of memory organisation for embedded multimedia system design", Kluwer Academic Publisher, 1998. ISBN 0-7923-8288-9.
9. G. Corre, N. Julien, E. Senn, E. Martin : Optimisation de la consommation des unités de mémorisation lors de la synthèse d'architecture, 3ème Colloque de CAO de Circuits et Systèmes Intégrés, Paris, Mai 2002.
10. P.Guitton-Ouhamou, C. Belleudy, M. Auguin: Power consumption Model for the DSP OAK Processor, proceedings of 11th IFIP International Conference on Very Large Scale Integration- System-On Chip 2001, Montpellier, France, pp73-78.
11. M.Auguin, L.Capella, F. Cuesta, E. Gresset : CODEF: A System Level Design Space Exploration Tool, Sophia-Antipolis forum on MicroElectronics, SAME 2000, France.

# Hardware Building Blocks of a Mixed Granularity Reconfigurable System-on-Chip Platform

K. Masselos<sup>1</sup>, S. Blionas<sup>1</sup>, J-Y. Mignolet<sup>2</sup>, A. Foster<sup>3</sup>, D. Soudris<sup>4</sup>, and S. Nikolaidis<sup>5</sup>

<sup>1</sup>INTRACOM SA, Peania 19002, Greece  
{kmas, sbli}@intracom.gr

<sup>2</sup>IMEC, Kapeldreef 75, B 3001, Leuven, Belgium  
jean-yves.mignolet@imec.be

<sup>3</sup>ST Microelectronics Belgium, Zaventem, Belgium  
alun.foster@st.com

<sup>4</sup>Democritus University of Thrace, Xanthi, Greece  
dsoudris@ee.duth.gr

<sup>5</sup>Aristotle University of Thessaloniki, Thessaloniki, Greece  
snikolaid@physics.auth.gr

**Abstract.** Due to the combination of flexibility and realization efficiency, reconfigurable hardware has become a promising implementation alternative. In the context of the IST-AMDREL project, a mixed granularity reconfigurable SoC platform targeting wireless communication systems has been developed. The platform's main building blocks are presented, including coarse grain reconfigurable unit, embedded FPGA, interconnection network and application specific reusable blocks. The combination of these blocks in platform instances is expected to lead to a good balance between implementation efficiency and flexibility. An AMDREL platform based reconfigurable SoC for a multi-mode wireless networking system is currently under development.

## 1 Introduction

Due to the combined need for flexibility (for adaptation to different evolving standards and operating conditions) and efficient realisation of wireless multimedia communications and also due to the increasing non-recurring engineering (NRE) costs related to system-on-chip development, reconfigurable hardware has become a promising implementation alternative for such applications.

A number of different FPGAs also with system level capabilities are commercially available. However their cost is prohibitive for large volume products. Commercial system ICs for the WLAN baseband processing are typically using architectures of single DSP or microcontroller core together with one or more algorithm specific fixed hardware accelerators. One solution applies hardware accelerators for transmit and receive filtering and channel encoding and decoding [11]. Another platform uses accelerators for channel decoding, FFT and synchronisation algorithms [12]. The fixed hardware accelerators require considerable large chip area due to demanding worst-case performance requirements. There are some promising results of using reconfigurable logic to enhance implementation quality of baseband processing archi-

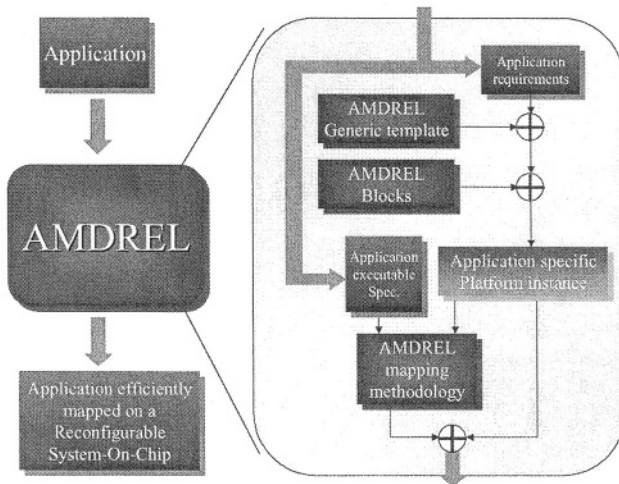


Fig. 1. The AMDREL platform

tures. DreAM is an example of configurable SoC architecture utilising integrated coarse-grain dynamically reconfigurable hardware [13].

To address the issues of reconfigurable System-on-Chip design the AMDREL project [10] develops a platform that includes: a) An architectural template for SoCs with reconfigurable blocks of mixed granularity. b) A set of hardware building blocks (embedded FPGA, coarse-grain, interconnect, custom blocks). c) Design methodologies to map applications on platform instances. Figure 1 depicts the components of the proposed platform. In this paper the hardware building blocks of the platform are presented. Detailed description of the mapping methods is out of the scope of this paper.

The rest of the paper is organized as follows: In section 2 the generic targeted platform architecture is briefly described. Section 3 and 4 present respectively the coarse-grain and fine-grain reconfigurable blocks. Section 5 depicts the interconnection network and application specific blocks are discussed in Section 6. Finally conclusions are drawn in section 7.

## 2 Target Architecture Model

A generic view of the targeted platform architecture is shown in Figure 2. The micro-processor controls the operation of the SoC including the re-configuration of the relevant blocks. More detailed description of the platform's reconfiguration mechanisms is out of the scope of this paper.

In contrary to most existing off-the-shelf reconfigurable hardware platforms the platform under consideration includes reconfigurable processing elements of different granularities combined with hardwired blocks for complex tasks not requiring flexibility. The inclusion of reconfigurable processing blocks of different granularities in the targeted platform is expected to lead to efficient mapping of the application tasks

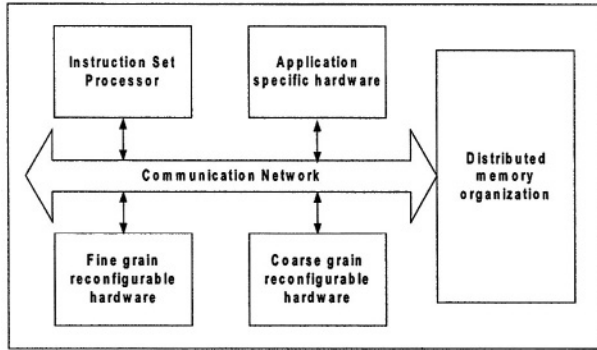


Fig. 2. Generic targeted architecture model

on the platform's processing elements without significantly sacrificing flexibility. Two major types of reconfigurable blocks are included:

- a) Coarse grain blocks of word-level granularity. Such blocks can be customized to specific group of tasks. For example a suitably connected array of multipliers and adders can be used for the realization of a number of DSP tasks such as FFT and FIR filters.
- b) Embedded FPGA including bit level logic units. This block is more flexible and can be used for a large number of tasks. It is particularly suited for control oriented functionality such as the time critical FSMs of WLAN MAC layer.

Application specific customized blocks are realized as reusable highly parameterized soft IP cores. This allows the use of the blocks in different platform instances. The different processing elements communicate through a reconfigurable interconnect network (by reconfigurable, it is meant that it will support dynamic rerouting of communication; the network will in fact be physically fixed in silicon). A communication network has been selected for on-chip communication instead of conventional bridged buses for performance, energy, flexibility and scalability reasons. Memory is included in the target platform for application and reconfiguration data storage and communication buffering.

### 3 Coarse Grain Reconfigurable Processing Elements

In coarse grain reconfigurable hardware some flexibility is traded-off for a potentially higher degree of optimisation in terms of area and power and the ability to reconfigure the platform at a rate, which is significantly faster than the changes of mode observed by a user of the application (not possible in most conventional FPGAs).

The architecture selected for the coarse grain reconfigurable elements of the targeted platform is a programmable VLIW data path engine whose internal architecture is customized to the set of tasks that will be realized on it in each case. (From a methodology point of view the key point is the identification of the tasks that will be jointly realized on the same reconfigurable element and the derivation of the most efficient organization for it). This architecture has been selected instead of an array of processing elements type of architecture since it leads to better hardware utilization.



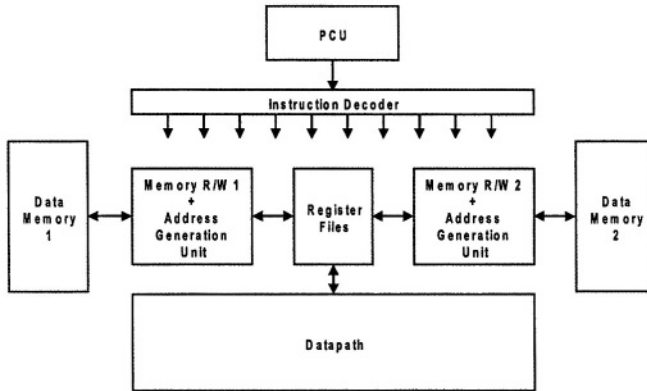


Fig. 3. Coarse grain reconfigurable hardware example

The architecture does not contain support for common data-types and/or operations, and contains specific hardware for implementing dedicated functions where such hardware implementation makes sense from a throughput point of view. The operation of the coarse gain elements is controlled by micro-code stored in a memory block. A simple view of the selected coarse grain elements architecture is shown in Figure 3.

The core of the coarse grain architecture consists mainly of register files to store temporary results, an address generation unit to read from / write to the source and destination memories, and a datapath with only the required operators, structured similarly to an ALU. It is equivalent to a Harvard based architecture with parallel instructions for the ALU and the memory accesses.

A coarse grain reconfigurable element has been developed for the mapper/interleaver and the corresponding demapper/deinterleaver functions of a wireless LAN baseband processing chain. These functions do not require concurrent execution due to the half-duplex nature of the application. The architecture of the coarse grain element follows the template described in Figure 3. The detailed architecture has been derived by first developing a high level C model of the tasks and then passing this model to the Chess/Checkers [14] package for retargetable processor design from Target Compilers Technologies.

The automatically generated by the Chess/Checkers package C-compiler is used for the generation of the micro-code for the targeted tasks. The final program (micro-code) realizing the selected tasks on the developed element consists of 170 instructions, running to 250 machine cycles. Assuming a 100MHz clock this leads to meeting the performance constraints for the processing of one OFDM symbol.

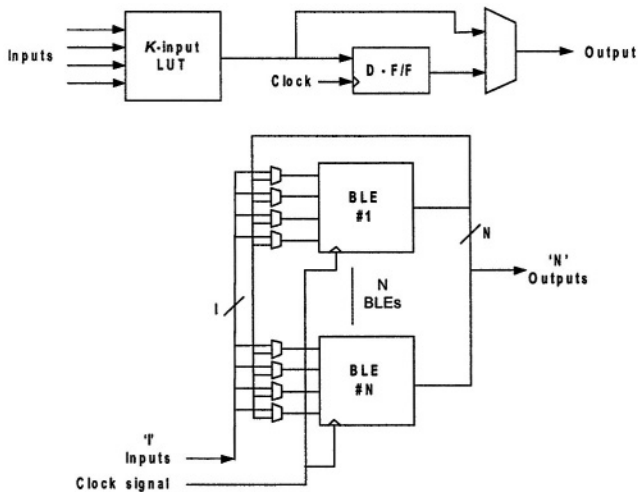
## 4 Embedded FPGA

A new embedded FPGA has been developed instead of using a commercially existing one for technology compatibility purposes with existing designs and also in order to aggressively apply optimizations for energy (which is very important in many wireless communications applications). The proposed embedded FPGA architecture is an

island-style one. The Configurable Logic Block (CLB) architecture is presented below. The CLBs interconnection scheme is not described due to lack of space.

#### 4.1 CLB Architecture

The choice of the CLB architecture is crucial to the CLB granularity, performance, density and power consumption. The CLB structure impacts the interconnect architecture and therefore the characteristics of the whole FPGA. The proposed CLB has a two-level hierarchy; it is composed of basic logic elements (BLEs) as seen in Figure 4.



**Fig. 4.** Proposed CLB Architecture: Basic Logic Element (BLE) and Cluster of BLEs

It has been proven that a LUT with four inputs, results in an FPGA with the minimum power consumption [3] and therefore this number of LUT inputs has been selected. The D-FF could play an important role in the performance, density and power consumption of the BLE. A double edge triggered FF with gated clock has been selected (Figure 5). Our simulation results with SPECTRE (Cadence) on STM 0.18u process, indicates that power consumption savings up to 77% are obtained when the FF is idle. Simulations also indicated that a gated clock signal at CLB level can save up to 83% power consumption when all FF are idle. The total energy dissipated by a BLE has been estimated to  $4.356 \times 10^{-13}$  J.

Finally, the number of BLEs in a CLB has been chosen to be 5, because it has been proven to be optimal for a number of benchmarks [3]. Additional experiments using Powermodel [3] with smaller benchmarks, confirmed this.

A tool chain has been developed to map RTL VHDL to the proposed FPGA. To our knowledge it is the most complete research flow available for HDL to FPGA mapping. It is based mostly on existing open-source academic tools.

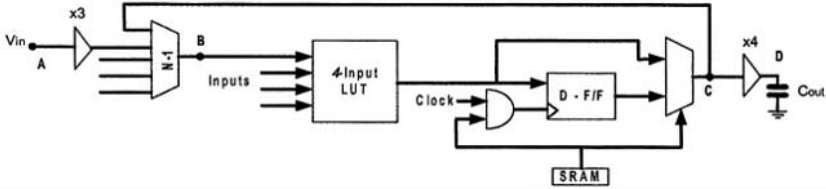


Fig. 5. Double edge triggered flip flop with gated clock

## 5 Communication Network

The central element of the proposed platform is the communication network since it allows the processing elements of the platform to communicate and to access the distributed memory organization. A packet-switched interconnect network has been selected. Networks-on-chips are to be preferred to bridged buses because they provide better throughput, better scalability, lower power consumption and allow for a structured chip wiring. In the context of dynamic reconfiguration, these networks should be flexible and efficient. Therefore they should provide a best-effort type of service, to provide a fair usage of the communication resources to the dynamically reconfigurable blocks. Guarantees on average message latency have been introduced in the network at low hardware expense by allowing the operating system to control the injection rate for every node in the network. For this purpose and also for debugging reasons the network provides visibility on the data flow to the platform's microprocessor. This control/debug path directly accesses the processing elements/router fixed interface without passing by the network. Two components are required to build the network-on-chip: a router and an interface between the router and the processing element, also called network interface component (NIC).

### 5.1 Router Design

The router is implemented as a parameterizable block, allowing for different number of input and output ports and therefore different topologies. The network is described by a set of connections and the respective routers are then parametrized automatically to fit in the selected topology. The connections between the routers can be uni- or bi-directional but between the routers and the processing blocks are by default bi-directional. The network uses virtual cut-through switching to provide the lowest latency possible in the case of non-uniform traffic as generally found in multimedia applications. The routing is deterministic or partially adaptive. A block diagram of a two-input two-output router is shown in Figure 6. More details on the router can be found in [1].

The present design is fully written in synthesisable VHDL and has been implemented to a Xilinx Virtex2Pro FPGA for test purposes. The router size as a function of the number of ports (inputs equal to outputs) is shown in Figure 7. The equivalent number of gates represents the logic and the memories used by the output buffers. The number of slices required only by the logic shows a quadratic increase with the num-

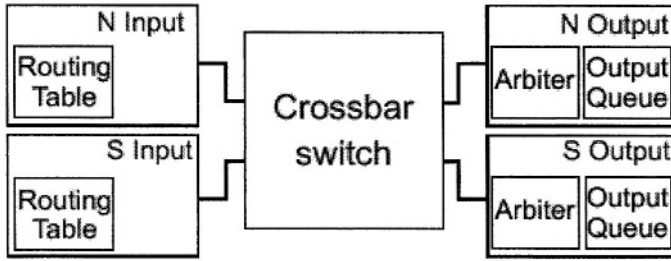


Fig. 6. Block diagram of a 2-inputs 2-outputs router

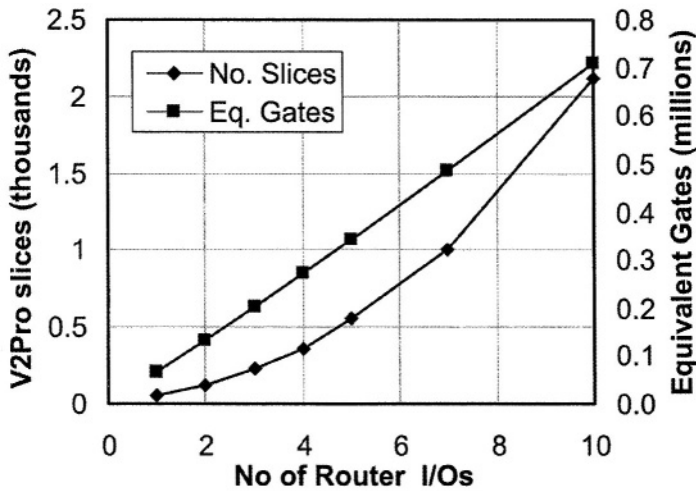


Fig. 7. Router size for an equal number of input output signals (Virtex2Pro slices)

ber of ports. The size of a router required for low dimensional topologies, excluding the buffers, is very small.

## 5.2 Network Interface Component (NIC) Design

The NIC implements several functions. First it buffers incoming and outgoing messages, and therefore isolates the processing elements from the network. Then it performs the translation from the logical destination of the messages towards the physical destination. It can also control the injection rate of the message on the network in order to provide some guarantee of service. Finally it implements some control and debug support on the processing element. To this end, the NIC is also connected to the platform's control local bus. More details on the NIC can be found in [2].

For research purposes, we implemented the NIC on an FPGA. On our target platform, the Virtex-II Pro 20, this area overhead amounts to 611 slices, or 6.58 percent of the chip per functional block.

## 6 Split Radix FFT Application Specific Reusable Block

The first application to be realized on the proposed platform is going to be a 5 GHz wireless LAN processor. A critical block of such a processor for which no run-time flexibility is required is an FFT. A split-radix FFT [8] IP core has been designed using reusable VHDL [9] so it can be implemented in application-specific hardware. The split-radix FFT algorithm has been selected because it requires fewer multiplications than conventionally used Radix-4 and Radix-2 algorithms [8].

The basic architecture of the FFT block is shown in Figure 8. It consists of the butterfly processing block, four RAM memories; two for storing the input data (real and imaginary data) and two for storing the results (real and imaginary), a ROM for storing coefficients (twiddle factors) and a control unit.

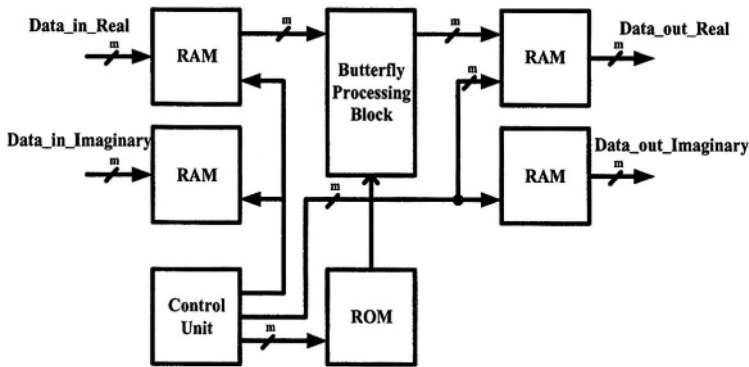


Fig. 8. Basic architecture of the FFT block

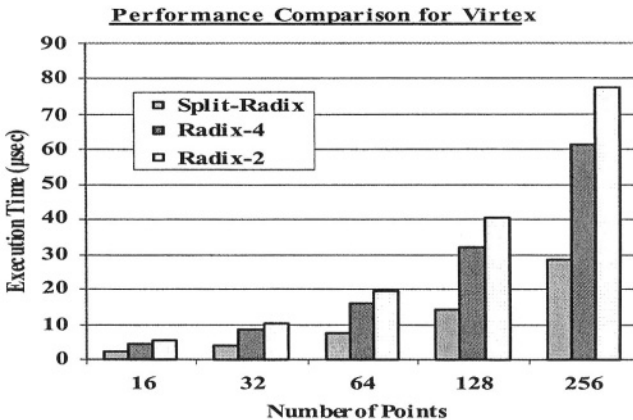


Fig. 9. Performance comparison among FFT implementations

Strictly for experimentation purposes, the split-radix IP core has been implemented on a Xilinx xc2v1000fg256 FPGA (instead of dedicated hardware). Performance (execution time) comparisons with Radix-4 and Radix-2 FFT implementations for

various numbers of points are presented in Figure 9. The developed Split Radix FFT core leads to performance improvement from 12% to 30% approximately in comparison to the Radix 4 core for realistic FFT sizes (64 to 256 points). Smaller but still important improvements are also obtained in terms of area and energy.

## 7 Conclusions

A mixed granularity reconfigurable System-on-Chip platform for wireless communications has been developed. The platform includes architecture template, hardware building blocks and design methods/tools. The hardware building blocks of the platform have been presented in this paper and include: a) Coarse grain reconfigurable blocks. b) Embedded FPGA. c) On-chip communication network. d) Application specific blocks. The realization of a complex dual mode wireless networking system (both MAC and physical layer functionalities are targeted) on a platform instance is currently on going. The reconfigurable SoC will be realized on 0.18  $\mu\text{m}$  STM technology.

**Acknowledgement.** This work was also supported by the EU through the IST project AMDREL, IST – 2001 0 34379.

## Disclaimer

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

## References

1. T.A. Bartic et al. "Highly Scalable Network on Chip for Reconfigurable Systems", International Symposium on System-on-Chip (SoC), Tampere, Finland, November 2003.
2. T. Marescaux et al. "Networks on Chip as Hardware Components of an OS for Reconfigurable Systems", Intl. Conf. on Field Programmable Logic and Applications (FPL), Lisbon, Portugal, September 2003.
3. Kara K.W. Poon: Power Estimation for Field-Programmable Gate Arrays. Master of Applied Science Dissertation, University of British Columbia (2002)
4. <http://www.mentor.com/leonardospectrum/datasheet.pdf>
5. [http://www.cbl.ncsu.edu/pub/Benchmark\\_dirs/LGSynth93/src/e2fmt/](http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LGSynth93/src/e2fmt/)
6. M. Sentovich et al.: SIS: A System for Sequential Circuit Synthesis. UCB/ERL M92/41 (1992)
7. V. Betz and J. Rose: VPR: A New Packing, Placement and Routing Tool for FPGA Research. in Proc. of FPL '97, London, UK (1997) 213-222.
8. P. Duhamel and H. Hollomann, "Split Radix FFT Algorithm", Electronic Letters, vol. 20, Jan. 1984, pp. 14-16.

9. P. Bricaud and M. Keating, "Reuse Methodology Manual for System-on-a-chip Designs", Kluwer Academic, Publishers, Boston (1999)
10. <http://vlsi.ee.duth.gr/amdrel/>
11. Tondelayo™ 1 Baseband Product Brief. Systemonic, Inc. 100 Century Center Court, Suite 205 San Jose, CA 95112 USA.
12. Thomson, J., et al. "An integrated 802.11a baseband and MAC processor". Solid-State Circuits Conference, 2002. Digest of Technical Papers. Volume: 2, 2002, pp. 92 – 93 and 414 - 415.
13. J. Becker, et. al, DReAM: Dynamically Reconfigurable Architecture for future Mobile Communication applications, FPL 2000, Villach Austria, August 2000.
14. <http://www.retarget.com/brfchschk.html>

# Enhancing GALS Processor Performance Using Data Classification Based on Data Latency

S. López<sup>1</sup>, O. Garnica<sup>1</sup>, and J.M. Colmenar<sup>2</sup>

<sup>1</sup> Departamento de Arquitectura de Computadores y Automática  
Universidad Complutense de Madrid, Spain  
{slopezal, ogarnica}@dacya.ucm.es

<sup>2</sup> C.E.S. Felipe II (Universidad Complutense de Madrid), Spain  
jmcolmenar@cesfelipesecondo.com

**Abstract.** This paper proposes a new approach for improving the performance of Globally Asynchronous Locally Synchronous (GALS) circuits. This approach takes advantage of the delay dependence of the input vectors to classify input data into several classes. Each class has a clock period associated, in such a way that a suitable clock is selected for each data. This technique has been applied to a GALS pipelined RISC processor based on DLX processor. Several programs were run over this processor performing different classifications, in order to check the viability of this new approach.

## 1 Introduction

Considering the number of transistors and clock frequencies in present digital systems, synchronization with a single clock source and negligible skew is extremely difficult, if not impossible. For this reason, alternative design approaches to synchronous design should be researched [2].

A natural approach to this problem is design multiple clock domains, globally-asynchronous, locally synchronous (GALS) systems [8]. GALS systems contain several independent synchronous blocks which operate with their own local clocks (possibly running at different frequencies) with asynchronous communication with each other. These systems do not have a global timing reference.

On the other hand, fully asynchronous design built using self timed circuits, replace the clock signal by local synchronization protocols. Some of their advantages are that they have no problems associated with the clock signal and circuit performance is the performance of the average case because a new computation can start immediately after the previous has finished [9]. Nevertheless, fully asynchronous design requires a considerable effort because of the absence of strong enough CAD tools that make the design feasible.

However, in terms of performance, fully asynchronous systems offer better results than GALS systems. In the first ones the output delay is highly dependent on the input pattern. Some input patterns will take significantly less time than others and just a few will take its critical path delay, the delay of the synchronous case. Hence, fully asynchronous performance is the performance of



the average case, as stated above. On the other hand, in GALS systems each synchronous block works with the critical path delay. In other words, every input pattern generates its correct output after the critical path delay because, locally, the system works as a synchronous design. Then, the performance for these synchronous blocks is the worst possible for each one. If these blocks had been implemented using fully asynchronous approach, just some of the input patterns would take the critical path delay and others would take less time.

Our work takes advantage of the input pattern delay dependence to modify the GALS approach. In our approach, input data are classified into several classes depending on its computation delay. We will see that these classification implies a very low overhead. Each class has a clock period associated, so that the most suitable clock period is selected depending on the class the input data belongs to. Our goal is to apply this methodology to design an asynchronous processor modifying a GALS processor into a GALS processor with Data Classification based on Data Latencies.

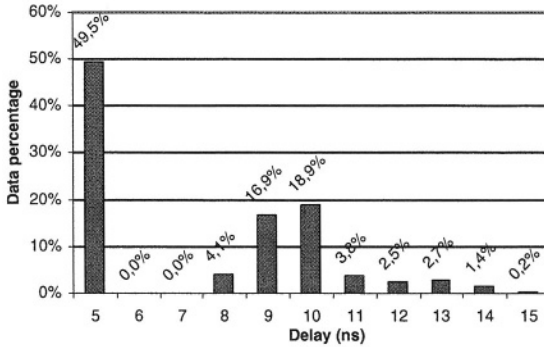
The rest of this paper is organized as follows. In Section 2 we explain our methodology design. Section 3 is devoted to explain a modified GALS processor applying Data classification based on Latency. In section 4 we present experimental results. Finally, in Section 5 we present conclusions and future work.

## 2 Data Classification Based on Data Latency (DCDL)

In previous work [10], several benchmarks were implemented as fully asynchronous circuits using the Delay Insensitive (DI) model. We run simulations using 1000 random input patterns over these circuits and we focused on the delay of every input. Fig. 1 illustrates the behavior of one of these circuits (apex2 benchmark from LGSynth95). This graph represents the percentage of the simulated inputs whose completion requires a given delay. Nearly 50% of the computations take just 5 ns. However, the worst case delay is 15 ns and this is the mandatory delay for the synchronous approach. Hence, half of the tested computations could be three times faster than the worst ones. In a fully asynchronous approach, the circuit average latency would be 7.5 ns, provided that all inputs were equiprobable, versus 15 ns average (and worst) latency of synchronous approach. However, as stated in section 1, the fully asynchronous approach requires a great design effort. Moreover, Delay Insensitive approach needs dual rail codification [6] and this fact increases power consumption.

In order to keep a good performance skipping the problems related to fully asynchronous approach, we propose a synchronous implementation with input data preclassification into classes attending to the computing delay of the logic. Each class will have an associated clock period. The period will be the most suitable for the inputs in the class so that, it will be taken the lower period available but large enough to finish the computation. Once a small logic has decided what class the input pattern belongs to, the associated clock is activated.

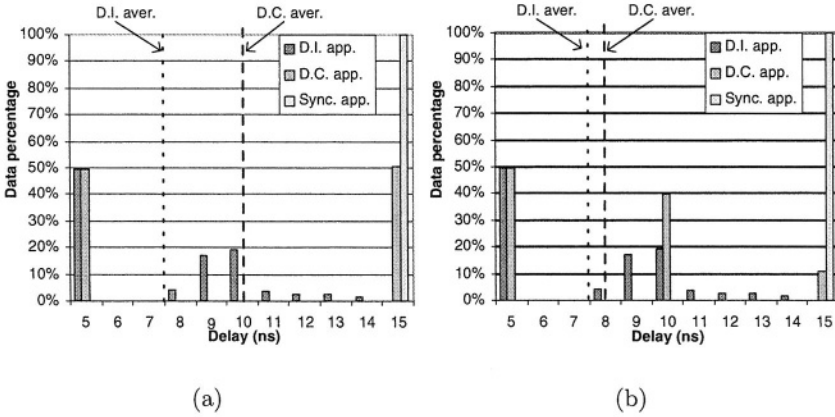
Going ahead with our example, we have classified inputs patterns into several classes. Fig. 2(a) represents results with two classes with their own clock



**Fig. 1.** Delay distribution of benchmark apex2.

period associated and Fig. 2(b) represents results with three classes with their own clock period. In both Fig. 2(a) and Fig. 2(b) the graph called D.I. app. represents the same that in Fig. 1. Also, graph called Sync. app. represents synchronous behavior: 100% of the input vectors take the worst time possible. In Fig. 2(a), graph D.C app. shows results of the DCDL approach where two classes have been defined. Since two classes have been defined, we also define two clocks. One of the periods is 5 ns (clock 1) and the other is 15 ns (clock 2). The circuit is implemented as a synchronous circuit but with a small logic added which decides if an input pattern computation will take 5 ns (class 1) or more than 5 ns (class 2). Those input patterns belonging to class 1 work with clock 1 (5 ns) and those belonging to class 2 work with clock 2 (15 ns). This way, in 1000 random equiprobable input patterns run over apex2 circuit, almost 50% of the computations take 5 ns and other 50% take 15 ns. The average performance with the DCDL approach (Fig. 2(b), dotted lines) is 10 ns versus 15 ns of synchronous approach and 7.5 ns of Delay Insensitive (DI) model (fully asynchronous approach). Average performance can be tuned adding more classes with different periods or varying the periods of the existing classes. Thus, we add a third class for those inputs which computation delay takes more than 5 ns and less than 10 ns (class 3)(see Fig. 2(b)). This class works with clock 3 (10 ns). With this classification, the average performance is 8 ns, versus the fully asynchronous approach whose delay is 7.5 ns (Fig. 2(b), dotted lines). So, the more classes are added, the more the performance comes close to the fully asynchronous performance. But the logic needed to all these classifications rises and, consequently, the power consumption increases too. Hence, a trade off is needed in the classification.

The clock period in the synchronous approach is obtained by conventional Static Timing Analysis (STA) that handles variability by analyzing a circuit at multiple process corners, and it is generally accepted that such an approach is inadequate for being overly pessimistic. An alternative approach is statistical static timing analysis (SSTA) [3], which treats delays not as fixed numbers, but as probability density functions (PDFs), taking the statistical distribution of parametric variations into consideration analyzing the circuit. There are other tech-



**Fig. 2.** Delay distribution of benchmark apex2 implemented as Delay Insensitive (D.I. app.), synchronous (Sync. app.) and with DCDL approach (D.C. app.). Fig. (a) presents a classification in 2 classes and (b) a classification in 3 classes.

niques, also based in PDFs, to estimate performance in asynchronous pipelines [4]. Our approach selects the appropriate time behavior not based in statistical analysis, but in deterministic analysis. DCDL finds patterns into the input with the same delay behavior and uses logic for the patterns detection.

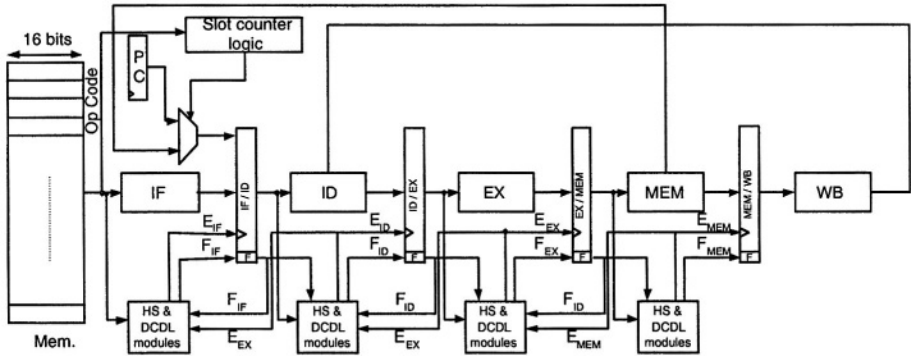
The main problem to be solved in DCDL approach is to find out the patterns that characterize the latency behavior of the input vectors. This problem can be solved in a pipelined processor whose input vectors entering each stage are well known and contain information about the instruction and the terms of operations. For example, in execute stage we can use a criterion that differentiate NOP instructions that will take less time than other instructions since no computation is done, in this stage, for this kind of instructions.

In next sections we will explain a RISC GALS pipelined processor implementation and its modification using DCDL.

### 3 Data Classification Applied to a RISC GALS Processor

We have implemented a GALS pipelined RISC processor based on a the DLX processor using VHDL. In [1] it was proposed a similar GALS DLX microprocessor implemented using Verilog.

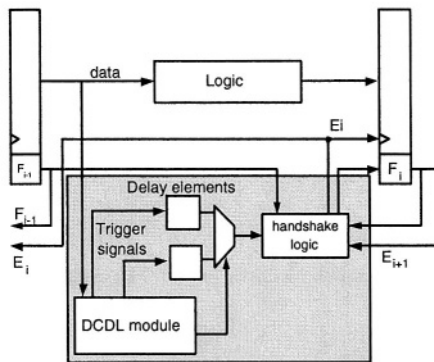
Fig. 3 shows a GALS RISC pipelined processor modified adding Data Classification approach. Fig. 3 illustrates the basic diagram of the pipeline. It posses five stages: Instruction Fetch (IF), Instruction Decode(ID), Execute (EX), Memory Access (MEM) and Write Back (WB). It also shows a HS&DC and a Slot Counter modules that we will explain in next subsection.



**Fig. 3.** Basic diagram of GALS RISC Pipelined Processor. The pipeline has 5 stages: Instruction Fetch (IF), Instruction decode (ID), Execute (EX) Memory Access (MEM) and Write Back (WB). It has a Slot Counter Logic to ensure the correct behavior in branch instructions case and the handshake and Data Classification logic (HS & DC ) (see section 3.1)

### 3.1 Handshake Protocol

Both in usual GALS processor and modified DCDL GALS Processor, different clock periods govern each stage. Therefore, it is required a handshake protocol to communicate stages. Fig. 4 shows a detailed view of the handshake logic and DCDL module (HS&DC module in Fig. 3). Let’s focus on the handshake logic.



**Fig. 4.** Detailed view of Handshake protocol and DCDL module.

Handshake protocol is based on the state of the registers between the stages. Hence, the registers can have *full* or *empty* state. Let’s focus in the ID/EX register (see Fig. 3). When this register stores a new data, this data is processed by the EX stage and, after  $t_{EX}$  time, the result is stored in EX/MEM register, provided that this register is *empty*. At this time, data in ID/EX register has

been *consumed* by the EX stage and register EX/MEM is *full*. ID/EX register is ready to store a new data so, this register is *empty*.

Thus, we have added two new signals, *Enable*,  $E$  signal and *Full*,  $F$  signal.  $E$  rising edge governs data storage in the registers.  $F$  is a flag which indicates if the register is *full* ( $F=1$ ) or *empty* ( $F=0$ ).

Since communication between stages is asynchronous, this handshake protocol must be implemented with a speed independent circuit, without conflicts in its Signal Transition Graph (STG). In order to ensure this, the protocol has been modelled as a Signal Transition Graph (STG) using Petrify tool [5]. This tool uses STGs as a formalism to specify the behavior of the circuits and returns the logic equations that represent the behavior of the circuit (see Eq. 1 and 2):

$$F_i = \overline{E}_{i+1}F_i + E_i \quad (1)$$

$$E_i = \overline{E}_{i+1}\overline{F}_iF_{i-1} \quad (2)$$

Subindex  $i$  refers to signals of the  $i$ th register. From Eq. 1 and 2, the *full* or *empty* state ( $F_i$  flag) and the value of Enable signal ( $E_i$ ) depends on the previous and next registers state. All these signals appear in Fig. 4.

This handshake protocol ensures communication with the previous and next stages. But in case of branch instructions, correct communication must be solved using a new logic block. As we see in Fig. 3, this pipeline includes a *slot counter logic*. The pipeline structure requires that three new instructions had entered in the pipeline before taking the branch. These instructions are required to update the registers before the branch is taken. So, the *slot counter logic* keeps the decision about the branch (probably solved before the incoming of the third instruction) till those three instructions have entered in the pipeline. In the synchronous case, this logic is not necessary because the branch decision is always taken when three instructions have entered in the pipeline. But in our case, this is not compulsory.

As Fig. 3 shows, Write Back (WB) stage does not have a register behind, so it does not have a clock associated. This stage just decides when data must be stored in the register file of Instruction Decode (ID) stage.

### 3.2 Data Classifications

DCDL logic is represented in Fig. 4 as well. Each stage has several delay elements. Data Classification based on Data Latency (*DCDL module*) decides what class the input data belongs to and, consequently, it chooses the delay the stage will use for this data. When the DCDL module makes the decision, the trigger signal rises and the delay element delays the signal rising the necessary time. Delay elements have been used in other works as well [12].

Several classifications can be done in this pipelined processor. We have chosen two simple classifications in order to probe the viability of this approach:

**Classification 1: NOP and no memory access class.** NOP instructions are empty instructions provided by the compiler to avoid control hazards. Almost 20% of the branch delay slots are filled with NOPs instructions in usual compilers [7]. This kind of instructions do not need any operation in execution (EX) stage and do not need any memory access in the memory (MEM) stage. Other instructions (arithmetic and branch instructions) do not have memory access in the memory (MEM) stage either. But in DLX processors every instruction has to be processed by every stage to avoid structural risks. In other words, no stages can be avoided. For these two kinds of instructions, a short clock period can be used while the usual stage clock (a longer one) can be used for other instructions.

Hence, in execution (EX) stage we define *class1* (NOP instructions) associated with *clock1* (short clock period) and *class2* (other instructions) associated with *clock2* (long clock period). The Execute (EX) stage distributed control (EXControl signal) contains information about the operations that must be executed in this stage. This signal is stored in ID/EX register and it provides EXControl=0000 when it has a *NOP instruction* to be processed and other values when others. Therefore, just four bits need to be driven into the data classification logic and a four inputs NAND gate is enough to achieve this classification.

In memory (MEM) stage we define *class1* (NOP instruction and no memory access instruction) associated with *clock1* (short clock period) and *class2* (other instructions) associated with *clock2* (long clock period). In usual programs memory access, load/store instructions, represents from 25% to 45% of the program instructions. In EX/MEM register, distributed control of this stage, provides MEMControl=000 when it has a *NOP instruction* or a *no memory access instruction* to be processed and other values when others. So, just three bits need to be driven into the data classification logic and a three inputs NAND gate is required to achieve this classification.

Applying this classification to other stages is pointless because they work in the same way for all instruction types and, consequently, they do not have significant differences in time.

**Classification 2: Multiplication Class.** In our pipelined processor implementation, multiplication function unit is not pipelined. Therefore, multiplications require more time than additions, so in the execution (EX) stage we can have two classes: *class1*, arithmetic instructions carry out multiplications with *clock1* (long period) and *class2*, the rest of the instructions with *clock2* associated (short period). Multiplication is coded in our pipeline with func=100001 and just this pattern must be identified.

## 4 Experimental Results

In order to obtain accurate results in simulations, we have made some assumptions about each stage delay. Table 1 shows some approximate values of RT level components delay [11]:

**Table 1.** RTL components delay.

Unit	Delay(ns)
Adder	15
Multiplier	24
Register	3
Local Memory	9

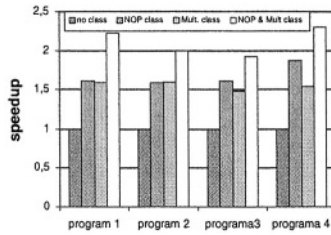
With these values we can assume the delays for each stage showed in Table 2. We can see that in execute (EX) stage we can obtain a time saving both with NOP and no memory access instruction classification and multiplication instructions classification. In the memory (MEM) stage we can save time with NOP and no memory access instruction.

**Table 2.** Stages delay. Second column shows clocks values of usual GALS pipeline (without any data classification). Third column shows local clock values for NOP or no memory access instructions when classification 1 is applied. Fourth column shows local clocks values in no multiplication instructions when classification 2 is applied. We show in bold the values that save time.

Stage	No Class. Delay(ns)	Class 1. Ins. Delay	Class 2. Ins. Delay
IF	9	9	9
ID	3	3	3
EX	24	<b>3</b>	<b>15</b>
MEM	9	<b>3</b>	9

In Fig. 5 we show some results obtained after running 4 different programs in four different cases. First, we have results for a usual GALS pipeline, with no classification (called *no class*). Second, we have obtained results for NOP and no memory access case classification (*NOP class*). Third, with Multiplication classification (*Mult. class*). And finally using both NOP and no memory access classification and Multiplication classification (*NOP & Mult. class*). The graph represents each classification improvement in terms of speedup, versus the absence of classification, usual GALS approach. We get improvements in all programs and them ranges from 1.5 to 2.3 times the usual GALS pipeline performance. Best results are obtained in case of two simultaneous classifications because more instructions are involved in classification. Applying two classifications implies that more logic is needed. But, as we have seen, in the classifications we have presented the logic is quite small.

In a pipeline with data classification the performance improvement is conditioned by percentage of classified instructions versus the non-classified instructions during the program execution and the amount of time that this classified instructions can save. So, other classifications can be applied in this pipeline. For example, programs usually have a high percentage of additions having one operand equal to 1. This additions take less time than other additions so they



**Fig. 5.** Speedup for four different programs applying Classification 1 (NOP and mem. Class.), Classification 2 (Mult. Class.) and both.

can be classified in a new clock period class. Multiplications by small values take less time well so they can be classified too.

So, the next step in Data Classification based on Latency is to classify the instruction operands. The time saved with this kind of classifications depends on the functional units implementation, so this classifications require to study the problem in depth in a future work.

## 5 Conclusions and Future Work

In this paper we have proposed an enhanced pipelined GALS processor implementation using Data Classification based on Data Latency (DCDL) approach. This new approach improves the GALS pipeline performance using two (or more) clocks in the stages where data classification is applied. Input vectors are classified depending on the delay that logic requires to compute them and each class has a suitable associated clock.

In order to check the viability of this new approach, we have implemented a GALS pipelined processor based on a DLX processor and we have applied different classifications. We have run four simple programs with good results, achieving an enhanced performance up to 2.3 times the usual GALS performance when two simultaneous classifications are applied. Hence, we can state that more classifications provide best performance.

The work we present in this paper is the first part of a larger work in which we will explore the possibilities of Data Classification based in Data Latency (DCDL). Currently, we are studying several benchmark using SimpleScalar in order to obtain statistical results about the most frequent instructions and operands to apply realistic and effective classifications. We are, also, analyzing several functional units implementation to apply this approach in operands classification. We are designing a dynamically scheduled processor in order to apply data classification on a high performance processor.

**Acknowledgment.** This research has been supported by Spanish Government Grant number TIC 2002/750.



## References

1. Manish Amde, Ivan Blunno, and Christos P. Sotiriou. Automating the design of an asynchronous DLX microprocessor. In *Proc. ACM/IEEE Design Automation Conference*, pages 502–507, June 2003.
2. Luca Benini and Giovanni De Micheli. Networks on chips: A new soc paradigm. *Computer*, 35(1):70–78, 2002.
3. M. Berkelaar. Statistical delay calculation, a linear time method. In *Proc. International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, December 1997.
4. J.M. Colmenar, O. Garnica, S. Lopez, J.I. Hidalgo, J. Lanchares, and R. Hermida. Empirical characterization of latency of long asynchronous pipelines with data-dependent module delays. In *Proc. EUROMICRO Conference on Parallel, Distributed and network-based Processing*, pages 112–119, 2004.
5. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. Technical report, Universitat Politècnica de Catalunya, 1996.
6. Scott Hauck. Asynchronous design methodologies: An overview. *Proceedings of the IEEE*, 83(1):69–93, January 1995.
7. John L. Hennessy and David A. Patterson. *Computer architecture (2nd ed.): a quantitative approach*. Morgan Kaufmann Publishers Inc., 1996.
8. Anoop Iyer and Diana Marculescu. Power and performance evaluation of globally asynchronous locally synchronous processors. In *Proceedings of the 29th annual international symposium on Computer architecture*, pages 158–168. IEEE Computer Society, 2002.
9. D. Kearney. Theoretical limits on the data dependent performance of asynchronous circuits. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 201–207, April 1999.
10. Sonia López, Óscar Garnica, Ignacio Hidalgo, Juan Lanchares, and Román Hermida. Power-consumption reduction in asynchronous circuits using delay path un-equalization. In Jorge Juan Chico and Enrico Macii, editors, *Power and Timing Modeling, Optimization and Simulation (PATMOS)*, volume 2799 of *Lecture Notes in Computer Science*, pages 151–160, September 2003.
11. Seong Yong Ohm, F.J. Kurdahi, and N.D. Dutt. A unified lower bound estimation technique for high-level synthesis. *IEEE Transactions on Computer-Aided Design*, May 1997.
12. Sheng Sun and Larry McMurchie and Carl Sechen. A high-performance 64-bit adder implemented in output prediction logic. In *Advanced Research in VLSI*. IEEE Press, 2001.

# Application Analysis with Integrated Identification of Complex Instructions for Configurable Processors

Nikolaos Kavvadias and Spiridon Nikolaidis

Section of Electronics and Computers, Department of Physics,  
Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece  
nkavv@skiathos.physics.auth.gr

**Abstract.** An extensible and configurable processor is a programmable platform offering the possibility to customize the instruction set and/or underlying microarchitecture. Efficient application analysis can identify the application parameters and instruction extensions that would influence processor performance. An application characterization flow is presented and demonstrated on the Wavelet/Scalar Quantization image compression application. In this context, novel application metrics are identified as the percentage cover, maximum cycle gain for each basic block and candidate-induced application speedup due to possible complex instructions. Furthermore, evaluating the instruction candidates during application analysis is proposed in order to establish a link with subsequent design space exploration steps.

## 1 Introduction

Embedded processors suitable for consumer applications, present interesting architectural refinements, in order to support power-hungry algorithms e.g. for high bandwidth wireless communications or video compression and decompression [1]. The portability of these systems makes energy consumption a critical design concern. For successfully implementing software applications on domain-specific processors under tight time-to-market constraints, requirements of high flexibility and programmability have also to be met.

The challenge of delivering the optimum balance between efficiency and flexibility can be met with the utilization of customizable processors. Most commercial offerings fall in the category of configurable and extensible processors [2],[3]. *Configurability* lies in either a) setting the configuration record for the core (regarding different cache sizes, multiplier throughput and technology specific module generation) or b) allowing modifications on the original microarchitecture template. In the first case, the end user selects the synthesis-time values for certain parameters of the processor core [4]. The second case requires that the basic architecture of the core is modifiable. For instance, the flexible pipeline stage model employing local control in [5] enables altering the pipeline depth of the processor. *Extensibility* of a processor comes in modifying the instruction set architecture by adding single-, multi-cycle or pipelined versions of complex instructions. This may require the introduction of custom units to the execution stage of the processor pipeline and this should be accounted in the architecture template of the processor. The instruction extensions are generated either

automatically or manually from a self-contained representation of the application code, assuming a structural and instruction set model of the processor [6].

Characterizing the application workload is a fundamental step in microprocessor design, since based on this analysis, the processor designer can decide the appropriate configuration and the required instruction extensions of a customizable core for achieving an advantageous performance-flexibility tradeoff. In this paper, an approach to application analysis is presented for extracting application parameters. The framework is based on the freely available SUIF/Machine SUIF (MachSUIF) compiler infrastructure [7]. Opposed to previous approaches, complex instruction candidates are identified at the stage of application analysis, since such information can be used for pruning the design space of possible instructions in an Application-Specific Instruction set Processor (ASIP) design flow. Static and dynamic characteristics of the application are also extracted and their impact on candidate identification is investigated. The metrics of percentage cover, maximum basic block cycle gain and candidate-induced application speedup that quantify the impact of including specific complex instructions are given. Overall, it is argued that generating an initial set of instruction candidates should be an integrated step of the application characterization flow to guide subsequent design space exploration steps.

The rest of this paper is organized as follows. The related work in application analysis and candidate instruction (template) identification is summarized in Section 2. Each step of the application characterization flow is described in Section 3 along with the use of existing and the associated in-house tools we have developed. Section 4 discusses the application of the proposed approach on the Wavelet/Scalar Quantization (WSQ) image compression algorithm and the corresponding results. Finally, Section 5 summarizes the paper.

## 2 Related Work

An important issue in domain-specific processor design is the task of application analysis extracting both static and dynamic metrics for the examined applications. Although important in ASIP synthesis, the effect of introducing candidate instructions to accelerate processor performance on a given application set is not adequately examined in context of application analysis in the vast majority of related work.

In [8] both the application and a specification of the processor are input to an estimation framework based on SUIF. A number of parameters characterizing the application are extracted: the average basic block size, number of multiply-accumulate operations, ratio of address to data computation instructions, ratio of I/O to total instructions, register liveness, and degree of instruction-level parallelism. Compared to [8], our approach searches for all candidate instructions by identifying fully-connected subgraphs in the DFG of each basic block, instead of restricting the search to a specific complex instruction type. Also, their tool has been designed for processor selection and not to assist ASIP synthesis, which explains the fact of using coarse parameters extracted from the instruction mix. These are intended as thresholds for selecting or rejecting a specific processor while our method performs the analysis in a much finer level.

A performance estimator using a parameterized architecture model has been developed in [9]. While the work presented is significant, the method has been constructed

with a specific processor type in mind. E.g. the assumed addressing modes are specific to DSP processors. Our method can identify non-DSP specific complex addressing schemes, as shifter-based addressing modes similar to those of the ARM7 processor.

Multimedia benchmark suites have been presented in [10],[11] along with their characterization profile. In [11] the popular MediaBench suite is introduced, characterized with metrics suitable for general-purpose processors. The benchmark suite in [10] is comprehensive with a thorough study, however also assuming a GPP template. Again, guidelines to finding the appropriate extension instructions suitable to multimedia-enhanced GPPs are not provided.

### 3 The Proposed Approach for Application Analysis and Characterization

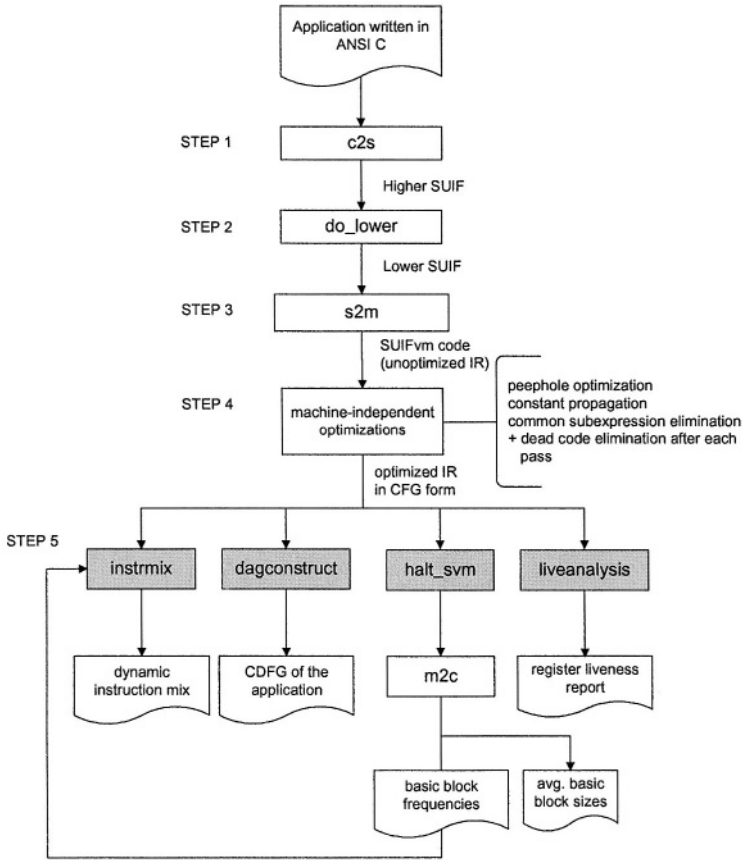
It is often at early stages in processor design, that the compilers and simulators for the entire range of applicable processor architectures one needs to consider, are not available [8]. In order for the application characterization results to be useful to the spectrum of evaluated microarchitectures, a common estimation platform is required. We propose using the MachSUIF intermediate representation (IR) for this purpose. In MachSUIF, the IR description uses the SUIF virtual machine instruction set (SUIFvm), which assumes that the underlying machine is a generic RISC, not biased towards any existing architecture.

In this case, the application is decomposed into its IR consisting of operations with minimal complexity, best known as primitive or atomic instructions. It is possible to organize the IR description of the application into Control Data Flow Graphs (CDFGs) with primitive instructions as nodes and edges denoting control and data dependencies. In MachSUIF, the executability of the original C program is retained at the level of the SUIFvm instruction set. The corresponding SUIFvm code consists the executable intermediate representation [12] of the benchmarked program. Dynamic characterization can be performed on the host machine by executing the resulting C program, generated by translating the SUIFvm back to C code.

The proposed application characterization flow is shown in Figure 1. Shaded blocks on the diagram distinguish our in-house tools from the available passes of the infrastructure. In the remaining paragraphs of this section, we detail the steps of the application characterization procedure.

In Step 1, the input C code for the application is passed through the *c2s* pass of the SUIF frontend. In this stage, the application is preprocessed and its SUIF representation is emitted. In the second step, *do\_lower*, several machine-independent transformations are performed as dismantling of loop and conditional statements to low-level operations. The resulting description is termed as lower SUIF in contrast to higher SUIF generated by *c2s*. Step 3 is needed to translate the lower SUIF code into the SUIFvm representation. For this task, the *s2m* compiler pass is used.

The IR code has not been scheduled and has not passed through register allocation, which is important so that false dependencies within the data flow graphs of each basic block are not created [13]. Step 4 performs architecture-independent optimizations on the IR, such as a) peephole optimization, b) constant propagation, c) dead



**Fig. 1.** The proposed application analysis and characterization flow

code elimination, and if decided, d) common subexpression elimination (CSE) to construct the optimized SUIFvm description.

Peephole optimization suppresses redundant move operations and is used to remove unnecessary type casting (CVT) operations that MachSUIF has the tendency to produce after the application of an active pass. The usefulness of CSE depends on the algorithm applied for complex instruction generation. Specifically, if overlapped templates are permitted during instruction generation, CSE will not prohibit the identification of any beneficial candidate. However, if a faster algorithm is used that only allows orthogonal covers, some opportunities will be missed. Assume a basic block with two instances of the same subexpression, e.g. a subgraph comprised of two primitive operations, placed in the core of two different single-cycle complex instructions consisting of three and four primitives respectively. If the second subexpression is eliminated, then the second instruction candidate could only consist of two primitives.

Finally, during Step 5, specific static and dynamic metrics are gathered. The corresponding analysis passes accept SUIFvm in CFG form.

The *dagconstruct* pass parses each node in the CFG and constructs the corresponding CDFG. Note that instructions involving memory operands (as in CISC-like machines) require additions to some libraries of the infrastructure. In this case, the *dagconstruct* pass should be updated to reflect these changes introduced to the *suiifvm* library.

A pass for generating the static instruction mix, *instrmix*, has also been developed. By using the execution frequencies for the basic blocks of the application, the dynamic instruction mix can be easily calculated. For calculating the execution frequencies, the SUIFvm code is translated to single-assignment style C using the *m2c* pass. Pass *halt\_svm* is used to instrument the C code by adding counters at the start of each basic block.

The *liveanalysis* pass is based on the *cfa* library and calculates the number and names of registers that are alive at basic block boundaries. The corresponding results help the designer decide the register file size.

## 4 Application Analysis for the Wavelet/Scalar Quantization Image Compression Algorithm

The case study application is based on a wavelet image compression algorithm [14] and is part of the Adaptive Computing Benchmarks [15], which are used to evaluate specific characteristics of reconfigurable architectures. Reportedly, the selected benchmark is used to stress reconfigurability by splitting execution time among several kernels. A compliant implementation of the WSQ algorithm is required to perform four standard steps: wavelet transform, quantization, run-length encoding and entropy coding (for the encoder part). The entropy encoding stage is realized with a Huffman encoder. In our paper, the application analysis framework is used to extract characteristics for both the encoding and decoding algorithms.

### 4.1 Instruction Mix

The dynamic instruction mix provides a classification of the application instructions into types based on the functional units used. Instructions are divided into integer and floating-point, while each of those has distinct subtypes: load and store, arithmetic, logical, shift, multiply, division, unconditional and conditional branch, call/return and remaining instructions. Figure 2 shows the instruction mix statistics for the *compress* and *decompress* applications, which correspond to the WSQ encoder and decoder, respectively. Note that WSQ is a pure integer application.

It is clear that arithmetic operations dominate the instruction mix of the applications. Also, *decompress* has higher computational complexity than *compress* since it requires higher amount of arithmetic and load instructions. The ratio of branches to the total instructions is very small (9.8%) which means that higher execution frequencies are encountered for relatively large basic blocks. This conclusion is supported by the results of Section 4.4.

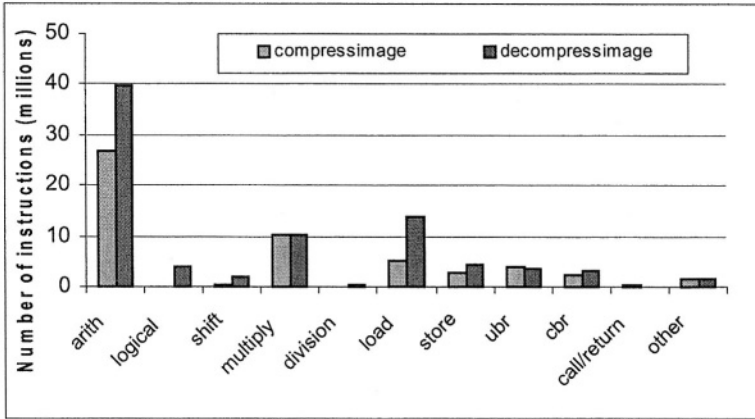


Fig. 2. Instruction mix statistics for the WSQ algorithm

## 4.2 Average Basic Block Size

The basic block sizes are easily calculated simultaneously to the static instruction mix. It is found that 3.98 and 4.48 instructions consist the average basic block for the *compress* and *decompress* applications, respectively. At a first glance, this result does not leave much room for performance benefits by exploiting complex instruction candidates within the same basic block. However, as it will be shown, heavily executed portions of the code comprise of rather large basic blocks.

## 4.3 Register Liveness Analysis

It is found that *decompress* has lower register pressure with a maximum of 8 saved registers while *compress* requires 11 saved registers. These results constitute a lower bound on the required local storage resources, more specifically the number of allocable registers of the architecture, for the WSQ algorithm.

## 4.4 Basic Block Frequencies

Figure 3 indicates the execution frequencies and sizes for the most heavily executed basic blocks for *compress* and *decompress*. Each basic block is assigned a unique name of the form: `<file_name>.<function_name>.<basic_block_number>`.

It is evident from Figure 3 that there exists space for achieving speedup in the performance critical basic blocks since their size is significantly above average.

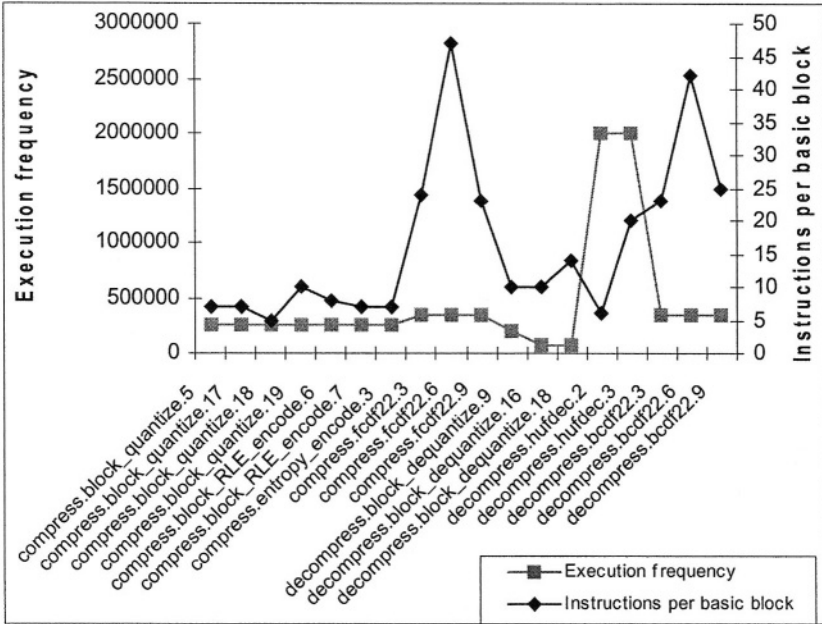


Fig. 3. Execution frequencies and sizes for the heavily executed basic blocks

### 4.5 Data Flow Graph Analysis for Identifying Candidate Instruction Extensions

The *dagconstruct* pass referred in the beginning of this section, generates DFGs for each basic block. Then, fully-connected subgraphs of these DFGs are identified as potential complex instructions. A measure of success for the selection of complex instructions using orthogonal covers is given by the *percentage cover factor* determined by the proportion of the number of instructions after selection to the number of instructions prior selection. A speedup factor, *maximum basic block cycle gain*, is also introduced and is calculated as the product of the maximum performance gain in cycles (assuming no data hazards and spills to memory) with the execution frequency of the specified basic block. An estimate on the performance impact of selecting a set of isomorphic patterns is given by the *candidate-induced application speedup* metric defined as the application speedup due to selecting the complex instruction. At this point, calculating the latter metric is not automated, and for this reason it is evaluated on the performance-critical basic blocks of the application. Since only 10 basic blocks incorporate the 85.3% of the instructions for *compress* and 8 basic blocks the 97.3% of the instructions for *decompress*, the extracted results are valid.

Table 1 shows the percentage cover factor and maximum cycle gain for the performance-critical basic blocks. In columns 2 and 3, the number of instructions prior and after complex instruction matching is given. The percentage cover and maximum gain values are given in columns 4 and 5 respectively. The average percentage cover is 83.2%.



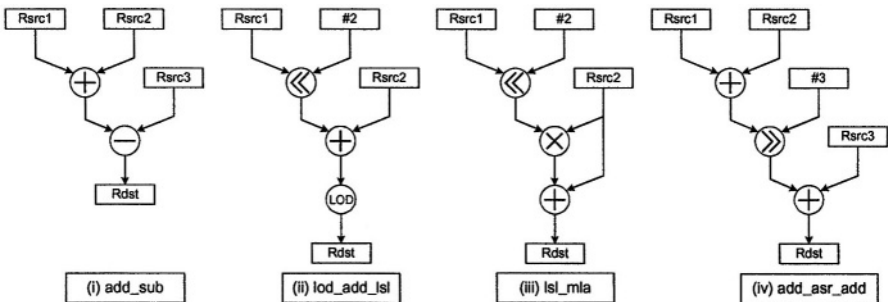
**Table 1.** Template selection results for the performance-critical basic blocks

Basic block ID	# Instr. (prior select.)	# Instr. (after select.)	% cover	Maximum cycle gain
compress.block_quantize.5	7	3	85.7	1048576
compress.block_quantize.17	7	3	85.7	1048576
compress.block_quantize.18	5	2	80.0	786432
compress.block_quantize.19	10	3	70.0	1806336
compress.block_RLE_encode.6	8	3	100.0	1290285
compress.block_RLE_encode.7	7	5	57.1	516096
compress.entropy_encode.3	7	4	57.1	786438
compress.fcdf22.3	24	9	87.5	5160960
compress.fcdf22.6	47	15	91.5	10895360
compress.fcdf22.9	23	9	82.6	4816896
decompress.block_dequantize.9	10	5	70.0	983040
decompress.block_dequantize.16	10	4	100.0	393216
decompress.block_dequantize.18	14	4	100.0	655360
decompress.hufdec.2	6	2	100.0	8000004
decompress.hufdec.3	20	11	70.0	18000000
decompress.bcdf22.3	23	9	82.6	4816896
decompress.bcdf22.6	42	13	92.9	9873920
decompress.bcdf22.9	25	10	84.0	5160960

In Table 2, candidate-induced application speedups are given in columns 2, 3 for the 23 unique (non-isomorphic) complex instructions that were identified. Estimates of the implementation details for these instructions are shown in column 4.

For both applications, load and store instructions as *lod\_add\_lsl*, *str\_add\_lsl* and *lod\_lsr* implementing shifter-based addressing modes provide the most significant speedup while not requiring any change to the memory access scheme. The generation of such specialized addressing modes although currently disallowed in [13], could be safely accounted in their DFG explorer.

Figure 4 shows a portion of the generated templates. A restriction of maximum 3 input and 1 output register operands has been applied to encompass for single register file limitations that apply to a generic RISC. The majority of these templates, for



**Fig. 4.** Candidate instruction examples

**Table 2.** Candidate-induced speedups for the compress and decompress applications

Application name	compress	decompress	
Candidate instruction	% candidate-induced speedup	% candidate-induced speedup	Possible implementation
mla	3.43	0.68	Multi-cycle/pipelined
lod_add_lsl	26.65	17.07	Single-cycle
stri_lsl	1.70	0.91	Single-cycle
bne_imm	0.57	0.08	Single-cycle
beq_inc	1.14	n.a.	Single-cycle
stri_add	1.73	n.a.	Single-cycle
mul_lsl	0.76	0.40	Multi-cycle/pipelined
str_lsl	15.85	5.72	Single-cycle
lsl_mla	1.51	n.a.	Multi-cycle/pipelined
add_sub	0.75	n.a.	Single-cycle
lod_lsl_inc	3.00	1.58	Single-cycle
lod_lsl_dec	2.25	n.a.	Single-cycle
add_asr_add	1.50	n.a.	Single-cycle
add_inc_mul	1.51	0.80	Multi-cycle/pipelined
bge_imm	n.a.	2.32	Multi-cycle/pipelined
ldc_and_sl	n.a.	2.32	Single-cycle
and_sli	n.a.	4.63	Single-cycle
lsl_inc	n.a.	2.32	Single-cycle
str_addi	n.a.	4.63	Single-cycle
add_asr_sub	n.a.	0.79	Single-cycle
add_add	n.a.	0.39	Single-cycle
mul_add_lsl	n.a.	1.18	Multi-cycle/pipelined
lod_lsrv	n.a.	6.95	Single-cycle

example (i), (ii), (iv), could be implemented as single-cycle instructions since 3 or more arithmetic (excluding multiplication and division), logical or shift-by-immediate operations would fit in a single cycle [13]. Complex instruction (iii) incorporates the multiply operation which almost certainly affects the processor cycle time. A multi-cycle or pipelined realization for this instruction is worthy investigating. Multi-cycle instructions may be acceptable even though the processing throughput against using their primitive instruction sequence may not be improved, since power consumption related to instruction fetch is significantly reduced.

It is possible that instruction templates can be merged into superset instructions that would be served on the same hardware. For example, templates (i), (ii), and (iv), make use of up to 2 adder/subtractors and 1 shifter, so that assuming appropriate control, a single instruction for these could be implemented. If a load/store operation is also part of the instruction, it must be performed on a different pipeline stage.

## 5 Conclusions

In this paper, an application analysis flow is proposed for evaluating the characteristics of applications running on configurable processor platforms. For this reason, an open-source compiler infrastructure is utilized to develop our in-house tools. Novel application parameters are introduced in the scope of application characterization as the percentage cover, maximum basic block cycle and candidate-induced application speedup due to the introduction of instruction extensions. An initial set of complex instructions is also generated in order to be used as a starting point in design space exploration iterations. To show the potential of the presented approach, the Wavelet/Scalar Quantization image compression application is used as a case study.

## References

1. Jacome, M.F., de Veciana, G.: Design challenges for new application-specific processors. *IEEE Design and Test of Computers*, Vol. 17, No. 2, (2000) 40-50
2. Gonzalez R.: Xtensa: A configurable and extensible processor. *IEEE Micro*, Vol. 20, No. 2, (2000) 60-70
3. Altera Nios: <http://www.altera.com>
4. Gaisler research: <http://www.gaisler.com>
5. Itoh, M., Higaki, S., Sato, J., Shiomi, A., Takeuchi, Y., Kitajima, A., Imai, M.: PEAS-III: An ASIP design environment. *IEEE Int. Conf. on Computer Design*, (2000) 430-436
6. A. Hoffmann et al.: A novel methodology for the design of application-specific instruction set processors (ASIPs) using a machine description language. *IEEE Trans. on Computer-Aided Design*, Vol. 20, No. 11, (2001) 1338-1354
7. Smith M.D., Holloway G.: An introduction to Machine SUIF and its portable libraries for analysis and optimization. Tech. Rpt., Division of Eng. and Applied Sciences, Harvard University, 2.02.07.15 edition, (2002)
8. Gupta T.V.K., Sharma P., Balakrishnan M., Malik S.: Processor evaluation in an embedded systems design environment., 13th Int. Conf. on VLSI Design, (2000) 98-103
9. Ghazal N., Newton R., Rabaey J.: Retargetable estimation scheme for DSP architecture selection. *Asia and South Pacific Design Automation Conf.*, (2000) 485-489
10. Kim S., Somani A.K.: Characterization of an extended multimedia benchmark on a general purpose microprocessor architecture. Tech. Rpt DCNL-CA-2000-002, DCNL (2000)
11. Lee C., Potkonjak M., Mangione-Smith W.H.: MediaBench: A tool for evaluating and synthesizing multimedia and communication systems. *Proc. of the IEEE/ACM Symp. on Microarchitecture*, (1997) 330-335
12. Leupers R., Wahlen O., Hohenauer M., Kogel T., Marwedel P.: An executable intermediate representation for retargetable compilation and high-level code optimization. *Int. Wkshp. on Systems, Architectures, Modeling and Simulation (SAMOS)*, Samos, Greece (2003)
13. Clark N., Zhong H., Tang W., Mahlke S.: Automatic Design of Application Specific Instruction Set Extensions through Dataflow Graph Exploration. *Int. Journal of Parallel Programming*, Vol. 31, No. 6, (2003) 429-449
14. Hopper T., Brislawn C., Bradley J.: WSQ Greyscale Fingerprint Image Compression Specification, Version 2.0, Criminal Justice Information Services, FBI, Washington, DC (1993)
15. Kumar S., Pires L., Ponnuswamy S., Nanavati C., Golusky, Vojta M., Wadi S., Pandalai D., Spaanenburg H.: A benchmark suite for evaluating configurable computing systems – Status, reflections, and future directions. *ACM Int. Symp. on FPGAs* (2000)

# Power Modeling, Estimation, and Optimization for Automated Co-design of Real-Time Embedded Systems

Amjad Mohsen and Richard Hofmann

Informatik7, Uni Erlangen, Martensstr. 3,  
91058 Erlangen, Germany

amuhsen@immd7.informatik.uni-erlangen.de, rhofmann@cs.fau.de

**Abstract.** This paper introduces a framework for low power co-design and optimization based on a library of pre-modeled components. The used library is enhanced with a set of features to make its use more efficient for automated co-design of real-time embedded systems. The library contains components of different architectures and types to make benefit of power-performance tradeoffs. The modeled components range from simple components to a more complex ones and IPs (Intellectual property). The modeled software components includes the instruction-level power model for selected processors and a set of optimized library routines widely used in selected areas such as in DSP. The co-synthesis process is performed based on an evolutionary algorithm for multi-objective design space exploration. To make the best utilization of the library features, the co-synthesis process is enhanced by an allocation/binding refinement step. This extra step allows trade off between performance and power. It refines the allocation/binding such that the required performance is satisfied without extra unnecessary power loss. Experiments are conducted to demonstrate the applicability of our approach using a set of benchmarks, part of them is taken from real-life examples.

## 1 Introduction

Power consumption has become a major design issue for digital embedded systems during the last few years. One reason is the wide spread of mobile electronics which depends in its operation on a battery that can supply limited amount of energy. Another reason is that the complexity of these systems is increasing rapidly with the exponentially raising capability of semiconductor technology.

Different approaches have been suggested and applied to handle the power/energy dissipation problem. At low abstraction levels, CAD supported tools such as SPICE and PowerMill simulators can be used to estimate the power consumption of a design. Based on these estimations the designer can modify the design to meet a set of design constraints. One drawback of this technique is the simulation time, especially when there is a huge number of design alternatives. So, this technique is limited for small systems. A promising approach which received wide interest during the last few years is to use high-level power/energy estimation and optimization. This technique may depend on a library of modeled components and can provide early power/energy estimations. So, automated high-level synthesis tools can evaluate a huge number of design alternatives early enough in the design process in an acceptable time.

The early power/energy estimation and evaluation of systems can lead to fewer and faster design cycles. At the same time, this method can be a source of large power and cost saving especially for systems with power and performance design constraints. In order to support a high-level design exploration and optimization, two things are required: Firstly, a specification language which can support automated implementation and verification of functional and temporal behavior of real-time systems [1]. Secondly, the required information has to be abstracted from lower levels and supplied in a proper way at the intended high level, possibly, in the form of a library.

The library of components can play an important role in an automated co-design methodology. It supports high-level power/energy estimation and evaluation on one hand and on the other hand it can help utilize design tradeoffs and yield a design with lower cost that still satisfies the design constraints. Modeling the consumed power by a component can be performed based on low-level simulations or even based on experimental measurements. Although, the modeling phase itself is time consuming and technology dependent but the modeled components are re-usable.

The rest of this article is organized in the following way: section 2 presents a short overview of selected power modeling and estimation approaches. Section 3 introduces the suggested library features and the modeling approach. Our co-synthesis methodology for low power is explained in section 4. Section 5 presents the experimental results obtained from the application of our suggested methodology for the used case studies. Concluding remarks and suggestions for future work are given in section 6.

## 2 Related Work

To realize the jump to high abstraction levels, it is necessary to model the power consumption of different components and processing units. Modeling the power consumption for processing units can be obtained by using top-down (analytical) methodologies [2] [3] or by using bottom-up (empirical) ones. The aim of the first methodology is to estimate the power consumed by a block based on its logic function. Although, this methodology can be used at high abstraction levels, it is very inaccurate. Our focus here is on the second methodology.

Several approaches have been reported for power macro-modelling for hardware components. A library of pre-characterized and parameterized micro-components such as adders, multipliers, registers, and controllers has been suggested in [4]. The power model of each library component has been derived from many layout exercises. The switching activities of the components inputs/outputs have been related to the power model. This library has been integrated in an RTL level power estimation methodology. A similar library as in [4] has been suggested in [5] to perform behavioural level power estimation. The authors have combined analytical and stochastic estimation techniques to perform power estimation and exploration.

The probably best known technique for software-level power estimation has been suggested by Tiwari et al. The work introduced in [6] and [7] suggested the use of instruction level power model to quantify the power cost of the software components in an embedded processor. Calibrating the model was based on measuring the current drawn by the microprocessor when executing an instruction. Estimating the power/energy consumed in the software (in a program) was then performed using these power models and the assembly/machine level code. Power analysis of a com-

mercial 32-bit RISC-based embedded microcontroller has been developed in [8] based on the instruction level power model. For system level evaluation purposes the speed of evaluation will be very slow because a huge number of design alternatives are available there.

A high level power estimation approach based on VHDL simulation has been presented in [9]. This approach was based on a library of pre-characterized blocks of the size of a single VHDL operator. Power models were developed based on information obtained from standard cell libraries. Estimating the power consumption based on VHDL simulation ignores all synthesis related effects and the possible logic optimization. Hence, this approach suffers from the inaccuracy in the estimated power.

ORINOCO tool was presented in [10] to compare different and functionally equivalent algorithms and bindings to RTL architectures with respect to power consumption. This tool reads in VHDL and C/C++ descriptions. The VHDL was simulated to determine the input data stream whereas, a control data flow graph was generated based on the C/C++ description. Based on the simulation data and the power models of the components, the power consumption was estimated.

A power estimation framework for hardware/software System-On-Chip (SOC) designs has been introduced in [11]. The approach was based on concurrent execution of different simulators for different parts of the system (hardware and software parts). Although, this approach can be fairly accurate it can be very slow especially for large systems when a huge number of design alternatives is available.

Although much of the previous work in the area of power estimation and low power synthesis was based on a library of pre-modelled components, a set of special features that are of vital importance for efficient hardware/software co-design has not been tackled. We suggest such features and conduct a set of experiments to show their effect. Also, most of the previous approaches considered hardware and software components separately. Our integrated methodology for low power co-design of real-time embedded systems explores the available design alternatives and yields power optimized design(s) that satisfy the constraints and have the lowest possible cost. We concentrate here on the benefit of the suggested library components on the overall power estimation accuracy and on handling power-performance-area tradeoffs.

### 3 The Components Library

A library that may be used in hardware/software co-design of real-time embedded systems should have some basic features to make its application efficient. In this context, efficiency means the evaluation of design alternatives at system level should be fast with relative accuracy as well as to make best use of design tradeoffs.

#### 3.1 Required Features

The first feature is that the modeled library components should enable its users to make benefit of power-performance tradeoffs. This feature could be achieved not only by supplying different components but also by offering different types of the same component to represent different architectures and technologies. Each type might be

optimized for power, performance, or area. The suggested allocation/binding refinement step can utilize such a feature to make extra power reduction.

Another feature is the component granularity which makes the trade off between estimation speed and accuracy. The modelled hardware components range from simple ones such as adders, multipliers to more complex components and IPs widely used by specific applications such as those used in DSP applications. Software components may include the basic instruction set of a processor and a set of pre-optimized library routines modelled for their power and performance requirements. The modelled fine grain components may only be used in estimating power and performance requirements of a program when no coarse-grain library function can be used for that purpose.

At the same time, when the library offers, to some extent, the hardware and software versions of similar functional components it enables fast comparison between hardware and software solutions.

### 3.2 Modeling Power Consumption

In order to model the power consumed by a hardware component, a set of experiments has been performed to extract the effect of input/output parameters on its consumed power. A model fitting phase is then required to extract the value that best fits the measured data (*power base*). A variety of model fitting techniques such as least-squares regression can be used [12]. Using, for example, the least-squares regression technique, the following equation (possibly in matrix form) is solved to extract the power base ( $P_{base}$ ) that minimizes the modelling error ( $E$ ) in the equation:

$$P_{observed} = CP_{base} + E \quad (1)$$

where:  $P_{observed}$  is the measured power consumption from each experiment,  $C$  is the parameter(s) values, which might be a vector. After modelling the power consumption of a component, the power model for that component is stored in the library. To obtain the consumed power by a component, the necessary parameters should be supplied as an argument and then combined with the corresponding  $P_{base}$ . The reported power consumption is given in mW/MHz to exclude the effect of the operational frequency on the power figures.

Our experimental results showed that many modules follow a simple linear model with respect to the input/output bus size ( $N$ ), such as adders, multiplexers, and buffers. Other modules follow a more complex model such as the multipliers (*where the power is a function of  $N^2$* ) and the MAC. The controller still follows a more complex model when implemented as a finite state machine (FSM). For this type of controllers, the consumed power is dependent on the number of inputs, outputs, and the number of states. Depending on the modelled component, the parameters affecting its consumed power are specified.

In the case of software components, a set of software library functions is modelled for power consumption. In addition to this, the instruction set of the selected processor is modelled for power consumption. Each instruction or library function is executed in a loop while simulating the target processor. At each loop cycle, new data set is supplied as operand(s) for the modelled component. Without supplying new test vectors at each iteration, the effect of data switching would not be included in the

power model in an actual way. Most of the previous work has ignored this point in modelling the power consumed in the software. To reduce the effect of the loop control, many instances of the same modelled instruction or function are used in the loop. All reported results for power are normalized by the processor frequency. When the instruction or the function requires more than one cycle, the number of cycles are multiplied by the average consumed power. So, the reported results represent the energy consumed by the processor when performing an instruction or a function. The start and end time of execution for software components have been experimentally specified by instantiating a counter.

Comparing our results with the instruction-level power estimation method, we have found experimentally that using the library functions or routines in the case of software reduces the estimation error. For example, in a program that consists of an arbitrary number of addition and multiplication operations, an estimation error of about 18% was calculated when using instruction level method while using coarser grain components reduces the estimation error drastically. More details about the modelling approach and modelled components are found in [15].

### 3.3 Component Architecture and Power-Performance Tradeoffs

A basic feature in the components library is that it contains more than one type of the same component. Different types of the same component represent different architectures and different technologies. These different types are suggested to utilize performance-power-area tradeoffs. Examples are: pipelined and non pipelined types, optimized architectures for power, area, or performance of the same component. The examples presented here to demonstrate our approach are optimized to be implemented in FPGA. The results are generated with the help of Quartus II software from Altera [13]. Synopsys is used when the components are implemented in ASICs [14]. We have selected the following examples to demonstrate the effect of different component architectures on power and performance. More detailed results are available in [15].

The first example is to show how different types of the same component can offer performance-power trade-off. Figure (1) shows the power consumption of two implementations of an adder: ripple and carry-look-ahead adder. The first is optimized for power consumption but it encounters more delay to perform an addition. For a 32-bit adder of these types, the obtained performance merit when using the performance-optimized component is (2.5) (2.5 times faster). On the other hand, a power reduction of (1.84) can be obtained when using the power-optimized type.

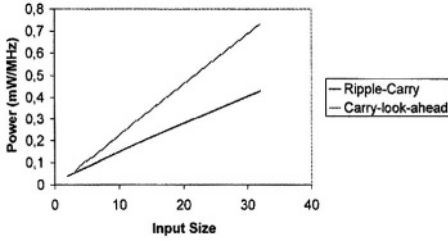
The other example is the design of the Finite Impulse Response (FIR) filter which is commonly used in signal processing applications. Its basic structure is a series of multiplications followed by an addition. Figure (2) shows the consumed power for pipelined serial and parallel versions of this filter<sup>1</sup>.

As a demonstration example for software components, we have chosen two different implementations of the multiplication instruction. Firstly, the multiplication instruction is implemented using the so called m-step serial multiplier, in the second implementation, a hardware multiplier is instantiated within the processor itself. The

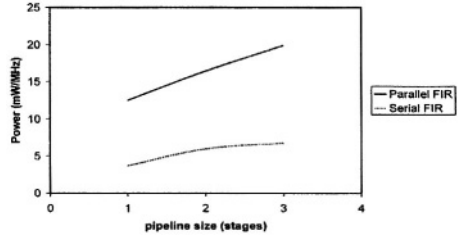
---

<sup>1</sup> The average power in this figure is not multiplied by the number of cycles needed to generate a valid output to show the difference in the average consumed power.





**Fig. 1.** Different implementations of an Adder



**Fig. 2.** Power consumed by FIR filter

former requires (140) cycles to perform a 32-bit multiplication with an average (dynamic) power consumption of (11.2) mW/MHz. Whereas the latter requires only (40) cycles and consumes (13.24) mW/MHz. Although, the former implementation is better in terms of average power, the latter is superior in terms of energy. These results have been obtained with the help of the SOPC design environment from Altera and the Nios processor has been selected for demonstration purposes.

## 4 Low Power System Synthesis

Reducing the time-to-market of large embedded systems is a basic goal of the integrated framework “Co-design and Rapid-Prototyping of Systems for Applications with Real-Time Constraints” (CORSAIR) [16]. CORSAIR supports the formal specification of embedded systems with the specification and description language (SDL) and message sequence charts (MSC). The components library has been integrated with CORSAIR. In this paper, we have replaced the tabu search in CORSAIR with an evolutionary algorithm based synthesizer. For complicated systems where the design space is very large, evolutionary algorithms can be applied [17]. The widely used evolutionary multi-objective optimizer SPEA2 [18], has been used in our framework. The optimization goal is to find design alternatives with pareto-optimal objective vectors. The overall design flow methodology is depicted in Figure (3).

Allocation obtained from the decoded implementation specifies generic components, such as: Processor, FFT, MULT ...etc. Further architecture details are refined by the allocation/binding refinement step. Infeasible allocations and bindings are handled by repair mechanisms and penalties. The scheduling step gives a valid schedule for the tasks given that the design constraints should be satisfied. Allocation refinement step has been suggested to search the “sub-space” of each allocation when different components’ types are available. This enables the synthesizer to generate a design with better quality by handling the power-performance-cost tradeoffs. So, allocation refinement searches to find the best type for each component instance that can satisfy the constraints without further power dissipation. This may allocate performance optimized components instances to execute the tasks on the critical path and power optimized types for other tasks. Power optimized types might increase the delay, but they might be allocated to exploit the available slack. So, performance is traded for power when the system throughput is maintained. This step deals with an ordered list of different types of the same component.

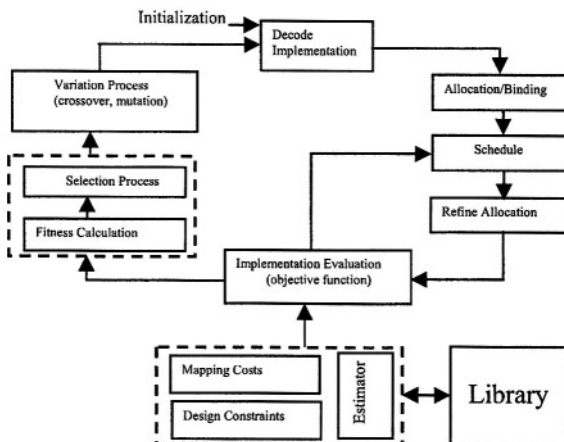


Fig. 3. Design flow

The power consumption, performance, and cost are estimated for each design alternative base on the library. The number of accesses for each component is combined with its power consumption which is obtained as explained before (see section 3.2). Design constraint violation is handled by using proper penalty functions for each design constraint.

## 5 Results

We have tested our methodology using four benchmarks to demonstrate the capability of our framework to produce design alternatives with low power/energy consumption without violating other design constraints on cost and performance. We focus here on the optimization of component allocation using different types available for selected components and on the mapping process. A general purpose processor is used to control the execution. We report the achieved power reduction in % compared to the case when no power optimized components are available. We have assumed that each application has a certain repetition rate and the processing should finish before a new cycle is started. Also, a stringent power budget is assumed, such as those in power critical applications.

The first benchmark is the Gaussian Elimination and it is taken from [19]. The total number of tasks in this Gaussian Elimination graph is:  $(n^2 + n - 2)/2$ , where  $n$  is the matrix size. The used architecture is a division unit and another unit to perform all basic operations and multiplication. A power reduction of 15% could be achieved when the division unit is replaced with a power optimized unit. This power optimized unit has been pipelined and then operated at low frequency (the frequency of the old system). It is possible to achieve more power reduction when all components are replaced by power optimized types but this increases the delay and cost to a limit that violates the forced constraint.

The second benchmark is taken also from [19]. This benchmark represents a recursive, one dimensional FFT algorithm. When an input vector of size  $n$  is used, there

are  $2 \times n - 1$  recursive call tasks and  $n \times \log_2 n$  butterfly operations. A power reduction of 17% is obtained when the butterfly operations are executed on a power optimized components. The power optimized components are design using adders and multipliers (pipelined) such as those described previously in this paper.

Another hypothetical benchmark (tgff1) is taken from [20]. A power reduction up to 19% could be achieved by using power optimized components. Although, power reduction could be achieved when refining the allocation, the cost is increased. Whether this cost increment is acceptable is strongly dependant on the application domain.

The last benchmark (OFD) is taken from a real-life example. This example is taken from a traffic monitoring system based on optical flow detection (OFD) algorithm. This algorithm is part of an autonomous model helicopter [21]. This algorithm originally runs on two DSPs with an average current of 760 mA at 3.3V. The OFD algorithm uses a repetition rate of 12.5 frames of  $78 \times 120$  pixels per second. The original data for power consumption and performance are partially taken from [20]. To demonstrate the effect of using different types on the OFD, we have used two DSPs and another extra hardware unit to perform convolution and other basic operations. So, the application parallelism is better utilized. The extra hardware component has different types. Power optimized types with extra delay and performance optimized types with extra power dissipation and extra cost. The power reduction obtained when using power optimized type for the extra hardware unit is about 6.3% without violating the performance constraints.

These examples could justify the need for and the applicability of using different types for selected components to achieve tradeoffs between power, performance, and area.

## 6 Conclusions and Further Work

Together with the suggested components library, the proposed technique in this paper can estimate and reduce the power dissipation of systems with stringent power constraints. Based on the components library which contains different types of the same component, the co-synthesis process can explore the design alternatives and refines the allocation in such a way the performance constraints are satisfied without extra power dissipation. The suggested features for the components library are of critical importance for automated co-design methodologies of real-time embedded systems. Such a library supports fast estimation and evaluation of different design alternatives at high abstraction levels. Experiments carried out on hypothetical and real-life examples show encouraging results in terms of power reduction of up to 19% without performance penalty. Based on this, power, performance, and area tradeoffs are better handled by automated co-synthesis tools by offering not only different components but also different types for selected components.

We plan to include in the future more accurate power models for communication components. Also, it is worth studying how performance optimized components can affect power reduction when integrated with systems that support dynamic voltage scaling.

**Acknowledgement.** We wish to thank Gruian, F, from Lund University, Sweden, and Schmitz, M., from Linköping, Sweden for providing their benchmark sets.

## References

- [1] Münzenberger, R., Dörfel, M., Hofmann, R., and Slomka, F.: A General Time Model for the Specification and Design of Embedded Real-Time Systems. *Microelectronics Journal* 34, (2003), pp. 989-1000.
- [2] Marculescu, R., Marculescu, D., and Pedram, M.: Information Theoretic Measures for Power Analysis. *IEEE trans. On Computer Aided Design of Integrated Circuits and Systems*, vol. 15, (1996), pp. 599-610.
- [3] Nemani, M., and Najm, F.: High-level Area and Power Estimation for VLSI Circuits. *IEEE trans. On Computer-Aided Design of Integrated Circuits and Systems*. (1999).
- [4] Landman, P., and Rabaey, J.: Architectural Power Analysis: the Dual Bit Type Method. *IEEE Trans. on VLSI*, (1995), pp. 173-187.
- [5] Mehra, R., and Rabaey, J.: Behavioural Level Power Estimation and Exploration. *Int. workshop on low power design*, California, USA, (1994).
- [6] Tiwari, V., Malik, S., and Wolfe, A.: Power Analysis of Embedded Software: A First Step towards Software Power Minimization, *IEEE Trans. on VLSI*, (1994), pp. 437-445.
- [7] Tiwari, V., Malik, S., Wolfe, A., and Lee, M.: Instruction Level Power Analysis and Optimization of Software. *Journal of VLSI Signal Processing*, (1996), pp. 1-18.
- [8] Tiwari, V., and Lee, M.: Power Analysis of a 32-bit Embedded Microcontroller, *VLSI Design Journal*, Vol. 7, No. 3, (1998).
- [9] Llopis, R.: High Level Power Estimation, *PATMOS'97*, Belgium, (1997).
- [10] Stammermann, A., Kruse, L., Nebel, W., and Pratsch, A.: System Level Optimization and Design Space Exploration for Low Power. *Proc. of the Int. Symp. on Systems Synthesis*, Canada, (2001).
- [11] Ljolo, M., Raghunathan, A., Dey, Lavagno, S., and Sangiovanni-Vincentelli, A.: Efficient Power Estimation Techniques for HW/SW systems. In *Proceedings of the IEEE VOLTA'99 International Workshop on Low Power Design*. (1999), Como, Italy, pp. 191-199.
- [12] Montgomery, D., Peck, E., and Vining, G.: *Introduction to Linear Regression Analysis*. John Wiley & Sons Inc, (2001).
- [13] Altera Documentations. Available at: <http://www.altera.com>.
- [14] Dalton Project. Available at: <http://www.cs.ucr.edu/dalton/tutorial/index.html>.
- [15] Mohsen, A., and Hofmann, R.: Characterizing Power Consumption and Delay of Functional/Library Components for Hardware/Software Co-design of Embedded Systems. *RSP'04*, (2004), Geneva. (*to appear*).
- [16] Slomka, F., Dörfel, M., Münzenberger, R., and Hofmann, R.: Hardware/Software Codesign and Rapid Prototyping of Embedded Systems. *IEEE Design & Test of Computers*, (2000), pp-28-28.
- [17] Blickle, M., Teich, J., and Thiele, L.: System-level Synthesis Using Evolutionary Algorithm. *Design Automation for Embedded Systems*, 3(1): (1998). pp. 23-58.
- [18] Zitzler, E., Laumanns, M., and Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. *Evolutionary Methods for Design, Optimization, and Control*, (2002), Spain.
- [19] Topcuoglu, H., Hariri, S., and Wu, M.: Performance-effective and Low-complexity Task Scheduling for Heterogeneous Computing. *IEEE trans. on Parallel and Distributed Systems*, Vol. 13, No. 3, (2002), pp 260-274.
- [20] Schmitz, M., Al-Hashimi, B., and Eles, P.: Synthesizing Energy-efficient Embedded Systems with LOPOCOS. *Design Automation for Embedded Systems*, 6, (2002), pp 401-424.
- [21] The Wallenberg Laboratory for Research on Information Technology and Autonomous Systems. Available at <http://www.ida.liu.se/ext/witas>.

# Mapping Computational Intensive Applications to a New Coarse-Grained Reconfigurable Data-Path\*

M.D. Galanis<sup>1</sup>, G. Theodoridis<sup>2</sup>, S. Tragoudas<sup>3</sup>, D. Soudris<sup>4</sup>, and C.E. Goutis<sup>1</sup>

<sup>1</sup> University of Patras,  
mgalanis@vlsi.ee.upatras.gr

<sup>2</sup> Aristotle University,

<sup>3</sup> Southern Illinois University, USA

<sup>4</sup> Democritus University, Greece

**Abstract.** In this paper, a high performance coarse-grained reconfigurable data-path, part of a mixed-granularity reconfigurable platform, is presented. The data-path consists of several coarse grained components that their universal type is shown to increase the system's performance due to significant reductions in latency. An automated methodology, which is based on unsophisticated algorithms, for mapping computational intensive applications on the proposed data-path is also presented. Results on DSP and multimedia benchmarks show an average latency reduction of approximately 21%, combined with an area consumption decrease, when the proposed data-path is compared with a high-performance one. The latency reduction is even greater, 43%, when the comparison is held with a data-path that instantiates primitive computational resources in FPGA hardware.

## 1 Introduction

Reconfigurable computing brings together the flexibility of software to the performance of hardware [1,2]. Due to the presence of coarse-grained units [1, 2], in mixed-granularity (hybrid) reconfigurable platforms, the performance is increased, while the power consumption and the reconfiguration overhead are decreased, when the coarse-grained hardware is used to implement an application instead of fine-grained (FPGA) hardware. Research activities in High-Level Synthesis (HLS) [3] and Application Specific Instruction Processors (ASIPs) [4, 5] have proven that the use of complex data-path resources instead of primitive ones (like single ALUs and multipliers) improves the performance of such applications. In these works, at the behavioral level, complex operations are used instead of groups of primitive ones, while at the architectural level, special hardware units called *templates* or *clusters* are used to implement these operations. A template may be a specialized hardware unit or a group of properly-designed chained units [3]. Chaining is the removal of the intermediate registers between the primitive units improving the total delay of the units combined. The templates can be either obtained by an existing library [3] or can be extracted from the application's Data Flow Graph (DFG) [4, 5].

---

\* This work was partially funded by the Alexander S. Onassis Public Benefit foundation.

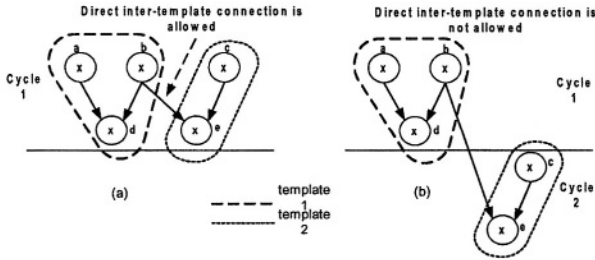


Fig. 1. Latency gain with direct inter-template connections

In this paper, a high-performance coarse-grained reconfigurable data-path for implementing computational (data-path) intensive applications, such as DSP and multimedia ones, is introduced. It consists of a set of Coarse-Grained Components (CGCs) implemented in ASIC technology, a reconfigurable interconnection network, and a centralized register bank. The data-path is a part of general mixed-granularity reconfigurable platform architecture. This platform consists of microprocessors(s), embedded FPGAs, data and program memories and the proposed coarse-grained data-path. Each CGC is able to realize complex operations. Direct inter-CGC connections exist to fully exploit chaining between nodes of different CGCs in contrast to existing template-based methods [3, 4, 5]. The offered flexibility allows the data-path to be efficiently used when a temporal partitioning approach is adopted, like the one in [6]. Furthermore, the stages of mapping are accommodated by unsophisticated, yet efficient, algorithms for scheduling and binding with the CGCs. A motivational example regarding the optimal chaining exploitation is shown in Fig. 1. If direct inter-template connections are permitted (as in the case of the CGC data-path) the chaining of operations is optimally exploited and the DFG is executed in one cycle (Fig. 1a). On the other hand, if direct inter-template connections are not allowed as in [3, 4, 5] the result of operation *b* is produced and stored in the register bank at the first clock cycle and it is consumed at the second cycle. Thus, the DFG is now realized in two cycles (Fig. 1b).

The rest of the paper is organized as follows. Section 2 describes the CGC data-path, while section 3 the features arisen from the design of the data-path. The mapping methodology for the proposed data-path architecture is described in section 4. Section 5 presents the experimental results, while in section 6 the related work is described. Finally, concluding remarks and future work are drawn in section 7.

## 2 CGC Data-Path Architecture

The proposed data-path architecture consists of: (a) the CGCs, (b) a centralized register bank, and (c) a reconfigurable interconnection network, which enables the inter-CGC connections and the connections between the CGCs and the register bank. The structure of the CGC is an  $n$ -by- $m$  ( $n \times m$ ) array of nodes, where  $n$  and  $m$  are the number of nodes per row and column, respectively. In Fig. 2a such a CGC (called hereafter as  $2 \times 2$  CGC) with 2 nodes per row and 2 nodes per column is illustrated. A  $2 \times 2$  CGC consists of four nodes whose interconnection is shown in Fig. 2a, four

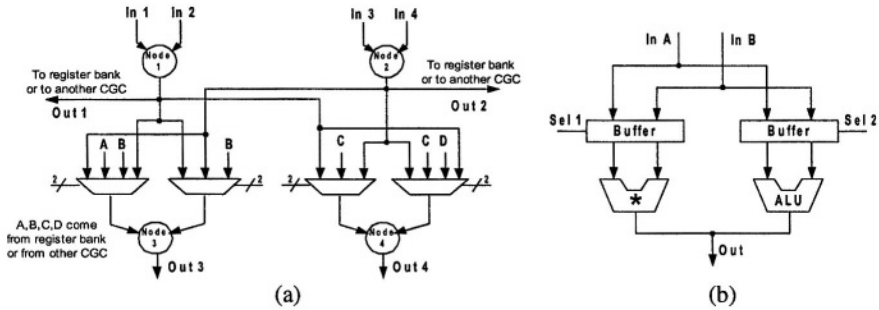


Fig. 2. Architecture of a 2x2 CGC (a) and the CGC node architecture (b)

inputs ( $in1, in2, in3, in4$ ) connected to the centralized register bank, four additional inputs ( $A, B, C, D$ ) connected to the register bank or to another CGC, two outputs ( $out1, out2$ ) also connected to the register bank and/or to another CGC, and two outputs ( $out3, out4$ ) whose values are stored in the register bank.

Each CGC node consists of two functional units that are a multiplier and an ALU as shown in Fig. 2b. Both units are implemented in combinational logic to exploit the benefits of the chaining of operations inside the CGC. The flexible reconfigurable interconnection among the nodes inside a CGC is chosen to allow in easily realizing any desired hardware template by properly configuring the existing steering logic (i.e. the multiplexers and the tri-state buffers). The ALU performs shifting, arithmetic (add/subtract), and logical operations. Each time either the multiplier or the ALU is activated according to the control signals,  $Sel1$  and  $Sel2$ , as shown in Fig. 2b. The data-bus width of the CGC is 16-bit, because such a bit-width is adequate for the majority of the DSP applications. The control-unit of the data-path is implemented in the fine-grained part (FPGA) of the mixed-granularity platform.

An  $n \times m$  CGC has an analogous structure. Particularly, the first-row nodes obtain their inputs from the register bank. All the other CGC nodes obtain their inputs from the register bank and/or a row with a smaller index from the same and/or another CGC. For the case of the CGC outputs, the last-row nodes store the results of their operations to the register bank. All the other nodes give their results to the register bank and/or to another CGC in the data-path. In previously published template methods, like in [3, 4, 5], templates with  $depth=2$  and  $width \leq 2$  (i.e.  $n=2, m \leq 2$  for the CGC) were mainly used due to two reasons: (a) larger templates introduce greater area and control overhead relative to a primitive resource data-path [3], and (b) templates consisting of two operations in sequence contribute the most to the performance improvements [4, 5]. So, CGCs with a value of  $n \leq 3$  and  $m \leq 3$  are adequate to be used in data-paths for achieving important speedup of applications.

The reconfigurable interconnection network is divided into two sub-networks. The first one is used for the communication among the CGCs. The second one is used for the communication of the CGCs with the register bank for storing and fetching data values. A crossbar interconnection network can be used to provide full connectivity in both cases. When a large number of CGCs is required to be present in the data-path (e.g. for supporting large amounts of parallelism), the crossbar network cannot be used since it is not scalable network. A scalable interconnection network is the fat-tree [7], which is a hierarchical network used extensively in multiprocessors systems. Particularly, an efficient interconnection network can be created by connecting CGCs

into clusters. Firstly,  $k$  CGCs are clustered and connected with a switch device, which provides the full connectivity among the  $k$  CGCs. Then,  $k$  clusters are recursively connected together to form a supercluster and so on. Any two CGCs can be connected with fewer than  $2\log_k N - 1$  switch devices, where  $N$  is the total number of CGCs in the data-path. So, the interconnect delay for implementing the full connectivity in a fat-tree interconnect network increases logarithmically with the number of CGCs in the data-path (since  $2\log_k N - 1$  switch devices are required) and not in quadratic manner as in a crossbar network, where  $O(N^2)$  switches are required. Although, a fat-tree like interconnection scheme can be also employed by the existing template methods [3, 4, 5], the interconnect delay of the CGC-based data-path is smaller as it consists of a smaller number of regular and uniform hardware resources (the CGCs) compared to the number of resources required in existing template-based data-paths.

### 3 CGC Data-Path Features

Compared with an equivalent CGC functionality realized by templates, like the ones of [3, 4, 5] the CGC's critical path increases due to the summations of the delays of levels of tri-state buffers and multiplexers. To have an indication for this increase an experiment has been performed. The delay of a 2x2 (2x3) CGC compared with a template of two multipliers in a sequence, is increased by 4.2% (4.7%), when both are synthesized at 0.13 $\mu$ m ASIC CMOS process. Thus, the performance improvements over the template-based data-paths are not negated, since the measured % delay increase (and thus the % increase in clock cycle period) is significantly smaller than % reduction of the clock cycles (see Tables 1 and 2) over the considered template data-path.

Although extra control signals are required to reconfigure the CGC, compared to a primitive or a template resource, the control-unit can be designed in a way that does not incur an increase in the critical path of the whole CGC data-path. This can be achieved when control signals are grouped together to define a subset of the state of the control-unit in a control-step (c-step). This way of synthesizing the control-unit is supported by current CAD synthesis tools [8], where the control-unit can be automatically synthesized under a given delay constraint. In this way, the delay of this control-unit is not increased relative to the one of a data-path consisting of templates and/or primitive resources. However, the area of the control-unit of the CGC data-path increases. This area redundancy - also due to the 2 functional units inside the CGC node - is a trade-off for achieving high-performance. So, since our priority is high-performance and not area consumption, this area increase is not a major consideration in this work. The area overhead, relative to a data-path composed by primitive resources, is also the case of data-paths composed by templates hardware units. In [3], an average increase of 66% in area consumption was reported.

Regarding the power consumption, each time an operation is covered by a CCG node, either the multiplier or the ALU is activated by properly controlling the corresponding tri-state buffers. When a CCG node is not utilized at a c-step, neither the multiplier nor the ALU are activated, thus reducing power consumption.

*Temporal partitioning* is a procedure for properly partitioning an application into a number of temporal segments, which are executed one after another. The ASAP leveling methodology of [9] can easily be adapted to the CGC data-path. So, it can be



an automated methodology for the temporal partitioning of applications to the CCG data-path. However, the ASAP leveling methodology does not consider the resource sharing of computational units (like the CGCs). It has been proved in [6] that a temporal partitioning which is combined with resource sharing of functional units reduces the latency, when compared with approaches like [9]. The development of an efficient temporal partitioning methodology, which takes into consideration the resource sharing of the CGCs, is a topic of future work.

## 4 Mapping Methodology

The introduced data-path offers advantages that enable the high-performance mapping of applications. As the CGC data-path consists of one type of resource (i.e the CGC), the mapping process can be realized by unsophisticated, though effective, algorithms. The universal and flexible structure of the CGC allows in achieving a full DFG covering using only CGCs. Also, the flexibility of the whole data-path provided by the CGCs and the interconnection network gives extra freedom to realize the DFG by a small number of CGC instances, while inter- and intra-CGC chaining is fully exploited.

The mapping methodology consists of (a) scheduling of DFG operations, and (b) binding with the CGCs. The input is an unscheduled DFG. For mapping Control Data Flow Graphs (CDFGs) the methodology is iterated through the DFGs comprising the CDFG of an application [10]. The design choice was to use a list-based scheduler [10]. In the CGC-based data-path, the list scheduler is simplified, since it handles one resource type, which is the CGC node. This happens since the input DFG consists of ALU and/or multiplication type of operations - which is the usual case in DSP benchmarks - (or it is transformed to be composed only by these operations) and each CGC node contains an ALU and a multiplier; thus each DFG node is considered of one resource type.

Due to the aforementioned features of the introduced CGC data-path a simple, though efficient, algorithm is used to perform binding. The pseudo-code of the binding algorithm is illustrated in Fig. 3. The input is the scheduled DFG, where each c-step has  $T_{prim}$  clock period, and this type of c-step is called from here after *c-step<sub>prim</sub>*. For example,  $T_{prim}$  can be set to the multiplier's delay, so each DFG node can occupy one *c-step<sub>prim</sub>*, since all DFG nodes have, in this case, unit execution delay. After binding, the overall latency of the DFG is measured in new clock cycles having period  $T_{CGC}$ , that is set for having unit execution delay for the CGCs.

```

do {
  for the number of CGCs
  for (CGC_index=0; CGC_index <n; CGC_index ++ )
    while (col_idx < number of ops in a row && col_idx < number of uncovered DFG nodes)
      map_to_CGC (DFG_node, CGC_index, col_idx)
    end while;
  end for;
end for;
} while (the DFG is covered)

```

Fig. 3. The binding algorithm

The CGC binding algorithm maps the DFG nodes to the CGC nodes in a row-wise manner. We define a term called *CGC\_index* that it is related to the new period  $T_{CGC}$  of the c-steps (i.e. the clock period), after the binding is performed. It represents the current level of component's operations that bind the DFG nodes. The *CGC\_index* takes the values from 0 and  $n-1$ , as the proposed component consists of  $n$  levels of operations. The algorithm covers the operations in  $c\text{-step}_{prim}$  ( $s$ ) for *CGC\_index* equal to 0 or until there are no DFG nodes left uncovered in a  $c\text{-step}_{prim}$ . Then it proceeds to the next value of *CGC\_index* till equal to  $n-1$ , if there are DFG nodes left to be covered. This procedure is repeated for every CGC in the data-path. The binding starts from the CGC assigned to the number 1, and it is continued, if it is necessary, for the next CGCs in the data-path. A CGC is not utilized, when they are no DFG nodes to be covered. Also, a CGC is partially utilized when there is no sufficient number of DFG nodes left and the mapping to CGC procedure (*map\_to\_CGC*) has already been started for this CGC. If there are  $p \times m$  CGCs in the data-path, the maximum number of operations per  $c\text{-step}_{prim}$  is equal to  $p \cdot m$ , as each CGC level consists of  $m$  nodes. Finally, after CGC binding, the register bank size is determined.

## 5 Experimental Results

A prototype tool has been developed in C++ for demonstrating the efficiency of the introduced CGC-based data-path methodology. The DFGs used in the experiments were obtained from representative benchmarks coded in behavioral VHDL and in C language. The first set of benchmarks is coded in VHDL and consists of data-flow structures used extensively in HLS research domain. The second set is coded in C language and consists of: (a) a set of files, from the Mediabench test suite [11] that implements DSP functions, and (b) an in-house JPEG encoder and IEEE 802.11a OFDM transmitter. The DFGs were obtained from the VHDL descriptions with the aid of the CDFG tool [12]. For extracting DFGs from the C code, the SUIF2 compiler [13] was used. Regarding the DFGs shown in the first column of Tables 1 and 2, the *ellip* to *wdf7* are extracted from HLS benchmarks, the *jpeg* to *ofdm* from the in-house benchmarks, and the *gsm\_enc* till the end from the Mediabench suite.

We have synthesized the control-units of small DFGs (*fir11*, *volterra* and *ellip*) for a data-path consisting of two 2x2 CGCs and we have measured the delay of these units. The specification of the control-units has been performed manually since by this time we do not support a method for automatically defining control-units. For the synthesis of the derived control-units, a 0.13 $\mu\text{m}$  CMOS process has been used. The average delay of the control-units is a small fraction (approximately 10% in average) of the delay imposed by a 2x2 CGC. This indicates that the delay of the control-unit does not affect the critical-path of the proposed data-path, which is defined by the CGC delay. Analogous results in the control-unit's delay are expected for DFGs consisting of large number of nodes (e.g the *gsm\_enc*).

A *second experiment* is performed that showed that a data-path with two 2x2 CGCs achieves an average clock cycles (latency) decrease of 58.1% when compared with a data-path composed by primitive resources. The clock cycle of the primitive resource data-path  $T_{prim}$  is set to the ALU delay. For this data-path, a multiplication operation takes two clock cycles to complete. As already mentioned, for the CGC data-path, the clock period is set for having unit execution delay. It is proved that the performance

of the proposed data-path is higher when  $T_{CGC} < 2.4 \cdot T_{prim}$  (1), where  $T_{CGC}$  and  $T_{prim}$  is the clock period for the CGC and the primitive resource based data-path, respectively. Equation (1) has been satisfied after comparing the 2x2 CGC delay with a 16-bit ALU delay, where both resources are implemented in structural VHDL and synthesized at a 0.13 $\mu$ m CMOS process. The results showed that  $T_{CGC} = 2.14 \cdot T_{prim}$ . If the physical layout of the CGC is manually optimized, then the  $T_{CGC}$  should become smaller. The same experiment was performed for other types of CGCs (e.g. the 2x3) and showed that the corresponding equations like (1) do not impose hard constraints.

A *third experiment* compares the performance and the area utilization of the introduced data-path with another high-performance data-path, which is composed by templates. The template library consists of the following templates: *multiply-multiply*, *multiply-alu*, *alu-alu*, and *alu-multiply*. These templates are chosen because they are proposed by the majority of the existing methods [3, 4, 5] to be used to derive high-performance data-paths for DSP and multimedia kernels. The CGC data-path consists of two 2x2 (case 1) and two 2x3 (case 2) CGCs, thus the chaining depth equals 2 in both cases. So, in case 1, four operations can be executed concurrently in each c-step, while in case 2, six operations can be executed concurrently. The assumptions of this experiment are: (a) template partial matching is enabled, and (b) the clock period for each synthesized template-based data-path is set to the delay of the *multiply-multiply* template (i.e. unit execution delay for all the templates). The template partial matching enables the full DFG covering, without needing extra primitive resources to be present in the data-path. As shown in Section 3, the delays of a 2x2 and a 2x3 CGC are marginally larger than a *multiply-multiply* template. Since, our design decision is to set the clock period  $T_{CGC}$  for having unit execution in a CGC data-path, the clock periods in both data-paths are virtually the same. Hence, assumption (b) is made so as the performance comparison in Table 1 (and in Table 2) is straightforward.

**Table 1.** Clock cycles (latency) results when the proposed data-path is compared with a template-based one (template partial matching are enabled)

DFG	# of DFG nodes	Template-based		CGC Based		% cycles decrease	
		4 res.	6 res.	two 2x2	two 2x3	4 res.	6 res.
ellip	34	9	9	6	6	33.3	33.3
fir11	31	7	7	6	6	14.3	14.3
nc	72	12	9	10	7	16.7	22.2
volterra	32	8	8	6	6	25.0	25.0
wavelet	67	12	9	9	7	25.0	22.2
wdf7	50	9	9	7	7	22.2	22.2
jpeg	646	99	66	81	54	18.2	18.2
ofdm	270	41	28	34	23	17.1	17.9
gsm_enc	822	120	80	103	69	14.2	13.8
gsm_dec	866	137	92	109	73	20.4	20.7
mpeg2	68	12	8	9	6	25.0	25.0
rasta	151	22	15	19	13	13.6	13.3
<b>Average values</b>						<b>20.4</b>	<b>20.7</b>

To derive the latency for the template-based data-path, binding (covering) with templates is performed in the unscheduled DFG and then scheduling is performed by

a list scheduler. The template binding is performed so as the available primitive computational resources (multipliers and/or ALUs) in each c-step is equal with the ones available from the CGC-based data-path. As illustrated in Table 1, the CGC data-path achieves better performance than the data-path consisting of templates, since fewer clock cycles are required to implement the considered benchmarks.

For this experiment an *area comparison* was also performed. The area consumption in the template-based data-path is the number of instantiated templates required for the mapping of all benchmarks with the scheduling and binding procedures previously presented. The number of multipliers and ALUs inside the CGCs and the templates has been enumerated. This number is a very good approximation of the area consumption for both data-paths, since they are considered resource dominated circuits, as they target DSP applications. There is an area increase of 62.5% (33.3%) in the multiplier's area and 37.5% (25.0%) for the case of the template-based data-path when compared with a CGC one consisting of the two 2x2 (2x3) components. So, the CGC data-path consumed less area, when a family of applications is mapped to it. Also, the number of the template instances for the template-based data-path has been enumerated. It has been found that, in average, approximately 7 (10) template instances are required when 4 (6) operations can be executed concurrently in each c-step. On the other hand, 2 CGCs instances are required in both cases. So, due the absence of flexible templates (like the CGCs) the generated template-based data-paths are realized by a large number of template instances preventing the adoption of direct inter-template connections and thus the inter-template chaining exploitation.

**Table 2.** Clock cycles (latency) results when the proposed data-path is compared with a template-based one (template full matching + FPGA primitive resources are enabled)

DFG	Template-based		CGC-based		% cycles decrease	
	4 res.	6 res.	two 2x2	two 2x3	4 res.	6 res.
ellip	12	12	6	6	50.0	50.0
fir11	11	9	6	6	45.4	33.3
nc	17	12	10	7	41.2	41.7
volterra	10	10	6	6	40.0	40.0
wavelet	19	14	9	7	52.6	50.0
wdf7	13	13	7	7	46.2	46.2
jpeg	134	91	81	54	39.5	40.6
ofdm	55	37	34	23	38.2	37.8
gsm_enc	196	103	103	69	47.5	33.0
gsm_dec	155	130	109	73	29.8	43.8
mpeg2	20	14	9	6	55.0	57.1
rasta	33	21	19	13	42.4	38.1
<b>Average values</b>					<b>44.0</b>	<b>42.6</b>

Finally, a *fourth experiment* (its results are shown in Table 2) was held to compare the performance of the CGC data-path with a template-based one, when the partial matching of templates is *not* enabled. So, in this case there are uncovered DFG nodes, since only the full matching of the templates is enabled, as in the templates of [4]. The uncovered nodes of the DFG are assumed to be implemented by primitive resources implemented in FPGA technology, as in [4]. The template library is the same as the

one in the third experiment. The clock period of the ASIC components  $T_{ASIC}$  (i.e. the templates) is set to the delay of the *multiply-multiply* template for each synthesized data-path. In the case of the FPGA hardware, the clock period  $T_{FPGA}$  is set to the delay of the *multiplier* unit. In this experiment we assume that  $T_{FPGA} = 2 \cdot T_{ASIC}$ , which a rather modest assumption for the performance gain of an ASIC technology compared to an FPGA technology. To simplify the synchronization problems between the FPGA and the ASIC hardware (due to the presence of two clocks), we assume: (a) a closely coupled ASIC template-based data-path and FPGA hardware, and (b) a clock period set to  $T_{ASIC}$ . So, the DFG operations mapped to the FPGA hardware have an execution delay of 2 clock cycles, and the ones mapped to the template data-path have unit-execution delay. As deduced from the results of Table 2, the performance improvement is even greater (approximately 43%) when the CGC data-path is compared with a template-based one that does not support partial matching.

## 6 Related Work

The Pleiades [2] architecture is a hybrid reconfigurable platform that combines an on-chip microprocessor with a number of heterogeneous reconfigurable units of different granularities connected via a reconfigurable interconnection network. These units are mainly MACs, ALUs, and an embedded FPGA. Although promising results, especially in power-consumption, have been reported, Pleiades suffers by a major drawback. Particularly, no systematic automated methodology exists for mapping an application or a family of applications on its architecture.

Corazao et al. [3], assuming a given library of optimally designed templates, proposed a methodology for selecting the proper templates to realize the critical path of the DFG and thus improving the performance. The reported results demonstrate high performance gains with an affordable area increase. Although many optimization techniques were utilized as part of the synthesis strategy, template selection had the largest impact on overall improvement in throughput.

Kastner et al. [4] addressed the automatic generation of complex instructions for an application domain. The templates implemented in ASIC technology are embedded in a hybrid reconfigurable system consisting of the templates and FPGA units. The clustering of operations is based on the frequency of appearance of successive operations e.g. multiplications followed by additions. They observe that the number of operations per template is small and conclude that simple pairs of operations are the best templates for DSP applications. However, as FPGA units implement the uncovered DFG operations, the system's performance is reduced and the power consumption is increased.

Cong et al. [5] addressed the problem of generating the application-specific instructions for configurable processors aiming at improving delay. The instruction generation considers only Multiple-Input Single Output (MISO) format, which cannot take into advantage register files with more than one write port. The pattern library is selected to maximize the potential speedup, subject to a total area constraint. Nevertheless, this does not exclude the generation of a large number of different patterns, which complicates the step of application mapping. The mapping stage is formulated as a minimum-area technology mapping problem and it is solved by a

binate covering problem, which is an NP-hard problem. So, the complexity of the application mapping is an important disadvantage of their approach.

## 7 Conclusions and Future Work

A high-performance reconfigurable coarse-grained data-path - part of a mixed granularity platform architecture - together with a methodology for mapping applications to this data-path, has been presented in this paper. Important performance gains have been achieved compared with primitive and template-based data-paths. On going work focuses on the development of a methodology for accelerating loop structures (like *for*, *while*, *do-while* structures) when these are mapped to a CGC data-path. In future work, an automated methodology for temporal partitioning tailored to the requirements of the CGC data-path, will be developed.

## References

1. Hartenstein, R.: A Decade of Reconfigurable Computing: A Visionary Retrospective. Proc. of Design and Test in Europe (DATE) (2001) 642-649
2. Wan, M., et al.: Design methodology of a low-energy reconfigurable single-chip DSP system. Journal of VLSI Signal Processing, vol.28, no.1-2. (2001) 47-61
3. Corazao, M. R., et al.: Performance Optimization Using Template Mapping for Datapath-Intensive High-Level Synthesis. IEEE Transactions on Computer Aided Design, vol.15, no. 2 (1996) 877-888
4. Kastner, R., et al.: Instruction Generation for Hybrid Reconfigurable Systems. ACM Transactions on Design Automation of Embedded Systems (TODAES), vol. 7, no.4 (2002) 605-627
5. Cong, J., et al.: Application-Specific Instruction Generation for Configurable Processor Architectures. Proc. of the ACM Int. Symposium on FPGA (2004) 183-189
6. Cardoso, J. M.: On Combining Temporal Partitioning and Sharing of Functional Units in Compilation for Reconfigurable Architectures. IEEE Trans. on Computers, vol. 52, no. 10 (2003) 1362-1375
7. Leiserson, C.E.: Fat-Trees: Universal Networks for Hardware Efficient Supercomputing. IEEE Transactions on Computers, vol. 43, no. 10 (1985) 892-901
8. Synopsys Design Compiler<sup>®</sup>: [www.synopsys.com](http://www.synopsys.com) (2004)
9. Gajjala Purna, K.M. and Bhatia, D.: Temporal Partitioning and Scheduling Data Flow Graphs for Reconfigurable Computers. IEEE Transactions on Computers, vol. 48, no. 6, (1999) 579-590
10. De Micheli, G.: Synthesis and Optimization of Digital Circuits. McGraw-Hill Publishers (1994)
11. Lee, C., et al.: MediaBench: a tool for evaluating and synthesizing multimedia and communications systems. Proc. of Int. Symposium on Microarchitecture (1997)
12. CDFG toolset: <http://poppy.snu.ac.kr/CDFG/cdfg.html> (2004)
13. Hall, M. W., et al.: Maximizing multiprocessor performance with the SUIF compiler. Computer, vol. 29 (1996) 84-89

# Power Estimation for Ripple-Carry Adders with Correlated Input Data

Kenny Johansson, Oscar Gustafsson, and Lars Wanhammar

Department of Electrical Engineering, Linköping University,  
SE-581 83 Linköping, Sweden

Tel: +46-13-28{4059, 4059,1344}, Fax: +46-13-139282  
{kennyj, oscarg, larsw}@isy.liu.se

**Abstract.** In this work modelling of the power consumption for ripple-carry adders implemented in CMOS is considered. Based on the switching activity of each input bit, two switching models, one full and one simplified, are derived. These switching models can be used to derive the average energy consumed for one computation. This work extends previous results by introducing a data dependent power model, i.e., correlated input data is considered. Examples show that the switching model is accurate, while there are some differences in the power consumption. This is due to the fact that not all switching in the ripple-carry adder is rail-to-rail (full swing) in the actual implementation.

## 1 Introduction

Power modelling and estimation of digital circuits have received considerable consideration for the last decade and more [1], [2]. Especially for DSP systems, power modelling with correlated data has been considered [3], [4].

A key component in almost all DSP systems is the binary adder. A basic adder structure is the ripple-carry adder, as shown in Fig. 1. The ripple-carry adder is based on full adder cells which adds the two input bits and the incoming carry bit to yield a result in form of a sum bit and an outgoing carry bit. Numerous low-power full adder cells have been presented during the years and recent comparisons are found in [5] and [6].

There are other adder structures providing higher speeds than the ripple-carry adder, for which the execution time is directly proportional to the data word length. Comparisons in terms of area, time, and power consumption for different adder structures are found in [7] and [8]. However, only numerical results are presented for the power consumption.

In [9] the average power consumption for the ripple-carry adder based on the average length carry chain was derived assuming random inputs. The method in [9] was extended in [10], where a better approximation was derived not only assuming the average length carry chain, but taking the average power for all possible lengths of the carry chains. It was concluded that this model followed the results of the power simulation tool HEAT [11] closely. Again, random inputs were assumed. Furthermore, models for three other adder structures were derived.

In this work we derive a power model for ripple-carry adders with correlated input data. In real world signals, the switching probabilities are different for different bit

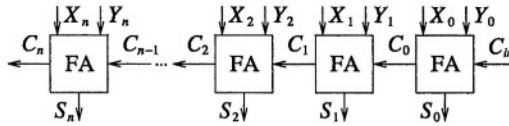


Fig. 1. Ripple-carry adder

positions [3], [4]. Hence, our proposed model provides a more relevant base for higher-level power modelling and estimation of DSP systems.

Two different models are proposed. One is based on a probability matrix, where each transition probability is multiplied with the corresponding switching energy. The other is using a simplified representation, where the average energy consumed by switching of the carry, sum, or both outputs is used. Assuming random inputs our second model will produce identical results to the model in [10], while further simplifications of the model will lead to the model in [9]. However, our model is not based on carry-chains so, apart from supporting correlated data, a different method is used in the derivation.

In the next section, the power model is introduced. Section 3 contains the main part of the contribution, this is where the switching activities are derived. In Section 4 some experimental results are presented and compared with [9] and [10], where applicable. It is shown that the proposed models gives accurate results in terms of switches, while the energy consumption is overestimated due to the fact that not all switching are rail-to-rail. However, this is a problem for the methods in [9] and [10] as well. Finally, in Section 5 some conclusions are drawn.

## 2 Power Model

The state of a full adder, as shown in Fig. 2, is defined as the logic value at its three inputs, in the order  $X_i$ ,  $Y_i$ , and  $C_{i-1}$ . The energy corresponding to every state transition combination at the inputs of a full adder is simulated and stored in a matrix. This is shown in Table 1 for the full adder used for the experimental results in Section 4.

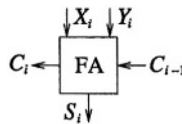


Fig. 2. Full adder used in a ripple-carry adder

In Table 2 the switching activity at the outputs is defined for every state transition. The symbol  $S$  means that the sum output switch,  $C$  means that the carry output switch,  $CS$  means that both the sum and the carry outputs switches, *none* means that the outputs does not switch, and *static* means that not even the inputs switches. The switching activities will be computed for a generate phase and a propagate phase, respectively. In the generate phase the data inputs are changed, and in the propagate phase the carry outputs are stabilized in the chain of full adders. This imply that transitions where both



**Table 1.** Energy, in fJ, corresponding to a transition from the state in the left column to the state in the heading row

	000	001	010	011	100	101	110	111
000	0.624	168.8	179.7	391.4	211.2	512.6	533.8	324.6
001	316.6	0.477	87.32	401.4	106.0	417.5	512.9	146.9
010	368.5	102.7	0.923	458.6	92.03	478.8	441.0	195.4
011	426.9	579.3	555.8	0.713	559.1	103.9	123.8	207.6
100	385.0	123.3	66.09	487.5	1.952	476.9	482.6	239.0
101	439.4	618.8	580.6	78.56	623.1	0.967	163.6	237.9
110	467.4	637.8	634.5	132.6	646.1	122.2	1.008	283.7
111	722.8	605.4	632.2	381.5	519.4	350.6	329.4	2.948

the data inputs and the carry input change are not possible. These transitions are in parentheses in Table 2. One exception from this is the first full adder, corresponding to the LSB, for which the carry input,  $C_{in}$ , may change in the generate phase.

**Table 2.** Resulting switching of the output signals for different transitions of the input signals

	000	001	010	011	100	101	110	111
000	<i>static</i>	<i>S</i>	<i>S</i>	<i>(C)</i>	<i>S</i>	<i>(C)</i>	<i>C</i>	<i>(CS)</i>
001	<i>S</i>	<i>static</i>	<i>(none)</i>	<i>CS</i>	<i>(none)</i>	<i>CS</i>	<i>(CS)</i>	<i>C</i>
010	<i>S</i>	<i>(none)</i>	<i>static</i>	<i>CS</i>	<i>none</i>	<i>(CS)</i>	<i>CS</i>	<i>(C)</i>
011	<i>(C)</i>	<i>CS</i>	<i>CS</i>	<i>static</i>	<i>(CS)</i>	<i>none</i>	<i>(none)</i>	<i>S</i>
100	<i>S</i>	<i>(none)</i>	<i>none</i>	<i>(CS)</i>	<i>static</i>	<i>CS</i>	<i>CS</i>	<i>(C)</i>
101	<i>(C)</i>	<i>CS</i>	<i>(CS)</i>	<i>none</i>	<i>CS</i>	<i>static</i>	<i>(none)</i>	<i>S</i>
110	<i>C</i>	<i>(CS)</i>	<i>CS</i>	<i>(none)</i>	<i>CS</i>	<i>(none)</i>	<i>static</i>	<i>S</i>
111	<i>(CS)</i>	<i>C</i>	<i>(C)</i>	<i>S</i>	<i>(C)</i>	<i>S</i>	<i>S</i>	<i>static</i>

In this work the power consumption will be estimated in two different ways. One method is to compute the probability for each specific state transition, and a simplified method is to compute the probabilities for each kind of transition. For the simplified method the average energies shown in Table 3 are used. As can be seen in this table, a static transition is negligible compared to the other types of transitions, and will therefore not be considered in the power model.

**Table 3.** Average energy, in fJ, corresponding to the different types of transitions

Transition	<i>CS</i>	<i>C</i>	<i>S</i>	<i>none</i>	<i>static</i>
Energy [fJ]	528.0	438.4	285.0	85.14	1.202

### 2.1 Timing Issues

In a full adder there are two computations performed, and the results are given at the carry and sum outputs, respectively. In a ripple-carry adder the carry output from each full adder (except the last full adder) is input to a subsequent full adder. The effect of this dependence is that a fast computation of the carry result in low switching activity at the sum outputs. This relation is shown in Fig. 3, where  $S_{del}$  is the delay corresponding to a sum computation and  $C_{del}$  is the delay corresponding to a carry computation. As a high-level tool, Model Sim™, was used, the switching activity in this figure is quantified so that a different value is obtained depending on the number of carry computations that is performed during one sum computation. Note that the relation of the delays does not effect the carry switching activities, since all carry bits will reach their final value in a certain order, i.e., from LSB to MSB.

In this work all carry switching will be assumed to effect the sum output, i.e.,  $0 < S_{del}/C_{del} \leq 1$  is assumed. This imply that the power consumption will be overestimated. However, this is a realistic model as every change at the carry input will in fact effect the sum output, although not always resulting in full swing switching.

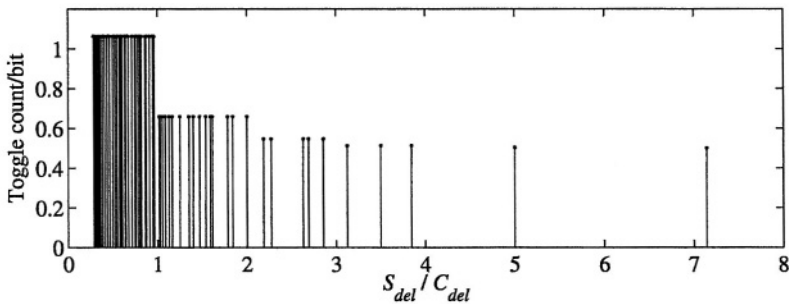


Fig. 3. Toggle count/bit at the sum output of an eight-bit ripple-carry adder.  $S_{del}$  is the delay corresponding to a sum computation and  $C_{del}$  is the delay corresponding to a carry computation

## 3 Switching Activity

In this section the switching activity in ripple-carry adders, as shown in Fig. 1, is computed. First the generate phase is considered, and then the propagate phase.

### 3.1 Switching Due to Change of Input

When new inputdata,  $X_i$  and  $Y_i$ , is applied to a full adder as shown in Fig. 2, the outputs,  $S_i$  and  $C_i$ , of the full adder may change. Note that the carry input,  $C_{i-1}$ , is constant during this phase. This is visualized in Fig. 4 using a state transition graph (STG), where the state represents the input values. Each transition between two different states represent a specific kind of switching activity at the outputs. For example, a transition between state 2 and state 6 result in a transition at both outputs,  $S_i$  and  $C_i$ . Note that the situations where no changes of the outputs occur can be divided into the case when the inputs does not change (*none*), and the case when both inputs change (*static*). One

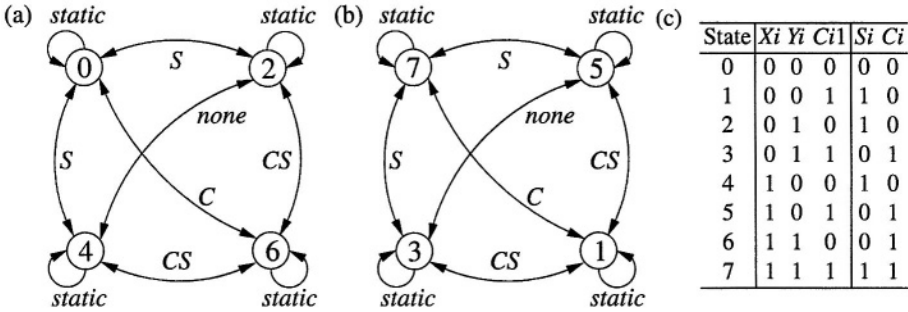


Fig. 4. State transition graph for a full adder. (a)  $C_{i-1} = 0$ , (b)  $C_{i-1} = 1$ , and (c) Truth table

exception from the STG in Fig. 4 is the first full adder (LSB), for which all three inputs may change during the generation phase.

**Computation of Transition Probabilities in the Generation Phase.** The goal with the following computations is to find the probability for each possible transition, i.e., obtain a probability matrix corresponding to Table 2. The row index  $j$ , column index  $k$ , and full adder index  $i$  will be used. The probability function,  $P(x)$ , will be used to state the probability that a signal,  $x$ , have the logic value one. The function,  $\alpha(x)$ , will be used to state the switching activity of the signal  $x$ . The input data,  $X_i$  and  $Y_i$ , are assumed to be random in the sense that the probabilities for zeros and ones are equal, i.e.

$$P(X_i) = P(Y_i) = \frac{1}{2} \tag{1}$$

The initial state (superscript  $I$ ) and final state (superscript  $F$ ) are defined by the input values as

$$\begin{cases} S_{i,j}^I = [X_{i,j}^I, Y_{i,j}^I, C_{i-1,j}^I], & X_{i,j}^I, Y_{i,j}^I, C_{i-1,j}^I \in \{0, 1\}, & 0 \leq j \leq 7 \\ S_{i,k}^F = [X_{i,k}^F, Y_{i,k}^F, C_{i-1,k}^F], & X_{i,k}^F, Y_{i,k}^F, C_{i-1,k}^F \in \{0, 1\}, & 0 \leq k \leq 7 \end{cases} \tag{2}$$

By applying the *xor* function to the state descriptions, a vector containing ones for changed and zeros for unchanged inputs is obtained as

$$\alpha_{i,j,k} = S_{i,j}^I \otimes S_{i,k}^F, \begin{cases} 0 \leq j \leq 7 \\ 0 \leq k \leq 7 \end{cases} \tag{3}$$

Due to the statement in (1), the carry input,  $C_{i-1}$ , is the only input that effects the state probabilities. Hence, there are only two different state probabilities, which are defined as

$$\begin{cases} P(S_i^0) = \frac{1}{4}(1 - P(C_{i-1})) \\ P(S_i^1) = \frac{1}{4}P(C_{i-1}) \end{cases}, \quad P(C_{i-1}) = \begin{cases} P(C_{in}), & i = 0 \\ \frac{1}{4} + \frac{1}{2}P(C_{i-2}), & i > 0 \end{cases} \tag{4}$$

where the superscripts, 0 and 1, indicates the logic value of the carry input.

Elements in the probability matrix are denoted by the variable  $q$ , with indices to describe full adder, row, and column. Each value,  $q$ , is obtained by multiplying the corresponding state probability, according to (4), with the input switching activities,  $\alpha(X_i)$  and  $\alpha(Y_i)$ . This is done with respect to the switching vector defined in , so that the complementary switching activity,  $1 - \alpha(X_i)$  and  $1 - \alpha(Y_i)$ , is used if the corresponding input does not change. For the first full adder (LSB) the carry input also has to be considered. Hence, each matrix element can be computed as

$$q_{i,j,k}^{gen} = (C_{i-1,j}^1 P(S_i^1) \overline{C_{i-1,j}^1} P(S_i^0)) \cdot \left\{ \begin{array}{l} 0 \leq j \leq 7 \\ 0 \leq k \leq 7 \\ \alpha(C_{i,j,k}^{gen}) = \begin{cases} \alpha(C_{in}), i = 0 \\ 0, i > 0 \end{cases} \end{array} \right. \cdot \left\{ \begin{array}{l} (\alpha_{i,j,k}(1)\alpha(X_i) + \overline{\alpha_{i,j,k}(1)}(1 - \alpha(X_i))) \\ (\alpha_{i,j,k}(2)\alpha(Y_i) + \overline{\alpha_{i,j,k}(2)}(1 - \alpha(Y_i))) \\ (\alpha_{i,j,k}(3)\alpha(C_{i,j,k}^{gen}) + \overline{\alpha_{i,j,k}(3)}(1 - \alpha(C_{i,j,k}^{gen}))) \end{array} \right. \quad (5)$$

### 3.2 Switching Due to Carry Propagation

In the propagate phase the data inputs,  $X_i$  and  $Y_i$ , are constant, while the carry input,  $C_{i-1}$ , may change several times. In Fig. 5 the STG corresponding to the propagate phase is shown. Note that all cases without changes of the outputs are static.

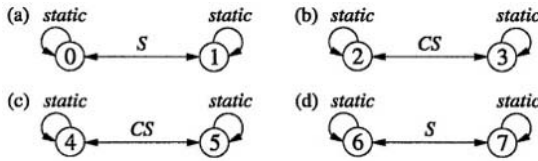


Fig. 5. State transition graph for a full adder when the carry input is active. (a)  $X_i = Y_i = 0$ , (b)  $X_i = 0, Y_i = 1$ , (c)  $X_i = 1, Y_i = 0$ , and (d)  $X_i = Y_i = 1$

**Computation of Transition Probabilities in the Propagate Phase.** In the following the propagate part of the probability matrix will be derived. As there are no transition at the inputs of the first full adder (LSB) in the propagate phase, the full adder index,  $i$ , is larger than zero in this section.

Let the variable  $\beta$  represent a weighted value, divided into equal parts for the eight different states, of the total carry switching activity at the carry input, i.e., both  $C$  and  $CS$  transitions are included.  $\beta$  is then computed using the probability matrix of the previous adder as

$$\beta_i = \frac{1}{8} \sum_{j,k} q_{i-1,j,k}, (j, k) \in \{C, CS\} \text{ where } q_{i,j,k} = q_{i,j,k}^{gen} + q_{i,j,k}^{pro}, \left\{ \begin{array}{l} 0 \leq j \leq 7 \\ 0 \leq k \leq 7 \end{array} \right. \quad (6)$$

Since only the carry input can change in the generation phase, transitions are limited within  $2 \times 2$  parts of the probability matrix. These transitions are described by

$$\begin{bmatrix} q_{i,j,k}^{pro} = iP(S_i^0) - \beta_i & q_{i,j,k+1}^{pro} = \beta_i \\ q_{i,j+1,k}^{pro} = \beta_i & q_{i,j+1,k+1}^{pro} = iP(S_i^1) - \beta_i \end{bmatrix}, \quad (j, k) \in \left\{ \begin{matrix} (0, 0), (2, 2) \\ (4, 4), (6, 6) \end{matrix} \right\}, \quad (7)$$

and all matrix elements,  $q_{i,j,k}^{pro}$ , not covered by (7) are zero.

### 3.3 Total Switching Activity

By adding the contributions from the generate and propagate phases, the total switching activity is obtained, as described in (6). The resulting probability matrix, for an arbitrary full adder (except for the first), is shown in Table 4. In this table the probabilities corresponding to static transitions are excluded as these are ignored in the power model. When the probability matrix for each full adder is derived, the total energy can be computed by summing all products obtained by multiplying every matrix element with the corresponding transition energy as presented in Table 1. This method will be referred to as the matrix model. As this is a comparatively complex method, a simplified model will be discussed in the following.

**Table 4.** Probability matrix for  $i > 0$ . The full adder index,  $i$ , is not included in this table

	000	001	010	011	100	101	110	111
000	<i>static</i>	$\beta$	$P(S^0)\alpha(Y)$ $(1-\alpha(X))$	0	$P(S^0)\alpha(X)$ $(1-\alpha(Y))$	0	$P(S^0)\alpha(X)$ $\alpha(Y)$	0
001	$\beta$	<i>static</i>	0	$P(S^1)\alpha(Y)$ $(1-\alpha(X))$	0	$P(S^1)\alpha(X)$ $(1-\alpha(Y))$	0	$P(S^1)\alpha(X)$ $\alpha(Y)$
010	$P(S^0)\alpha(Y)$ $(1-\alpha(X))$	0	<i>static</i>	$\beta$	$P(S^0)\alpha(X)$ $\alpha(Y)$	0	$P(S^0)\alpha(X)$ $(1-\alpha(Y))$	0
011	0	$P(S^1)\alpha(Y)$ $(1-\alpha(X))$	$\beta$	<i>static</i>	0	$P(S^1)\alpha(X)$ $\alpha(Y)$	0	$P(S^1)\alpha(X)$ $(1-\alpha(Y))$
100	$P(S^0)\alpha(X)$ $(1-\alpha(Y))$	0	$P(S^0)\alpha(X)$ $\alpha(Y)$	0	<i>static</i>	$\beta$	$P(S^0)\alpha(Y)$ $(1-\alpha(X))$	0
101	0	$P(S^1)\alpha(X)$ $(1-\alpha(Y))$	0	$P(S^1)\alpha(X)$ $\alpha(Y)$	$\beta$	<i>static</i>	0	$P(S^1)\alpha(Y)$ $(1-\alpha(X))$
110	$P(S^0)\alpha(X)$ $\alpha(Y)$	0	$P(S^0)\alpha(X)$ $(1-\alpha(Y))$	0	$P(S^0)\alpha(Y)$ $(1-\alpha(X))$	0	<i>static</i>	$\beta$
111	0	$P(S^1)\alpha(X)$ $\alpha(Y)$	0	$P(S^1)\alpha(X)$ $(1-\alpha(Y))$	0	$P(S^1)\alpha(Y)$ $(1-\alpha(X))$	$\beta$	<i>static</i>

The aim of this simplified method is to compute the probabilities for each kind of transition, and then compute the power consumption using the average energies presented in Table 3. The different types of switching activities are named,  $\alpha(\text{transition}_i)$ , where *transition* correspond to the symbols used in Table 2. By summing all  $C$  transitions (see Table 2) in Table 4, the switching activity for this type is obtained as

$$\alpha(C_i) = 2 (P(S_i^0) + P(S_i^1)) \alpha(X_i)\alpha(Y_i) = \frac{1}{2} \alpha(X_i)\alpha(Y_i), \quad i > 0, \quad (8)$$

for each full adder. The switching activity corresponding to  $S$  transitions is computed as

$$\begin{aligned} \alpha(S_i) &= 4\beta_i + 2(P(S_i^0) + P(S_i^1))(\alpha(X_i)(1 - \alpha(Y_i)) + \alpha(Y_i)(1 - \alpha(X_i))) = \\ &= 4\beta_i + \frac{1}{2}(\alpha(X_i) + \alpha(Y_i)) - 2\alpha(C_i), \quad i > 0 \end{aligned} \tag{9}$$

Finally, it is established that the switching activities corresponding to the two remaining types of transitions are obtained directly from (8) and (9) as

$$\alpha(\text{none}_i) = \alpha(C_i), \quad \alpha(CS_i) = \alpha(S_i), \quad i > 0. \tag{10}$$

In (9) the variable  $\beta_i$  is required. To compute this value according to (6) it is necessary to produce the probability matrix. However, the expression in (6) can now be simplified to

$$\beta_i = \frac{1}{8}(\alpha(C_{i-1}) + \alpha(CS_{i-1})), \quad i > 0. \tag{11}$$

The transition activities are now fully specified for all full adders except the first (LSB). All elements in the probability matrix corresponding to the first full adder are computed according to (5), as there are no transitions in the propagation phase. From the probability matrix, the different types of transition activities are obtained as

$$\begin{cases} \alpha(C_0) = \frac{1}{2}(\alpha(X_0)\alpha(Y_0) + \alpha(X_0)\alpha(C_{in}) + \alpha(Y_0)\alpha(C_{in}) - 3\alpha(X_0)\alpha(Y_0)\alpha(C_{in})) \\ \alpha(S_0) = \frac{1}{2}(\alpha(X_0) + \alpha(Y_0) + \alpha(C_{in}) - 3\alpha(X_0)\alpha(Y_0)\alpha(C_{in})) - 2\alpha(C_0) \\ \alpha(\text{none}_0) = \alpha(C_0) \\ \alpha(CS_0) = \alpha(S_0) + \alpha(X_0)\alpha(Y_0)\alpha(C_{in}) \end{cases} \tag{12}$$

Note that the switching activities in (12) are equal to the corresponding formulas in (8), (9), and (10) if  $\alpha(C_{in})$  and  $\beta_i$  are zero.

### 3.4 Uncorrelated Input Data

If uncorrelated input data,  $X_i$  and  $Y_i$ , are assumed, i.e.,  $\alpha(X_i) = \alpha(Y_i) = 1/2$ , the transition activities can be simplified. Under this assumption the transition activities in the first full adder can be derived from (12) as

$$\begin{cases} \alpha(C_0) = \frac{1}{8}(1 + \alpha(C_{in})) \\ \alpha(S_0) = \frac{1}{8}(2 - \alpha(C_{in})) \end{cases}, \quad \begin{cases} \alpha(\text{none}_0) = \alpha(C_0) \\ \alpha(CS_0) = \frac{1}{8}(2 + \alpha(C_{in})) \end{cases} \tag{13}$$

From (8) and (9) the switching activities corresponding to  $C$  and  $S$  transitions for the remaining full adders are computed as

$$\alpha(C_i) = \frac{1}{8}, \quad \alpha(S_i) = 4\beta_i + \frac{1}{4}, \quad i > 0 \tag{14}$$

By summing the contribution from all full adders, a closed-form expression of the total switching activity corresponding to  $C$  transitions,  $\alpha(C_{tot})$ , is obtained as

$$\alpha(C_{tot}) = \alpha(C_0) + \sum_{i=1}^n \alpha(C_i) = \frac{1}{8}(1 + n + \alpha(C_{in})), \quad (15)$$

where  $n + 1$  is the number of full adders, as shown in Fig. 1.

In (14) the variable  $\beta_i$  is required. This value is computed according to (11) as

$$\begin{cases} \beta_i = \frac{1}{8}(\alpha(C_0) + \alpha(CS_0)) = \frac{1}{64}(3 + 2\alpha(C_{in})) \\ \beta_i = \frac{1}{8} \left( \frac{1}{8} + \beta_{i-1} + \frac{1}{4} \right) = \frac{1}{64} + \frac{\beta_{i-1}}{2}, \quad i > 1 \end{cases} \quad (16)$$

The total switching activity corresponding to  $S$  transitions,  $\alpha(S_{tot})$ , is obtained by summing the contribution from all full adders as

$$\begin{aligned} \alpha(S_{tot}) &= \alpha(S_0) + \sum_{i=1}^n \alpha(S_i) = \alpha(S_0) + \frac{n}{4} + 4 \sum_{i=1}^n \beta_i = \\ &= \alpha(S_0) + \frac{n}{4} + 4 \left( \frac{3n}{32} + 2 \left( \beta_i - \frac{3}{32} \right) (1 - 2^{-n}) \right) = \\ &= \frac{1}{8} \left( 2 + 5n - \alpha(C_{in}) + (2\alpha(C_{in}) - 3) (1 - 2^{-n}) \right) \end{aligned} \quad (17)$$

Finally, the switching activity for the two remaining types of transitions are obtained as

$$\alpha(none_{tot}) = \alpha(C_{tot}), \quad \alpha(CS_{tot}) = \alpha(S_{tot}) + \frac{1}{4}\alpha(C_{in}). \quad (18)$$

**Uncorrelated Carry Input.** In this section earlier presented models are deduced by simplifying this new model. If the carry input,  $C_{in}$ , is assumed to be random, i.e.  $\alpha(C_{in}) = 1/2$ , the total switching activities in (15), (17), and (18) are reduced to

$$\begin{cases} \alpha(C_{tot}) = \frac{1}{16}(3 + 2n) \\ \alpha(S_{tot}) = \frac{1}{8} \left( 5n - \frac{1}{2} + 2^{1-n} \right) \end{cases}, \quad \begin{cases} \alpha(none_{tot}) = \alpha(C_{tot}) \\ \alpha(CS_{tot}) = \alpha(S_{tot}) + \frac{1}{8} \end{cases} \quad (19)$$

In the model presented in [10], an equation comparable to (19) was given. If, for example,  $C_{in}$  is constant, i.e., 0 (adder) or 1 (subtractor), the value of  $\alpha(C_{in})$  is zero. This case is handled by the closed-form expressions in (15), (17), and (18), but not by (19).

For uncorrelated input data it can be shown that

$$\begin{aligned} \lim_{i \rightarrow \infty} \beta_i &= \frac{3}{32} \Rightarrow \lim_{i \rightarrow \infty} \alpha(S_i) = \lim_{i \rightarrow \infty} \alpha(CS_i) = \frac{5}{8}, \\ \text{and } \lim_{i \rightarrow \infty} \alpha(C_i) &= \lim_{i \rightarrow \infty} \alpha(none_i) = \frac{1}{8}. \end{aligned} \quad (20)$$

The values (20) in was used in the model presented in [9], where the power estimation consequently was stated as a function of the word length without considering each full adder individually.

### 3.5 Summary

The differences between the discussed models are summarized in Table 5, where the model name TMMM stands for Transition Model based on Matrix Model and denotes the simplified model.

**Table 5.** Comparison of different power models for ripple-carry adders

Model	Equations	Energy	Inputs	Carry
Carry chain [9]	(20)	Table 3	Random	–
Carry chain II [10]	(19)	Table 3	Random	Random
TMMM Random	(15), (17), (18)	Table 3	Random	Corr.
TMMM Corr.	(8) – (12)	Table 3	Corr.	Corr.
Matrix	(2) – (7)	Table 1	Corr.	Corr.

## 4 Experimental Results

To validate the model, transistor level simulations were performed for an AMS 0.35  $\mu\text{m}$  CMOS process using Spectre<sup>TM</sup>. The full adder used was a *mirror adder* from [12]. The delay relation, which was discussed in Section 2.1, for this full adder was  $1 < S_{del}/C_{del} < 2$ . Hence, not all switching in the transistor level simulation will be rail-to-rail.

The energy consumed for each state change of the full adder is shown in Table 1, while the average energy for each change of the outputs is shown in Table 2. In this characterization the sum output was loaded with an inverter, while the carry output was loaded with the carry input of an identical full adder. All simulation results are average values over 1000 pairs of input samples at a data rate of 50 MHz.

### 4.1 Uncorrelated Data

The first validation is for uncorrelated input data. Here, we consider both constant and random  $C_{in}$ . The results are shown in Table 6, where it is also compared with the results obtained from [9] and [10].

It is worth noting that for random  $C_{in}$  the method in [10] and the proposed simplified (TMMM) method gives identical results, as previously discussed. The differences between the proposed methods and the Spectre<sup>TM</sup> results are due from the non rail-to-rail switching present in the transistor level simulation. In [10] the results were compared to those obtained by HEAT [11], and a close correspondence were shown. However,



**Table 6.** Average power consumption, in  $\mu\text{W}$ , for a ripple-carry adder. Simulated and derived by the proposed methods, as well as from [9] and [10]

Bits ( $n + 1$ )	Constant $C_{in} = 0$			Random $C_{in}$				Any $C_{in}$ Carry chain
	Sim.	TMMM	Matrix	Sim.	Carry chain II	TMMM	Matrix	
4	73.74	86.13	87.10	76.69	92.97	92.97	92.88	114.7
8	168.2	199.1	199.6	168.9	206.5	206.5	205.9	229.4
12	258.8	313.7	313.7	260.0	321.1	321.1	320.1	344.1
16	347.3	428.4	428.0	352.5	435.8	435.8	434.3	458.9
20	440.1	543.1	542.2	443.1	550.6	550.6	548.5	573.6
24	527.2	657.8	656.4	533.1	665.3	665.3	662.7	688.3
32	710.1	887.2	884.9	715.0	894.7	894.7	891.2	917.7

**Table 7.** Average power consumption, in  $\mu\text{W}$ , for individual full adders in a four-bit ripple-carry adder with random  $C_{in}$ 

FA no.	Sim.	Carry chain [9]	Carry chain II [10]	TMMM	Matrix
1	15.94	28.68	15.83	15.83	15.99
2	19.03	28.68	23.60	23.60	23.54
3	21.05	28.68	26.14	26.14	26.05
4	20.62	28.68	27.41	27.41	27.30

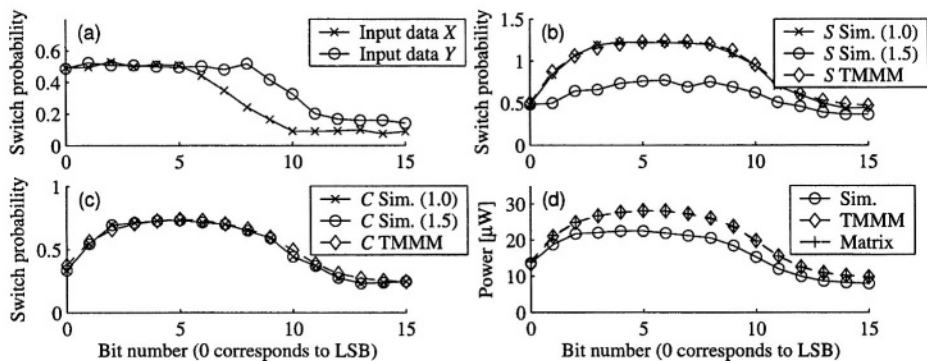
HEAT is based on switching activities as well. Hence, all switching are assumed to be full swing in HEAT.

When each individual full adder is considered the results in Table 7 are obtained for a four-bit ripple-carry adder. The differences for the most significant full adder is partly due to that the carry output is now connected to an inverter. Note that the proposed methods are close to the simulation results for the first full adder, due to the fact that all switching in this full adder is rail-to-rail.

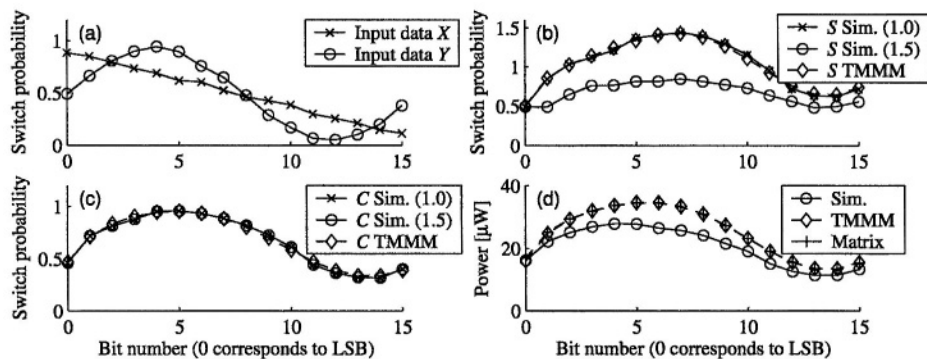
## 4.2 Correlated Data

The main advantage of the proposed method over previous methods is the ability to estimate the power consumption for correlated data. Here, two different cases of correlated input signals are presented.

First, two input signals with typical two's-complement correlation, present in many real world applications are applied. The switching probabilities for the input signals are shown in Fig. 6(a). In Figs. 6(b) and (c) the resulting switching probabilities for each sum and carry signals are shown, respectively. These are compared with simulated results from a VHDL model in Model Sim<sup>TM</sup>, using two different delay relations. For the simulation using the same delay relation as the model was based upon, there is a good fit for the switching activities at the outputs. The difference of the switching activities at the sum outputs obtained in Model Sim<sup>TM</sup> is due to the number of full swing switches. Hence, the simulation with  $S_{del}/C_{del} \approx 1.0$  shows the total number of switches, while



**Fig. 6.** Switching activities of single bits for (a) input signals, (b) sum output, and (c) carry output. (d) Power consumption at 50 MHz. In (b) and (c) the value in parentheses for the simulations is the delay relation,  $S_{del}/C_{del}$ , which was discussed in Section 2.1



**Fig. 7.** Switching activities of single bits for (a) input signals, (b) sum output, and (c) carry output. (d) Power consumption at 50 MHz. In (b) and (c) the value in parentheses for the simulations is the delay relation,  $S_{del}/C_{del}$ , which was discussed in Section 2.1

the simulation with  $S_{del}/C_{del} = 1.5$  shows the number of full swing switches. As was stated in Section 2.1, the delay relation is shown not to effect the switching activities at the carry outputs. Finally, in Fig. 6(d), the estimated power consumption for each full adder cell is shown together with the transistor level simulation results. As can be seen in Fig. 6(d) the power consumption is overestimated. Again, the differences are due to not all switching in the transistor level simulation being full swing. The differences between the full matrix method and the simplified TMMM method is small for this case, mainly due to the symmetry between the  $X$  and  $Y$  inputs of the used full adder. Hence, the average energy consumption in Table 3 is useful. However, for full adders with non-symmetric  $X$  and  $Y$  inputs one can expect a larger deviation between the proposed methods, with the matrix method producing more accurate results.

To further validate the model, two more unrealistic input signals are applied. The switching probabilities for these signals are shown in Fig. 7(a). The switching activity

of the sum and carry outputs are shown in Figs. 7(b) and (c), respectively. Again, there is a close match between the simulated results and the proposed method. Figure 7(d) shows the estimated and simulated power consumption.

## 5 Conclusions

In this work a data dependent switching activity model was proposed for the ripple-carry adder. Previous work have only considered random input data, but for most applications the input data is correlated. Hence, the proposed method could be included in high-level power estimation to give more accurate results. It was shown by examples that the proposed model was accurate in estimating the switching activity of the output signals. However, the power was overestimated as not all switching in the implemented adder were rail-to-rail (full swing). Further work include data dependent power models for other adder structures.

## References

- [1] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. VLSI Systems*, vol. 2, no. 4, pp. 446–455, Dec. 1994.
- [2] E. Macii, M. Pedram, and F. Somenzi, "High-level power modeling, estimation, and optimization," *IEEE Trans. Computer-Aided Design*, vol. 17, no. 11, pp. 1061–1079, Nov. 1998.
- [3] M. Lundberg, K. Muhammad, K. Roy, and S. K. Wilson, "A novel approach to high-level swithing activity modeling with applications to low-power DSP system synthesis," *IEEE Trans. Signal Processing*, vol. 49, no. 12, pp. 3157–3167, Dec. 2001.
- [4] P. E. Landman and J. M. Rabaey, "Activity-sensitive architectural power analysis," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 6, pp. 571–587, June 1996.
- [5] A. M. Shams, T. K. Darwish, and M. A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adders cells," *IEEE Trans. VLSI Systems*, vol. 10, no. 1, pp. 20–29, Feb. 2002.
- [6] M. Alioto and G. Palumbo, "Analysis and comparison on full adder block in submicron technology," *IEEE Trans. VLSI Systems*, vol. 10, no. 6, pp. 806–823, Dec. 2002.
- [7] T. K. Callaway and E. E. Swartzlander, Jr., "Estimating the power consumption of CMOS adders," *Proc. 11th Symp. on Comp. Arithmetic*, 1993, pp. 210–216.
- [8] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst.-II*, vol. 43, no.10, pp. 689–702, Oct. 1996.
- [9] L. A. Montalvo, J. H. Satyanarayana, and K. K. Parhi, "Estimation of average energy consumption of ripple-carry adder based on average length carry chains," *Proc. IEEE Workshop on VLSI Signal Processing*, 1996, pp. 189–198.
- [10] R. Freking and K. K. Parhi, "Theoretical estimation of power consumption in binary adders," *Proc. IEEE Int. Symp Circuits Syst.*, Hong Kong, 1998, vol. II, pp. 453–457.
- [11] J. H. Satyanarayana and K. K. Parhi, "Power estimation of digital data paths using HEAT," *IEEE Design & Test of Computers*, vol. 17, no. 2, pp. 101–110, April 2000.
- [12] J. M. Rabaey, *Digital Integrated Circuits - A Design Perspective*, Prentice Hall, 1996.

# LPVIP: A Low-Power ROM-Less ALU for Low-Precision LNS

Mark G. Arnold

Lehigh University, Bethlehem PA 18015, USA,  
marnold@eecs.lehigh.edu

**Abstract.** A new low-precision ALU suitable for a variety of multimedia and DSP applications performs Logarithmic Number System (LNS) additions and most LNS subtractions without using a ROM. An example application is described in which the Low Precision Very Insignificant Power (LPVIP) ALU described here uses fewer gates than an equivalent fixed-point ALU to accomplish a visually similar video output for MPEG decoding. The switching activity of the proposed LNS circuit is an order of magnitude lower than that of a general-purpose fixed-point multiply-accumulate circuit, and about half of the switching activity of a specialized fixed-point video circuit.

## 1 LNS

The signed Logarithmic Number System (LNS) [5] represents a real value by a sign bit and the base- $b$  logarithm of its value. Typically  $b$  is 2. The logarithm field is a finite-precision truncation of the exact logarithm. Multiplication, division and powers work on the same principle as the slide rule, and introduce no additional error. This error behavior is superior to that of conventional fixed- and floating-point arithmetics.

Mitchell [8] suggested a very low-cost approach to approximating the base-2 logarithm and antilogarithm without needing a table or iteration. Mitchell's technique (described in Section 3) is accurate to about four bits. There are more accurate techniques [3], but none are as economical as Mitchell's method.

Logarithm and antilogarithm approximations have been employed as a replacement for a fixed-point multiplier in the method of the slide rule: take the logarithms of the numbers to be multiplied, add the logarithms and finally take the antilogarithm of the sum to form the fixed-point product. Later accumulation of products, as occurs in DSP applications [15], happens in conventional fixed-point. The problem with the log-add-antilog approach is two fold: it is cumbersome (many repetitive log- and antilog-calculation steps) for more complex algorithms, and it fails to capitalize on the information-compression properties of the logarithm function. These properties have been recognized as reducing bit-switching activity and thereby the power consumption of LNS-based memory systems [9].

In order to capitalize on the advantages of LNS, it is necessary to keep numbers in logarithmic representation throughout the entire computation. Thus,

the logarithm function is used only at input, and the antilogarithm function is used only at output.

The issue then is how addition and subtraction work if they cannot be done in fixed point. In fact, a clever bit of nineteenth-century mathematics proposed by Leonelli [7] deals with LNS addition and subtraction. Given  $x = \log_b(X)$  and  $y = \log_b(Y)$ , we can compute the logarithm of the *sum* with  $\log_b(X + Y) = y + s_b(x - y)$  and we can compute the logarithm of the *difference* with  $\log_b |X - Y| = y + d_b(x - y)$ , where

$$s_b(z) = \log_b(1 + b^z) \quad \text{and} \quad d_b(z) = \log_b |1 - b^z|. \quad (1)$$

Thus, we have reduced the problem to that of approximating these functions.

Early LNS implementations [5] used ROM to lookup  $s_b$  and  $d_b$ . When designed for  $F$ -bit accuracy, a simple  $s_b$  ROM requires about  $F^2 \cdot 2^F$  bits. Visual evidence [4] suggests that low-precision LNS ( $2 \leq F \leq 5$ ) may produce acceptable results for Motion Picture Experts Group (MPEG) video decoding using the Inverse Discrete Cosine Transform (IDCT).

This paper introduces a novel Low Precision Very Insignificant Power (LPVIP) ALU that is based on a combination of Mitchell's method, Leonelli's addition algorithm, and minimal logic to mitigate the errors in the proposed approximation. No one has described such a combination like LPVIP. The closest designs are low-precision modified interpolators. Taylor suggested a low-precision, multiplierless interpolator [14] that yields three to four extra bits using two ROMs. Arnold [1] suggested an improved interpolator of similar accuracy using one ROM. Both [14] and [1] dealt only with  $s_b$ . In contrast, the LPVIP approach disclosed here uses no ROM, except for values of  $d_b$  near zero.

## 2 Power Consumption

Complementary Metal Oxide Semiconductor (CMOS) circuits consume power in two ways. First, regardless of frequency, the circuit consumes some power. Second, there is the power consumed that varies according to frequency. This dynamic power consumption for a circuit composed of simple gates (such as ANDs/ORs) can be estimated [6,9] as

$$P = f_{clk} \cdot V_{dd}^2 \cdot \sum_i a_i \cdot C_i, \quad (2)$$

where  $a_i$  is the average switching activity (transitions per clock cycle) on the wire(s) that connects to gate  $i$ ,  $C_i$  is its capacitance,  $V_{dd}$  is the supply voltage and  $f_{clk}$  is the clock frequency (Hz).

One way to compare the merit of alternative designs is to measure the switching activity of each gate using a simulation. This paper will present a gate-level Verilog simulation of fixed-point and LNS ALUs in the context of IDCTs using real MPEG data. In each case, the switching activity on the inputs to each gate are tallied. If we assume that  $C_i$ ,  $V_{dd}$  and  $f_{clk}$  are the same in every design, the

sum of the switching activities will provide a relative figure of merit to judge the simulated designs.

Paliouras and Stouraitis [9] considered LNS power consumption using methods similar to those used to analyze fixed point [6,10,11]. Sullivan [12] noted that by using a random logic  $s_b$  unit, power savings of 28% was possible. This paper gives a novel  $s_b$  approximation suitable for applications with the similar precision requirements as Sullivan's. The proposed approximation provides much greater power savings than the best design given by Sullivan.

Comparative power analysis of fixed-point and LNS requires defining how fixed point and LNS can be considered equivalent. This is not straightforward, as the relative accuracy of LNS and the absolute accuracy of fixed point are different. The literature has proposed several ways to establish such equivalency. Paliouras and Stouraitis [9] define LNS to be equivalent to fixed point when both systems have the same Average Relative Representational Error (ARRE) and the LNS covers at least as much dynamic range as the fixed-point system. Sullivan [12] assumed slightly different criteria: the Signal-to-Noise Ratios (SNR) of the two systems are the same and the LNS covers at least as much dynamic range. The criterion used here for MPEG decoding is more relaxed: LNS and fixed point are equivalent as long as the video output looks roughly the same [4]. This criterion has an advantage over the others: such LNS takes significantly fewer bits than an equivalent fixed-point system, and the shorter LNS word saves power in the memory system as well as in the ALU.

One factor determining power consumption is the effective capacitance, which is roughly proportional to the area of ROMs or PLAs. ROMs or PLAs tend to dominate the area in prior Leonelli-based LNS ALUs. The proposed LPVIP ALU can approximate  $s_b$  without any ROM and  $d_b$  with minimal ROM, thus greatly reducing the area and power consumption for applications, like MPEG decoding, which tolerate the low-precision nature of the proposed approximation. Since the minimal  $d_b$  ROM is used infrequently, its address can be latched (and thereby its contribution to switching activity can be made almost zero).

### 3 Mitchell's Method

Mitchell [8] proposed two related methods: one for the logarithm,  $\log_2(X)$ , and one for the antilogarithm,  $2^x$ . Like many approximations, Mitchell's methods are restricted to a limited range of arguments.

In the case of the logarithm, Mitchell uses  $\log_2(X) \approx X - 1$  in the range  $1 \leq X \leq 2$ . In a realistic application,  $X$  may fall outside this range. In such cases, it is possible to determine  $\text{int}(\log_2(X))$  by finding the position of the leading significant bit in  $X$  and then shifting appropriately<sup>1</sup>. Thus,  $1 \leq 2^{-\text{int}(\log_2(X))} \cdot X < 2$ , and in general  $\log_2(X) \approx \text{int}(\log_2(X)) + 2^{-\text{int}(\log_2(X))} \cdot X - 1$ .

In the case of the antilogarithm, Mitchell uses  $2^x \approx x + 1$  in the range  $0 \leq x \leq 1$ . Again a shifter can be employed for arguments outside this range so that in general  $2^x \approx 2^{\text{int}(x)} \cdot (\text{frac}(x) + 1)$ .

<sup>1</sup> Here  $\text{int}(X) + \text{frac}(X) = X$  and  $0 \leq \text{frac}(X) < 1$ .

Mitchell uses two logarithm circuits, one adder and one antilogarithm circuit (totaling three shifters, two leading-bit detectors and one adder) to implement his approximate multiplier. The width of the antilogarithm shifter is determined by the dynamic range required which, because of the information-compression property of the logarithm function, is likely to be much wider than the width of the LNS representation. Thus, much more switching activity occurs in Mitchell’s multiplier than is desirable.

### 4 Novel LPVIP $s_b$ Approximation

In order to keep numbers in LNS format during addition, it might be tempting to approximate  $s_b$  by applying Mitchell’s antilogarithm approximation, adding one and then using Mitchell’s logarithm approximation. Unfortunately, this obvious approach incurs error with both approximations and incurs extra switching activity due to the antilogarithm word width.

Instead, the novel approach described here uses the fact that

$$s_2(z) \approx 2^z \tag{3}$$

for  $z \leq 0$ . (Commutativity [5] of  $X + Y$  allows us to have  $z = -|x - y|$ .) The error in (3) is  $E_s(z) = 2^z - s_2(z)$ . The argument that produces the largest error,  $z_s = \log_2(1/\ln(2) - 1) \approx -1.17561488$ , can be found by solving  $E'_s(z_s) = 0$ , which in turn gives  $2^{z_s} \approx 0.442695$  and  $s_2(z_s) \approx 0.52876637$ . Thus, at worst (3) produces an error of about -0.086, which is good to four bits.

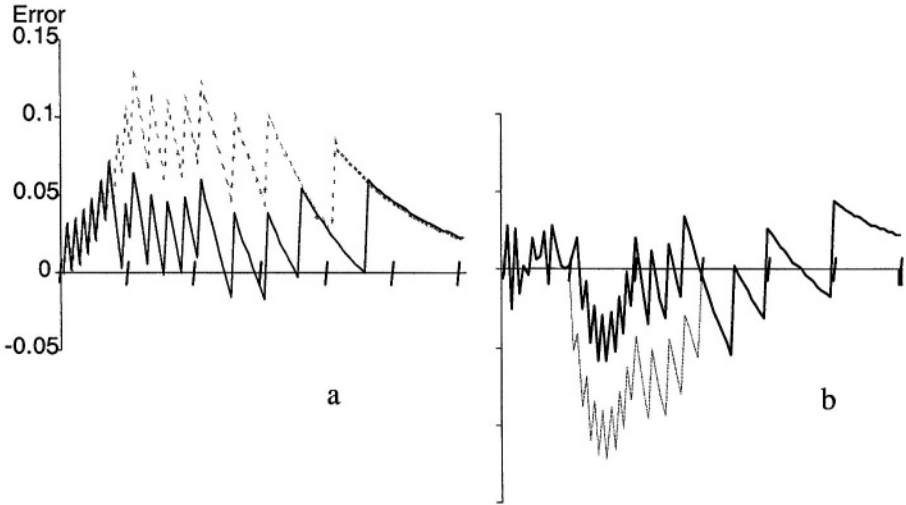
By combining (3) with Mitchell’s antilogarithm approximation, we have

$$s_2(z) \approx 2^{\text{int}(z)} \cdot (\text{frac}(z) + 1). \tag{4}$$

This approximation is exact at  $z = 0$  and as  $z$  approaches  $-\infty$ . It is larger otherwise.

Aside from  $E_s$  derived above, the total error in (4),  $\hat{E}_s = 2^{\text{int}(z)} \cdot (\text{frac}(z) + 1) - s_2(z)$ , results from the finite size of  $z$  and the errors inherent to Mitchell’s method. The dotted line in Figure 1a shows  $\hat{E}_s$ . Although this appears quite inaccurate, experiments indicate it may be acceptable for MPEG decoding. The bottom tips of this saw-tooth curve roughly follow the values of  $E_s$ , peaking with about a value of 0.08. Note that the solid line (explained below) obscures a portion of the dotted line. The zigzag of the dotted line is due to Mitchell’s approximation. The slight waviness of the smooth line segments that make up the zigzag is due to quantization of  $z$  (four bits after the radix point in this case). At its height,  $\hat{E}_s$  exceeds 0.1, which is a larger error than desired.

In order to mitigate this error, a novel correction term is added to (4). For the four-bit precision shown in Figure 1a, this term equals  $1/16$  for  $-13/16 \geq z \geq -56/16$  and for  $-65/16 \geq z \geq -72/16$ , and equals 0 otherwise. This correction is a compromise between accuracy and cost. The solid line in Figure 1a shows the LPVIP approximation after this correction is added.



**Fig. 1.** **a.** Error in LPVIP  $s_2(-z)$ . **b.** Error in LPVIP  $-d_2(-z)$  approximation. Dotted line has no correction; solid line uses correction logic.

Since  $\text{frac}(z) < 1$ , the addition of  $\text{frac}(z) + 1$  in (4) occurs at no cost by concatenation. Thus, the only cost of the proposed LPVIP  $s_b$  approximation is a shifter whose size (the number of bits in  $\text{frac}(z)$ ) is significantly smaller than would be required for the shifter size (the number of bits in  $z$ ) used in conversion to fixed point with Mitchell’s antilogarithm method. Unlike previous  $s_b$  units, the proposed circuit requires no ROM.

### 5 Approximating $-d_b$

The proposed circuit will approximate  $-d_b(z)$  rather than  $d_b(z)$  so that the output of the LPVIP approximation unit is always positive. For highly negative values of  $z$ ,  $-d_b(z) \approx s_b(z)$ ; however, for  $z$  near zero the magnitude of these two functions differ widely because  $-d_b(z)$  approaches  $\infty$  as  $z$  approaches zero. Because of this, we need to treat  $-1 < z \leq 0$  specially. Before considering that special case, note that

$$-d_2(z) \approx 2^{z+1} \tag{5}$$

for  $z \leq -1$ . The error in (5) is  $E_d(z) = 2^{z+1} + d_2(z)$ . The argument that produces the largest error,  $z_d = \log_2(1 - 1/(2 \ln(2))) \approx -1.8434611$ , can be found by solving  $E'_d(z_d) = 0$ , which then gives  $2^{z_d+1} \approx 0.55730496$  and  $-d_2(z_d) \approx 0.4712336285$ . Thus, at worst (5) produces an error of about 0.086, which, like (3), is good to four bits.

By combining (5) with Mitchell’s antilogarithm approximation, we can use the following in the case  $z \leq -1$ :

$$-d_2(z) \approx 2^{\text{int}(z+1)} \cdot (\text{frac}(z + 1) + 1). \tag{6}$$



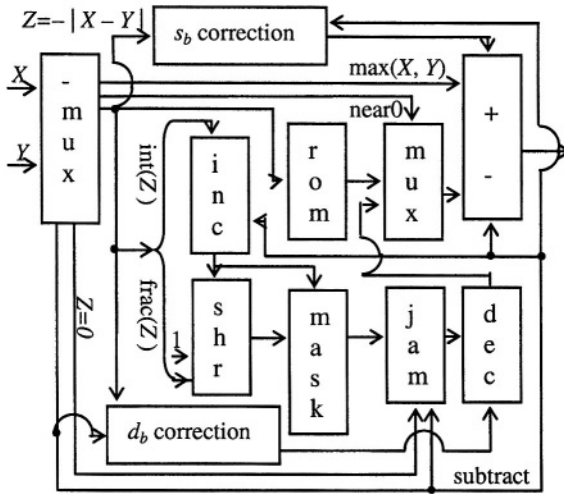


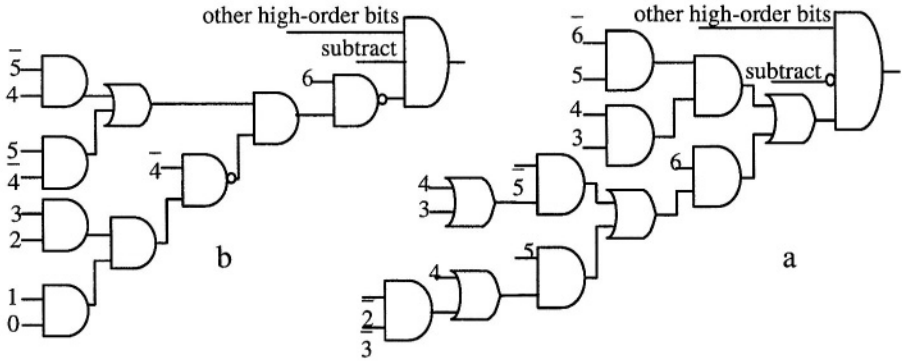
Fig. 2. LPVIP ALU.

This approximation is exact at  $z = 1$  and as  $z$  approaches  $-\infty$ . Of course  $\text{frac}(z) = \text{frac}(z + 1)$ , so a further simplification is possible. By including an incrementor on its input, the same LPVIP hardware that computes  $s_2(z)$  can compute  $-d_2(z)$  for  $z \leq -1$ . For values of  $z > -1$ ,  $-d_2(z)$  will be the best rounded value looked up from a ROM. The dotted line in Figure 1b shows  $\hat{E}_d = 2^{\text{int}(z+1)}(\text{frac}(z) + 1) + d_2(z)$  and the solid line shows the error after a correction term of  $-1/16$  is added in the case  $-18/16 \geq z \geq -48/16$ .

## 6 Implementation

Figure 2 shows a possible implementation of the LPVIP ALU. Since  $\log_b(X + Y) = \max(x, y) + s_b(z)$ , addition begins by determining  $\max(x, y)$  and  $z = -|x - y|$ . The same idea works for subtraction with  $d_b$ . Like prior LNS ALUs,  $\max(x, y)$  and  $z$  can be formed by a pair of subtractors ( $x - y$  and  $y - x$ ), a mux that selects the larger of  $x$  and  $y$  based on the carry-out of one of the subtractors and another mux that selects the negative difference. The block “-mux” implements this (other implementations are possible, such as a more economical but slower circuit consisting of a single subtract, a conditional negation and a mux). It is easy for this block to determine when  $z = 0$  by ANDing together the complement of the high-order bit of each subtractor. This “-mux” block also includes logic that generates a “subtract” signal that alters the behavior of the ALU so that the LPVIP technique approximates  $-d_2$  when the signal is asserted and  $s_2$  otherwise. Aside from the “near0” signal described below, all of the elements of this “-mux” module have been shown in the literature, e.g.,[12].

The bus  $z$  is split into its integer part “int( $z$ )” and its fractional part “frac( $z$ ).” An incrementor (labeled “inc”) adds one to the value of int( $z$ ) when



**Fig. 3.** **a.** Correction logic for  $s_2$  used in Figures 1a and 2. **b.** Correction logic for  $d_2$  used in Figures 1b and 2. The  $z$  input bits are labeled in little endian order. The bar indicates negation.

the subtract signal is asserted. The possibly incremented integer acts as the shift count for the shifter (“shr”). Since the output of “inc” is irrelevant when  $\text{int}(z) = -1$ , its logic can be simplified somewhat.

The bus being shifted by “shr” is formed by concatenating 1 onto the left of  $\text{frac}(z)$ . This shifter is controlled by the least-significant bits of  $\text{int}(z)$  (or  $\text{int}(z + 1)$  for subtraction). Because of this, the logic of the incrementor can be reduced further. Two raised to the number of shifter-control bits should be at least as large as the number of bits in  $\text{frac}(z)$ . For the example here, where there are four bits in  $\text{frac}(z)$ , this implies two bits for the shifter control, which in turn allows the shifter to recognize the cases in which  $\text{frac}(z)$  is shifted one, two, three or four places to the right, corresponding to  $\text{int}(z) = -1$ ,  $\text{int}(z) = -2$ ,  $\text{int}(z) = -3$  and  $\text{int}(z) = -4$ , respectively. The case of  $\text{int}(z) < -4$  is not handled by the shifter directly, but rather by masking logic connected to the output of the shifter. The masking logic passes the shifter output through in the cases described above. These cases are recognized by the fact the high-order bits of  $\text{int}(z)$  are all ones. If any of the high-order bits of  $\text{int}(z)$  are zero, the masking logic outputs zero.

Naturally, the masking logic returns 0 for the case of  $z = 0$ , which is wrong for both addition and subtraction. Thus, the case of  $z = 0$  is not handled by the shifter and masking logic, but rather by jamming logic following the masking logic. When  $z = 0$  the jamming logic returns either the largest possible value  $\approx -d_2(0) = \infty$ , or  $s_2(0) = 1.0$ . The order in which the “dec” and “jam” modules are connected could be reversed since “jam” only is used when  $z = 0$ , and “dec” is only used for subtract when  $z < -1$ .

Unless the system is performing a subtract with a value  $0 > z \geq -1$ , the output of the jamming logic is passed through the final “mux” to the adder/subtractor (“add/sub”). This unit adds the  $s_2$  output from the final mux when the subtract signal is not asserted, but subtracts the  $-d_2$  output when

this signal is asserted—thereby forming the desired logarithm of the sum or difference.

In order to improve the accuracy of the result, novel logic is connected to the  $z$  bus. This logic generates a value which is added (carry in,  $c_i$ ) to the final result. In the four-bit-precision example described here, the output is 1 (representing  $1/16$ ) when the subtract signal is not asserted and  $-13/16 \geq z \geq -56/16$  or  $-65/16 \geq z \geq -72/16$ . Otherwise the output is 0. A possible implementation of this using only two-input gates is given in Figure 1a. This approach uses only seven AND gates and four OR gates. To correct the  $-d_2$ ,  $-1/16$  is added for  $-18/16 \geq z \geq -48/16$ . Figure 1b illustrates a possible implementation using eight AND gates, one OR gate, and two inverters. In Figure 2, a decrementor (labeled “dec”) adds  $-1/16$  to the fractional part of the output from the jamming logic when the output of the novel logic shown in Figure 1b is asserted.

The special case of subtract with  $z$  near zero is indicated by a signal (“near0”) that controls the final “mux.” There is a small ROM addressed by  $\text{frac}(z)$  which holds values of  $-d_2(z)$  for  $0 \geq z \geq -1$ . If the subtract signal is asserted when  $0 > z \geq -1$ , the output of the ROM is passed through the final “mux” to the adder/subtractor. The near0 signal is generated inside the “-mux” block by ANDing the high-order bits of  $z$ .

The values stored in the ROM are not completely arbitrary. In the four-bit case used as an example here, for  $-1 \geq z \geq -13/16$ , the best rounded value for  $-d_2(z)$  can be formed exactly by concatenating 1 to  $\text{frac}(z)$ . Thus only twelve words of ROM are required (eleven if we ignore the singularity at  $z = 0$ ), which is a ten-fold reduction compared to the 128 words required by prior techniques [5,13] for a similar level of accuracy.

## 7 Switching Simulation

A gate-level simulation of the 10-bit ( $F = 4$ ) LPVIP ALU described in the previous section together with a  $10 \times 10$  LNS multiplier was implemented in Verilog. For comparison, a 16-bit ( $F = 4$ ) general-purpose fixed-point  $16 \times 16$  multiply-accumulate circuit and a specialized fixed-point  $4 \times 16$  multiply-accumulate circuit were implemented in similar Verilog. Table 1 shows how these circuits were exercised with non-zero data from an actual MPEG video [4].

The activities for each of the four basic combinational gate types are shown. From this, the total combinational switching activity is derived (assuming XOR and Mux are three times a simple gate). The flip-flop activity assumes a multiply-

**Table 1.** Switching Activity Observed in LPVIP Simulations.

Circuit	AND	OR	XOR	Mux	Total Comb.	Flip Flop
FX $4 \times 16$	61,677,377	11,453,481	61,677,377	0	258,162,989	130,829,472
FX $16 \times 16$	347,115,545	19,419,597	347,115,545	0	1,407,881,777	203,498,444
LNS $10 \times 10$	26,795,462	10,525,344	18,185,636	8,268,541	116,683,333	38,153,336

accumulate register (10 bits for LNS; 16 bits for FX) composed of D-type master-slave circuits clocked in each cycle. The ROM in the LPVIP circuit is used only 2,098 times out of the total 141,186 multiply-accumulate iterations, or roughly 1%. Therefore, the power consumption of the small ROM used in those infrequent cases can be neglected because a latch isolates it from the rest of the circuit.

The  $10 \times 10$  LNS and the  $16 \times 16$  FX circuits allow the dynamic range of -2,048 to 2,047 for both multiplicand and multiplier. As such, these circuits are general-purpose and suitable for a wide variety of DSP and multimedia applications, where the factor-of-ten improvement for power consumption shown in Table 1 may be possible. Because the absolute value of the cosine coefficients for the IDCT are all less than one, a more specialized fixed-point circuit is possible, using a  $4 \times 16$  multiply, which significantly reduces its switching activity. Even considering this, the general-purpose LPVIP circuit is about twice as efficient as the special-purpose fixed-point one for this MPEG application (considering total activity from logic and flip flops). It is interesting to note that the LNS flip flops consume only one third of what the LNS combinational logic consumes. In contrast, the fixed-point flip flops consume one half of what the specialized fixed-point combinational logic consumes, pointing out that there are system-wide advantages (beyond the ALU) from choosing LNS.

## 8 Conclusions

The LPVIP approach offers a much simpler ROM-less ALU for computing  $s_2$  with accuracy of about  $F = 4$ , and a much-reduced-ROM ALU for computing  $d_2$ . The approach seems well suited to multimedia applications, like MPEG decoding. In the context of portable devices with smaller, lower-resolution displays, the proposed MPEG decoder using LPVIP LNS should be visually acceptable and offers significant power savings.

## References

1. M. G. Arnold, T. A. Bailey, and J. R. Cowles, "Improved Accuracy for Logarithmic Addition in DSP Applications," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp.1714-1717, 1988.
2. Mark G. Arnold, "LNS for Low-Power MPEG Decoding," *Advanced Signal Processing Algorithms, Architectures and Implementations XII*, SPIE, Seattle, Washington, vol. 4791, pp. 369-380, 11 July 2002.
3. M. Arnold, T. Bailey, J. Cowles and J. Cupal, "Error Analysis of the Kmetz/Maenner Algorithm," *Journal of VLSI Signal Processing*, vol. 33, pp. 37-53, 2003.
4. Mark G. Arnold, "Reduced Power Consumption for MPEG Decoding with LNS," *Application-Specific Systems, Architectures, and Processors*, IEEE, San Jose, California, pp. 65-75, 17-19 July 2002.
5. N. G. Kingsbury and P. J. W. Rayner, "Digital Filtering Using Logarithmic Arithmetic," *Electronics Letters*, vol. 7, no. 2, pp. 56-58, 28 January 1971.

6. P. Landman and J. R. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," *IEEE Transactions on VLSI*, vol. 3, no. 2, pp. 173-187, June 1995.
7. Z. Leonelli, *Supplément Logarithmique*, a reprint of Leonelli's 1803 manuscript with a biography by J. Houël, Gauthier-Villars, Paris, 1875.
8. J. N. Mitchell, "Computer Multiplication and Division using Binary Logarithms," *IEEE Transactions on Electronic Computers*, vol. EC-11, pp. 512-517, August 1962.
9. V. Paliouras and T. Stouraitis, "Logarithmic Number System for Low-Power Arithmetic," *PATMOS 2000: International Workshop on Power and Timing Modeling, Optimization and Simulation*, Göttingen, Germany, pp. 285-294, 13-15 September 2000.
10. S. Ramprasad, N. Shanbhag and I. Hajj, "Analytical Estimation of Signal Transition Activity from Word-Level Statistics," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 16, no. 7, July 1997.
11. J. R. Sacha and M. J. Irwin, "The Logarithmic Number System for Strength Reduction in Adaptive Filtering," *International Symposium on Low Power Electronics and Design*, Monterey, California, pp. 10-12, August 1998.
12. T.J. Sullivan, *Estimating the Power Consumption of Custom CMOS Digital Signal Processing Integrated Circuits for Both the Uniform and Logarithmic Number Systems*, Ph.D. Dissertation, ESSRL-93-7, Electronic Systems and Signals Research Laboratory, Department of Electrical Engineering, Washington University, Saint Louis, Missouri, 1993.
13. E. E. Swartzlander and A. G. Alexopoulos, "The Sign/Logarithm Number System," *IEEE Transactions on Computers*, vol. C-24, pp. 1238-1242, Dec. 1975.
14. F. J. Taylor, "An Extended Precision Logarithmic Number System," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-31, pp. 231-233, February 1983.
15. O. Vainio and Y. Neuvo, "Logarithmic Arithmetic in FIR Filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-33, No. 8, pp. 826-828, August 1986.

# Low Level Adaptive Frequency in Synthesis of High Speed Digital Circuits

Leonardo Valencia

STMicroelectronics, Central R&D, 850 rue Jean Monnet, 38920 Crolles

**Abstract.** Many techniques are used to control the speed of digital designs at architecture level [1] or to stop the clock at gate level, mainly to save power [2]. Some high speed circuits speed up the clock on portions of the design which were not critical otherwise [3]. Asynchronous designs are fine grain speed adaptive, but this solution is not easy to use [4]. This article presents an efficient and easy to use technique to increase performance of synthesized digital circuits with low level adaptive speed – i.e. at gate level.

## 1 Introduction

During the synthesis of synchronous digital circuits, the optimization for speed of the logic between sequential elements often ends up by getting limited by a group of critical paths. Then, there are mainly two ways of optimizing the speed of operation of such a circuit: either a microarchitecture re-design or a standard cell library speed improvement. These solutions are time consuming, it is sometimes not possible to change the microarchitecture by adding pipeline stages, for instance, and the standard library may already be tuned for high speed synthesis, especially on the critical paths, as presented in [5]. So the circuit speed is limited by a certain portion of the design but not all of it. Then it is possible to run at faster clock rate on some group of paths, when looking each path at gate level –i.e with a fine grain analysis.

The timing critical portion of a digital design is assumed to be activated during typical application. Which is not always the case, if we refer to algorithms used to benchmark the CPU speed we can notice that most of the operations are not activating the critical path. For instance, in an addition operator, the critical path is often the carry computation, but if assume that all combination of arguments ( $a + b$ ) are equiprobable then for one case in four we can replace the carry logic computation by a single AND gate. Indeed, if the most significant bits of “a” and “b” arguments are equal to logic one state then the carry has to be one also. We can do the same assumption when “a” and “b” most significant bits are equal to zero, carry has to be zero.

The technique described here takes advantage of faster design portion; it enables to dynamically choose a faster clock when the critical path is functionally a faster path than the worst critical path of the whole design. Then the circuit does not operate at a fixed frequency but at an average frequency which can be evaluated for a given application or functional pattern.

## 2 Structural Description of the Technique

The structural description of the design using the fine grain adaptive frequency can be split in two part.

The first part is the generations of the signals that will trigger dynamic switching from one clock to a faster clock which is a few gates that observe the data coming to the critical path.

The second part is the clock selection with a mux gate which selection signal is generated by the first part.

The generation of the clock selector signal is design dependent but it can be also an activity monitoring over sequential elements just like the clock gating techniques. This last type of signals is often available in designs dealing with massive clock gating. The selection from one clock to another is then done dynamically as the state of the sequential elements allows faster clocks.

Since the clock speed can vary during the application from one cycle to another, the working frequency of the circuit is no more the slowest clock but an average of all the clock speed used taking into account the utilization of each of them.

The cost of this technique relies on the timing constraints used to drive the synthesis. Since there could be multiple timing constraints set for each clock instead of one set. The re-design cost at rtl level is almost none, since there is no modification of the rtl functionality. There is only additional rtl for the description of the observation points.

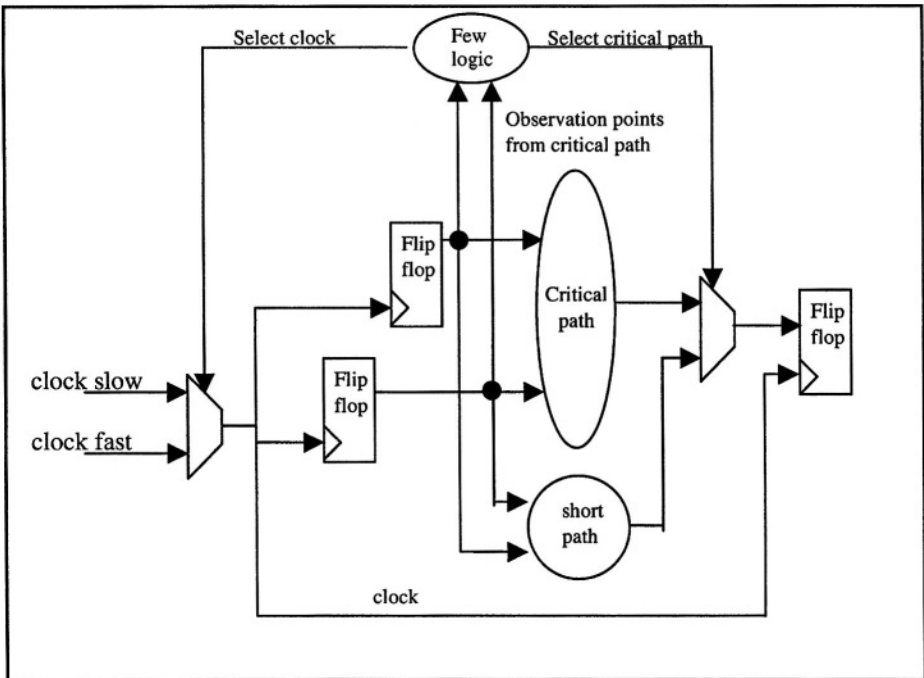


Fig. 1. Example of structural description of the fine grain adaptive frequency technique.

Note that if the short path is in fact a single wire coming from the capture flip flop then this technique is similar to a recirculation mux which can be reverted to clock gating technique. The clock gating for power saving is a particular case of the technique and the clock enable signal can be used for clock selection between fast and slow clock.

Note also that in fig1, only two clock clocks are represented since only one path is considered. But there can be, “n” clocks as we consider “n” groups of critical path in a design. And if “n” is large, close to the maximum number of path in the design, then this design behaves almost like an asynchronous design – i.e. the clock selector would then be equivalent to a handshake or acknowledge signal. Then we understand that this technique brings some of the asynchronous advantage to a synchronous design implementation.

In this example also, the observation points are taken on the flip flop output but they can be taken on the flip flop input, this steals some time on the previous cycle. This is very convenient to detect transition on flip flops, a XOR gate between the input and output of the flop turns to one just before clock capture edge when the next state for the flip flop is not stable. This can be very helpful to monitor the activity on the launching flip flops of the critical path. And select the fast clock when there is not activity on the critical path.

So, we have reviewed many implementations of the technique for observing the critical paths and generating select signal for the faster clocks. Some implementation are more automated since we can recycle clock gating information some are less automated since we have to provide the description of the short path in the rtl to be synthesized.

### 3 Measure of the Performance Increase

To compare the efficiency of each implementation versus the standard one clock design, we need to define an application or set of patterns that are representative of many standard applications. It is difficult to choose one pattern rather than another.

In order to cover a wide range of functional patterns, the benchmark used to compare the design implementations performance will be a function of the instruction mix of a processor design.

The measure of performance is the number of instructions executed per cycle. In the case of a standard, one clock, implementation, the performance indice is 1, like one instruction per cycle.

More generally, if we consider a calculation unit with “n” types of instructions, the instruction mix of a given functional pattern being a vector of coefficients  $\langle a(1), a(2), \dots, a(n) \rangle$ , representing the occurrence of each instruction, then the performance indice is :



$$\text{Relative performance} = \frac{1}{\text{Min}_i(\text{Freq}(i)) \cdot \sum_{i=1}^n \frac{a(i)}{\text{Freq}(i)}}$$

$$\text{Considering: } \sum_{i=1}^n a(i) = 1$$

For a given design implementation, “n” is known,  $\text{Freq}(1), \text{Freq}(2), \dots, \text{Freq}(n)$  are known as well, they are all the clocks frequencies used in the design for each instruction. Then the relative performance can be calculated for a given pattern and instruction mix which provide the  $a(1), a(2), \dots, a(n)$  coefficients.

The relative performance is the ratio between the worst clock period and the average time for one instruction, any ratio superior to 1 means a speed up gain of the design using this technique as compared to the reference design with the performance indice of 1.

## 4 Automation and Drawbacks

We already mentioned that the cost of the technique had an impact on the automation flow for synthesis.

Indeed the synthesis of the design will have to take into account different path groups with each ones its own constraint for clock frequency. The cost in time, for design constraint re-writing, depends on the CAD tool used for synthesis. For instance this study here has used SYNOPSIS tool suite for synthesis.

If we choose to limit the range of clock frequency usable in a design to only integer multiples, we can then use *multi\_cycle* type of constraints. This helps the constraint writing task since we deal with only one clock and its multiples.

The cost in area is very design dependent since it depends on the activity monitoring strategy. A few gates are added for monitoring the activity but most of the area penalty is linked to the higher timing constraint for almost each path of the design; this penalty can increase very fast with frequency constraint.

The cost in power is linked to the area increase for the leakage power and for the rest it is almost unchanged if we consider the energy per cycle. Otherwise if we do not consider the speed increase in the power consumption the circuit will consume more power as it runs faster, of course.

The design cost at system level is linked to the clock generation. As we migrate from a design paradigm with one clock to multiple clocks we need to generate them with more complex frequency generator or with an on chip PLL/DLL. Also a tradeoff has to be found for each design implementation between the optimal clocks frequency and the ones the on chip hardware allows to generate.

## 5 Example of Implementation and Speed Increase

The design chosen is a 64 bit ALU with an 8 register file. It has been synthesized with a cmos logic standard cell library in 0.13um technology. The operators or instructions of the design are: arithmetic shifts (sar and sal), 64 bit integer multiplication (imul), 64 bit integer division, 64 bit addition (add) , increment (inc), 32 bit load immediate (loadih, loadil), arithmetic operators (xor,and,or) and 64 bit store register operation (store). The design has 8 registers of 64 bit. This design is implemented with one clock for the entire chip, two clocks and three clocks. The one clock design (design implementation A) is taken as reference. The two clocks design (design implementation B) has one clock for the integer division and one clock for the rest of the design. The three clocks design (design implementation C) has one clock for the integer division, one clock for the integer multiplication, and one clock for the rest of the design.

Design implementation	Clock frequencies obtained after synthesis in 0.13um technology	Area Penalty w.r.t. design A	Power Penalty (leakage and dynamic) w.r.t design A
A	Freq(1)=29Mhz	1.0	1.00
B	Freq(1)=29Mhz Freq(2)=256Mhz	1.14	1.64
C	Freq(1)=29Mhz Freq(2)=256Mhz Freq(3)=463Mhz	1.15	2.07

We consider three classes of instructions, the ones that can compute only in a Freq(1) cycle (class 1) , the ones that can be evaluated in a Freq(1) or Freq(2) cycle but not in a Freq(3) cycle (class 2) and the ones can be evaluated in a Freq(3) cycle (class 3).

For a given instruction mix  $\langle a_1, a_2, a_3 \rangle$  corresponding respectively to the proportion of class1, class2 and class3 instructions in the mix, we can then calculate the relative performance indice.

Here below are the figures of relative performance between design C and design A:

Scenarios of Instruction mix	a(1)	a(2)	a(3)	Relative performance indice
Instruction mix 1	0.1	0.1	0.8	6.194
Instruction mix 2	0.2	0.1	0.7	3.919
Instruction mix 3	0.2	0.4	0.4	3.699
Instruction mix 4	0.3	0.2	0.5	2.825
Instruction mix 5	0.3	0.5	0.2	2.709
Instruction mix 6	0.5	0.2	0.3	1.893
Instruction mix 7	0.5	0.3	0.2	1.830
Instruction mix 8	0.7	0.1	0.2	1.381

Depending on the scenario the speed up is then varying from ~519% to ~38%.

Of course in this design the integer division is a big penalty, and we may consider what the speed up is if the ALU does not implement the integer division operator. We

design this variant without the integer division, there is no more the class1 instruction, and the results are:

Scenarios of Instruction mix	a(1)	a(2)	Relative performance indice
Instruction mix 1	0.2	0.8	1.557
Instruction mix 2	0.3	0.7	1.455
Instruction mix 3	0.6	0.4	1.218
Instruction mix 4	0.5	0.5	1.288
Instruction mix 5	0.8	0.2	1.098
Instruction mix 6	0.7	0.3	1.155
Instruction mix 7	0.8	0.2	1.098
Instruction mix 8	0.8	0.2	1.098

When removing the integer division of the design the speed up is still interesting since it varies from ~55% to ~10%.

## 6 Conclusion

The efficiency of the technique has been demonstrated with a typical design and across many instruction mixes. It brings an interesting speed up as compared to drawbacks such as reasonable area increase, power consumption increase and dealing with more complex synthesis constraints.

The other advantage brought by this technique is also a gain on the design time, since the architecture does not need to be modified to get the speed up gain. For instance, the design shown in the example required less than half a day of design time.

As compared to techniques of dynamic microarchitecture reconfiguration [6] the gain in design time per speed up increase is interesting.

## References

1. "Intel 80200 Processor Based on Intel Xscale Microarchitecture", <http://developer.intel.com/design/iio/manuals/273411.htm>
2. W. Kuo, T.Hwangand A.Wu , "Decomposition of Instruction Decoder for Low Power Design" Design Automation and Test in Europe 2004.
3. D. J. Delegates, M. Barnay, G. Geannopoulos, K. Kreitzer, A. P. Singh, S. Wijeratne "Low-Voltage-Swing Logic Circuits for a 7GHz X86 Integer Core", Intel Hillsboro , IEEE International Solid-State Circuits Conference ISSCC 2004.
4. F. Burns, D. Shang, A. Koelmans and A. Yakovlev, "An Asynchronous Synthesis Toolset Using Verilog" , Design Automation and Test in Europe 2004.
5. H. Chen, D. H.C. Du, L. Liu, "Critical Path Selection for Performance Optimization", Department of Computer Science -University of Minnesota, Minneapolis.{wolf,jbf}@ccrc.wustl.edu .
6. A. Iyer, D. Marculescu, "Run-time Scaling of Microarchitecture Resources in a Processor for Energy Savings", Electrical and Computer Engineering Department Center for Electronic Design Automation, Carnegie Mellon University, {aiyer,dianam}@ece.cmu.edu.

# A Novel Mechanism for Delay-Insensitive Data Transfer Based on Current-Mode Multiple Valued Logic

Myeong-Hoon Oh and Dong-Soo Har

Department of Information and Communications  
GIST (Gwangju Institute of Science and Technology)  
1 Oryong-dong Puk-gu Gwangju, 500-712, South Korea  
{mhoh, hardon}@gist.ac.kr

**Abstract.** By conventional delay-insensitive data encodings, the number of required wires for transferring  $N$ -bit data is  $2N+1$ . To reduce the required number of wires to  $N+1$ , and thus, reducing complexity in designing a large scaled chip, a novel data transfer mechanism based on current-mode multiple valued logic is proposed. Effectiveness of proposed data transfer mechanism is validated by comparisons with conventional data transfer mechanisms using dual-rail and 1-of-4 encodings at the  $0.25\text{-}\mu\text{m}$  CMOS technology. Simulation results of 32-bit data transfer with 10 mm wire length demonstrate that proposed data transfer mechanism reduces the time-power product values of dual-rail and 1-of-4 encoding by 55.5% and 8.5%, respectively, in addition to the reduction of the number of wire by about half.

## 1 Introduction

In an SoC design based on a globally asynchronous locally synchronous (GALS) system, delay-insensitive (DI) data transfers are more desirable than bundled data protocol [1] with matched delay lines, because of design complexity and timing uncertainty of wires among massive function blocks. However, conventional DI data transfer mechanisms, dual-rail data encoding [2] and 1-of-4 data encoding [3], suffer from increased wire cost, because of  $2N+1$  physical wires for transferring only  $N$ -bit data.

The data transfer based on multiple-valued logic (MVL) [4] is an effective choice to reduce wire cost. The MVL circuits, however, have critical drawbacks such as slower switching speed caused by the increased complexity, and larger power consumption due to more logic levels than those of the binary logic. It is, therefore, reasonable not to apply the MVL circuit to the entire binary logic system.

MVL circuits are classified into voltage-mode MVL (VMMVL) circuits and current-mode MVL (CMMVL) circuits according to the way of representing multiple values. Although the VMMVL circuit has been the main interest of earlier studies, the VMMVL circuits are complicated to construct since their

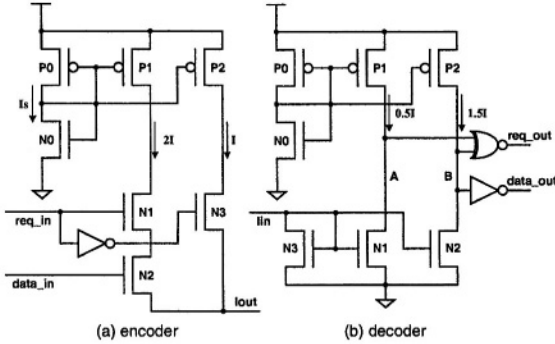


Fig. 1. Schematic of basic MVL circuit

dynamic range and noise margin strongly depend on the supply voltage. On the other hand, the CMMVL circuits [5] can guarantee the stable operation even at the lower supply voltage by freely controlling the amount of internal current. Therefore, the CMMVL circuit rather than the VMMVL circuit is discussed in this paper.

In order to reduce wire cost incurred by DI data transfer mechanism, a CMMVL-based global interconnection scheme where N-bit data is transferred with only N+1 wires is suggested. In addition, performance and power consumption of proposed CMMVL circuits are compared with those of the conventional DI data transfer mechanisms.

## 2 Structure of Encoder and Decoder in MVL Circuit

In order to apply a CMMVL circuit to digital system design, the mutual conversion between voltage and current values is required. Encoder and decoder modules in CMMVL circuits convert a voltage to a current value and a current to a voltage value, respectively.

In general, in each local module of a GALS system, all data are formatted to fit for single-rail. Thus, interfacing signals of the proposed MVL circuit with external environment are based on the single-rail bundled-data handshake protocol [1] which is the most commonly used for asynchronous circuit design. To the best knowledge of authors, DI data transfer scheme using a 2-phase signaling has not been published yet. Due to the complexity in implementing completion detection, conventional DI data transfer schemes, dual-rail and 1-of-4 encodings employ a 4-phase signaling. In this paper, the 4-phase signaling between encoder and decoder modules is used for DI data transfers.

Figure 1(a) shows a schematic of the encoder module for one bit input. P0 and N0 in a current source (CS) generate constant current  $I_s$ . Since the voltage  $V_{GS}$  between gate and source is identical to the voltage  $V_{DS}$  between drain and source, P0 and N0 always operate in the saturation region and the constant

**Table 1.** Current mapping to combination of input signals

<i>req_in</i>	0	1	1
<i>data_in</i>	X	0	1
current level	<b>I</b>	0	<b>2I</b>
(A, B)	(0,1)	(1,1)	(0,0)

current **I**s flows through their drains. The current **I**s is duplicated to drains of P1 and P2 working as a current mirror (CM). This drain current can be scaled by varying size of P1 and P2. The encoder's two binary logic signals, *req\_in* and *data\_in* are inputs to a voltage switch current generator (VSCG) composed of pass transistors, N1, N2, and N3. These transistors determine the current levels which are mapped to the combination of input signals by selecting the drain current of the CM. The current levels used for the MVL circuit are shown in Table 1. The *data\_in* signal is valid, only when the *req\_in* signal keeps logical '1' value in the 4-phase bundled protocol. Therefore, current levels are assigned 0 and **2I** according to the *data\_in* '0' and '1', respectively. Current level **I** is mapped to return-to-zero phase that is initiated by logical '0' of the *req\_in* signal. Following the current mapping in Table 1, the designer should make the current through N1 and N2 roughly twice as large as that flowing through N3. Those two individual current levels through N2 and N3 are added together to form the combined current *I<sub>out</sub>*.

Encoded current levels are transformed to the previously defined voltage levels in a decoder module. A decoder's schematic is illustrated in Fig. 1(b). Transistors N0, P0, P1, and P2 work as CS and CM modules in the same manner as in the encoder. The three-valued input current *I<sub>in</sub>* is applied to N3. The transistor N3 then copies *I<sub>in</sub>* to drains of N1 and N2 which jointly act as a current comparator (CC) through a coupling with P1 and P2. The CM composed of P1 and P2 should create threshold currents **0.5I** and **1.5I** in order to generate the differential current according to input current *I<sub>in</sub>*. Nodes **A**, **B** at drains of N1, N2, respectively, take logical '1' as long as *I<sub>in</sub>* is 0, because N1 and N2 do not pull any current. When input current level **I** is applied, **A** has '0' voltage level, since N1 consumes all of the threshold current **0.5I** to drive **I** current while **B** still takes logical '1' voltage level due to the remaining differential current (**1.5I** - **I** = **0.5I**) at node **B**. Based on similar reasoning, it is found that both **A** and **B** have '0' voltage for the input current **2I**. These combinations of voltage values are used to reconstruct original voltage-mode input signals with standard CMOS logic gates. Table 1 lists the logical voltage values at nodes **A** and **B** according to each current level. The corresponding equations of each decoded output signal are  $req\_out = AB + !A!B = XNOR(A,B)$  and  $data\_out = !B = INV(B)$ .

### 3 Design and Simulation Results

One of the most important factors in designing the schematics in Fig. 1 is the reference current **I**, which can vary depending upon the transistor's size. The

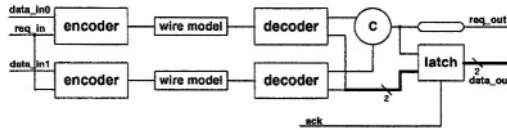


Fig. 2. Simulation environment including MVL circuit

reference current  $I$  critically affects the performance of the CMMVL circuit. With larger  $I$ , bigger differential current is created at a CC block in the decoder such that the switching speed becomes higher, and the wasted current flowing into the ground of CS and CM blocks increases, resulting in increased power consumption. Therefore, a metric reflecting both time delay and power dissipation simultaneously is needed and tuning the size of internal transistors in the CMMVL circuit for an optimal result is necessary. With the simulation environment involving the CMMVL circuit as in Fig. 2, the time-power product  $T^*P$  is employed as the performance metric and measured over various wire lengths.

The data type used for simulation environment is 2-bit data and the simulation environment is composed of two distinct pairs of encoder and decoder modules, a 2-input C-element for the completion detection of data transfer, and a latch block in a receiver. The relevant wire model for simulation is distributed RC model [6] at the 3rd metal layer of ANAM 0.25- $\mu\text{m}$  5-metal CMOS technology [7]. In the transistor level, the average delay from *data\_in* to *data\_out* was measured by making use of HSPICE and Root Mean Squared (RMS) dissipated power was observed by using NanoSim tool. The test vectors were generated in such a manner that average input data rate is set to a quarter of the maximum data transfer rate in on-chip bus architectures [8], [9].

The approaches to minimize the power consumption in the CMMVL circuit without deteriorating delay performance can be summarized as follows.

- To minimize the wasted current in CS blocks in both encoder and decoder modules, CS keeps generating current  $I_s$  as small as possible. However, CM should scale up  $I_s$  to make reference currents  $I$  and  $2I$ .
- Since the CMMVL circuit initially guarantees the stable operation at low supply voltage, 2.5 V at the given 0.25- $\mu\text{m}$  technology was not needed. In this paper, we use 2 V as supply voltage to the area of the CMMVL circuit.

The  $T^*P$  value was obtained while tuning internal transistors was carried out repetitively. With the simulation environment in Fig. 2 for a 10 mm wire model, minimal  $T^*P$  value was obtained with 4.01 ns of delay and 1041.06  $\mu\text{W}$  of RMS power when we generate about 54  $\mu\text{A}$  as the reference current  $I$  at 2 V supply voltage. Table 2 lists *Width (W)/Length (L)* ratios of transistors in the encoder and the decoder used for simulations, while typical PMOS and NMOS transistors of [7] have *W/L* ratios of 1.74  $\mu\text{m}/0.24 \mu\text{m}=7.25$  and 0.58  $\mu\text{m}/0.24 \mu\text{m}=2.42$ , respectively.

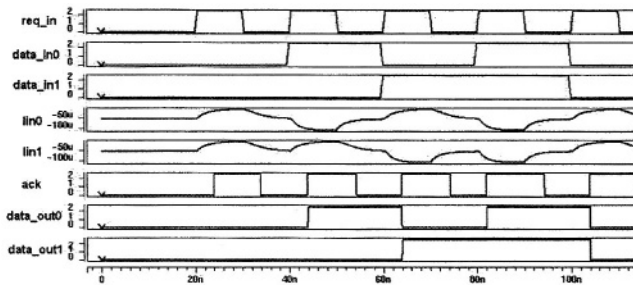
To verify the correctness of the CMMVL circuit in the environment with power supply noise which is the one of the largest noise sources in typical digital

**Table 2.** *W/L* ratios of transistors in Fig. 1

transistor	P0	N0	P1	P2	N1	N2	N3
encoder	0.25	0.25	14.5	4.83	2.42	2.42	2.42
decoder	0.25	0.25	2.42	8.46	20.83	20.83	20.83

**Table 3.** Delay and reference current according to supply voltage variation

supply voltage (V)	1.7	1.8	1.9	2	2.1	2.2	2.3
delay (ns)	–	4.55	4.23	4.01	4.26	4.48	–
<b>Iref</b> ( $\mu$ A)	26	36	45	54	64	75	86



**Fig. 3.** Results obtained from the simulation environment in Fig. 2

systems, we observed delay and reference current as we changed the supply voltage of the encoder from 1.7 V to 2.3 V by a step increment of 0.1 V. Note that the original supply voltage in our implementation is 2 V. In Table 3, it is seen that a new reference current **Iref** is depending on the supply voltage and delay is varying because of the amount of differential current between threshold current of decoder and changing **Iref**. The current **Iref** should be more than  $0.5I=27 \mu A$  and less than  $1.5I=81 \mu A$  to guarantee correct operations of the CMMVL circuit, and therefore, the noise margin of power supply can be estimated. In Table 3, it is observed that **Iref** values between 1.8 V and 2.2 V satisfy the above condition of  $27 \mu A < Iref < 81 \mu A$  and this 0.2 V voltage margin for the original supply voltage (2 V) is reasonable range with the given 0.25- $\mu m$  technology, according to the experiments in [10].

The functionality of the CMMVL circuit is validated, as seen in Fig. 3. It is seen that 2-bit input data (*data\_in0*, *data\_in1*) is transformed into a certain current level (*Iin0*, *Iin1*) and the original data is reconstructed to output data (*data\_out0*, *data\_out1*) correctly with all the available 2-bit data patterns, i.e., ‘00’, ‘01’, ‘10’, and ‘11.’

### 4 Comparisons with Other Schemes

To evaluate the effectiveness of the CMMVL circuit, two DI schemes the dual-rail encoding and the 1-of-4 encoding scheme were compared. It is also assumed in



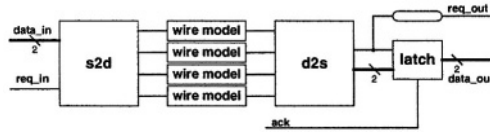


Fig. 4. Simulation environment for dual-rail and 1-of-4 encoding schemes

Table 4. Area comparisons

	Encoder		Decoder		Total	
	# of Tr.	Area ( $\mu\text{m}^2$ )	# of Tr.	Area ( $\mu\text{m}^2$ )	# of Tr.	Area ( $\mu\text{m}^2$ )
CMMVL	18	5.62	56	21.42	74	27.04
Dual-rail	28	8.91	26	9.74	54	18.64
1-of-4	40	14.48	24	10.30	64	24.78

Fig. 4 that the transferred data width is 2-bit for the simulation environment of dual-rail and 1-of-4 schemes. The **s2d** module converts single-rail 2-bit data symbols into DI 4-bit dual-rail or 1-or-4 data encodings. The **d2s** module, converting back to single-rail 2-bit data symbols, includes a 2-input C- element for the completion detection of each 2-bit data. Note that the number of required wires in Fig. 4 is twice as many as that of the CMMVL circuit in Fig. 2 for 2-bit data.

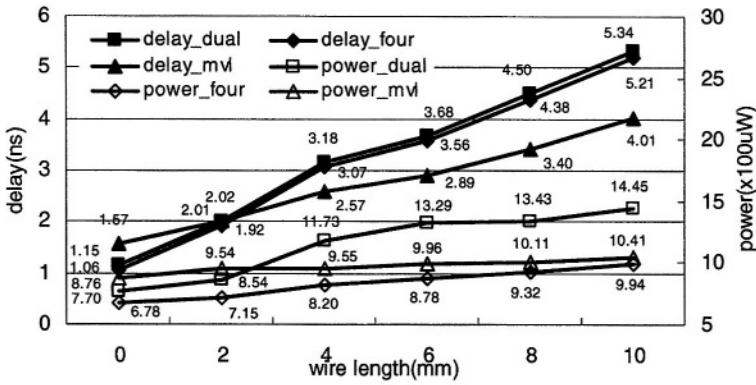
The area comparisons involving three schemes are summarized in Table 4. The area (width \* length) as well as the number of transistors except for commonly used blocks such as wire model, latch block and delay cell, were measured for the encoder and the decoder. The area of the CMMVL scheme is the largest among three schemes. However, this numerical value is not very large compared with the 1-of-4 scheme, since the area of the CMMVL scheme is 1.09 times that of the 1-of-4 scheme.

#### 4.1 Comparisons with Non-buffered Dual-Rail and 1-of-4 Schemes

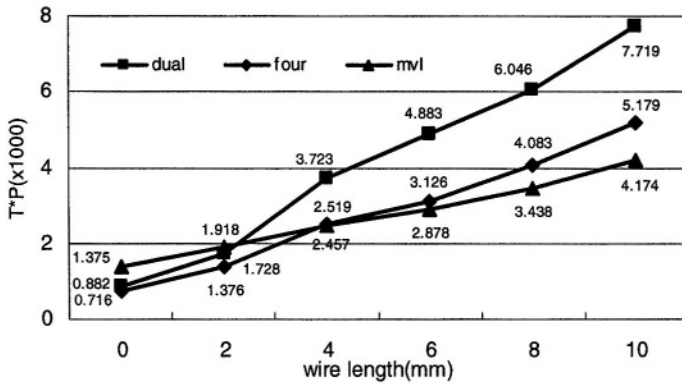
The plots of Fig. 5(a) illustrate simulation results of delay and power dissipation of the CMMVL scheme (delay\_mv1, power\_mv1), the dual-rail scheme (delay\_dual, power\_dual), and the 1-of-4 scheme (delay\_four, power\_four) according to the variation of wire length ranging from 0 mm to 10 mm by a step increment of 2mm.

As the wire length becomes larger, it is assured that results of delay and power consumption generally increase due to enlarged resistance and capacitance in the transmission line. Measured results of delay in dual-rail and 1-of-4 schemes are nearly equivalent due to the simplicity of their data conversion logics. The slope of ascending for dual-rail and 1-of-4 schemes is larger than the counterpart of the CMMVL scheme. Consequently, the CMMVL scheme demonstrates superior delay performance to the others with wire lengths of 2 mm or more.

The amount of power consumption in the dual-rail scheme is larger than that of the 1-of-4 scheme over all range of wire length, since the dual-rail scheme



(a) Comparisons of delay and power according to wire length



(b) Comparisons of T\*P

**Fig. 5.** Comparisons with non-buffered dual-rail and 1-or-4 schemes

requires twice more signal transitions for the data conversion than does the 1-of-4 scheme. On the other hand, the variation of power dissipation in the CMMVL scheme, in general, is smaller than those of two other schemes. This observation is due to the constant reference current generated from CS and CM blocks in the CMMVL circuit, regardless of the wire length. Over 4 mm wires, the power consumption of the CMMVL scheme is relatively small to that of the dual-rail scheme.

Figure 5(b) shows the product of delay and power corresponding to three different schemes. It is seen in the figure that the CMMVL scheme is more efficient than the others with wire lengths over 4 mm. For example, the CMMVL scheme was able to reduce  $T*P$  value by about 45.9% and 19.4% with 10 mm wire length, compared with those of the dual-rail scheme and the 1-of-4 scheme, respectively.

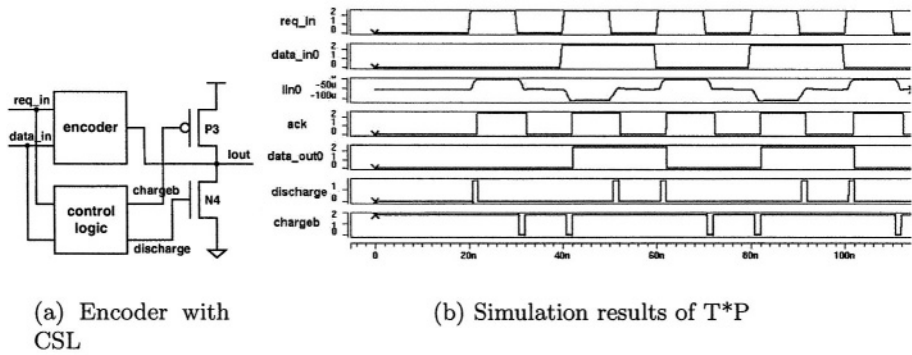


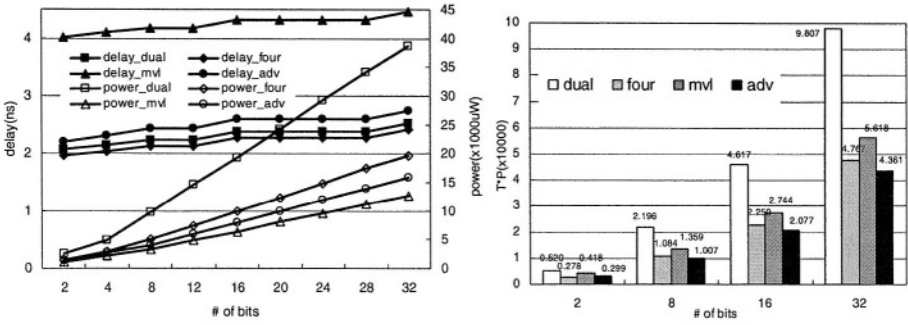
Fig. 6. CMMVL circuit for high performance

### 4.2 Comparisons with Buffered Dual-Rail and 1-of-4 Schemes

In general, the transmission delay decreases by inserting buffers in the middle of the long wire line such as the wire of 10 mm or more, when designing voltage-mode binary circuits. To compare the CMMVL scheme with the implementation of dual-rail and 1-of-4 schemes for high performance, the simulation was performed in the 10 mm wire model with four buffer cell libraries [7] of which the driving capabilities are one, two, seven, and ten times larger than those of a basic buffer cell. Inserting the buffers in the wire, various effects of the buffers on resulting performances were considered by changing the type of the buffers, the number of the buffers, and the place of insertion. It was found that the optimal result of the dual-rail scheme with buffers in 10 mm wire was obtained when delay and power consumption were 2.08 ns and 2504.62  $\mu\text{W}$ , respectively. In case of the 1-of-4 scheme, measured values of delay and power consumption were 1.97 ns and 1409.76  $\mu\text{W}$ , respectively. The incremental area due to the inserted buffers is about 11.3  $\mu\text{m}^2$  for both schemes.

The transmission delay in current-mode circuits is influenced by long wires, as well. The fundamental cause of the transmission delay is that input currents of  $I_{in0}$  and  $I_{in1}$  take a little time to reach given current levels, as is pictorially depicted in Fig. 3. In other words, owing to the capacitance load existing in long wires, constant delay is needed when entire amount of reference current from CS and CM in the encoder is copied to CM in the decoder. However, when the reference current is varying, time for its reconstruction in decoder can be reduced by charging and discharging small amount of current in long wires in advance.

We have designed a current steering logic (CSL) to accelerate the data transfer in the CMMVL circuit with long wires. In Fig. 6(a), P3 and N4 connected to final output of the encoder perform charge and discharge operations. A new **control logic** creates *chargeb* and *discharge* signals in the form of pulses to adjust the amount of current by using initial input signals, *req\_in* and *data\_in*. To guarantee correct detection, the amount of current for charge or discharge operation should not exceed the given level of reference current. Such limitation



(a) Comparisons of delay and power according to data width

(b) Comparisons of T\*P

Fig. 7. Comparisons with buffered dual-rail and 1-or-4 schemes

of the amount of current is effectively carried out by managing the resistance of P3 and N4 and pulse width of *chargeb* and *discharge* signals in **control logic**.

Figure 6(b) shows the simulation results of the CMMVL scheme with the CSL in the simulation environment of Fig. 2 with 10 mm wires. It is confirmed by the results that the transmission delay decreases due to the fast arrival time of reference current *Iin0* as compared to that of Fig. 3. The measured values of delay and power are 2.20 ns and 1360  $\mu$ W. In spite of the area overhead (about 21.67  $\mu$ m<sup>2</sup>) of **control logic**, the power consumption does not increase significantly, compared with the value 1041.06  $\mu$ W obtained without the CSL. This can be accounted for by the short circuit currents in XNOR gate and inverter of Fig. 1(b) shrunken drastically. Since the level of reference current is determined earlier by the CSL, node **A** and **B** experience much shorter transition of current level change.

As we note this practical data transfer scenario, the width of transferred data is extended up to 32-bit. Figure 7(a) illustrates simulation results of four different schemes, including the CMMVL scheme with the CSL, indicated by *delay\_adv* and *power\_adv* with 10 mm wires. As the number of bits increases, delays of four schemes increase with the unit of  $\log_2(\#ofbits)$ , which is due to the addition of stages in the C-element tree. Also, the power consumption increases linearly with the number of bits, maintaining a constant difference between individual schemes because the increment of power consumption is proportional to the number of wires.

In Fig. 7(b), the CMMVL scheme seems to be more advantageous than the dual-rail buffered scheme and it is competitive to the 1-of-4 scheme with the relatively large data size transmitted over long wires. In addition, compared with dual-rail and 1-of-4 schemes for 32-bit data transfer, the T\*P values of the CMMVL scheme with the CSL are decreased by about 55.5% and 8.5%, respectively.

## 5 Conclusions

We have designed DI data transfer mechanism using CMMVL circuits with only  $N+1$  wires for transferring  $N$ -bit data for which conventional DI data transfer mechanisms require  $2N+1$  wires. Proposed CMMVL circuits were thoroughly compared with traditional dual-rail and 1-of-4 DI schemes through the delay and power simulation in the transistor level with  $0.25\text{-}\mu\text{m}$  CMOS technology.

With non-buffered dual-rail and 1-of-4 schemes according to variable lengths of wire, experiments for 2-bit data demonstrate that the CMMVL scheme is superior to the others, leading to lower  $T^*P$  values with wire length over 4 mm. In the simulation environment with more practically designed dual-rail and 1-of-4 schemes, where buffers are inserted to wires for high-speed data transfer and data width is relatively large, the CMMVL scheme result in considerably lower  $T^*P$  values. Specifically, with 32-bit data and 10 mm wires, the CMMVL scheme reduces  $T^*P$  values of dual-rail and 1-of-4 schemes by 55.5 % and 8.5 %, respectively.

**Acknowledgments.** This work was supported in part from Brain Korea 21 Project and IC Design Education Center (IDEC).

## References

1. Furber, S. B., Day, P.: Four-phase micropipeline latch control circuits, IEEE Trans. VLSI Systems, Vol.4, No.2, (1996), 247-253
2. Sparso, J., Furber, S. B.: Principles of asynchronous circuit design: a system perspective, Kluwer Academic Publishers, (2001)
3. Bainbridge, W. J., Furber, S. B.: Delay insensitive system-on-chip interconnect using 1-of-4 data encoding, Proc. of the Symp. on Asynchronous Circuits and Systems, (2001), 118-126
4. Smith, K. C.: Multiple-valued logic: a tutorial and appreciation, IEEE Computer, Vol.21, (1988), 17-27
5. Jain, Atul K., Bolton, Ron J., Abd-El-Barr, Mostafa H.: CMOS multiple-valued logic design-part I: circuit implementation, IEEE trans. Circuits and Systems I: Fundamental Theory and Applications, Vol.40, No.8, (1993), 503-514
6. Weste, Neil H. E., Eshraghian, K.: Principles of COMS VLSI Design, Addison-Wesley publishing company, (1992)
7. CNU-IDEC cell library data book, IDEC Chungnam National University, (1999)
8. Advanced microcontroller bus architecture specification, Advanced RISC Machines Ltd, (1999)
9. CoreConnect, processor local bus architecture specifications, IBM Corp., (2000)
10. Jiang, Yi-Min, Cheng, Kwang-Ting: Vector generation for power supply noise estimation and verification of deep submicron designs, IEEE trans. VLSI Systems, Vol.9, No.2, (2001), 329-340

# Pipelines in Dynamic Dual-Rail Circuits

Jing-ling Yang<sup>1</sup>, Chiu-sing Choy<sup>2</sup>, Cheong-fat Chan<sup>2</sup>, and Kong-pong Pun<sup>2</sup>

<sup>1</sup> Department of Electrical and Electronic Engineering,  
The University of Hong Kong

<sup>2</sup> Department of Electronic Engineering,  
The Chinese University of Hong Kong

jlyang@eee.hku.hk, {cschoy, cfchan, kppun}@ee.cuhk.edu.hk

**Abstract.** This work investigates the inherent relations among latch-free dynamic pipelines (LFDP). Approaches to self-checking (SC) LFDP design are also presented.

## 1 Introduction

A pipeline is common paradigm for very high-speed computation. A pipeline provides high speed because it supports the parallel execution of many operations.

In static circuits, pipeline needs both storing and processing elements [1]. Unlike static circuits, dynamic pipeline can be designed without latches between pipeline-stages. Local events of the pipeline stages are used to control the dynamic computational stages to enter precharge or evaluation phases. The clever control sequences allow the pipeline-stages themselves to function as implicit latches [2].

Since LFDP is governed by local events, not an external clock. It is also called self-timing or asynchronous pipeline [3].

Self-timing control circuits are known to have some inherent SC characteristics [4], such as circuit activities cease in the presence of single and multiple stuck-at faults at gate inputs and outputs [5, 6, 7, 8].

Some authors have also demonstrated that single stuck-at, stuck-closed and stuck-open faults in most transistors in any Differential Cascade Voltage Switch Logic (DCVSL) circuit [9] yield either correct values or cause loss of complementarity at the outputs [10]. Such a property holds in multi-level DCVSL circuits in addition to single-level DCVSL circuits [11]. This inherent security against faults, of DCVSL circuits, allows them to be designed as efficient, strongly SC circuits [12].

LFDP, consisting only of asynchronous handshake control circuits and DCVSL computational blocks, can thus have the possibilities to be SC [13, 14, 15]. This work examines the relations among these existed LFDPs. Approaches to SC LFDP design are also discussed.

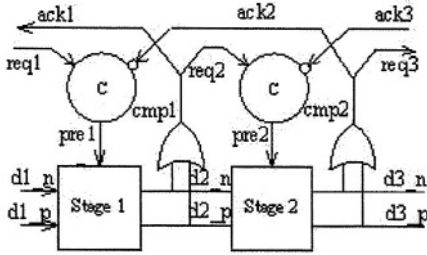


Fig. 1. Classical DAP Architecture

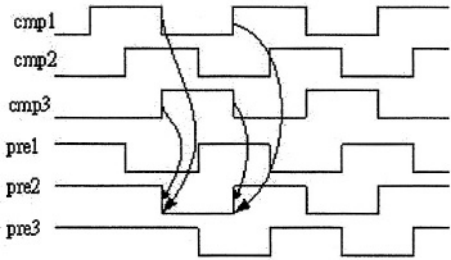


Fig. 2. Classical DAP Timing Diagram

## 2 Dynamic Asynchronous Pipeline (DAP)

The classical architecture of a DAP, using DCVSL blocks, event-driven latches and C elements, obeys a conservative handshake protocol that is commonly used in a static asynchronous Datapath [15].

Fig. 1 shows the handshake part of the classical architecture of a DAP. The “complete” signal of each pipeline stage is resolved from the dual-rail output. The stage block contains event-driven latches and DCVSL function unit

The working sequence of the classical DAP is: stage N performs evaluation when stage N-1 is full (stage N-1 has finished evaluation) while stage N+1 is empty (stage N+1 has finished precharge). Stage N enters precharge when stage N-1 is empty (stage N-1 has finished precharge) and stage N+1 is full (stage N+1 has finished evaluation). Fig. 2 shows this timing relationship, in which *cmp1*, *cmp2* and *cmp3* are complete signals from stage1, stage2 and stage3 respectively, and *pre1*, *pre2*, *pre3* are local control signals for stage1, stage2 and stage3.

The complete cycle (throughput) for a pipeline stage, as traced above, involves two evaluations, two precharge and two completion detections. The analytical pipeline cycle time *T* is:

$$T = E_1 + E_2 + E_3 + E_4 + \max(DE_0, DE_4) + T_c$$

Here  $E_1, E_2, E_3, E_4$  are the evaluation time of stage 1,2,3,4 respectively.  $DE_0$  and  $DE_4$  are stage 0 and 4’s evaluation completion detection time.  $T_c$  is the delay through the C-element.

## 3 William’s Pipeline

William’s pipeline [2] applies self-timing to avoid the need for high-speed clocks by directly concatenating precharged function blocks without latches. The self-timed control introduces no serial overhead, making the total chip latency equal to just the combinational logic delays of the data elements.

Fig. 3 shows William’s pipeline architecture. Each pipeline stage consists of a dynamic DCVSL function block, a completion detector and an inverter. The completion detector indicates the validity or absence of data at the outputs of the associated func-

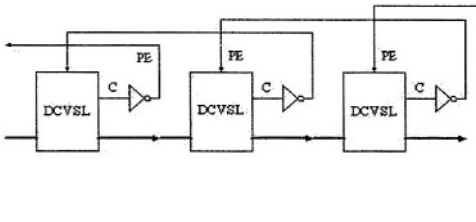


Fig. 3. Williams's Pipeline Architecture

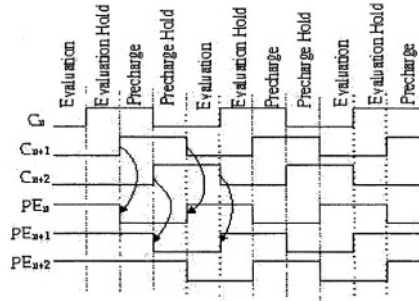


Fig. 4. William's Pipeline Timing Diagram

tion. Each pair of output data is checked by NAND gate, and then a C-element is used to combine all the results, and thus generate the complete signal.

The sequence of the pipeline control is: stage N is precharged when stage N+1 finishes evaluating; stage N evaluates when stage N+1 finishes precharge, whereas the actual evaluation commences only after valid data inputs have also been received from stage N-1. Fig. 4 is the handshake timing of William's pipeline.

The complete cycle for the first pipeline stage from one evaluation to the next evaluation is: (1) stage 1 evaluation; (2) stage 2 evaluation; (3) after stage 2 has finished evaluation, stage 1 is precharged, stage 2 remains in evaluation, stage 3 evaluates, and the three stages work in parallel; (4) after stage 3 has finished evaluating, stage 2 begins to be precharged; (5) stage 2 precharges; (6) when stage 2 has finished being precharged, stage 1 begins to evaluate again

The complete cycle (throughput) for a pipeline stage, as traced above, involves two evaluations, two precharge and two completion detections. The analytical pipeline cycle time T is:

$$T = E_1 + E_2 + E_3 + E_4 + DE_4 + T_{inv}$$

Here  $E_1, E_2, E_3, E_4$  are the evaluation time of stage 1, 2, 3,4 respectively,  $DE_4$  is stage 4's evaluation completion detection time.  $T_{inv}$  is the delay through the inverter.

### 3.1 Relationship Between Classic DAP and William's Pipeline

Fig. 1 shows the classical DAP architecture, the figure indicates that the stage N can enter evaluation when the stage N-1 is full (stage N-1 finishes evaluation) and the stage N+1 stage is empty (stage N+1 finishes precharge). The function block is dual-rail-coded, such that each data bit has only two types of data: 00 means that the data are idle, 01 or 10 means that the data are valid. When input data are idle, DCVSL maintains the precharged state. When the data are valid, DCVSL begins to evaluate. Therefore, the explicit request signal from stage N-1 can be replaced by the implicit dual-rail coded input data of the N stage. William's pipeline is implemented by this mechanism.



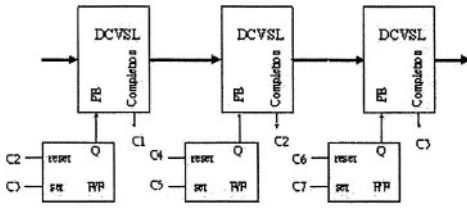


Fig. 5. Matsubara's Pipeline Architecture

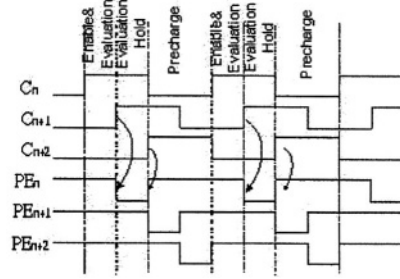


Fig. 6. Matsubara's Pipeline Timing Diagram

### 4 Matsubara's Pipeline

Matsubara proposed a zero-overhead structure, in which the precharge signal to the DCVSL function stage is removed sufficiently before the arrival of the effect data at the stage.

Fig. 5 shows Matsubara's pipeline architecture [16]. Each pipeline stage includes a dynamic DCVSL function block with completion indication, a flip-flop with the reset port connected to the complete signal from the N+1 stage; the set port is connected to the complete signal from the N+2 stage, and the output port Q is connected to the precharge /Evaluation (PE) port of the N stage.

The sequence of pipeline control is as follows: stage N is precharged when stage N+1 finishes evaluation; stage N evaluates when stage N+2 finishes evaluation. The actual evaluation begins only after valid inputs have also been received from stage N-1. Fig. 6 presents the handshake timing of Matsubara's pipeline.

The complete cycle for the first pipeline stage from one evaluation to the next evaluation is: (1) stage 1 evaluation; (2) stage 2 evaluation; (3) after stage 2 has finished evaluating, stage 1 begins be precharged, stage 2 remains in evaluation, stage 3 evaluates, and the three stages work in parallel; (4) after stage 3 finishes evaluating, stage 1 begins to evaluate again.

The complete cycle (throughput) for a pipeline stage, as traced above, involves two evaluations, one precharge and two completion detections. The analytical pipeline cycle time T is:

$$T = E_1 + E_2 + E_3 + DE_3 + T_{F/F}$$

Here  $E_1, E_2, E_3$  are the evaluation time of stage 1, 2, 3 respectively.  $DE_3$  is stage 2 and 3's evaluation completion detection time.  $T_{F/F}$  is the delay through the flip/flop.

#### 4.1 Relationship Between William's and Matsubara's Pipelines

Unlike William's pipeline, Matsubara's pipeline allows the precharge signal to be removed before the valid data arrive. So its cycle time does not include  $P_2$ , which is stage 2's precharge time.

### 4.2 Design Constraints

But Matsubara’s pipeline must also satisfy the precharge width requirement. That is equation  $(DE_2 + P_1 + DP_1 < E_3 + DE_2)$  must be satisfied. Here,  $E_3$  is the evaluation time of stage 3.  $DE_2$  and  $DE_3$  are stage 2 and 3’s evaluation complete detection time, respectively.  $D_1$  is stage1’s precharge complete detection time, and  $P_1$  is the precharge time of stage 1.

## 5 Montek’s Lp3/1 Pipeline

Montek’s LP3/1 [17] seeks to provide high throughput, without degraded latency.

Fig. 7 shows Montek’s 3/1 pipeline. Each pipeline stage includes a dynamic DCVSL function block with completion indication, a NAND with one positive input port connected to a complete signal of stage N+1, another negative input port connected to the complete signal of the N+2 stage and an output port connect to the PE port of the N stage.

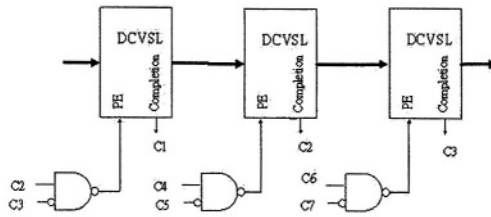


Fig. 7. Montek’s pipeline

The handshake arrangement and the timing diagram of Montek’s LP3/1 pipeline are exactly the same as those of Matsubara’s pipeline.

The analytical pipeline cycle time T is:

$$T = E_1 + E_2 + E_3 + DE_3 + T_{NAND}$$

Here  $E_1, E_2, E_3$  are the evaluation time of stage 1, 2, 3 respectively.  $DE_3$  is stage 3’s evaluation completion detection time.  $T_{NAND}$  is delay through the NAND gate.

### 5.1 Relationship Between Matsubara’s and Montek’s LP3/1 Pipelines

The handshake arrangement and the timing diagram of Montek’s LP3/1 pipeline are exactly the same as those of Matsubara’s pipeline. The only difference is that the handshake cell uses a NAND gate with a positive and a negative input port replacing the S/R flip-flop.

### 5.2 Design Constraints

The correct operation of the evaluation phase of LP3/1 pipelines requires that stage 2’s precharging is completed before stage 3’s ( $C_2 = 0$  arrives before  $C_3 = 0$ ). This requirement ensures a glitch-free evaluation phase when the NAND is used.

The reference time 0 is set when stage 2 has just completed evaluation. The time when  $C_3$  is asserted low is given by  $(E_3 + DE_3 + P_2 + DP_2)$ . Similarly,  $C_2$  is asserted low at time  $(DE_2 + P_1 + DP_1)$ .

Therefore, to maintain uninterrupted evaluation, the  $C_2 = 0$  should arrive at least a setup time  $T_{setup}$ , before  $C_3$  is asserted. That is,

$$DE_2 + P_1 + DP_1 + T_{setup} \leq E_3 + DE_3 + P_2 + DP_2$$

## 6 Locally Distributed Asynchronous Pipeline

The locally distributed asynchronous pipeline (LDAP) is a new pipeline scheme that was presented in [17]. It can be used in a fine-grain dynamic datapath in which the delay of each processing stage is less than that of the completion detector.

Fig. 8 shows the LDAP architecture. Each pipeline consists of a dynamic DCVSL block and a completion detector at each stage’s input port.

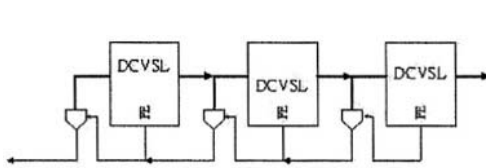


Fig. 8. LDA Pipeline

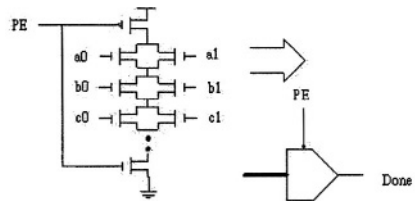


Fig. 9. LDA Completion Detector

A modified completion detector is required to generate the “early completion” signal. See Fig. 9. The completion signal is generated in parallel with the actual pre-charge or evaluation process of the associated function block, instead of after completion.

A complete cycle for the first pipeline stage from one evaluation to the next evaluation is: (1) stage 1 evaluation; (2) stage’s completion detector detects “early done” of stage 2’s evaluation (in parallel with stage 2’s evaluation), thereby asserts the precharge control of stage 1; (3) stage 3’s completion detector detects “early done” of stage 3’s evaluation (in parallel with stage 2’s evaluation), thereby asserts the precharge control of stage 2; (4) stage 2’ completion detector detects “early done” of stage 2’s precharge (in parallel with stage 2’s precharge), enables stage 1 to evaluate again in the next cycle.

The cycle time of the pipeline is:

$$T = E_1 + DE_2 + DE_3 + DP_2$$

## 6.1 Relationship Between William's Pipeline and LDAP

LDAP pipeline uses the same protocol as that of William's, a different method of completion detection. The complete signal is generated from the input signals instead of output signals.

## 6.2 Design Constraints

In LDAP, the following timing constraints must be satisfied to support the "early done" scheme,

$$E_N \leq DE_N + P_{N-1}$$

That's

$$DE_N \geq E_N - P_{N-1}$$

$E_N$  is the evaluation time of stage N,  $DE_N$  is the evaluation complete detection time of stage N, and  $P_{N-1}$  is the precharge time of stage N-1.

According to this constrain, the completion detector cannot be "too fast".

## 7 Self-Checking Requirements

The above discussion indicates that different ways to implement LFDP are available, but only two running sequences are used. One is four steps per cycle (they are evaluation, evaluation-hold, precharge and precharge-hold) without any assumption about time, as in classical LFDP and Williams's pipeline. LDA is also a four steps LFDP without any assumption about timing; its protocol is the same as William's. However the complete signal is generated from input signals rather than the output signals. Another is three steps per cycle (they are evaluation, evaluation-hold, precharge) based on some assumption about timing, as in Matsubara's and Montek's LP3/1 Pipeline.

The three-steps running mode is more efficient than the four-steps running mode, but its implementations need to satisfy some timing constraints, which will cause the related SC implementation difficult to design and analyze. So the following self-checking design scheme is based four-steps running mode which has no timing assumption.

In LFDP each pipeline stage involves a computation block and a handshake circuit. The connection between them is complete signals from computation block and local clock control signals from handshake circuits. A SC LFDP could be composed by a SC computation block and a SC handshake cell. A SC computation block should function as: any stuck-at fault in data signals will cause the complete signal can not be generated. A SC handshake circuits should function as: any stuck-at fault in complete signals from computation block will cause a local clock signal can not be generated. A SC computation block could be designed by using DCVSL circuits [10]. In the following we only consider the implementation of SC handshake circuit.

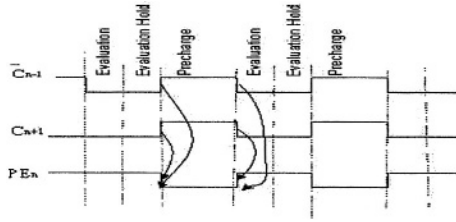


Fig. 10. Timing Diagram for Self-Checking Design

To achieve SC handshake cell, the following timing diagram, Fig. 10, could be used in designing an LFDP with self-checking ability. The signal in Fig. 10 is defined as,  $C_{n-1}$  is the complement of  $C_{n-1}$ , which is the completion signal of stages N-1.  $C_{n+1}$  presents completion signals from stages N+1 of a pipeline.  $CP_n$  is the local clock signal of stages N.

The sequence of pipeline control is as follows: stage N enters the precharge phase with the preconditions that stage N-1 has finished precharging, and the evaluation of stage N+1 has completed. Stage N enters the evaluation phase with the precondition that stage N-1 has been evaluated, and stage N+1 has been precharged.

Consider the timing diagram in Fig. 10. Its transition diagram can be expressed as follows:

$$PE_n \downarrow = \overline{C_{n-1}} \uparrow \cdot C_{n+1} \uparrow$$

$$PE_n \uparrow = \overline{C_{n-1}} \downarrow \cdot C_{n+1} \downarrow$$

These two expressions indicate that (1) if  $\overline{C_{n-1}}$  is stuck-at 0,  $PE_n \downarrow$  will not occur; if  $C_{n-1}$  is stuck-at 1,  $PE_n \uparrow$  will not occur. Therefore, a stuck-at fault in N-1 will stop stage N. (2) If  $C_{n+1}$  is stuck-at 1,  $PE_n \downarrow$  will not occur; if  $C_{n+1}$  is stuck-at 0,  $PE_n \uparrow$  will not occur. Therefore a stuck-at fault in N+1 will also stop stage N.

Here, we can conclude that a stuck-at fault in the completion signal of stage N will stop the operation of stages N-1, and N+1, which will further halt the entire pipeline.

Fig. 11 shows the implementation of a self-checking LFDP. Each pipeline stage includes a dynamic DCVSL function block with completion indication, a handshake cell with inputs connected to the complete signals from stages N-1 and N+1, and outputs connected to the PE control signal of stage N.

The sequence of the pipeline control is: stage N is precharged when stage N+1 finishes evaluating and stage N-1 finishes precharge; stage N evaluates when stage N+1 finishes precharge and stage N-1 finishes evaluation.

The complete cycle for the first pipeline stage from one evaluation to the next evaluation is: (1) stage 1 evaluates; (2) stage 2 evaluates; (3) after stage 2 has finished evaluation, stage 0 has finished precharge, stage 1 precharge; stage 2 remains in evaluation status; stage 3 is evaluating. So the three stages work in parallel; (4) after stage 0 has finished evaluation, and stages 1 and 2 have finished precharge, stage 1 begins to evaluate again.

The complete cycle (throughput) for a pipeline stage, as traced above, involves two evaluations, two precharges and two completion detections. The analytical pipeline cycle time T is:

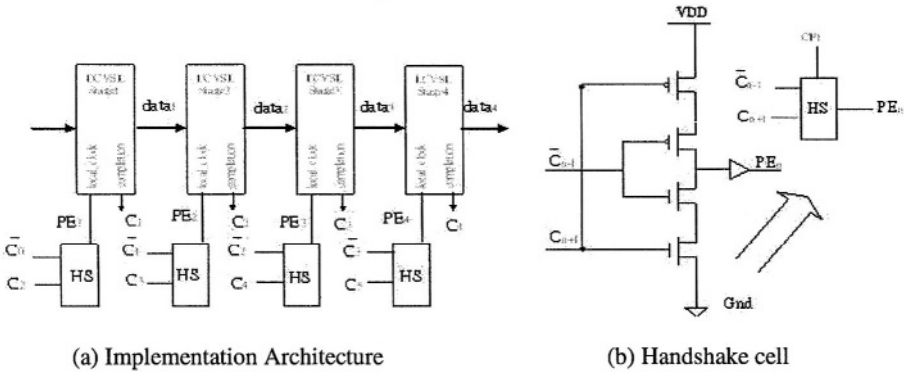


Fig. 11. Four-Step Self-Checking LFDP

$$T = E_1 + E_2 + E_3 + \max(DE_0, DE_3) + T_{HS}$$

Here  $E_1, E_2, E_3$  are the evaluation time of stage 1, 2, 3 respectively.  $DE_3$  is stage 3's evaluation completion detection time.  $T_{HS}$  is delay through the handshake cell.

### 8 Conclusions

The DCVSL circuit provides many opportunities for asynchronous pipeline to improve its handshake implementation. This work summarizes the relationships among the state-of-art LFDP designs. The improvements in performance and self-checking ability of LFDP designs are also discussed.

**Acknowledgement.** We would like to thank Prof Alfred, K. K. Wong of the University of Hong Kong for his encouragement and support.

Prof Wong can be contacted at awong@fortis-systems.com.

### References

1. I. E. Sutherland, "Micropipelines", communications of the ACM, vol.32, no.6, pp.720-738, June 1989.
2. Ted E. Williams, Mark A. Horowitz, "A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider", IEEE Journal of Solid-State Circuits, VOL.26, No.11, pp. 1651-1661, November 1991.
3. Al Davis, Steven M. Nowick, "An introduction to asynchronous circuit design" University of Utah. Technical "report, 1997.
4. John Wakerly, "Error Detecting Codes, Self-Checking Circuits and Applications ", Elsevier North-Holland, Inc., 1978.
5. H. Hulgaard, S.M. Burns, and G. Borriello, "Testing Asynchronous Circuits: A Survey", Integration, the VLSI J., vol. 19, pp.111-131, Nov. 1995.

6. P. Beerel and T.Y. Meng, "Semi-Modularity and Testability of speed-independent Circuits", *Integration, TheVLSIJ.*, Vol. 13, Sept, 1992.
7. Alain J. Martin and Pierter J. Hazewindus, "Testing Delay-insensitive Circuits", *Advanced Research in VLSI: Proceedings of the 1991 UC Santa Cruz Conference*, pp. 118-132. The MIT Press, Cambridge, MA, 1991.
8. V.I. Varshavky, editor, "Self-timed Control of Concurrent Processes", Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.
9. Chu, LK. M., and Puffrey, D.L. : "Design Procedure for Differential Cascade Voltage Switch Circuits", *IEEE J. Solid-State Circuits*, 1986, 21, (6), pp. 1082-1087.
10. Niraj K. Jha, "Testing for Multiple Faults in Domino-CMOS Logic Circuits", *IEEE Transaction on Computer-aided Design*, Vol 7, No. 1, January 1988.
11. Niraj K. Jha, "Strong Fault-Secure and Strong Self-checking Domino-CMOS Implementations of Totally Self-Checking Circuits", *IEEE Transactions on Computer-aided Design*, Vol., 9, NO. 3, March 1990.
12. Nick Kanopoulos and Nagesh Vasanthavada, "Testing of Differential Cascade Voltage Switch (DCVS) Circuits", *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 3, June 1990.
13. Jing-ling Yang, Chiu-sing Choy, Cheong-fat Chan, Kong-pong Pun, "Design for self-checking and self-timed datapath", *Proceedings of 21<sup>st</sup> VLSI Test Symposium*, 2003.
14. Ilana David, Ran Ginosar, Michael Yoeli, "Self-Timed is Self-Checking", *Journal of Electronic Testing: Theory and Application*, 6,219-228, 1995.
15. Divid A. Rennels and Hyeongil Kim, "Concurrent Error Detection in Self-timed VLSI", *FTCS-24*, Austin, Texas, June 15-17, 1994.
16. Gensoh Matsubara, Nobuhiro Ide, "A Low Power Zero-Overhead Self-Timed Division and Square Root Unit Combining a Single-Rail Static with a Dual-Rail Dynamic Circuits", *Proceedings of the Third International Symposium on Advanced Research in Asynchronous Circuits and System*, 1997, pp.198-209.
17. Montek Singh and Steven M. Nowick, "High-Throughput Asynchronous Pipelines for Fine-Grain Dynamic Datapath", *Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and System*, 2000, pp.198-209.

# Optimum Buffer Size for Dynamic Voltage Processors

Ali Manzak<sup>1</sup> and Chaitali Chakrabarti<sup>2</sup>

<sup>1</sup> Suleyman Demirel University, Isparta 32260, Turkey,  
manzak@mmf.sdu.edu.tr

<sup>2</sup> Arizona State University, Tempe, AZ 85287 USA,  
chaitali@asu.edu

**Abstract.** This paper addresses the problem of calculating optimum buffer size for a dynamic voltage scaling processor. We determine the minimum required buffer size giving minimum energy solution for periodic (single, multiple) or aperiodic tasks. The calculations are based on information about data size (maximum, minimum), execution time (best case, worst case), and deadlines.

## 1 Introduction

Dynamic voltage scaling is one of the most effective ways of reducing energy in a system with varying computational loads [1]-[4]. In this paper we consider the problem of optimum buffer size calculation for a dynamic voltage scaling processor. Inclusion of buffer is particularly important in multimedia applications since buffering stabilizes the fluctuating inter-arrival of input frames and allows for a constant processor speed. Since buffer is an additional hardware component that occupies real estate, finding the minimum buffer size that reduces the overall energy consumption is clearly an important problem. This problem is made harder by the fact that future data size and/or the execution time of the task is not known until the task is actually executed.

Use of buffers to *average* workloads was first presented in [5]. The assumption was that the exact workloads of all buffered tasks are known priori. More recently, [6] proposed a rate selection algorithm to distribute the workload under similar assumptions. They consider a practical scenario where the dynamic voltage system provides quantized voltage values instead of continuous values. The proposed algorithm evaluates the workload of all the tasks in the buffer and selects the appropriate quantized rate value. While task configurations with deadlines are not considered in [5] and [6], [7] considers multimedia applications where the task deadlines are relaxed. The scheme in [7] relies on the best case and worst case execution times (instead of exact workloads) to calculate the minimum buffer size to achieve maximum energy savings for single task, multiple sub-task and multi-task configurations. The calculations are based on the fact that maximum energy savings is obtained when the processor is not idle. While this fact is true for soft tasks, it is not valid for tasks with hard deadlines.



If latency can be tolerated (as in multimedia applications), a method to adjust task deadlines for further reducing energy consumption has also been presented.

In this paper we address the problem of finding the optimum (minimum) buffer size that minimizes energy. The minimum energy configuration corresponds to the case when the processor is not idle, the deadlines are not violated and all the task instances are assigned the same voltage. While for tasks with soft timing constraints, the idling condition is sufficient, for tasks with hard timing constraints, it is the task deadlines that determine the buffer size.

The rest of the paper is organized as follows. Section 2 describes the problem and the condition for energy minimization. Section 3 describes how to calculate the buffer sizes for periodic and periodic tasks. Section 4 concludes the paper.

## 2 Preliminaries

### 2.1 System Configuration

The basic architecture for a dynamic voltage scaling system is shown in Fig 1. The input data is first stored in the buffer and then sent to the processor core. The task scheduling algorithm specifies the operating voltage and frequency based on the task load.

The DC/DC converter has a sensitivity of  $\Delta V$  and can provide voltages in the range  $V_{min}$  to  $V_{max}$ . The oscillator provides a clock frequency that is related to voltage  $V$  by  $1/f = k' C_L \frac{V}{(V - V_t)^2}$ , where  $V_t$  is the threshold voltage,  $C_L$  is the load capacitance and  $k'$  is a device parameter (which depends on the transconductance and the width to length ratio).

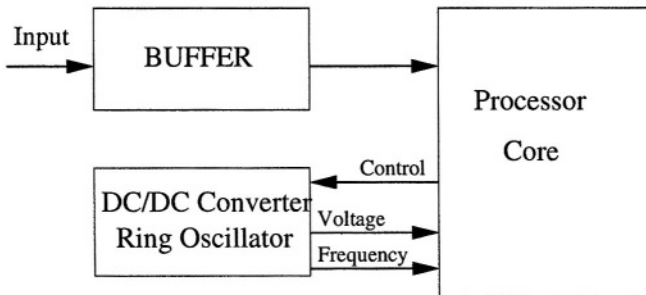


Fig. 1. System architecture

### 2.2 Definitions

The following parameters has been used in the rest of the paper. Task  $j$  has arrival time  $a_j$ , finish time  $f_j$ , deadline  $d_j = f_j - a_j$ , service time  $s_j$ , period  $P_j$ , worst case execution time of  $WCET_j$ , best case execution time of  $BCET_j$ , average case execution time of  $ACET_j$ , maximum data size of  $maxdata_j$ , minimum

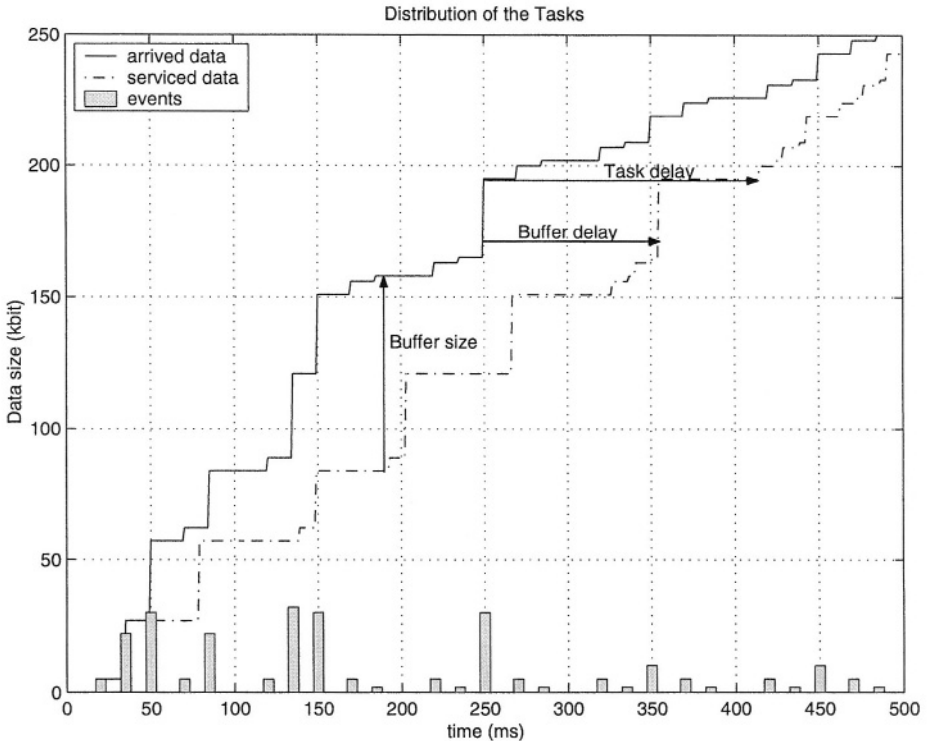


Fig. 2. Arrival and service curve for a simple example

data size of  $mindata_j$ , and average data size of  $avedata_j$ . The service time is defined as the execution time + maximum service time of the previous task, its arrival time}. In addition,  $Data_j(k)$  is the data size of the  $k^{th}$  instance of task  $j$ .

Next we explain the concept of arrival time, service time, arrival curve, service curve, buffer delay and buffer size with the help of the following simple example.

**Simple example:** Figure 2 shows the arrival times and sizes of individual events which are input to the system. For instance, data  $j$  is input at time  $a_j = 200ms$  and is of size 30 kbits. Furthermore, there is a delay associated with the buffer. This is defined as the difference between the time when data is input and when the data gets serviced. In Figure 2, data  $j$  which arrives at  $250ms$  is serviced at time  $s_j = 354ms$ . Thus the buffer delay is  $354 - 250 = 104ms$  for data  $j$ . If it takes 60ms to execute data  $j$ , then the finish time of task  $j$ ,  $f_j \geq 354 + 60 = 414ms$  for feasible assignment ( $f_j$  is the time where execution of task  $j$  is completed).

Figure 2 also describes the arrival and service curve for this example. The distribution of the event data is used to generate the arrival curve. Associated with each point in the arrival curve is arrival data,  $a(t)$ , defined as the total data that is input to the system at time  $t$ . Associated with each point in the service curve is serviced data,  $s(t)$ , defined as the total data that is sent to the processor

for execution at time  $t$ . In the example in Figure 2, at time  $t = 180ms$ ,  $a(t) = 157kbit$  and  $s(t) = 84kbit$ . Thus at time  $t$ , the system has to be able to store data of size  $a(t) - s(t) = 73kbit$ . For feasibility, the buffer size  $B \geq \max\{a(t) - s(t)\}$ , for all  $t$ .

In order to generate the service curve in Figure 2, all the tasks are executed at 2V (reference voltage  $V_{ref} = 3.3V$ ). This voltage assignment corresponds to the minimum energy solution. However this comes at the expense of a buffer size  $B$  of 80kbit and a buffer delay (maximum) of 170ms. If the given constraints (the buffer size and delay) are less than these values, it would not be possible to execute the tasks at 2V and the resulting energy reduction would be smaller.

### 2.3 Problem Statement

In this paper we address the following problem.

- Given a set of tasks,  $\gamma_1, \gamma_2, \dots, \gamma_n$ , each with its arrival time, deadline, period, data size (maximum and minimum data sizes), and execution time (WCET, BCET in cycles), find the optimum (minimum) buffer size such that energy is minimum.

The minimum energy configuration corresponds to the case when the tasks are assigned equal voltages [8](provided the other constraints are satisfied). Thus energy saving increases as buffer size increases, since tasks have a better chance of being assigned similar voltages. This statement is only valid when the tasks are soft. If the tasks have hard deadlines, there is an optimum buffer size and if the given buffer size is larger than the optimum value, the energy savings do not necessarily increase since deadline constraints still need to be satisfied. If the buffer size is smaller than the optimum value, the processor can be forced to be idle resulting in the task voltages not being optimally assigned.

## 3 Calculating the Optimum Buffer Size

In this section we describe a procedure to calculate the ‘optimum’ buffer size. The optimum buffer size is defined as the *smallest* buffer required for a minimum energy assignment. Recall that the minimum energy assignment corresponds to the case where all the tasks are being executed using the same voltage. Thus larger the buffer larger is the potential of obtaining the minimum energy assignment (ignoring the effect of additional energy required to access larger buffers). In this section we describe how to calculate the smallest or optimum buffer size for single periodic tasks (section 3.1), multiple periodic tasks (section 3.2) and aperiodic tasks (section 3.3). We consider two cases - one in which the execution time can be derived from the data size and one in which the execution time is independent of the data size. While the first case is more common, the second case occurs in video applications where the execution time is a function of the frame type or the extent of compression, and does not depend on the datasize.

In the derivation of the optimum buffer size, we consider two factors

1. *The processor should never be idle.* This is because energy savings are larger when the processor operates at lower voltage instead of idling.
2. *Task deadlines should be taken into account.* When deadline constraints are given, tasks should first satisfy the deadline constraints. While large deadline constraints give more opportunities for the tasks to be assigned closer voltages, they also require a larger buffer. Smaller deadline constraints require tasks to be executed earlier, thereby decreasing the number of tasks inside the buffer resulting in smaller buffer size.

**Optimum buffer size:** Minimum energy is obtained when the processor is never idle, and all the task instances are assigned the same voltages. In a system with *soft* deadline constraints such as in multimedia applications, preventing idling is a necessary condition for energy minimization. However, in a system with *hard* deadline constraints, the deadline constraints determine the buffer size. Let  $B_{idle}$  be the minimum buffer size that prevents idling and let  $B_d$  be the minimum buffer size that satisfies the deadline constraints. Choosing buffer size bigger than  $B_d$  is a waste, since all the tasks have to be executed before their deadlines and there won't be any situation where there is more than  $B_d$  data in the buffer. These results conclude that for a system with hard deadline constraints, the optimum buffer size  $B_{opt}=B_d$ .

On the other hand, in a system with soft deadline constraint there is no optimum buffer size. Bigger the buffer, greater chance of tasks in the buffer being assigned equal voltages. Preventing idling can be considered as a major factor for determination of buffer size for a soft task system.

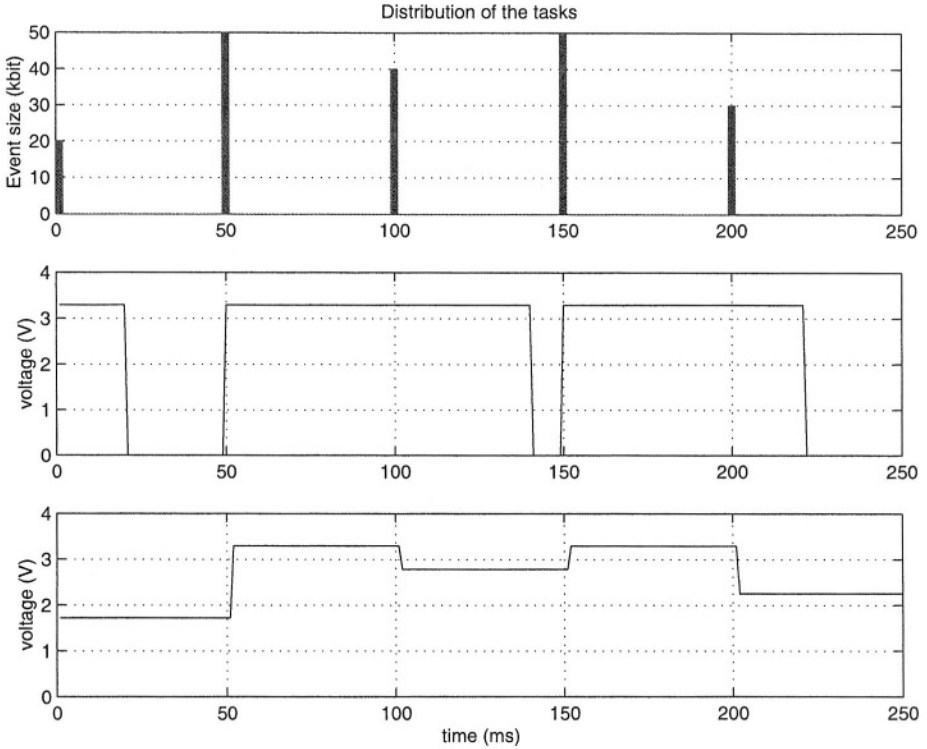
### 3.1 Single Periodic Task

We consider two cases, one in which the execution time can be derived from the data size and one in which the execution time cannot be derived from the data size.

**Execution time is proportional to data size:** We assume that maximum data size (maxdata) and minimum data size (mindata) are known and that the execution time can be determined from the data size.

**Prevent Idling:** Fig.3 describes a single periodic task with period equal to deadline and a period of 50ms (Fig.3).

The data sizes for different instants of the task vary from 20kbit to 50kbit (Fig.3a) and the corresponding execution times vary from 20ms to 50ms. When the processor executes all instances of the task at the reference voltage of 3.3V, we find that the processor is idle when the task size is small (Fig.3b). The slack can be exploited and the supply voltage of the processor can be reduced such that the processor is never idle as in Fig.3c. This implies that the current task should be completed before the next data arrives in the buffer. The minimum



**Fig. 3.** Example for single periodic task assignment where period=deadline: a) task size and arrival times b) task execution profile when processor operates at  $V_{ref} = 3.3V$  c) task execution profile when slack is utilized.

buffer size that prevents idling,  $B_{idle}$ , is then given by  $B_{idle} = 0$ . If, however, we require all tasks to be stored before their execution, the required buffer size is

$$B_{idle} = maxdata.$$

Thus, in this example, if  $B = 50kbit$ , the processor is never idle. The energy saving for the assignment in Fig.3c is 32% compared to the case when the processor runs at 3.3V and is idle at times (for the assignment in Fig.3b).

**Deadline constraint:** Now let us consider the case when the deadline of a task is equal to  $nP$ . For the minimum energy assignment, the most pessimistic case occurs when the execution of a task instance takes time  $nP$ . The buffer should be capable of storing the  $(n-1)$  data received during  $n$  periods. Then the required buffer size is  $(n - 1) * maxdata$ .

In general, if  $d'$  is the deadline, then the minimum buffer size that minimizes energy satisfying deadline constraints is given by

$$B_d = maxdata * (\lceil \frac{d}{P} - 1 \rceil)$$

**Optimal buffer size:** The optimal buffer size for hard real-time tasks is

$$B_{opt} = B_d = maxdata * (\lceil \frac{d}{P} - 1 \rceil) \tag{1}$$

**Execution time is not proportional to data size:** The parameters *maxdata*, *P*, *BCET* and *WCET* are known; the execution time of the data varies between *BCET* and *WCET* and is not known.

**Prevent Idling:** The processor can be idle when a task completes its execution early (*BCET*) and there is no task ready in the buffer to be executed. This situation can be easily prevented by arranging the supply voltage of the processor such that even the best case execution time takes *P* (period) sec. However, if the task is executed in *WCET*, then the above voltage arrangement will result in the task finishing computation at  $P * WCET/BCET$  sec. Then the buffer size should be large enough to store data coming in time  $P * WCET/BCET$ . Since only one task arrives every *P* sec, the required minimum buffer size is

$$B_{idle} = maxdata * \lceil \frac{WCET}{BCET} - 1 \rceil$$

A similar expression has been derived in [7].

**Effect of deadline constraint:** Let the voltage of the  $i^{th}$  instance of the task be such that *BCET* takes *P* sec. If the task is executed at *WCET*, the task finishes its computation at time  $P(WCET/BCET)$  sec. which should be earlier than its deadline. Thus

$$d \geq P(\frac{WCET}{BCET})$$

If  $d < P(\frac{WCET}{BCET})$ , we can not guarantee that the processor will not be idle and the buffer size is then given by

$$B_d = maxdata * (\lceil \frac{d}{P} - 1 \rceil)$$

But if  $d = P$  and  $P \geq WCET$ , the required buffer size is  $B = maxdata$  (theoretically there is no need for buffer). Note that no feasible solution can be guaranteed if  $P < WCET$ .

When deadline constraint is higher than  $P(\frac{WCET}{BCET})$ , the buffer size should be large enough to accept all the tasks received before the deadline. Note that if the current task execution is completed earlier than *WCET*, extra slack can be used for the remaining instances.

**Optimal buffer size:** The optimal buffer size for a single task with variable execution time is the same as that of a single task with execution time proportional to data size.

$$B_{opt} = B_d = maxdata * (\lceil \frac{d}{P} - 1 \rceil) \tag{2}$$

However there is a difference during scheduling of these two cases as we will see in Section 4.1. Since the exact execution time of the tasks are not known, the voltages of the tasks are assigned according to *WCET*. While this guarantees a feasible solution, it overestimates the voltages and decreases the energy savings compared to the case when execution time is proportional to data size.

### 3.2 Multiple Periodic Tasks

**Execution time is proportional to data size:** We assume that  $P$ ,  $maxdata$ , and  $mindata$  values of the tasks are known. The execution time can be derived from the data size.

**Prevent idling:** When there are multiple tasks with different periods and execution times, each task requires a different buffer size to prevent idling. The minimum calculated buffer size ensures that the processor is never idle. So, the task with the smallest buffer requirement determines the required buffer size. Ideally,  $B_{idle} = 0$  since the arrival time of the future data is known. However since the buffer can be overloaded when the different tasks arrive at the same time, the required buffer size to prevent idling is

$$B_{idle} = \sum_{j=1}^n maxdata_j$$

**Effect of deadline constraint:** The maximum data that can be stored in the buffer when the deadlines are satisfied is

$$B_d = \sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil)$$

This is because there can be maximum of  $\lceil \frac{d_j}{P_j} \rceil$  instances in the buffer.

**Optimal buffer size:** The optimal buffer size for a system with hard deadline constraints is given by

$$B_{opt} = B_d = \sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil) \quad (3)$$

**Execution time is not proportional to data size:** We assume that  $P$ ,  $maxdata$ ,  $mindata$ , WCET and BCET values of the tasks are known. The exact execution time is not known.

**Prevent idling:** Let  $P_{min}$  be the minimum period among all the tasks. This implies that the processor can be idle for atmost  $P_{min}$ . Idling can be prevented when the task with BCET takes  $P_{min}$  by operating at a lowered voltage. However, if any task is executed at WCET instead of BCET, the buffer should be large enough to hold all the new coming data during the execution of that task. The following equation gives the total data that can arrive at the buffer if task execution takes WCET (BCET is  $P_{min}$ ).

$$B_{idle} = \sum_{j=1}^n maxdata_j * (\lceil \frac{(WCET/BCET)_{max} P_{min}}{P_j} \rceil)$$

**Effect of deadline constraint:** The minimum buffer size that allows tasks to use all available deadlines is the same as the case where the data sizes are proportional to execution times.

$$B_d = \sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil)$$

**Optimal buffer size:** The optimal buffer size is for a system with hard deadlines is then

$$B_{opt} = B_d = \sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil) \quad (4)$$

### 3.3 Aperiodic Tasks

For aperiodic tasks, if no information on task data size, arrival and deadline time is available, the buffer size can not be calculated. If, on the other hand, maximum data size, arrival and deadline time are known, buffer size can be calculated for minimum energy.

**Prevent idling when execution time is proportional to data size:**

This is similar to the case of single periodic task. If  $maxdata$  is the maximum data size of all tasks, then

$$B_{idle} = maxdata$$

**Prevent idling when execution time is not proportional to data size:**

The maximum execution time that ensures that the processor is not idle for task  $j$  is given by

$$e_j = \left(\frac{WCET}{BCET}\right) * t_r$$

where  $t_r (= a_{j+1} - a_j)$  is the time the next data arrives. The buffer should essentially be able to store all the data that arrived during  $e_j$ .

$$B_{idle} = maxdata * [max\{number of data arrived during  $e_j\} - 1]$$$

**Effect of deadline constraint:**

Deadline constraint determines the optimal buffer size in both cases (execution time is or is not proportional to data size). Basically, the buffer should be capable of storing all the data that arrives during the execution of task  $j$  in the processor. Since this might take upto  $d_j$ , the deadline of task  $j$ ,  $B_d$  is given by

$$B_{opt} = B_d = maxdata * [max\{number of data arrived during  $d_j\} - 1]$$$

### 3.4 Summary and Results

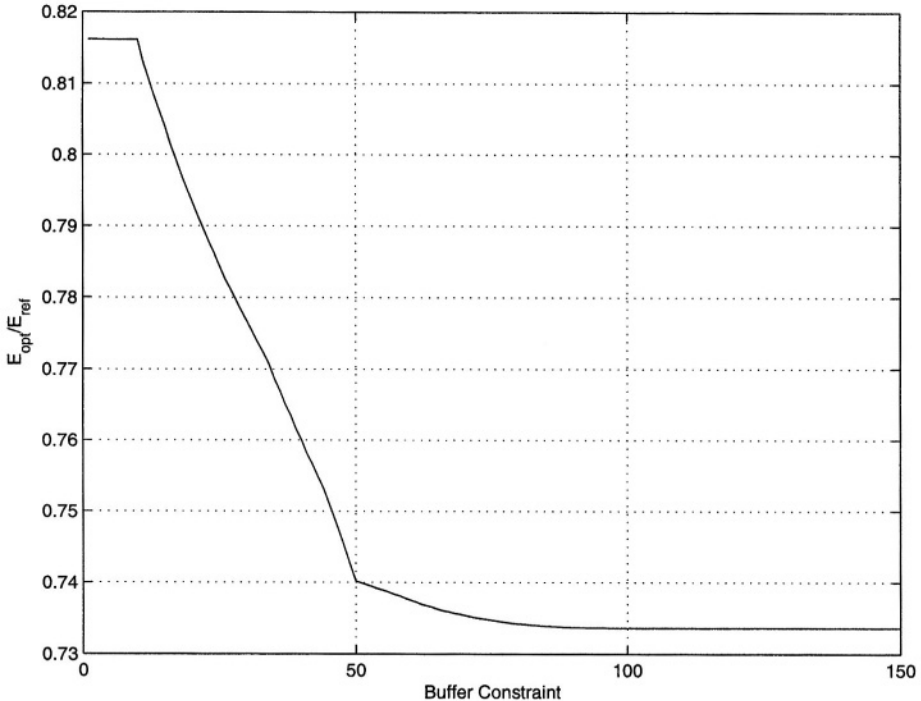
Table 1 summarizes the buffer sizes for single and multiple periodic tasks and aperiodic tasks.

**Table 1.** Summary of buffer size calculation

Tasks	$e$ prop datasize	Soft deadline, $B_{idle}$	Hard deadline, $B_d$
Single periodic	yes	$maxdata$	$maxdata * (\lceil \frac{d}{P} - 1 \rceil)$
	no	$maxdata * \lceil \frac{WCET}{BCET} - 1 \rceil$	
Multiple periodic	yes	$\sum_{j=1}^n maxdata_j$	$\sum_{j=1}^n maxdata_j * (\lceil \frac{d_j}{P_j} \rceil)$
	no	$\sum_{j=1}^n maxdata_j * (\lceil \frac{WCET/BCET}{P_j} \rceil_{max P_{min}})$	
Aperiodic	yes	$maxdata$	$maxdata * [max\{\# of data arr. in d_j\} - 1]$
	no	$maxdata * [max\{\# of data arr. in e_j\} - 1]$	

To verify the correctness of the table an experiment for single periodic tasks with task execution time  $e$  is proportional to data size has been done. Simulation parameters are chosen as following:  $period = p = 50$ ,  $maxdata = 50$  and  $deadline = 3 * p$ . Task voltages optimally assigned using the optimal offline scheduling algorithm in [8].





**Fig. 4.** Normalized energy versus buffer size constraint curve for 10 task assignment problem for deadline=3P.

We experiment with 1000 different task configurations, where each configuration consists of 10 different tasks with task execution times generated randomly between  $0.1maxdata$  and  $maxdata$ . Normalized energy versus buffer constraint curve is shown in Fig. 4, where  $E_{opt}$  is the optimum energy and  $E_{ref}$  is the energy when tasks are executed at reference voltage (3.3V). Buffer size is changed from 0 to 150 and average optimum normalized energy is calculated for each buffer size. As a result of theoretical calculation the optimum buffer size is 100. Figure 4 also confirms that no further energy savings is possible if buffer size is bigger than 100.

When buffer is less than 10, each task is scheduled in  $p = period$ . No task will be stored in buffer, since the smallest task is 10. Task starts scheduling whenever task arrives and finishes its scheduling before the next task arrives. When buffer size is bigger than 10, we start storing next data in buffer and more savings can be done arranging voltages of tasks optimally. When buffer is bigger than 50, next two tasks can be stored in buffer and more savings can be done. However energy savings do not improve much. As expected no more energy savings is possible when buffer is bigger than 100.

## 4 Conclusions

In this paper we addressed the problem of optimum buffer size selection for a dynamic voltage scaling processor. Our contribution is:

Determination of the optimum buffer size for periodic (single and multiple) and aperiodic tasks.

Our study shows that the energy savings increase with larger buffer sizes and relaxed deadline constraints. This is because in both cases, the chances of the tasks being assigned the same voltage are higher. However this improvement is very limited when certain buffer size has been reached. Also large buffers result in larger access times and larger energy consumption. These factors have not been considered in our analysis.

The task configurations that we have considered assume hard deadlines. However, in a real-time environment, there may be missed deadlines, missed tasks etc. that can be part of the QoS specification. We plan to extend our current minimum energy scheduling algorithms to maximise QoS.

## References

1. Yao, F., Demers A., Shenker, S.: A scheduling model for reduced CPU energy. *IEEE Annual Foundations of Computer Science*. (1995) 374–382
2. Ishihara T., H. Yasuura, H.: Voltage scheduling problem for dynamically variable voltage processor. *Proc. of the Int. Symp. on Low Power Design*. (1998)
3. Pering, T., Burd, T., Brodersen, R.: The simulation and evaluation of dynamic voltage scheduling algorithms. *Int. Symp. on Low Power Electronics and Design*. (1998) 76–81
4. Manzak, A., Chakrabarti, A.: Variable voltage task scheduling for minimizing energy. *Proc. of the Int. Symp. on Low Power Design*. (2001) 279–282
5. Gutnik, V.: Variable supply voltage for low power DSP. Master's thesis, Massachusetts Institute of Technology. (1996)
6. Chandrasena, L. H., Liebelt, M. J.: A rate selection algorithm for quantized undithered dynamic voltage scaling. *Int. Symp. on Low Power Electronics and Design*. (2000) 213–215
7. Im, C., Kim, H., Ha, S.: Dynamic voltage scheduling technique for low-power multimedia applications using buffers. *Proc. of the Int. Symp. on Low Power Design*. (2001) 34–39
8. Manzak, A., Chakrabarti, C.: Voltage Scaling for Energy Minimization with QoS Constraints. *Proc of Int. Conf. on Comp. Design*. (2001) 438–446

# Design Optimization with Automated Cell Generation

A. Landrault<sup>1</sup>, N. Azémard<sup>2</sup>, P. Maurine<sup>2</sup>, M. Robert<sup>2</sup>, D. Auvergne<sup>2</sup>

<sup>1</sup> LEAT, UMR 6071 CNRS, 254 rue A. Einstein, 06560 Sophia Antipolis, France.

<sup>2</sup> LIRMM, UMR 9928 CNRS, 161, rue Ada F-34392 Montpellier cedex 5 France.

**Abstract.** It is well recognized that designs based on automated standard cell flow have been found slower and larger in area than comparable designs manually generated or optimized. On the other hand it becomes necessary for designers to quickly prototype IP blocks in newly available processes. This paper describes an approach combining a performance optimization by path classification (POPS) tool with a transistor level layout synthesis tool ( $I^2P^2$ ) dedicated to CMOS synchronous design fast generation. Validations are given on a 0.18  $\mu\text{m}$  CMOS process by comparing standard cell approach to the proposed approach.

## 1 Introduction

Standard cell libraries have been successfully used for years as very efficient alternative to design very large circuits without transistor level verification. However with the increasing complexity of designs, this concept becomes less and less attractive and results in not so high quality designs. This is particularly true for what concerns the circuit speed and area [1]. It appears that most of the time standard cells are too generic and not well suited to the block being created. As a result the final design is not well optimized in terms of timing, power and area. As a solution to overcome the limitations of standard cell based design a library free technology mapping has been proposed [2]. It allows generating on the fly any digital cell at the size imposed by the timing constraint. The step to be solved is then the timing estimation and optimization of the design.

In this paper, we describe an investigation towards the possibility of combining a transistor level layout synthesis ( $I^2P^2$  [3]) and a performance optimization by path classification (POPS [4]). The main contribution of this paper is to link automatic layout generation of digital circuits and path performance estimation and optimization to generate high quality designs, by directly evaluating the interconnection load from the generated layout. This is an alternative to solve the timing closure problem.

The paper is organized as follows. In section 2, we describe the standard cell based approaches. In section 3, we present the new flow based on the combination of a transistor level layout synthesis (and the associated software engine integrated in the  $I^2P^2$  tool) with a delay/power performance optimization tool. In section 4, we describe the implementation of the interface between both tools. In section 5, we analyze the results obtained with this new “transistor level layout synthesis” that enables transistor level optimization before to conclude in section 6.

## 2 State of the Art: Design Methods Using Standard Cell Based Approach

The overall goal of the “Standard Cell” flows [5] is to generate the layout of the design from a behavioral description. The Boolean equations obtained after logical optimization are mapped onto a target pre-defined standard library. Afterwards the final layout is obtained after placement and routing of the footprint of the individual pre-characterized cells. All these standard cell based approaches involve a well-understood trade-off, and each cell of the library is speed and power characterized. Furthermore, the existing Industrial EDA tools are relatively well adapted to this approach.

Meanwhile some negative aspects emerge.

- With new UDSM technologies where interconnects are of significant importance it is more difficult to predict the cell drive and to satisfy the timing constraints. Timing closure is one of the most critical steps to satisfy in circuit design. Various iterations are needed between each point-tool of the flow. This can seriously damage the flow convergence.
- In fact, it is well known that the quality of designs highly depend on the library to be used [6] and of the variety of functionalities and sizes for each primitive gate [7]. Most of the designers recognize that standard cell libraries contain “too generic” cells [8], and that significant improvements can be achieved just by resizing some existing cells (drive continuity) or by adding a few customized cells (design dependency) in the initial library.

These drawbacks persist for new emerging approaches (such as “Liquid Library” or tools built around a unified data model) because the effectiveness of these alternative methodologies is highly dependent on the standard library development and process migration, which are costly.

## 3 Optimization at Transistor Level Using Layout Synthesis

We propose a standard cell independent based approach, working at transistor level using performance optimization by path classification tool. The idea beyond this approach is to realize the performance optimized final layout of a circuit, starting from its structural HDL description.

The main motivations to link these two tools are twofold.

- Firstly to avoid to support the library and standard cell generation using this adaptive library concept based on layout synthesis. The dependency to a standard library is very costly in term of time (around 6 months to develop and characterize a library) and in term of human resources (around 5 people/month) while it only requires less than a day, using  $I^2P^2$ , to migrate to a new process by creating a new technology file.
- Secondly to optimize the performance using a deterministic method based on path classification at transistor level. With the combined use of POPS and  $I^2P^2$ , the transistors can be individually sized or re-sized continuously (no discrete size limitations) using a fast and accurate modeling of the performance of CMOS gates.

To realize the technology mapping, a logical synthesis commercial tool is supplied with a functionality set called the “virtual library”. We associate to each function a virtual cell that can be considered as a set of connected transistors (only at symbolic level, no layout generated). The complete set of functionalities appearing in the virtual library is expected to imply less transistor count in the design. Actually, virtual cells appear as an interface between synthesis, place and route and the layout generation.

The proposed flow takes as input a behavioral description of the circuit function (VHDL or Verilog). It is composed of the two following steps.

- First a high-level logic synthesis and optimization, and a technology mapping are realized on a virtual library based approach.
- Followed by a transistor level layout synthesis step, at which the layout of the circuit is generated, using as inputs a set of constraints, a structural description of the design and the content of the virtual library. This includes placement, routing and physical layout generation. Timing analysis and optimization are realized using POPS.

The two developed tools ( $I^2P^2$  and POPS) achieve this last step. As it is illustrated in Fig.1, the “transistor level layout synthesis” tool starts from a structural description of the circuit and generates the layout in CIF or GDSII format.

This is developed around three major steps:

- creation of the virtual cells from their functionality, resulting in an associated transistor network,
- placement and routing based on a physical estimation obtained from the targeted layout style,
- timing analysis and optimization realized with POPS.

These features enable more optimized and refined performance results regarding the initial design constraints. Information exchanged between these tools is made through the common data structure of  $I^2P^2$  and the interface with POPS. The physical layout of each row is then automatically generated, with full respect of the technology rules.

The starting point of this flow is to provide a transistor net list with an optimal number of transistors corresponding to the required functionality, then to estimate and optimize the timing and power characteristics of the circuit before the final layout availability. The last point is to generate a dense layout of the net list, based on well-specified technology rules.

### 3.1 Layout Synthesis Tool

#### Layout Style

We have chosen, for the transistor-level generation, a layout style derived from the well-known “linear-matrix” [9] style, which is mainly characterized by the minimization of the space between the two N and P diffusion zones. The layout of the circuit is constructed by a sequence of NMOS and PMOS transistor rows and the routing is realized over the diffusion zones (gain in terms of area) and avoids the use of metal 2 to save the porosity of each row and to facilitate the routing step.

As described in Fig. 2, all the ports are placed at the center between N and P diffusions. This layout style has been chosen because it’s perfectly adapted to software implementation (regular style with vertical placement of the poly grid) [10]. With

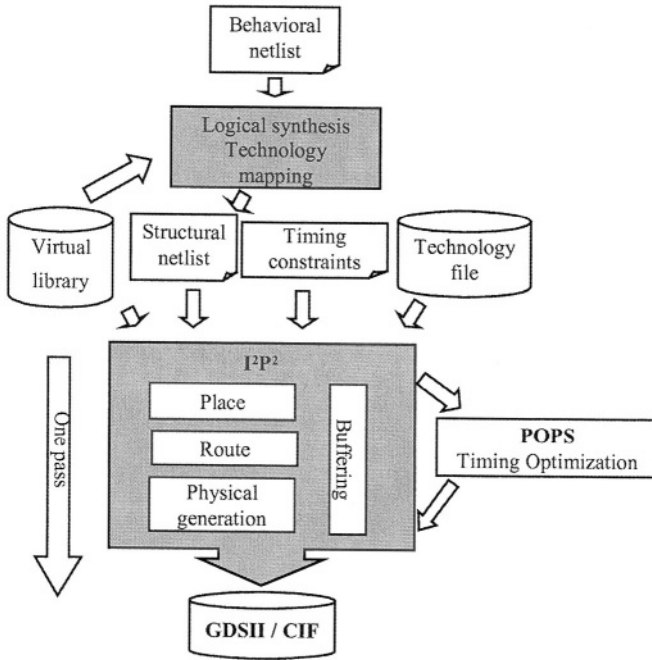


Fig.1. Flow description

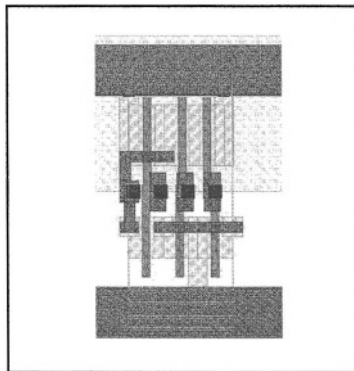


Fig. 2. Layout Style

respect to this style, a temporary layout can be very quickly generated for each virtual cell needed in the design. This layout used only for the prediction of the cell size and the port location will then be deleted. The final detailed layout of the design will be done only at the very end of the flow. Place, route and timing analysis tools will then use these accurate estimations of the width, the height and the port position during all the block generation steps.

### Concept of “Virtual Library”

The proposed flow presented here is based on the possibility to work directly at transistor level. By using transistors generated (at symbolic level first to fill the data structure) on the fly instead of using a static set of pre-characterized cells, we can perform performance optimization by resizing transistor instead of using a set of pre-packaged functionalities. We have to keep in mind that, no layout is generated before the final generation. Layout will be generated only for cell size prediction (temporary and then deleted) and at the very end of the flow at row-level.

The set of cells constituting the “Virtual Library” is mostly composed of complex gates or flexible cells. As a consequence the number of available cells is virtually unlimited (in terms of logical functionality). The constraints are only fixed by the target technology and performance limitations (maximum number of serial transistors).

In addition, as the final layout is automatically generated, the driving capability of all these “virtual cells” can be continuously modified using the POPS tool (by updating the transistor net list), at the contrary of the discrete drive possibility offered in usual standard cell design.

### Logic Optimization / Technology Mapping

“Virtual Libraries” are constituted of a large number of cells, compared to standard library, because the number of different logic functions available in CMOS technology are only limited by the maximum number of N and P transistors in series. For instance, for a maximum of four serial transistors (for each plan, N and P) we may dispose of a library with 3503 different functionalities with the added possibility of continuously sizing each logical function. This implies more flexibility to achieve the mapping step on a given circuit and results in less transistor numbers compared to the standard approach.

The virtual library also includes specific functionalities such as DFF or Muller cells. These functionalities are associated to Spice files, describing the transistor network.

## 3.2 Place and Route, Layout Generation

Place and route and layout generation tools implemented in  $\mathcal{I}^2\mathcal{P}^2$  have been vastly presented in previous work [3]. In this section, we briefly summarize the strategy of these tools.

### Virtual Cell Place and Route

Placement and routing steps are performed from the “Virtual Cell” structural net list representation using predictive performance analytical models.

- Firstly, a placement based on an iterative partitioning and global routing between each partition is performed [11]. Then cells are placed in each partition to allow rows creation. A global routing step is lastly done to generate virtual channels.
- Secondly, the symbolic view of each virtual cell (using the Euler’s trail solution [12]) is created. Afterward the symbolic transistors rows are generated then optimized (flipping, merging) and routed (“inner Row” maze routing).

- Finally, all the remaining connections are routed by a virtual channel router using detail routing algorithms (constraint graphs and multi-layer maze router).

### Layout Generation

The final layout generation consists in creating from the symbolic view of each row a constraint graph that enables to obtain a “compacted” layout in the “linear-matrix” style for any given technology rules. The main advantage of this procedure is to guarantee technology independence.

### 3.3 Timing Estimation and Optimization: POPS

Once the circuit transistor net list has been generated the essential steps of transistor sizing and timing analysis is performed. The efficiency of the proposed layout synthesis flow presented in Fig.1, and the ability to obtain an optimal circuit is conditioned by the following assumptions.

- The initial phase needs to produce a layout as optimal as possible.
- The modifications performed during the incremental processes need to be efficient and constructive, while impacting the minimal number of transistors, to allow a quick convergence.

Consequently, two new problems need to be addressed. Firstly, as there is no characterization involved in the Layout Synthesis flow, an accurate predictive model for speed and power is required.

For that we use a tool developed (POPS: Performance Optimization by Path Selection) for analyzing and optimizing the paths of a combinatorial circuit in submicronic processes. The following properties are targeted:

- path analysis and enumeration in a speed or power performance order, or in a speed/power trade off order
- accurate performance modeling (delay and power) considering submicronic effects [13],
- definition of local optimization protocols in order to reduce the problem complexity [14] by applying simple optimization criteria defined from explicit performance modeling.

The objective of this tool is to allow nearly ideal profiling of the speed/power distribution on the paths of a circuit. This ideally would impose that all the paths be at the targeted delay constraint. Speeding up or down the longest or shortest paths is obtained by operating on a restricted number of gates [14]. The selection on critical paths of poor drive capability or lightly loaded gates reduces the global optimization problem [15,16] to local optimizations allowing effective management of the circuit delay and power. Moreover minimizing the size of gates belonging to non-critical paths (shortest paths for example) is the natural alternative of power saving implementation techniques [17]. In this way, in a reasonable CPU time, it appears possible to manage speed or power on significant circuits.

Realistic delay evaluation based path evaluation and ordering gives facilities in circuit verification and optimization. This can be used for path classification in increasing or decreasing order of delays as well than for implementing power saving techniques or in trading speed for power. The application given in [4] shows clearly the



interest in controlling the path number for circuit optimization. It has been clearly demonstrated that if delay constraints can be satisfied with regular transistor sizing, local sizing on selected gates results in power minimized implementation. Considering the longest and shortest paths it has been clearly shown that circuit optimization can be obtained by sizing few paths of a circuit.

## 4 Implementation and Tools Interfacing

As previously discussed, in order to converge most efficiently during the optimization phase, the architecture of  $I^2P^2$  is based on a unique data-structure done in C++. This prototype integrates plug and play facilities to ease the exchange of the different “engines” under development such as POPS that has also been developed in C++.

The interfacing between POPS and  $I^2P^2$  tool is realized using the Spice format. Once the layout of a given circuit have been placed, routed and generated,  $I^2P^2$  produces the Spice format back-annotation of the circuit.

This Spice description file is then used as input for the external performance optimization tool. After the optimization step, POPS gives as output a net list (Spice format) containing the updated sizes of the transistors as shown in Fig.3

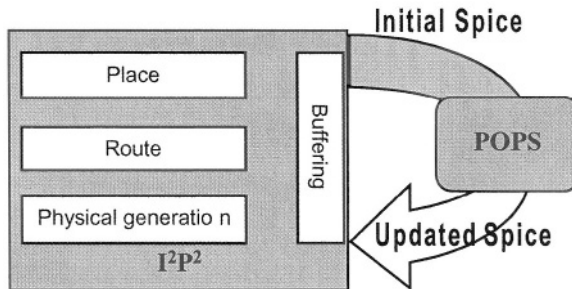


Fig. 3. POPS/  $I^2P^2$  interface

As illustrated, POPS modifies the initial Spice circuit net list, generated by  $I^2P^2$  in updating the transistor sizes of the critical path. This new sizes of transistors are then modified in the data structure of the  $I^2P^2$  software to enable a new final layout generation in CIF or GDSII format.

## 5 Results and Validation

Validations have been done in a  $0.18\mu\text{m}$  CMOS technology. The comparison has been done with respect to the standard flow using an In-House library. We have run our prototype on ISCAS85 circuits. We proceed to various comparisons on transistor number (logic synthesis efficiency), transistor density (area) and especially timing/power performances (transistor width) to evaluate the proposed approach.

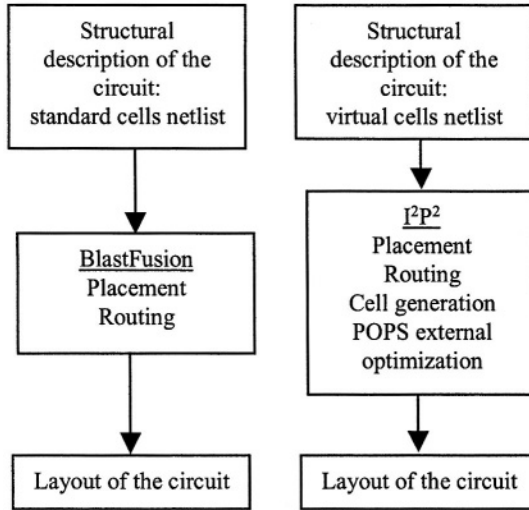


Fig. 4. Standard flow versus  $I^2P^2/POPS$  flow.

We validate the complete flow (POPS combined with  $I^2P^2$ ) with respect to the standard approach, on a set of benchmarks of circuits of various complexities (up to fifteen thousand transistors). We target, for each approach, the best achievable timing constraint. We reduce on purpose the number of functionalities in the standard library as the extension from simple gates (such as the Inv-Nand-Nor) to complex combinatorial gates delay representation is obtained using serial transistor array reduction technique. The benchmark metric is the sum of the transistors width, the timing and the final area. The two design flows are compared in Fig.4.

The standard flow is realized with BlastFusion from Magma.

### Transistor Number and Area Results

First, we compare the transistor number required to realize our benchmarks circuits and the associated area of the final layout. Results are summarized in Table 1.

Table 1. Transistor number and area comparison.

Circuit	Standard		$I^2P^2/POPS$	
	Transistor number	Area ( $\mu m^2$ )	Transistor number	Area ( $\mu m^2$ )
C17	26	242	26	216
C432	1270	51500	1258	45000
C880	1448	1147000	1438	120000

As shown, we obtain the same number of transistors, which is not surprising as we use the same set of functionalities to perform the mapping. The values obtained with the proposed flow ( $I^2P^2/POPS$ ) results in a smaller area /power implementation.

### Timing and Power Validation

By connecting the optimization (POPS) and the layout synthesis tools (I2P2), we target a performance improvement with respect to designs based on automated standard cell flow that have been found slower than comparable designs manually generated.

In Table 2 we compare the circuit performance in terms of timing and average transistor width. As illustrated the standard cell approach (Blastfusion) always result in a larger area implementation and a nearly equivalent delay. Next step, under development will be to compare the implementation area for an identical timing constraint. But considering the results given in Table 2 we can expect a much greater decrease in area using our automated synthesis tool.

**Table 2.** Timing and average transistor width comparison.

Circuit	Standard		I <sup>2</sup> P <sup>2</sup> /POPS	
	Sum of Transistor width (μm)	Timing (ns)	Average Transistor width(μm)	Timing (ns)
C17	40.4	0.178	30.4	0.176
C432	1950	1.18	1522	1.21
C880	3200	0.99	2450	0.98

This demonstrates that working at transistor level with the I<sup>2</sup>P<sup>2</sup>/POPS flow, give the possibility to resize in a deterministic way the transistors belonging to the cells to be sped-up (specific and continuous transistor resizing). This implies significant reduction of the global width of the transistors. As a direct effect, we can expect to obtain a substantial power dissipation reduction compared to standard cell approach.

### Run Time Analysis and Migration Facilities

These results concerning area, timing and power are quite encouraging mostly if we consider the facility obtained in generating and migrating macro-blocks in very short time (3minutes for the c880 circuits). Moreover, apart from the time necessary to update the technology file of I2P2 and to update the calibration of POPS (few hours), the circuits have been migrated from one process to another in a very short time (few minutes). This reinforces one of the main advantages offered by this approach: the possibility to quickly prototype IP blocks.

## 6 Conclusion

In this paper we have presented an original alternative to the classical standard cell based flow integrating a layout synthesis tool combined to a delay/power performance optimization tool. In this “virtual cell library” methodology the physical generation is obtained at the transistor level, integrating the different steps of physical layout generation and performance estimation. Optimization step is realized with the POPS tool that associates selective sizing technique to circuit path classification based on an incremental technique This approach may give great facilities in quickly evaluating performances (delay/power thanks to fast and accurate modeling) and prototyping different flavors of IP Blocks by using the latest available technology.

We have shown that by combining a performance optimization by path classification tool to the transistor level synthesis tool, it was possible to develop a prototype able to handle simple blocks (thousand of transistors). Optimization of hundred thousand transistor blocks is under development in a 0.13 $\mu\text{m}$  process.

## References

- [1] W. J. Dally, A. Chang, "The role of custom design in ASIC chips", proc. 37<sup>th</sup> design automation Conference, pp. 643-647,2000.
- [2] Reis, R. Reis, D. Auvergne, M. Robert, "The Library Free Technology Mapping Problem", IWLS, Vol. 2, pp. 7.1.1-7.1.5, 1997.
- [3] Landrault, L. Pellier, A. Richard, C. Jay, M. Robert, D. Auvergne, "An I.P. Migration and Prototyping Strategy Using Transistor Level Synthesis", DCIS'03, XVIII Design of Circuits and Integrated Systems Conference, Ciudad Real, Espagne, 19-21 November 2003,pp266-271.
- [4] N. Azemard, D. Auvergne, "POPS : A tool for delay/power performance optimization ", Journal of Systems Architecture, Elsevier, n°47, pp375-382, 2001.
- [5] D. McMillen, M. Butts, R. Composano, D. Hill, T.W. Williams, "An Industrial View of Electronic Design Automation", IEEE Transactions on Computer, Vol. 19, No 12, December 2000, pp. 1428-1448.
- [6] K. Keutzer, K Scott, "Improving Cell Library for synthesis", Proc. Of the International Workshop on Logic Synthesis, 1993.
- [7] K. Keutzer, K. Kolwicz, M. Lega, "Impact of Library Size on the Quality of Automated Synthesis", ICCAD 1987, pp. 120-123.
- [8] P. de Dood, "Approach makes most of synthesis, place and route - Liquid Cell ease the flow", EETimes, September 10, 2001, Issue: 1183.
- [9] A.D.Lopez, H.S.Law, "A Dense Gate-Matrix Layout for MOS VLSI", IEEE Transactions on Electron Devices, Vol. ED-27, No. 8, August 1980, pp. 1671-1675.
- [10] F.Moraes, R.Reis, L.Torres, M.Robert, D.Auvergne, "Pre-Layout Performance Prediction For Automatic Macro-Cell Synthesis", IEEE-ISCAS'96, Atlanta (USA), Mai 1996, pp. 814-817.
- [11] M. Fiduccia, R. M. Mattheyses, "A linear-time heuristics for improving network partitions.", Proceedings of the 19th Design Automation Conference, pages 175-181, 1982.
- [12] M.A. Riepe, K.A. Sakallah, "Transistor level micro-placement and routing for two dimensional digital VLSI cell synthesis", University of Michigan, Ann Arbor ISPD '99 Monterey CA USA.
- [13] M. Maxfield, "Delay effects Rule in Deep-Submicron Ics", Electronic Design, pp.109-121,1995
- [14] S.W.Cheng, H.C.Chen, D.H.C.Du,A.Lim, "the Role of Long and Short Paths in Circuit Performance Optimization", IEEE trans. On CAD of I.C. and Systems, vol. 13, n°7, pp.857-864, July1994.
- [15] R. Murgai, "On the Global Fanout Optimization Problem", In IWLS, Granlibakken, USA, 1999.
- [16] C.L. Berman, J.L. Carter, K.F. Day, "The Fanout Problem: From Theory to Practice", In C.L. Seitz editor, Advanced Research in VLSI: Proceedings of the 1989 Decennial Caltech Conferences, pp. 69-99, MIT Press, March 1989.
- [17] H.C.Chen, D.H.C.Du and L.R.Liu, "Critical Path Selection for Performance Optimization", IEEE trans. On CAD of Integrated Circuits and Systems, vol. 12, n°2, pp. 185-195, February 1995.

# A New Transistor Folding Algorithm Applied to an Automatic Full-Custom Layout Generation Tool\*

Fabricio B. Bastian<sup>1</sup>, Cristiano Lazzari<sup>1</sup>, Jose Luis Guntzel<sup>2</sup>, and Ricardo Reis<sup>1</sup>

<sup>1</sup> UFRGS - Universidade Federal do Rio Grande do Sul - Instituto de Informática  
Porto Alegre - RS, Brazil

{fbastian,clazz,reis}@inf.ufrgs.br

<sup>2</sup> UFPEL - Universidade Federal de Pelotas - Depto. de Matemática, Estatística e Computação  
Pelotas - RS, Brazil  
guntzel@ufpel.edu.br

**Abstract.** This paper presents a new folding algorithm applied to an automatic layout generation tool. Most of transistors sizing algorithms propose continuous sizing. Nevertheless, in row-based layout synthesis, the variation of transistor sizes may cause non-uniform cell heights that may lead to significant waste of layout area. The proposed folding approach leads to a very simple algorithm that is able to obtain very good results.

## 1 Introduction

Delay optimization of combinational blocks is still a very important issue in the design of digital circuits due to the continuous shrinking in device dimensions offered by the CMOS technologies [1]. In order to meet all the timing requirements, transistors with various current driving capabilities are required. Judicious increase of certain transistor sizes can reduce the circuit delay at the expense of additional chip area.

In row-based layout synthesis, the heights variation due to transistor sizing may cause non-uniform cell heights that may lead to a significant waste in layout area. In order to utilize the chip area more efficiently, a transistor folding scheme should be introduced.

Gupta and Hayes [2,3] and Krzysztof and Berezowski [4] proposed good solutions for cell generator tools. Although, the folding problem in a row-based layout tool is different, as we explain latter. Besides, Kim and Kang [5] developed an algorithm for row-based CMOS layout design which consider the folding before the transistor placement. Since our algorithm is applied to a timing-driven automatic layout generation tool and we do not desire to insert any extra diffusion gaps to the layout, the reorder of transistors is not an advantage.

This paper introduces a new transistor folding algorithm that can find the optimal folded transistors arrangement for a given transistor placement, given the desired transistors sizes and given the PMOS and NMOS transistors basic sizes (finger sizes). The proposed algorithm is able to fold transistors in a way that the initial transistor placement is not modified as in previously methods. All the folded transistors remain together in the row avoiding metal connections between them. This means that the P-plane and the

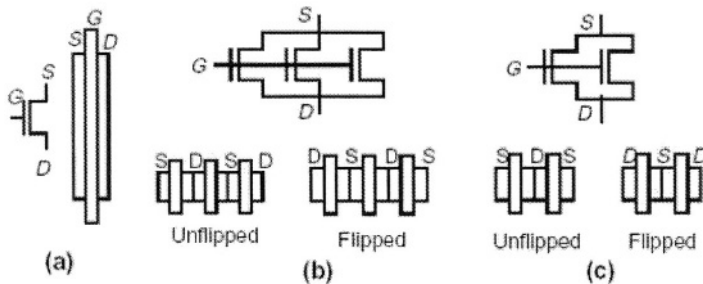
\* This work is supported by CAPES, CNPq and FAPERGS Brazilian Agencies

N-plane of a given cell can be folded independently, neither changing the layout topology nor increasing the design complexity (when folded transistors are connected using metal layers). Because of this flexibility, logics with non-complementary planes can be folded by our algorithm.

As this approach is supposed to be applied to an automatic full-custom layout generation tool based on a linear matrix style, the transistor placement is performed to maximize diffusion sharing not just inside cells but also between them. By increasing diffusion sharing, we are reducing the routing complexity, which is an important problem when using full-over-the-cell routing tools. Also, the transistor placement is the same for P-plane and N-plane, this way, the same poly line can be used to connect the entries of the complementary transistor gates.

## 2 Problem Overview

To attend timing constrains, transistors are sized to optimize circuit performance. The sizing process often results in non-uniform transistor heights, which can lead to a waste of layout area. Transistor folding is a technique that consists of breaking a transistor into smaller ones (legs or fingers). These legs are connected in parallel and must be placed together to reduce connections complexity.

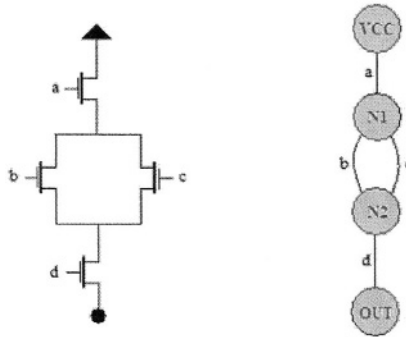


**Fig. 1.** (a) A transistor and its folding into (b) an odd (three) and (c) an even (two) number of legs

As illustrated in fig. 1, folded transistors may have different terminal layout, depending on the folding parity. Because of this, diffusion sharing can be affected. When a transistor with source and drain nets (S,D) (fig.1a) has to be folded to an odd number of legs (fig.1b), the nets at its ends remain (S,D). Nevertheless, when the transistor should be folded to an even number of legs (fig.1c) the end nets depends on the transistor's orientation: (S,S) if the transistor is placed unflipped, and (D,D) if the transistor is placed flipped, at this point, the diffusion sharing will certainly become affected.

The transistor folding is usually considered to be a constraint to chain formation. Although, it can also be used to reduce the number of diffusion gaps in the final layout as shown below.

Fig.2 shows a P-Plane (PMOS transistor plane is used on the examples without losing the generality) of an arbitrary cell. The presented example does not have an Euler path,



**Fig. 2.** A P-Plane of an arbitrary cell and its respective undirected graph

because it is not possible to walk over its entire respective undirected graph passing just once by every graph edge. It means that it is necessary to insert a diffusion gap in the transistor placement.

To demonstrate how the folding technique can reduce the number of diffusion gaps lets assume that the chosen placement is VCC-a-N1-b-N2-d-OUT-[GAP]-N2-c-N1. Folding the transistor d in two legs the diffusion gap in the transistor placement can be removed, causing a reduction on transistors' perimeters, which makes the capacitances smaller and the cell faster. So, by inserting transistors instead of diffusion breaks, a better performance and area are guaranteed.

As mentioned before, the folding problem on a row-based approach is different than on a standard-cell approach. Since the proposed algorithm is to be applied to an automatic full-custom layout generation tool based on a linear matrix style, and the sizing tool considers the routing connections, the cell placement cannot be changed at the expense of invalidating the previous transistor sizing. In the standard-cell approach, each cell is sized, folded and optimized individually. So, many of the folding algorithms [2,3,4] used to generate cell layouts do reorder of transistors during the folding. However, in a row-based layout, a cell that needs to be resized using the folding algorithm is inserted in a row and probably shares its diffusion areas with their neighbor cells. Since it is not desirable to insert extra diffusion gaps, and the complementary transistors share the same polysilicon layer, the transistor placement should not be changed.

Although, some important observations about the transistor placement should be done:

- The transistor placement algorithm may always try to start the path with VCC (GND) node and ends it with VCC (GND) node too. It is important because the only way that two neighbor cells have to share diffusion is through the VCC (GND) node.
- If the cell does not have an Euler Path and a diffusion gap has to be inserted, the first transistor after the gap needs to have at least one node in common with the last transistor placed before the gap. If this condition is not satisfied, the transistor folding will not be able to remove the gap when the restriction "all folded transistors may remain together in the row" is applied.

### 3 The Folding Algorithm

Considering the problem explained above, a folding algorithm was developed to attend all these constrains:

- The folded transistors must remain together in the row;
- The transistor placement should not be modified;
- And no extra gap must be inserted.

To satisfy the requirements, a dynamic folding with static placement approach was chosen, since the transistor placement should not be changed. So, given the placement, the desired size and the initial size of the transistors, the algorithm is able to determine the number of legs and the orientation for each transistor.

The folding is executed in two steps, first statically and then dynamically. In the static part the number of legs of each transistor is determined simply dividing the size of the resized transistor ( $S_{resized}$ ) by the folding size ( $S_{folding}$ ):

$$N_{legs} = S_{resized} / S_{folding}$$

The static folding may introduce diffusion sharing problems, consequently adding extra gaps to the layout. Fig.3a shows the P-MOS plane of a cell that should have its height duplicated, fig.3b shows its respective undirected graph and fig.3d illustrate the row fragment that the cell is inserted, a diffusion gap is necessarily inserted because the cell does not end by a VCC node, transistors X and Y represents the neighbor cell's transistors. In fig.3c the cell had its height duplicated by static folding. As previously stated, the folded transistors have to be placed contiguously, so the static folding will imply the insertion of a new gap (VCC-a-N1-a-VCC-b-N1-b-VCC-c-N1-c-VCC-[GAP]-N1-d-S-d-N1-[GAP]). To exemplify that reordering of transistors cause no improvements, a new

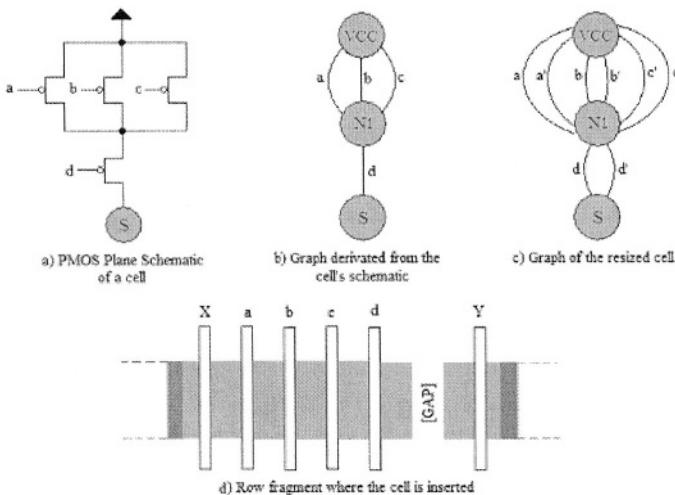


Fig. 3. An example of a situation that static folding can cause diffusion sharing problems



path is chosen (an Eulerian Path) [GAP]- N1-a-VCC-a-N1-b-VCC-b-N1-c-VCC-c-N1-d-S-d-N1-[GAP], and the two gaps were obtained again, so the problem was not solved, but shifted. It is important to stand out that if all transistors need to be folded by an odd number of legs, the static folding is sufficient in most of cases, because no extra gap is inserted. So, if the layout has no removable gaps and all transistors must be folded by an odd number of legs, the dynamic folding do not need to be executed (as stated before, the nets at the ends of the transistors remain equal).

After the static folding, the dynamic folding is executed. The main idea of the dynamic part is to classify the transistors according to its parity and respecting it in such a way that no new diffusion gaps are inserted. The parity of the transistors are considered as follows:

- An even transistor is a transistor that, after folded, should provide a path to return to a node. An even transistor can be seen as there is no path between two nodes.
- An odd transistor is a transistor that, after folded, should provide a path to go to another node. An odd transistor can be seen as there is just an edge between two nodes.

An even transistor will always be resized by an even multiplicity of legs as well as an odd transistor will always be resized by an odd multiplicity.

To classify the transistors, the algorithm walks over the undirected graph applying the definitions explained above for each transistor, considering the given placement and the transistor chain. The classification of a transistor depends on how it is connected to its neighbor transistors. During the classification, the transistors can receive a temporary classification, which can be changed along the analysis, as exemplified below.

Considering the placement VCC-a-N1-b-VCC-c-N1-d-S-[GAP] (fig.3d), the steps to classify the transistors represented in the graph of the fig.3c (resized cell with folded transistors):

1. *Passing by transistor a*: it is odd, because it gives a path to go from VCC to N1 (go from one node to another).
2. *Passing by transistor b*: now the analysis is made to the pair of transistor a and b. This pair of transistor together are even, because they give a path to go from VCC to VCC, in other words, they give a path to return to VCC node. So, transistor a is reclassified as an even transistor, and transistor b is classified as an even transistor.
3. *Passing by transistor c*: it is odd, because it gives a path to go from VCC to N1.
4. *Passing by transistor d*: it is odd, because it gives a path to go from N1 to S. Just at this part of the algorithm transistor c is definitively marked as an odd transistor.

As the placement is not changed, the algorithm can be simplified to walk over a list instead of a graph, like shown below by the following pseudo-code program.

```
s = begin of the list;
r: if s not equal to the end of the list{
    buffernode = s;
    s++;
    atualtransistor = s;
    classify currenttransistor as odd;
```

```

    if the orientation of the currenttransistor is not
        defined{
            orientation of the currenttransistor = unflipped;}
s++;
currentnode = s;
while s not equal to the end of the list{
    buffertransistor = currenttransistor;
    s++;
    currenttransistor = s;
    s++;
    if currenttransistor = [GAP]{
        Remove the gap if possible and return to r;}
    if s = buffernode{
        buffernode = null;
        currentnode = s;
        buffertransistor is classified as even;
        currenttransistor is classified as even;
        if the orientation of the currenttransistor is not
            defined, define it;
    }else{
        buffernode = atualnode;
        currentnode = s;
        currenttransistor is classified as odd;
        if the orientation of the currenttransistor is not
            defined, define it;
    }
}
}
}

```

On the above pseudo-code program, *s* represents a pointer variable that walks over the list, *buffertransistor* means the anterior transistor and *currenttransistor* is the transistor that is been classified an oriented. The list used is the sequence of nodes and transistors that represents the transistor placement. Considering the placement of the fig.3d the list obtained is VCC-a-N1-b-VCC-c-N1-d-S-[GAP].

The orientation of each next transistor is determined by the parity and the orientation of its anterior transistor:

- If the buffertransistor has even parity and unflipped order, the currenttransistor will be placed flipped;
- If the buffertransistor has even parity and flipped order, the currenttransistor will be placed unflipped;
- If the buffertransistor has odd parity and unflipped order, the currenttransistor will be placed unflipped;
- If the buffertransistor has even odd and unflipped order, the currenttransistor will be placed unflipped;

As stated at the end of section 2, the diffusion gaps can only be removed if the transistor placement satisfies the requirements, otherwise the gaps will remain in the layout. The pseudo-code program below shows how the diffusion gaps are removed.

```

if currenttransistor = [GAP]{
  tempgap1 = s;
  tempgap2 = buffernode;
  if tempgap1 = tempgap2{
    if the buffertransistor is even and flipped{
      reclassify the buffertransistor as odd;}
    if the buffertransistor is odd and unflipped{
      reclassify the buffertransistor as even;}
    define the order of the nexttransistor as unflipped;
  }else{
    tempgap2++;
    tempgap2++;
    if tempgap1 = tempgap2{
      if the buffertransistor is even and unflipped{
        reclassify the buffertransistor as odd;}
      if the buffertransistor is odd and unflipped{
        reclassify the buffertransistor as even;}
      define the order of the nexttransistor as unflipped;
    }else{
      tempgap1++;
      tempgap1++;
      if tempgap1 = tempgap2{
        if the buffertransistor is even and unflipped{
          reclassify the buffertransistor as odd;}
        if the buffertransistor is odd and flipped{
          reclassify the buffertransistor as even;}
        define the order of the nexttransistor as flipped;
      }else{
        tempgap2--;
        tempgap2--;
        if the buffertransistor is even and flipped{
          reclassify the buffertransistor as odd;}
        if the buffertransistor is odd and flipped{
          reclassify the buffertransistor as even;}
        define the order of the nexttransistor as
          flipped;}
    }
  }
}
}
}

```

The variable *nexttransistor*, on the pseudo-code program above, represents the next transistor that will be classified and oriented (next *currenttransistor*).

Considering the undirected graph of the fig.3b with the placement VCC-a-N1-d-S-[GAP]-VCC-b-N1-c-VCC, a gap is obtained. Supposing the height of all cell must be duplicated, after a static folding the obtained placement, in the better case, will be VCC-a-N1-a-VCC-[GAP]-N1-d-S-d-N1-b-VCC-b-N1-c-VCC-c-N1-[GAP], with the insertion of one extra gap. But applying the proposed algorithm the new obtained placement is VCC-a-N1-a-VCC-a-N1-d-S-d-N1-b-VCC-b-N1-c-VCC-c-N1-c-VCC with no gaps and no placement modification. The transistors classified as odd are folded in three legs and the transistors classified as even are folded in two legs. As the obtained placement shows, an extra transistor was inserted for each gap, which reduces the transistors' perimeters, consequently reducing the capacitances, the area and making the cell faster.

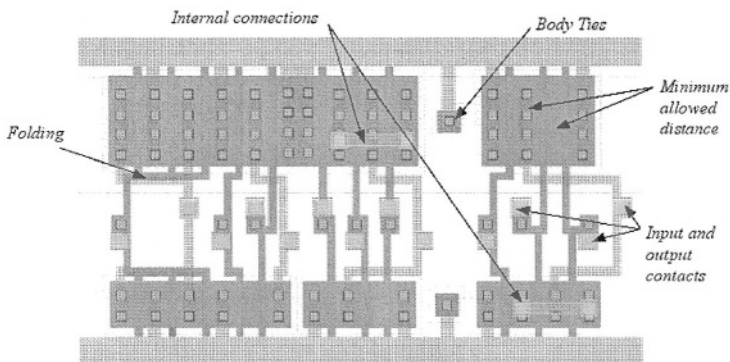
The algorithm has just one while structure which makes its complexity small and its execution time almost linear to the number of transistors that must be classified.

## 4 Automatic Full Custom Layout Generation

A full automatic layout generator called Parrot Punch is presented in [6] where a layout generation strategy is given for three metal layer CMOS technologies. This tool is now extended to any number of layers and the folding technique presented in this work is applied. Parrot Punch associated with Tictac::Sizing [7] is able to generated circuits optimized in delay.

Standard cell methodology can be used for ASIC design because the layout process is fully automated. However, standard-cell approaches are based on a limited number of cells and it imposes restrictions in the synthesis. Moreover, different versions of each cell are required to drive different strengths and it increases the cell libraries to hundred or thousands of cells.

On the other hand, automatic full custom layout generators develop layouts based in a list of transistors and design rules provided by the foundry. It means that cell libraries are not used and any kind of cell can be designed on demand. In addition, this method is completely flexible to create cells optimized to each location where they are inserted.



**Fig. 4.** An example of layout generated by Parrot Punch

Fig.4 shows an example of layout generated by Parrot Punch. It is illustrated a row in which transistors are placed based on the linear matrix style. The main characteristics are input/output placed between PMOS and NMOS transistors, internal connections performed in metal 1 and the attempt for using minimum values of a technology. Folded transistors are also shown in the example where a transistor is divided in three legs.

## 5 Results

Table 1, Table 2 and Table 3 show the results of the proposed algorithm over different circuits. Cases where the delay reduction does not correspond to the delay requirement show limitations of the gate sizing method used on the layout generation tool. As the algorithm allows PMOS Plane and NMOS Plane sizing independently, the number of PMOS and NMOS folded transistor may be different.

Table 3 shows important results: the circuit area has been reduced after the folding. It happens because the folding inserts extra transistors, which give more space to the

**Table 1.** Results of the proposed algorithm applied to circuit csa8 (288 transistors) with the required delay optimizations

Delay Requirement	N.Opt.	10%	30%
Area	$6500\mu^2$	1%	3%
Delay Reduction		7%	20%
Sized Gates		4	9
NMOS folded transistors		12	25
PMOS folded transistors		2	37

**Table 2.** Results of the proposed algorithm applied to circuit bw (628 transistors) with the required delay optimizations

Delay Requirement	N.Opt.	10%	20%	30%
Area	$16068\mu^2$	4%	6%	10%
Delay Reduction		10%	20%	25%
Sized Gates		4	8	23
NMOS folded transistors		4	12	42
PMOS folded transistors		2	13	30

**Table 3.** Results of the proposed algorithm applied to circuit alu2 (1390 transistors) with the required delay optimizations

Delay Requirement	N.Opt.	10%	20%	30%
Area	$40716\mu^2$	-2%	-9%	-5%
Delay Reduction		10%	20%	27%
Sized Gates		5	8	22
NMOS folded transistors		19	26	59
PMOS folded transistors		13	24	56

routing tool, that do not need to insert empty spaces to complete the routing, reducing the final area.

The results present that good delay optimizations were done with a small area penalty. This proves the algorithm effectiveness.

## 6 Conclusions

A new folding algorithm was presented as well as some results when applied with the automatic layout generation tool called Parrot Punch.

The algorithm uses a new approach for folding, based on the parity of the Transistors and avoiding diffusion gaps. So, reducing the transistors perimeters, which make the cells faster.

Results show that good delay optimizations can be done with small area penalty, which means that many transistors can be resized with almost no impact to the layout area of the circuit.

## References

1. Reis, R. Power and Timing Driven Physical Design Automation. PATMOS2003 - 13th International Workshop on Power and Timing Modeling, Optimization and Simulation, Torino, September 10-12, 2003. LNCS Springer Verlag
2. Gupta, A., Hayes, J.P.: Optimal 2-D Cell Layout with Integrated Transistor Folding. ICCAD (1998) 128-135
3. Gupta, A., The, S-C, Hayes, J.P.: XPRESS: A Cell Layout Generator with Integrated Transistor Folding. European Desing & Test Conf. (1996) 393-400
4. Berezowski, K.S.: Transistor Chaining with Integrated Dynamic Folding for 1-D Leaf Cell Synthesis. EUROMICRO Digital Systems Design Symposium (2001)
5. Kim, J., Kang, S.M.: An Efficient Transistor Folding Algorithm for Row-Based CMOS Layout Design. Design Automation Conference (1997) 456-459
6. Lazzari, C., Domingues, C. V., Güntzel, J., Reis, R.: A New Macro-cell Generation Strategy for Three Metal Layer CMOS Technologies. Vlsi-Soc. Darmstad-Alemanha (2003) 193-197
7. Santos, C. L., Wilke, G., Lazzari, C., Güntzel, J., Reis, R.: A Transistor Sizing Method Applied to an Automatic Layout Generation Tool. Sbcci. São Paulo (2003) 303 - 307

# A Novel Constant-Time Fault-Secure Binary Counter

Dimitris Karatasos<sup>1</sup>, Athanasios Kakarountas<sup>1</sup>,  
George Theodoridis<sup>2</sup>, and Costas Goutis<sup>1</sup>

<sup>1</sup> VLSI Design Laboratory, Dpt. of Electrical & Computer Eng., University of Patras,  
Patras, GR – 26500, Greece

{dkaratasos, kakaruda, goutis}@ee.upatras.gr  
<http://www.vlsi.ee.upatras.gr>

<sup>2</sup> Electronics Dvn., Dpt. of Physics, Aristotle University of Thessaloniki,  
Thessaloniki, GR – 54006, Greece  
theodor@physics.auth.gr

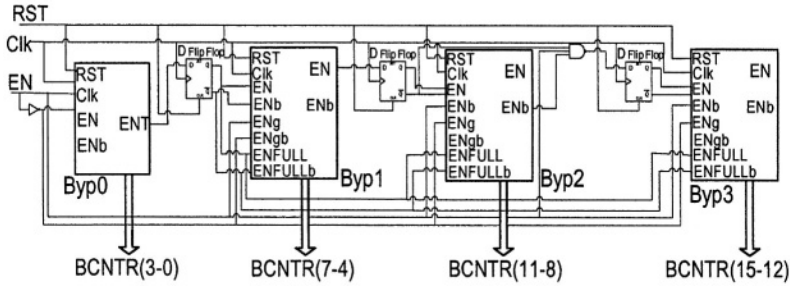
**Abstract.** A novel constant-time fault-secure binary counter is presented. This counter is appropriate for performance aware safety critical systems. This is achieved due its ability to read its binary output on-the-fly. The safety scheme is based on parity prediction technique achieving concurrent on-line testing. The experimental results exhibit high levels of safe operation, keeping the performance of the Constant Time Binary Counter almost linear to its size increase, while area overhead is lower than that of any counter based on a multi-channel technique.

## 1 Introduction

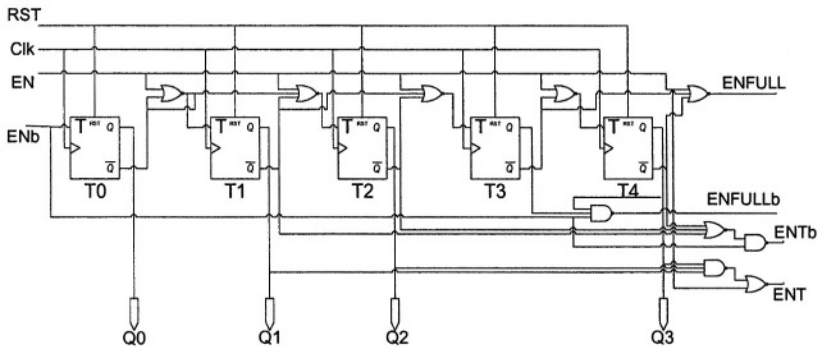
In every computational system, counters are fundamental elements, which are mainly used to generate synchronization signals and/or provide sequences of states. Implementation of large counters, when needed in a system can generate critical design regarding the overall performance. Typical counter implementations are output's bitwidth dependant. Thus, special structures have been proposed to enable counting independent of the counter's size [1],[2]. These counters are characterized as constant time counters. Additionally such counters are expected to have a binary output, which can be read on-the-fly.

Although there are several implementations of constant time binary counters, they cannot be used as is for safety critical systems. Such systems require on-time error detection and in the case of concurrent on-line testing, detection must be performed in near real-time. Thus, the design of fault-secure counters turns to be extremely important for safety critical systems. However, modification of components into fault-secure introduces significant delay; degrading fault-secure component's overall performance. Thus, a high speed, or a constant-time counter is considered ideal to be turned into fault-secure, in order to exploit its performance behavior and reduce the resulted time penalty.

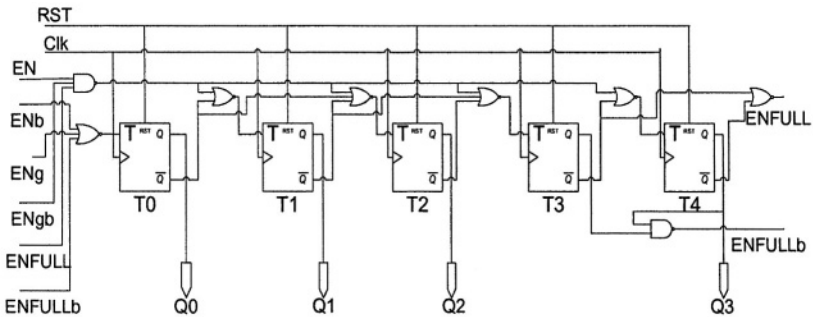
The basic principles for designing fault-secure binary counters have been presented in [3]. Although the proposed solution in [3] is satisfactory, it addresses a slow counter. A high-speed implementation was presented in [4] however it isn't a binary



**Fig. 1.** The structure of a 16-bit non safe Constant-Time By-Pass Counter, constructed by four segments. The first segment is the initial module (*Byp0*) while the rest modules are replicas of module (*Byp1*)



**Fig. 2.** The structure of the initial module (*Byp0*) used as the first segment of the Constant-Time By-Pass Counter



**Fig. 3.** The structure of the module (*Byp1*) used for the rest of the segments of the Constant-Time By-Pass Counter



counter, which doesn't enable binary read on-the-fly. Alternative implementations are based on multi-channeled structures [5]. This technique is based on the use of several replicas of the non-safe component and continuous monitoring of their outputs. A case of disagreement indicates erroneous functionality. In this case, area and power overhead are proportional to the number of counter's replicas. However this structure is considered as the one presenting the higher performance.

In this paper a fault-secure architecture of a constant-time binary counter is introduced. Fault-secureness is achieved by adopting a parity prediction scheme. In each counter state, the parity of the counter's output is predicted for the next state and in the next clock cycle it is compared to the generated output's parity. Synchronous operation of a fault-secure binary counter makes reading the counter's value difficult; reducing counting rate proportionally to counter's length. Various configurations of counter output length have been designed and compared to the non-safe in terms of area and performance. The experimental results show high performance, with a critical path almost constant for every counter variation, while area overhead is lower than that of counter based on a double-channeled approach.

This paper is organized as follows: In Section 2 the structure of the selected constant-time counter is briefly presented. In Section 3 the special circuit that predicts the next state is presented. In Section 4 the proposed fault-secure architecture is presented, while in Section 5 the experimental results are given.

## 2 Constant-Time By-Pass Binary Counter

In Fig. 1 the architecture of a 16-bit non safe constant-time counter is illustrated. The architecture relies upon an efficient counter segmentation aiming at achieving high-clock frequency independent of the counter's length [2]. Exploiting special properties of the binary system, prescaling techniques are adopted to take advantage of the reduced clock frequency presented by the higher order bits [1]. Thus, the counter is segmented in modules and a high-order module is "clocked" by a low frequency signal derived by a lower order module. Since the length of the first module is smaller than the counter's length constant-time operation is achieved.

The architecture of the first module (Byp0) is illustrated in Fig. 2 while in Fig. 3 the module repeated for the rest segments of the circuit are illustrated. As shown in these figures, critical path equals to the delay of a 3 input NAND gate plus the delay of a 3 input NOR gate.

The above mentioned constant delay throughout the presented counter is the main benefit of this counter. Another beneficial characteristic of the counter is its low design complexity. These two design characteristics are exploited to turn this Constant-Time counter into a fault-secure one.

## 3 Next State Circuit Predictor

In both modules similar structures are observed, where T flip-flop are placed in successive order and their outputs correspond to the outputs of the module. For each T flip-flop holds:

$$T_i = Q_1 \cdot Q_2 \cdots Q_{i-1} \text{ for } i > 2. \tag{1}$$

where  $T_i$  and  $Q_i$  correspond to the input and the output of the  $i^{\text{th}}$  flip-flop, respectively. The characteristic equation of a T flip-flop that determines the output of the next state is given by:

$$Q^{t+1} = T^t \oplus Q^t. \tag{2}$$

The parity of the output of the next state can be predicted as a function of the parities of the inputs to the flip-flops and their outputs as described in [3]:

$$PQ^{t+1} = PT^t \oplus PQ^t. \tag{3}$$

The parity predictor circuit which is derived from Eq. (3) is illustrated in Fig. 4. Considering Fig. 2 and 3, it can be observed that the counter’s output corresponds to the outputs of  $T_0$ ,  $T_1$ ,  $T_2$  and  $T_4$  flip-flops. The parity is computed for the inputs and outputs of these flip-flops, while the XOR function between the parity functions gives the parity of the outputs for the next state.

The outputs can be checked comparing the predicted parity delayed at one state with the current parity module’s outputs. We use the  $\bar{Q}$  output of the flip-flop as the predicted parity, so it can easily be used in a double rail checker.

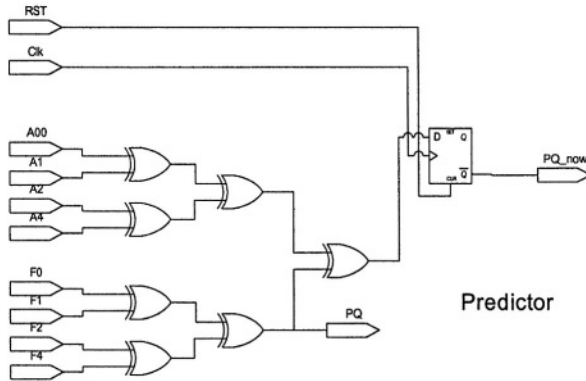


Fig. 4. Parity predictor circuit of the Constant-Time By-Pass Counter next state

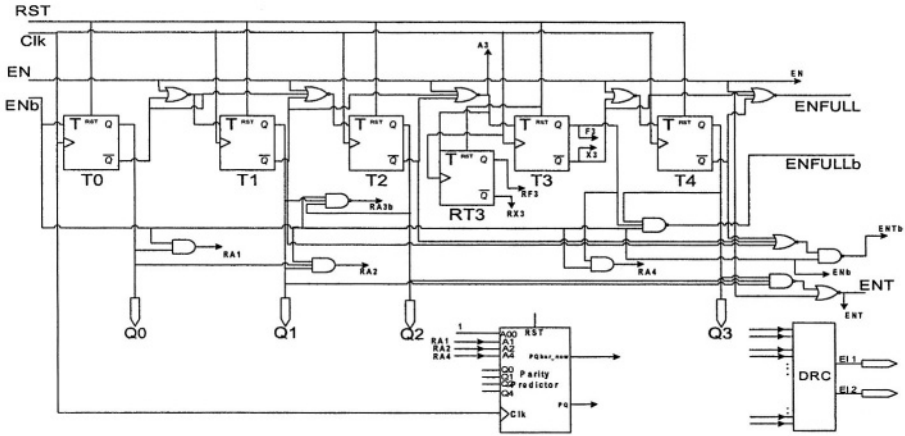
### 4 Fault-Secure By-Pass Counter

In this section, the Constant-Time By-Pass Counter, which is presented in [2], is modified to become fault-secure. Due to the low design complexity of the presented segments, this counter is appropriate to become fault-secure. In this section, every modification for each segment is presented in detail. Concluding, the structure of the fault-secure counter is given, using the modified segments. The benefits of the selection of a segmented counter are the locality of the error detection mechanisms and the constant delay at each segment, observed on the parity prediction mechanism. The

latter offer a novel implementation of a Constant-Time fault-secure binary counter, based on information redundancy.

### 4.1 Fault-Secureness of the Initial Segment

In Fig. 5 the first module of the counter is redesigned to achieve fault-secureness as shown. The previously described predictor is used to predict the parity of the next state.  $T_0$  corresponds to the  $\overline{EN}$  input signal of the module, which is ‘true’ when the module is counting. The outputs of the predictor are driven to a TRC (Two-Rail Checker) that checks signals using their actual value and their complementary one.



**Fig. 5.** The structure of the fault-secure initial module (*Byp0*) used as the first segment of the Constant-Time By-Pass Counter

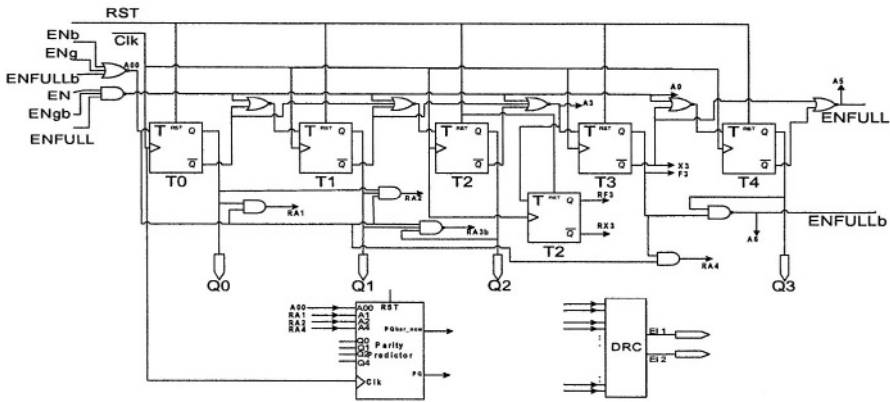
The NOR gates have not been duplicated because these gates end in only one flip-flop [3]. Therefore, a single fault at the output of the NOR gates will always propagate only in one output altering the output parity once. Thus, the error is detected. The signals driven to the predictor have been produced from the complementary logic to achieve dissociation from the outputs of the flip-flops. In the opposite case, a fault in the output of a NOR gate would propagate so at the T flip-flop as to the predictor. This results a double change of the predicted parity and therefore the error will not be detected.

The  $T_3$  flip-flop has been duplicated as it can't be checked by the predictor. Thus, the outputs of  $T_3$  with their complements are driven to the TRC. We follow the same approach also for the NOR gate that ends at the input of  $T_3$ . The complement of the NOR gate generated also with a double rail form. To avoid undetectable interconnection faults the input,  $EN$  has been driven to the TRC with its complement. As shown in the Fig. 5,  $EN$  signal ends to the TRC after driving the inputs of the rest gates, so if a stuck open fault takes place in the interconnection line, it will be detected by the TRC. In an opposite case, it would produce a multiple error and in the case of even multiplicity is impossible to be detected by parity codes.

The output signal  $ENT$ , which will result to the enable signal for the next module is also being checked by the dissociation from the complement signal.

### 4.2 Fault-Secureness of the Repetitive Segment

In Fig. 6 the second fault-secure module, which is repeated for the construction of a counter of variable width, is illustrated. The same technique described above for achieving fault-secureness is used. Parity predictor accepts the same inputs with the T flip-flops (created with a complementary logic) and the outputs of the module generating so the predicted parity as the present parity of the outputs.  $T_3$  flip-flop is duplicated to check for errors in its outputs, while the NOR gate attached in  $T_3$  input is being driven to the TRC with its complement. At the TRC we drive, eventually, the *ENFULL* output with its complement.



**Fig. 6.** The structure of the fault-secure module (*Bypl*) used for the rest of the segments of the Constant-Time By-Pass Counter

### 4.3 Fault-Secureness of the Constant-Time By-Pass Counter

By examining closely the top-level design of the counter we can ensure fault-secureness with the implementation proposed in Fig. 7. The signals *Error Indication 1* and 2 of each module are driven in an overall TRC to obtain the detection of any error in any module. That is true, for any fault occurs inside a module, the internal TRC will produce either (00) or (11) in the module’s outputs causing error detection by the overall TRC.

In the proposed architecture we duplicated the flip-flops before each module. In addition, in the TRC enter the positive output of D flip-flops with the negative output of redundant D flip-flops, for the correct state of the D flip-flops to be ensured. Here, we do not check the case of the negative output of D flip-flops with the positive output of redundant D flip-flops. If the positive and negative outputs have the same value, the error will be detected inside the module that checks *enable* and *enable* signals. Thus, only checking the state of the flip-flops is necessary to ensure fault-secureness.

The output of NOR gates is also being driven to TRC with its complement produced using double rail structures. To detect interconnection faults, we attach *EN* and *EN* signals at the TRC after driving them throughout the rest of the gates.

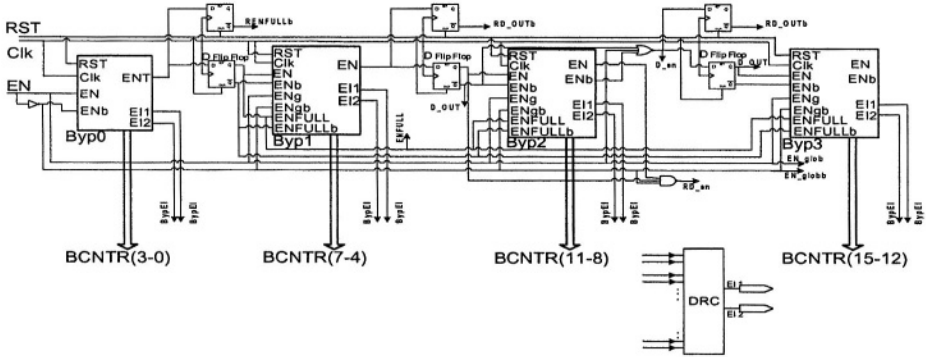


Fig. 7. The structure of the fault-secure Constant-Time By-Pass Counter

### 5 Experimental Results

The proposed fault-secure counter and the original counter were described in VHDL and synthesized using Leonardo Spectrum using AMS 0,6μ (3,3V) technology. Table 1 and Fig. 8 show the delay comparisons between the two counters. It is concluded that the Fault-secure counter is slower than the original one due to additional required circuit. To eliminate the dependence of the counter size we placed the maximum number of checks inside the modules to create a standard overhead.

Table 2 present area values of the original, the proposed and the multi-channel fault-secure counter. It is concluded that the area of the proposed fault-secure counter is larger than that of the original one but much lower than the multi-channeled fault-secure counter. As multi-channel fault-secure counter a counter with full duplication of the whole circuit and the TRC checkers is denoted.

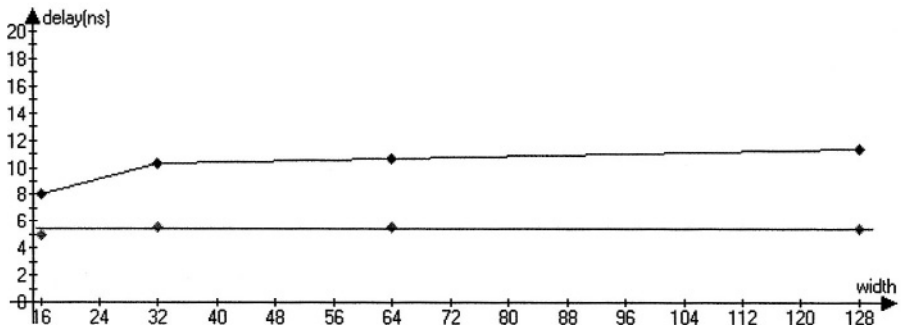


Fig. 8. The structure of the fault-secure module (Byp1) used for the rest of the segments of the Constant-Time By-Pass Counter

**Table 1.** Delay of the original and Fault-secure counter (ns) for various output bitwidths

counter \ length	16	32	64	128
<b>Original</b>	4.98	5.62	5.51	5.49
<b>Fault-secure</b>	8.04	10.35	10.62	11.32
<b>Penalty</b>	61.4%	84.1%	92.7%	106.1%

**Table 2.** Area of the original and Fault-secure counter (sq.mils) for various output bitwidths

counter \ length	16	32	64	128
<b>Original</b>	62	126	254	511
<b>Fault-secure</b>	160	320	639	1278
<b>Penalty</b>	158%	153.9%	151.5%	150%
<b>Multi-channel</b>	230	470	950	1912

## 6 Conclusions

A fault-secure scheme has been implemented for the Constant-Time By-Pass binary counter with the use of parity prediction principles. The counter keeps its high performance while area cost is acceptable compared to multi-channel techniques. The low design complexity of the original non safe Constant-Time By-Pass counter, presented in [2], is exploited. The segmentation of the counter is used to locally generate checking signals and minimize delay. The results showed a near constant behaviour, which was not observed in other implementations, as they were presented in [3] and [4]. Thus, the novel fault-secure counter, which was presented above, can satisfy high-performance requirements of safety-critical applications.

## References

1. Stan, M.R., Tenca, A.F., Ercegovac, M.D.: Long and Fast Up/Down Counters. *IEEE Transactions on Computers*. 47 (1998) 722–735
2. Kakarountas, A.P., Theodoridis, G., Papadomanolakis, K.S., Goutis, C.E.: A Novel High-Speed Counter with Counting Rate Independent of the Counter's Length. In: *Proceedings of IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. (2003)
3. Karaolis, E., Nikolaidis, S., Goutis, C.E.: Fault-secure Binary Counter Design. In: *Proceedings of IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. (1999)1659–1662
4. Papadomanolakis, K.S., Kakarountas, A.P., Sklavos, N., Goutis, C.E.: A Fast Johnson-Mobius Encoding Scheme for Fault-secure Binary Counters. In: *Proceedings of Design, Automation and Test in Europe (DATE)*. (2002)
5. von Neumann, J.: *Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components*. Princeton University Press (1956)

# Buffer Sizing for Crosstalk Induced Delay Uncertainty

Dimitrios Velenis<sup>1</sup> and Eby G. Friedman<sup>2</sup>

<sup>1</sup> Illinois Institute of Technology, Electrical and Computer Engineering Department,  
Chicago, IL 60616, USA,

velenis@ece.lit.edu

<sup>2</sup> University of Rochester, Electrical and Computer Engineering Department,  
Rochester, NY 14627, USA

friedman@ece.rochester.edu

**Abstract.** On-chip feature size scaling has aggravated the importance of crosstalk among interconnect lines. The effect of crosstalk on the delay of signals propagating along interconnect lines is investigated in this paper. It is shown that delay uncertainty is proportional to the amount of coupling among interconnect lines. A methodology to reduce delay uncertainty by increasing the size of interconnect buffers is presented. In addition, the effect of increasing buffer size on the power dissipation is discussed. A power efficiency metric is introduced that characterizes the trade off between the reduction in delay uncertainty and the increase in power dissipation.

## 1 Introduction

The rapid scaling of on-chip geometric dimensions supports the system-on-chip integration of multiple subsystems, greatly increasing the functionality of an integrated circuit. The continuous quest for higher circuit performance has pushed clock frequencies deep into the gigahertz frequencies range, reducing the period of the clock signal well below a nanosecond. These strict timing constraints require tight control on the delay of signals within a high speed synchronous integrated circuit. The deviation of a signal delay from a target value can cause incorrect data to be latched within a register, resulting in a system malfunctioning. These variations in signal delay are described as *delay uncertainty*. The sensitivity of a circuit to delay uncertainty has become an issue of fundamental importance to the design of high performance synchronous systems.

Significant research effort has therefore been focused on characterizing and reducing delay uncertainty [1, 2]. A primary research target is the statistical characterization [3, 4] of process parameter variations and delay uncertainty in order to specify the minimum timing constraints for synchronizing high speed circuits [5,6]. In addition, design methodologies for clock distribution networks [7] have been developed to reduce uncertainty in the clock signal delay [8,9,10].

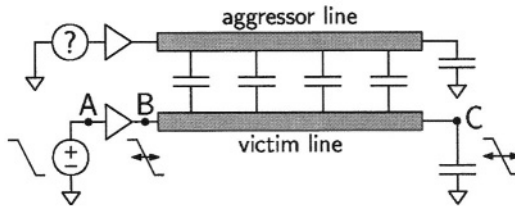
Another important effect that introduces delay uncertainty is crosstalk among interconnect lines [11, 12, 13]. Scaled on-chip feature size reduces the distance between adjacent interconnect lines. In addition, the thickness-to-width

wire aspect ratio has increased with each technology generation [14]. These effects result in a significant increase in coupling capacitance and crosstalk among interconnect lines [15,16,17,18]. The effect of interconnect crosstalk however, has been primarily described as variations in the voltage and current rather than uncertainty in the signal propagation delay.

The effects of interconnect crosstalk on the signal delay are investigated in this paper. It is shown that uncertainty in the signal delay is introduced due to variations in the effective capacitive load of an interconnect line. The effect of these variations on signal delay can be alleviated by increasing the size of a buffer driving a critical delay line, as described in Section 2. Increasing coupling among interconnect lines increases the delay uncertainty, as discussed in Section 3. It is demonstrated in this paper, that the delay uncertainty due to increased coupling can be reduced by increasing the size of the interconnect buffer. In Section 4, the effect of increased buffer size on power dissipation is discussed. A power efficiency metric is introduced to describe the trade off between delay uncertainty and power dissipation. Finally, some conclusions are presented in Section 5.

## 2 Effects of Buffer Size on Delay Uncertainty

Capacitive coupling among interconnect lines introduces uncertainty in the effective capacitive load of the buffers driving these lines. The variation in effective load creates uncertainty in the delay of a signal that propagates along a buffer and an interconnect line. An example of a pair of capacitively coupled interconnects is shown in Fig. 1. The propagation delay from point A to point C along the *victim line* shown in Fig. 1 is affected by the switching of the capacitively coupled *aggressor line*. The signal propagation delay on the victim line can be divided into two parts: the buffer delay from point A to point B, as shown in Fig. 1, and the interconnect delay between points B and C.



**Fig. 1.** Model for capacitive coupling between two interconnect lines

The uncertainty of the switching activity of the aggressor line shown in Fig. 1 introduces delay uncertainty in the overall signal propagation along the victim line. Possible signal transitions of the aggressor line during a signal switching on the victim line are:

- i) switch in phase (*i.e.*, the same direction) with the victim line.
- ii) switch out of phase (*i.e.*, the opposite direction) with the victim line.
- iii) no switching (*i.e.*, remain at a steady state, either high or low).



Any uncertainty in the signal transition of the aggressor line introduces uncertainty in the effective capacitive load of the buffer driving the victim line. A relationship describing the delay of an inverter driving an effective capacitive load  $C_L$  is presented in [19],

$$t_D = \left( \frac{1}{2} - \frac{1 - v_T}{1 + \alpha} \right) t_T + \frac{C_L V_{DD}}{2I_{D0}}, \quad (1)$$

where

$$v_T = \frac{V_{TH}}{V_{DD}}, \quad (2)$$

and  $t_D$  is the signal propagation delay,  $\alpha$  is the velocity saturation index [19],  $t_T$  is the transition time of the signal at the input of the inverter, and  $V_{DD}$  is the supply voltage.  $V_{TH}$  is the threshold voltage of the active transistor during a signal transition and  $I_{D0}$  is the drain current flowing through that transistor (defined at  $V_{GS} = V_{DS} = V_{DD}$ ).  $I_{D0}$  is often used as an index of the *drive strength* of a MOSFET transistor and depends upon the transistor width  $W$ .

The delay uncertainty of a signal propagating along a buffer due to variations in the effective capacitive load can be determined by differentiating (1) with respect to  $C_L$ ,

$$\partial t_D = \frac{V_{DD}}{2I_{D0}} \partial C_L. \quad (3)$$

As shown in (3), the variation in delay  $\partial t_D$  is proportional to the variation in the effective load of the buffer  $\partial C_L$ . In addition, it is shown in (3) that increasing the size of a buffer (*i.e.*, increasing  $I_{D0}$ ) reduces the effect of variations in the capacitive load on the signal propagation delay.

In order to evaluate the delay uncertainty of a signal on the victim line, a coupled pair of interconnect lines is simulated using spectre@simulator\* for any possible switching activity of the aggressor line. The interconnect structure being evaluated is illustrated in Fig. 2.

The uncertainty in the signal propagation delay along the buffer and the interconnect wire of a victim line is shown in Fig. 3 for different switching patterns of the aggressor line. In the example shown in Fig. 3, the signal at the input of the driver of the victim line switches from high-to-low. The size of the driving buffer for both the victim and the aggressor line is the same. The two lines are coupled along  $\frac{1}{4}$  of their total length. The uncertainty of the signal propagation delay along the buffer of the victim line is 23.8 picoseconds as shown in Fig. 3(a). In addition, the uncertainty of the signal delay along the interconnect of the victim line is 2.5 picoseconds, as illustrated in Fig. 3(b). The corresponding delay along the buffer and interconnect wire for an uncoupled line is also shown in Figs. 3(a) and 3(b) for comparison.

As described in (3), increasing the buffer size reduces the sensitivity of the buffer delay to variations in the load capacitance. This effect is demonstrated in Fig. 4(a), where the size of the aggressor line buffer equals one while the size

\* Spectre® is a registered trademark of Cadence Design Systems.

- Metal layer : m1
- Feature size  $\lambda = 0.09\mu m$ 
  - minimum line width  $W_{min} = 3\lambda = 0.27\mu m$
  - minimum line spacing  $S_{min} = 3\lambda = 0.27\mu m$
- Line length  $L = 400\mu m$
- Line resistivity  $R_L = 296\mu\Omega/\mu m$
- Line capacitance:

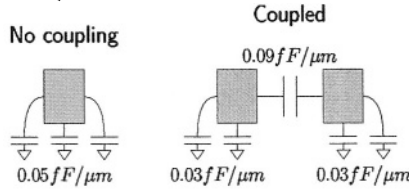
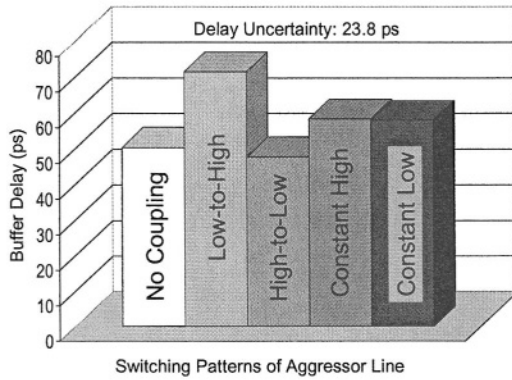
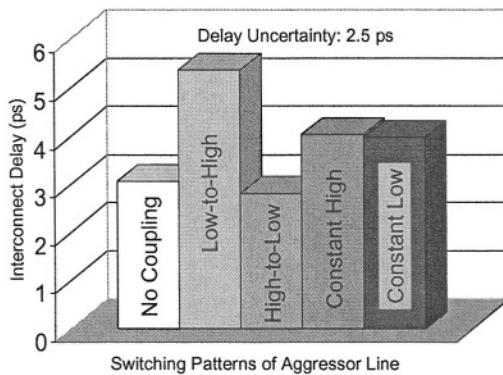


Fig. 2. Simulation setup of capacitively coupled interconnect lines

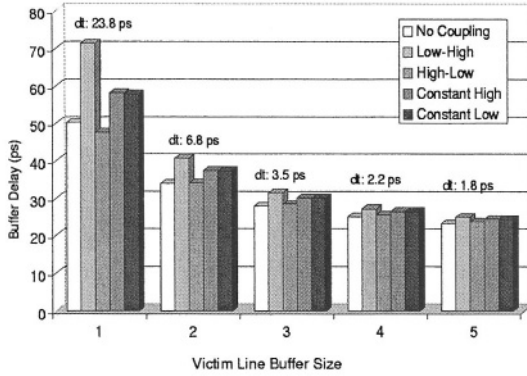


(a) Uncertainty in the signal delay along the interconnect buffer

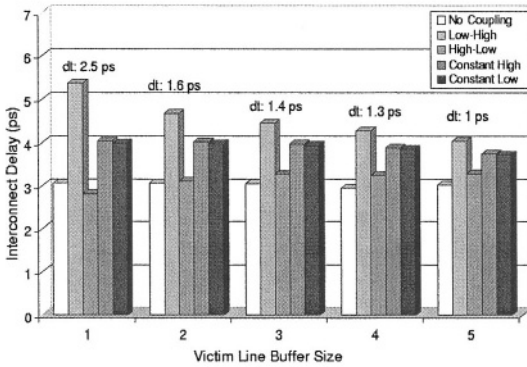


(b) Uncertainty in the signal delay along the interconnect wire

Fig. 3. Uncertainty of the signal delay propagation along the victim line due to different switching activities of the aggressor line



(a) Reduction in delay uncertainty of the buffer driving the victim line with increasing buffer size



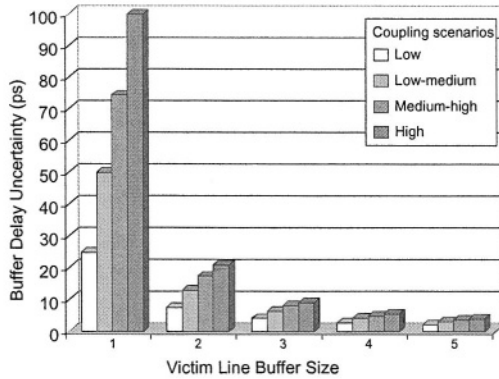
(b) Reduction in delay uncertainty of the interconnect line with increasing buffer size

**Fig. 4.** Reduction in delay uncertainty along the victim line with increased buffer size

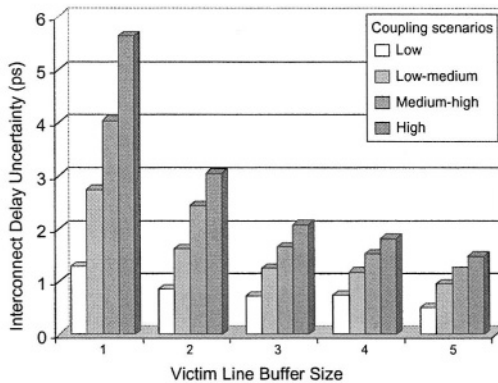
of the victim line buffer is increased by up to five times. In addition, as shown in Fig. 4(b), the increased current flowing through the driver buffer reduces the delay uncertainty of a signal propagating along the wire of the victim line (between points *B* and *C* as shown in Fig. 1), for the same increase in the size of the victim line buffer.

### 3 Effects of Coupling on Delay Uncertainty

As described in the previous section, the uncertainty in the signal delay along the interconnect lines depends upon the relative size of the buffers driving these lines. It is demonstrated that increasing the size of an interconnect buffer reduces the effects of variations in the effective load. However, the effective capacitive load



(a) Change in delay uncertainty of the driving buffer with increasing coupling and buffer size



(b) Change in delay uncertainty of the interconnect line with increasing coupling and buffer size

**Fig. 5.** Delay uncertainty increases proportionally with capacitive coupling between the lines

is increased with increasing coupling between the interconnect lines. Variations of the effective load due to uncertain switching activity of the aggressor line can therefore introduce a greater amount of delay uncertainty on the victim line.

The simulation setup illustrated in Fig. 2 is used to investigate this effect. Four different coupling scenarios are considered, depending upon the amount of coupling between the interconnect lines:

1. Low coupling: the lines are coupled along  $\frac{1}{4}$  of the total length.
2. Low-medium coupling: the lines are coupled along  $\frac{1}{2}$  of the total length.
3. Medium-high coupling: the lines are coupled along  $\frac{3}{4}$  of the total length.
4. High coupling: the lines are coupled along the entire wire length.

Investigating these coupling scenarios demonstrates that the effect of reducing the delay uncertainty by increasing the buffer size becomes more significant as the coupling between the aggressor and victim line increases. As shown in Fig. 5(a), the uncertainty of the signal propagation delay along the buffer driving the victim line increases proportionally with increasing capacitive coupling between the two lines. The delay uncertainty, however, is reduced exponentially with increasing buffer size. Similar trends in the delay uncertainty of the signal propagation delay along the interconnect line are illustrated in Fig. 5(b). The delay uncertainty increases proportionally with increased coupling between the lines and is reduced with increasing buffer size.

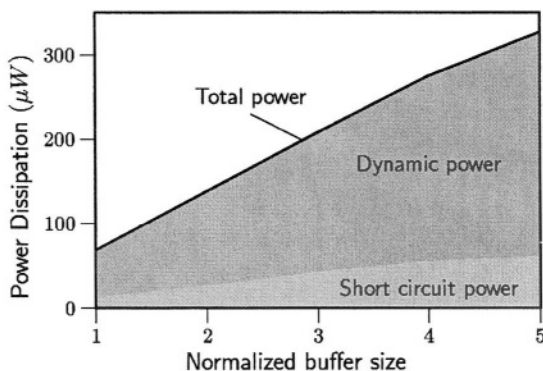
## 4 Power Dissipation Trade Offs

Increasing the size of an interconnect buffer reduces the delay uncertainty introduced due to crosstalk between interconnect lines. Increasing buffer size, however, also increases the buffer area and power dissipation. With increasing die area and the scaling of on-chip feature size, the requirement for increased buffer area is not of primary concern. The increase in power dissipation, however, is a significant effect that imposes practical constraints on the buffer size. In this section, design trade offs between power dissipation and delay uncertainty are discussed.

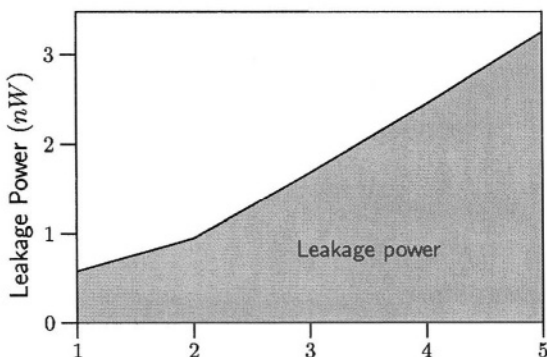
**Table 1.** Transistor size and power dissipation components for different buffer sizes.

Buffer Size	NMOS tran. size	PMOS tran. size	Dynamic power ( $\mu W$ )	Short circuit power ( $\mu W$ )	Leakage power ( $nW$ )	Total power ( $\mu W$ )
1	0.9 $\mu m$	2.34 $\mu m$	55.5	12.7	0.5	68.2
2	1.8 $\mu m$	4.68 $\mu m$	112.2	26.2	0.9	138.4
3	2.7 $\mu m$	7.02 $\mu m$	165.5	42.7	1.6	208.2
4	3.6 $\mu m$	9.36 $\mu m$	220.3	55.0	2.4	275.3
5	4.5 $\mu m$	11.7 $\mu m$	266.9	60.3	3.2	327.2

The effect of the buffer size on power dissipation is listed on Table 1. In the second and third columns of Table 1, the sizes of the NMOS and PMOS transistors are listed for different buffer sizes. The power dissipation for a signal transition cycle between high-to-low and low-to-high is listed in columns four and five. The dynamic power is dissipated while charging and discharging the transistor gates. The short-circuit power is dissipated due to the current that flows directly from  $V_{dd}$  to ground during a signal transition, when both the PMOS and NMOS transistors are on. The increase in short-circuit and dynamic power with buffer size is illustrated in Fig. 6(a). The leakage power listed in column six of Table 1 represents the power dissipated due to leakage current flowing through the buffer transistors while the buffer input is at a steady state voltage. The increase in leakage power with larger buffer size is illustrated in



(a) Short-circuit, dynamic, and total power dissipation with increasing buffer size

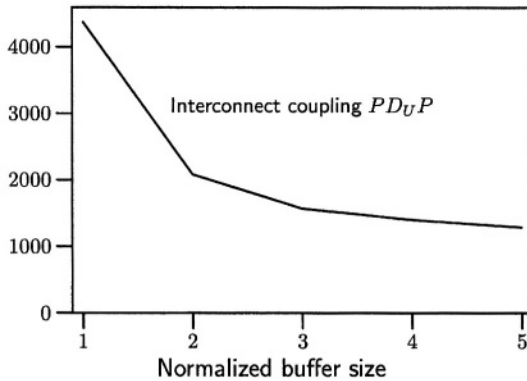


(b) Increase in leakage power with increasing buffer size

**Fig. 6.** Increase in power dissipation with buffer size

Fig. 6(b). The total power dissipation is listed in column seven of Table 1 and is illustrated in Fig. 6(a).

A qualitative trade off between the reduction in delay uncertainty and increased power dissipation with increasing buffer size can be described by the *Power-Delay-uncertainty-Product* ( $PDUP$ ). The  $PDUP$  is introduced to compare the rate of *decreasing* delay uncertainty with respect to the rate of *increasing* power dissipation, as the buffer size is increased. Decreasing  $PDUP$  with increasing buffer size indicates that the reduction in delay uncertainty is higher than the increase in power dissipation. Alternatively, an increase in  $PDUP$  with increasing buffer size demonstrates a faster increase in power dissipation than the decrease in delay uncertainty. The  $PDUP$  with increasing buffer size is illustrated in Fig. 7 for the delay uncertainty introduced due to capacitive coupling



**Fig. 7.** Power-Delay uncertainty Product ( $PD_{UP}$ ) for different sizes of victim line drivers

among interconnects. As shown in Fig. 7, increasing the buffer size is a power efficient way to reduce delay uncertainty due to interconnect coupling, since decreasing  $PD_{UP}$  indicates that the power dissipation increases slower than the delay uncertainty is reduced.

## 5 Conclusions

The uncertainty in the signal delay due to crosstalk among interconnect lines is investigated. In addition, the effect on the delay uncertainty of different switching patterns among capacitively coupled lines is evaluated. It is shown that delay uncertainty increases with coupling among the lines and decreases with increasing buffer size. The effect of increasing the buffer size on the power dissipation is also examined. A power efficiency metric, the Power-Delay-uncertainty-Product ( $PD_{UP}$ ), is introduced to quantify the trade off between delay uncertainty and power dissipation. It is shown that increasing buffer size is a power efficient method to reduce delay uncertainty due to capacitive coupling among interconnects.

## References

1. P. Zarkesh-Ha, T. Mule, and J. D. Meindl, "Characterization and Modeling of Clock Skew with Process Variations," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 441–444, May 1999.
2. S. R. Nassif, "Modeling and Analysis of Manufacturing Variations," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 223–228, May 2001.
3. M. Orshansky, J. C. Chen, and C. Hu, "Direct Sampling Methodology for Statistical Analysis of Scaled CMOS Technologies," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 12, No. 4, pp. 403–408, November 1999.

4. V. Mehrotra, S. L. Sam, D. Boning, A. Chandrakasan, R. Vallishayee, and S. Nassif, "A Methodology for Modeling the Effects of Systematic Within-Die Interconnect and Device Variation on Circuit Performance," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 172–175, June 2000.
5. S. Zanella, A. Nardi, A. Neviani, M. Quarantelli, S. Saxena, and C. Guardiani, "Analysis of the Impact of Process Variations on Clock Skew," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 13, No. 4, pp. 401–407, November 2000.
6. S. Sauter, D. Schmitt-Landsiedel, R. Thewes, and W. Weber, "Effect of Parameter Variations at Chip and Wafer Level on Clock Skews," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 13, No. 4, pp. 395–400, November 2000.
7. I. S. Kourtev and E. G. Friedman, *Timing Optimization Through Clock Skew Scheduling*, Kluwer Academic Publishers, Norwell Massachusetts, 2000.
8. J. L. Neves and E. G. Friedman, "Optimal Clock Skew Scheduling Tolerant to Process Variations," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 623–628, June 1996.
9. M. Nekili, C. Bois, and Y. Savaria, "Pipelined H-Trees for High-Speed Clocking of Large Integrated Systems in Presence of Process Variations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 5, No. 2, pp. 161–174, June 1997.
10. D. Velenis, M. C. Papaefthymiou, and E. G. Friedman, "Reduced Delay Uncertainty in High Performance Clock Distribution Networks," *Proceedings of the IEEE Design Automation and Test in Europe Conference*, pp. 68–73, March 2003.
11. K. T. Tang and E. G. Friedman, "Interconnect Coupling Noise in CMOS VLSI Circuits," *Proceedings of the ACM International Symposium on Physical Design*, pp. 48–53, April 1999.
12. A. Vittal, L. H. Chen, M. Marek-Sadowska, K.-P. Wang, and S. Yang, "Crosstalk in VLSI Interconnections," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 12, pp. 1817–1824, December 1999.
13. S. S. Sapatnekar, "A Timing Model Incorporating the Effect of Crosstalk on Delay and its Application to Optimal Channel Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 5, pp. 550–559, May 2000.
14. X. Huang, Y. Cao, D. Sylvester, S. Lin, T.-J. King, and C. Hu, "RLC Signal Integrity Analysis of High Global Interconnects," *Proceedings of the IEEE International Electron Devices Meeting*, pp. 731–734, December 2000.
15. K. T. Tang and E. G. Friedman, "Delay and Noise Estimation of CMOS Logic Gates Driving Coupled Resistive-Capacitive Interconnections," *Integration, the VLSI Journal*, Vol. 29, No. 2, pp. 131–165, September 2000.
16. J. Cong, D. Z. Pan, and P. V. Srinivas, "Improved Crosstalk Modeling for Noise Constrained Interconnect Optimization," *Proceedings of the ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pp. 14–20, December 2000.
17. M. Kuhlmann and S. S. Sapatnekar, "Exact and Efficient Crosstalk Estimation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 20, No. 7, pp. 858–866, July 2001.
18. T. Sakurai, "Closed-Form Expressions for Interconnection Delay, Coupling and Crosstalk in VLSI's," *IEEE Transactions on Electron Devices*, Vol. 40, No. 1, pp. 118–124, January 1993.
19. T. Sakurai and A. R. Newton, "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas," *IEEE Journal of Solid State Circuits*, Vol. 25, No. 2, pp. 584–593, April 1990.



# Optimal Logarithmic Representation in Terms of SNR Behavior

P. Vouzis<sup>1</sup> and V. Paliouras<sup>2</sup>

<sup>1</sup> Computer Engineering and Informatics Department  
University of Patras, Greece  
vouzis@ceid.upatras.gr

<sup>2</sup> Electrical and Computer Engineering Department  
University of Patras, Greece  
paliouras@ee.upatras.gr

**Abstract.** This paper investigates the Signal-to-Noise Ratio (SNR) performance of the Logarithmic Number System (LNS) representation against the SNR performance of the fixed-point representation. Analytic formulas are presented for the evaluation and the comparison of the two aforementioned representations, and the superiority of the LNS representation is demonstrated. It is shown that the base  $b$  of the logarithmic representation has a major impact onto the SNR performance, the SNR dependance on the variations of the standard deviation of the analog signal, and the memory requirements for logarithmic addition and subtraction. In addition, step-by-step procedures are introduced to compute the base  $b$  that optimizes all these performance measures.

## 1 Introduction

The LNS has been studied for several years as an alternative to the linear fixed-point number system [1]. The main advantage of the LNS is the simplification of the arithmetic operations of multiplication, division, roots and powers. Although high-precision LNS-based implementations have been proposed [2], the uses of the LNS are primarily focused on DSP applications, that require relatively short word-lengths [3] due to the exponential increase of the memory requirements for logarithmic addition and subtraction as the word-length increases.

The parameters of the LNS representation, i.e., the base  $b$  and the number of integral  $k$  and fractional  $l$  bits, can be chosen by the designer to meet certain signal quality specifications with minimal cost. In [4] Paliouras and Stouraitis introduce analytic formulas for the calculation of  $k$  and  $l$ , given the base  $b$ , so that the logarithmic representation demonstrates a dynamic range and an average representational error at least as good as an  $n$ -bit linear fixed-point system. In [5] it is shown that departing from the usual choice of base  $b = 2$  an optimal base can be selected to minimize the number of words needed to be stored for logarithmic addition and subtraction, while maintaining the equivalence to an  $n$ -bit linear number system as defined in [4].

In DSP applications, an important measure of signal quality is the SNR that a certain representation demonstrates. Kwa *et al.* [6] present a stochastic technique to analyze the SNR characteristics for a logarithmic data encoder when excited by either Laplacian distributed or sinusoidal inputs. In [6], though, the effect of overflow at the output of the A/D converter is not considered. Jayant and Noll prove that the A/D overflow is of major importance both in the logarithmic and the linear domain [7].

This paper investigates the impact of the logarithmic base  $b$  onto i) the SNR of an LNS representation ( $\text{SNR}_{\log}$ ), ii) the memory requirements for logarithmic addition and subtraction, and iii) the dependance on the variations of the standard deviation of the analog signal. In particular, analytic formulas are introduced that encompass all effects (“dead-zone” and overflow) that contribute to the total error budget of the representation. It is shown that when the LNS and the linear fixed-point system are compared in terms of the SNR, the proper choice of  $b$  can result in considerable savings in memory requirements. For instance, for the case of  $k + l = 8$ , the memory reduction is 33% in comparison to the base-choice according to [5]. The optimal base values are different than those obtained when the optimization criteria are the average representational error and dynamic range equivalence [5].

## 2 SNR of Linear and Logarithmic Representation

If the sampling period of an Analog-to-Digital (A/D) converter is  $T$ , then on each time step  $t_m = mT$ ,  $m = 0, 1, 2, \dots$ , an output is produced which is a mapping of the analog input  $x(t)$ , sampled at time  $t_m$ , to a binary word of finite precision  $x(m)$ , thus resulting in a quantization error equal to

$$e(m, t) = x(m) - x(t) \text{ ,} \tag{1}$$

at the interval  $t \in [mT, (m + 1)T)$ . In [7] it is shown that, if the Probability Density Function (PDF)  $P(x(t))$  of the amplitude of  $x(t)$  exhibits zero mean, the variance  $\sigma_e^2$ , or the power, of the quantization noise, at the output of an A/D converter, for the region that can be covered by the dynamic range  $D = [x_0, x_{N-1}]$  of a binary representation, is

$$\sigma_e^2 = \sum_{i=0}^{N-2} \int_{x_i}^{x_{i+1}} [x(m) - x(t)]^2 P(x(t)) dx(t) \text{ ,} \tag{2}$$

where  $N$  is the number of the distinct quantization steps and  $[x_i, x_{i+1}]$  is the interval of any arbitrary quantization step. If the input  $x(t)$  has a nonzero mean  $\mu$ , it can be removed by subtracting  $\mu$  from the input and adding it back after quantization. Alternatively, this procedure can be interpreted as a shift of the center of the quantizer to the mean value. A zero mean is henceforth assumed without loss of generality [7].

Analytic computation of (2) proves to be very awkward, because the number of quantization steps,  $N$ , depends exponentially on the number of bits,  $n$ ; i.e.,  $N = 2^n$ . Instead, Jayant and Noll [7] present the approximation

$$\sigma_e^2 = \frac{q^2}{12} \int_{x_{\min}}^{x_{\max}} \frac{P(x(t))}{\hat{C}(x(t))^2} dx(t) , \tag{3}$$

where  $q$  is the width of the quantization step,  $D = [x_{\min}, x_{\max}]$  is the dynamic range of the representation, and  $C(x(t))$  is the companding characteristic used in the quantization process. Equation (3) is an approximation, valid for  $n \geq 6$ , and as wordlength of interest exceed six, is henceforward adopted.

### 2.1 The Fixed-Point Representation

The application of (3) to compute the noise variance for a linear fixed-point system is straightforward, since the companding characteristic is  $C(x) = x$  and thus  $\hat{C}(x) = 1$ . For an  $n$ -bit two's-complement representation with  $k$  integral and  $l$  fractional bits, the dynamic range is  $D_{\text{fxp}} = [-2^{k-1}, 2^{k-1} - 2^{-l}]$ , the quantization step is  $q = 2^{-l}$ , and hence

$$\sigma_{D_{\text{fxp}}}^2(k, l) = \frac{2^{-2l}}{12} \int_{-2^{k-1}}^{2^{k-1} - 2^{-l}} P(x(t)) dx(t) . \tag{4}$$

However, the effect of overflow, which imposes signal peak-clipping, has also to be taken into account. Peak-clipping arises due to the limited maximum input amplitude that can be handled by an A/D converter; any input signal larger than the input amplitude limit saturates the system. Input values outside the interval  $D_{\text{fxp}}$  are rounded either to the minimum or to the maximum representable number,  $-2^{k-1}$  and  $2^{k-1} - 2^{-l}$  respectively, depending on the sign of the input values. Hence, the error-power due to peak-clipping is computed as

$$\begin{aligned} \sigma_{P_{C_{\text{fxp}}}}^2(k, l) = & \int_{-\infty}^{-2^{k-1}} [x(t) - (-2^{k-1})]^2 P(x(t)) dx(t) + \\ & \int_{2^{k-1} - 2^{-l}}^{+\infty} [x(t) - (2^{k-1} - 2^{-l})]^2 P(x(t)) dx(t) . \end{aligned} \tag{5}$$

Concluding, the total error-power,  $\sigma_{e, \text{fxp}}^2$ , due to conversion from a real value to a binary linear two's-complement representation is

$$\sigma_{e, \text{fxp}}^2(k, l) = \sigma_{D_{\text{fxp}}}^2(k, l) + \sigma_{P_{C_{\text{fxp}}}}^2(k, l) . \tag{6}$$

### 2.2 The Logarithmic Representation

In the case of the logarithmic representation, the computation of the noise variance is more involved. For an input value  $X$  to be mapped to the LNS, the triplet  $(z, s, x)$  is defined as follows

$$X \xrightarrow{\text{LNS}} (z, s, x = \log_b |X|) , \tag{7}$$

where  $z$  is a single-bit flag the assertion of which denotes that  $X$  is zero,  $s$  is the sign of  $X$ , and  $b$  is the base of the logarithmic representation. The output

of a base- $b$  LNS A/D converter is a two's complement binary number with  $k$  integral and  $l$  fractional bits representing the amplitude of the absolute value of the input  $X$ , concatenated with the zero and sign bits. Thus the conversion from LNS to the fixed-point system is

$$X = (1 - z)(-1)^s b^x . \tag{8}$$

In the following, the quantization process performed by the logarithmic A/D is described. The process consists of two steps: first the logarithm of the input, sampled at  $t = mT$ , is calculated, and second the quantization is performed on the logarithmic value. Equation (8) denotes that for every arbitrary logarithmic value  $(z, 0, x)$ , the value  $(z, 1, x)$  exists, thus the logarithmic representation is symmetric around zero. If also, it is assumed that  $P(x(t))$  is an even function, the error power contribution of the positive and the negative input values is equal. Thus, it is sufficient to calculate the error power only for positive inputs and double the result to compensate for the negative ones. For input values spanning the positive dynamic range of a  $(k, l, b)$  two's-complement logarithmic representation with  $b > 1$ , i.e.,  $D_{\log} = [b^{-2^{k-1}}, b^{2^{k-1}-2^{-l}}]$ , equation (3) can be written as

$$\sigma_{D_{\log}}^2(b, k, l) = \frac{2^{-2l}(\ln b)^2}{12} \int_{b^{-2^{k-1}}}^{b^{2^{k-1}-2^{-l}}} x^2(t)P(x(t)) dx(t) , \tag{9}$$

since  $\dot{C}(x) = (\log_b x)' = 1/(x \ln b)$ .

However, since  $\lim_{x \rightarrow 0} \log_b |x| = -\infty$  and the smallest representable number is  $-2^{k-1}$ , assuming  $k$  integral bits, there is a so called "dead zone" around zero for  $X \in I_{dz} = [0, b^{-2^{k-1}})$ . The rounding policy assumed for the A/D converter for the interval  $I_{dz}$  is to round down to 0 input values falling into  $[0, b^{-2^{k-1}}/2)$  by asserting the zero flag in (7), and to round up to  $b^{2^{k-1}}$  input values falling into  $[b^{-2^{k-1}}/2, b^{-2^{k-1}}]$ , resulting in an error power given by

$$\begin{aligned} \sigma_{D_{Z_{\log}}}^2(b, k) = & \int_0^{b^{-2^{k-1}}/2} x^2(t)P(x(t)) dx(t) + \\ & \int_{b^{-2^{k-1}}/2}^{b^{-2^{k-1}}} [x(t) - b^{-2^{k-1}}]^2 P(x(t)) dx(t) . \end{aligned} \tag{10}$$

Additionally, inputs  $X \in (b^{2^{k-1}-2^{-l}}, +\infty)$  saturate the logarithmic A/D converter and are rounded to  $b^{2^{k-1}-2^{-l}}$ , i.e., the largest representable number. Peak-clipping error-power contribution is thus

$$\sigma_{P_{C_{\log}}}^2(b, k, l) = \int_{b^{2^{k-1}-2^{-l}}}^{+\infty} [x(t) - b^{2^{k-1}-2^{-l}}]^2 P(x(t)) dx(t) . \tag{11}$$

Summing up (9)–(11) the total error-power budget, of a base- $b$  binary logarithmic representation, can be calculated as follows:

$$\sigma_{e, \log}^2(b, k, l) = 2(\sigma_{D_{\log}}^2(b, k, l) + \sigma_{D_{Z_{\log}}}^2(b, k) + \sigma_{P_{C_{\log}}}^2(b, k, l)) . \tag{12}$$

Finally, given the variance  $\sigma_x^2$  of the input signal  $x(t)$ , the SNR of any finite representation (linear or logarithmic) is equal to

$$\text{SNR} = 10 \log_{10} \frac{\sigma_x^2}{\sigma_e^2} . \quad (13)$$

Equations (4)–(13) can now be used to evaluate and compare the fixed-point and logarithmic representations as far as the SNR is used as a signal quality measure. Additionally, the exploration of the design space of the LNS representation can be conducted in terms of the SNR behavior.

### 3 Comparison of Linear and Logarithmic Representation in Terms of Maximum Achievable SNR

The distribution  $P(x(t))$  used to evaluate the SNR either in the linear or the logarithmic domain, depends on the nature of the A/D converter's input. Throughout this paper the normal or Gaussian distribution, encountered in many natural processes, and given by

$$P_N(x(t), \mu, \sigma_x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x(t)-\mu)^2}{2\sigma_x^2}} \quad (14)$$

is adopted, where  $\sigma_x$  and  $\mu$  denote the standard deviation and the expected value respectively. From this point forward  $\mu = 0$  is assumed.

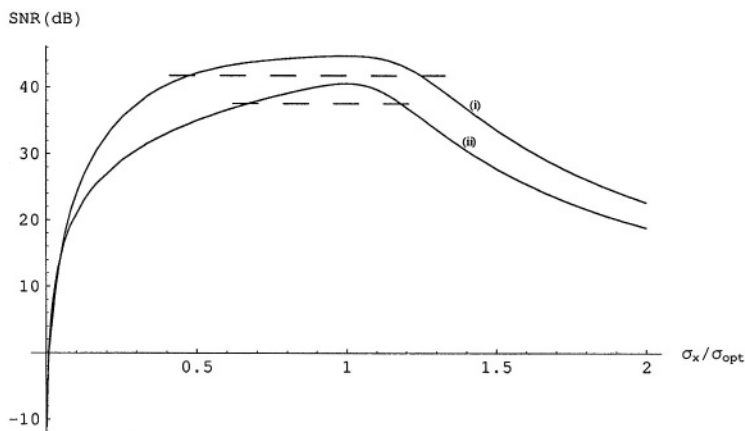
Jayant and Noll in [7], present a comparison between the fixed-point and the logarithmic representations following the  $\mu$ -Law and A-Law, regarding the achievable SNR in relation to the analog input's standard deviation and its variations. It is shown that for any  $(k, l)$  representation exists an optimal value of  $\sigma_x$ ,  $\sigma_{\text{opt}}$ , for which the maximum SNR is achieved, and any variation from this value results in the degradation of the SNR. The comparison in [7] exhibits the superiority of the logarithmic representations against the linear one in both the maximum achievable SNR, and the dependance of the SNR on the variations of  $\sigma_x$ .

However, when the conversion to the logarithmic domain is conducted in order to perform arithmetic manipulations, equation (7) has to be followed. In this case, the achievable  $\text{SNR}_{\log}$  is subject not only to  $\sigma_x$ , but also to the value of the base  $b$  of the representation. The conducted numerical exploration of the design space for the values of  $b$  proves that there exists an optimal base,  $b_{\text{opt}}$ , for every  $(k, l)$  two's complement logarithmic representation, which when adopted by an A/D converter results in the maximum achievable  $\text{SNR}_{\log}$ ,  $\text{SNR}_{\log, \text{max}}$ .

The study of cases of  $k+l$ ,  $8 \leq k+l \leq 16$ , proves that the base- $b_{\text{opt}}$  logarithmic representation of an analog signal exhibiting  $\sigma_{\log, \text{opt}}$ , outperforms the fixed-point representation of a signal exhibiting  $\sigma_{\text{fp}, \text{opt}}$ , in terms of maximum achievable SNR for equal word length  $n$ . Numerical analysis shows that for a given  $k+l$ ,  $\text{SNR}_{\log, \text{max}}$  remains constant for any combination of  $(k, l)$ ; thus Table 1 presents the comparison between the logarithmic and the linear representation for the cases of  $k = 1$  and  $l = n - 1$ . The use of a logarithmic arithmetic unit with

**Table 1.** Maximum SNR comparison between the logarithmic and the fixed-point representation for the case of  $k = 1$  and  $l = n - 1$

$n$	$\text{SNR}_{\text{fxp,max}}(\text{dB})$	$\sigma_{\text{fxp,opt}}$	$\text{SNR}_{\text{log,max}}(\text{dB})$	$\sigma_{\text{log,opt}}$	$b_{\text{opt}}$
8	40.57	0.254	44.77	2.354	10.799
9	46.03	0.237	50.01	2.847	13.732
10	51.55	0.222	55.32	3.452	17.465
11	57.11	0.210	60.68	4.195	22.205
12	62.71	0.199	66.09	5.109	28.224
13	68.35	0.190	71.54	6.242	35.867
14	74.00	0.182	77.03	7.633	45.530
15	79.70	0.175	82.55	9.360	57.811
16	85.40	0.168	88.10	11.51	73.443



**Fig. 1.** The SNR behavior of the logarithmic (i) and the linear (ii) representations, for the case of  $k + l = 8$ , as a function of the variation of  $\sigma_x$ . The dashed lines indicate the  $-3\text{dB}$  levels

$n > 16$  would have excessive memory requirements for the implementation of the logarithmic addition and subtraction, hence only the cases of  $n \leq 16$  are studied.

A logarithmic quantizer, except for the higher  $\text{SNR}_{\text{log,max}}$ , decreases the dependence of  $\text{SNR}_{\text{log}}$  on the variations of  $\sigma_x$ , compared to a linear A/D converter. In [7] the superiority of the  $\mu$ -Law and A-Law companding functions is presented concerning the aforementioned dependence property. The use of (7) does not eliminate this property. Figure 1 depicts the dependence of the SNR on the variations of  $\sigma_x$  of the input for the case of  $n = 8$ , and for the logarithmic quantizer using the optimal base  $b_{\text{opt}}$ . It is shown that departing from the optimal standard deviation, the SNR degrades both in the LNS and the linear representation. The interval, though, where the SNR remains above  $-3\text{ dB}$  is obviously longer in the logarithmic than in the linear case,  $[0.475, 1.249]$  and  $[0.671, 1.182]$  respectively. Moreover, the intervals where  $\text{SNR}_{\text{log}} \geq \text{SNR}_{\text{fxp,max}}$ ,

and where the LNS remains above the  $-3$  dB level of the linear representation, despite the variations of  $\sigma_x$ , are  $[0.409, 1.289]$  and  $[0.302, 1.379]$  respectively.

#### 4 Optimal Logarithmic Base- $b$ Computation Based on SNR Performance

The simplification of the operations of multiplication, division, roots, and powers, in the LNS, is counterbalanced by the increased complexity of logarithmic addition and subtraction. If  $x = \log_b|X|$ ,  $y = \log_b|Y|$  and the logarithmic addition and subtraction of  $X$ ,  $Y$  are denoted by  $z_a$  and  $z_s$  respectively, then

$$z_a = \max\{x, y\} + f_a(d) \quad (15)$$

$$z_s = \max\{x, y\} + f_s(d) \quad (16)$$

where  $f_a(d) = \log_b(1+b^{-d})$ ,  $f_s(d) = \log_b(1-b^{-d})$  and  $d = \max\{x, y\} - \min\{x, y\}$ . Several sophisticated methods, utilizing numerical approximations, have been proposed for the evaluation of  $f_a(d)$  and  $f_s(d)$  [8]. For small word lengths, though, e.g. less than 16 bits, functions  $f_a(d)$  and  $f_s(d)$  can be evaluated by means of simple look-up tables (LUTs).

For a  $(k, l, b)$  two's complement logarithmic representation, the effective-zero value of  $f_a(d)$ ,  $e_a = -\log_b(b^{2^{-l}} - 1)$ , result in  $f_a \leq 2^{-l}$  and thus  $f_a$  is rounded to zero [9]. Consequently, the number of non-zero entries needed to be stored in a LUT to perform logarithmic addition is

$$w_{\text{add}}(b, l) = \left\lceil \frac{e_a - 2^{-l}}{2^{-l}} \right\rceil \quad (17)$$

Similarly, for logarithmic subtraction effective zero is equal to  $e_s = -\log_b(1 - b^{-2^{-l}})$  and the words needed, after algebraic manipulation, are

$$w_{\text{sub}}(b, l) = w_{\text{add}}(b, l) + 1 \quad (18)$$

In [5] a formula is presented for the calculation of the optimal base  $b$  that minimizes the number of words needed to be stored for logarithmic addition and subtraction, regarding a  $(k, l, b)$  LNS representation equivalent to an  $n$ -bit linear one, according to the criteria of the average representational error and the dynamic range. However, when the measure of equivalence between a fixed-point and a logarithmic representation is the SNR, the optimal base  $b$  is different.

Table 1 shows that  $\text{SNR}_{\log, \max} > \text{SNR}_{\text{fxp}, \max}$ , for equal word lengths. Hence, departing from the value of  $b_{\text{opt}}$ , the  $\text{SNR}_{\log}$  degrades and the solution of

$$\text{SNR}_{\log}(b, k, l, \sigma_{\log, \text{opt}}) = \text{SNR}_{\text{fxp}, \max}(k, l) \quad (19)$$

gives the  $b_{\text{eq}}$  sought for SNR equivalence between the LNS and the linear number system representations. Equation (19) has two solutions, denoted  $b_1$  and  $b_2$ ,  $b_1 < b_2$ . The choice of  $b_1$  or  $b_2$  has an impact on the values of  $w_{\text{add}}(b, l)$  and

**Table 2.** When the SNR is the equivalence criteria between the logarithmic and the fixed-point representation considerable memory saving can be achieved

$(k, l)$	Proposed			According to [5]			Memory Savings
	$b_{eq}$	$w_{add}$	$SNR_{log}(dB)$	$b$	$w_{add}$	$SNR_{log}(dB)$	
(1,7)	63.5	105	40.57	20.8	157	43.24	33.1%
(1,8)	83.8	234	46.03	29.7	325	48.31	28.0%
(1,9)	109.0	511	51.55	42.3	672	53.49	24.0%
(1,10)	140.8	1104	57.11	60.0	1380	58.72	20.0%
(1,11)	179.9	2357	62.71	85.1	2826	64.05	16.6%
(1,12)	227.1	5002	68.35	120.5	5770	70.39	13.3%
(1,13)	287.6	10529	74.00	170.6	11753	74.84	10.4%
(1,14)	355.9	22125	79.70	241.4	23894	82.10	7.4%
(1,15)	444.9	46155	85.40	341.5	48497	84.88	4.8%

$w_{sub}(b, l)$  according to (17) and (18), respectively. Since,  $\partial w_{add}(b, l)/\partial b < 0$  and  $\partial w_{sub}(b, l)/\partial b < 0$ , the larger the value of  $b$  the larger the memory savings are. Consequently, the adoption of the larger  $b_2$  root of (19) results in memory savings in the LUTs used in the logarithmic addition and subtraction. Table 2 presents the memory requirements when  $b_{eq}$  is adopted, together with the saving compared to the base choice according to [5].

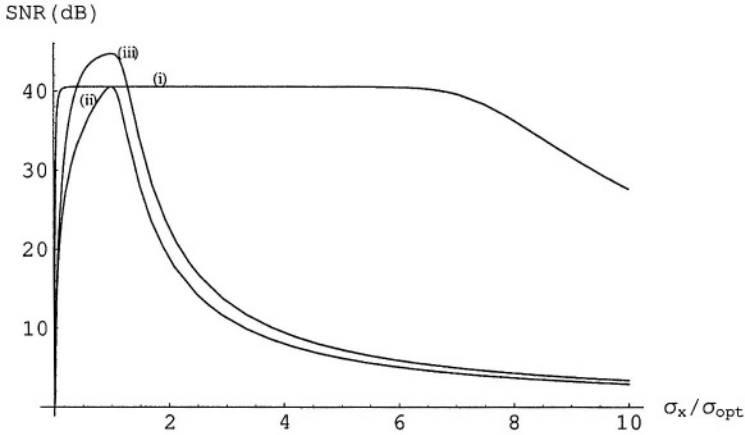
What is more, by selecting root  $b_2$  of (19) as the base  $b$  of the logarithmic A/D converter, apart from the aforementioned memory requirements reduction, the dependence of  $SNR_{log}$  on the variations of  $\sigma_x$  also decreases. Figure 2 illustrates this property for the case of  $k + l = 8$  and  $b = b_{eq}$ . Standard deviation  $\sigma_x$  of the input spans the interval [0.7,4.19] and  $SNR_{log}$  still remains above  $SNR_{fxp,max}$ , while the standard deviation of the fixed-point representation must not have the slightest variation from  $\sigma_{fxp,opt}$  to achieve the same SNR performance. In situations such as speed coding, the exact value of the input variance is not known in advance; and in addition, it tends to change with time. In such situations, a signal-to-quantization noise ratio that is constant over a broad range of input variances can be obtained by using a logarithmic companding law [7].

The results presented in Table 1 and Fig. 2 lead to the following step-by-step procedure, which can be used to calculate  $b_{eq}$  of a  $(k, l)$  logarithmic representation in order to be equivalent to  $(k, l)$  fixed-point one.

**Optimization Problem 1** *To find the value of  $b$  that must be used by a  $(k, l)$  two’s-complement logarithmic A/D converter in order its  $SNR_{log}$  performance to be equal to the SNR performance of a  $(k, l)$  two’s-complement linear A/D converter,  $SNR_{fxp,max}(k, l)$ , is sought then:*

1. From Table 1 determine the  $\sigma_{log,opt}$  for the given  $(k, l)$ .
2. Solve equation  $SNR_{log}(b, k, l, \sigma_{log,opt}) = SNR_{fxp,max}(k, l)$  with regard to  $b$ .
3. From the two solution produced in step 2 select the larger one to have smaller memory requirements for logarithmic addition and subtraction and weaker dependence on the variations of the standard deviation of the input signal.





**Fig. 2.** The adoption of  $b_{eq}$  results in an  $SNR_{log}$  (i) at least as good as  $SNR_{fxp}$  (ii), and also to considerable weaker dependence on the variations of the standard deviation of the input signal  $\sigma_x$ . Curve (iii) depicts  $SNR_{log}$  when  $b = b_{opt}$

When the SNR equivalence between an LNS and a linear representation is not the objective, but instead a particular  $SNR_{log}$  performance is required,  $SNR_{log,req}$ , the choice of base  $b$  is of major importance.

**Optimization Problem 2** *To achieve a particular SNR,  $SNR_{log,req}$ , using a two's-complement logarithmic A/D converter then:*

1. From Table 1 determine the smallest word length that an LNS representation must have in order  $SNR_{log,max} \geq SNR_{log,req}$  and the corresponding  $\sigma_{log,opt}$ .
2. Solve equation  $SNR(b, k, l, \sigma_{log,opt}) = SNR_{log,req}$  with regard to  $b$ .
3. From the solutions of  $b$  in step 2 chose the biggest one, to be used by the A/D converter, in order to have the smallest memory requirements for logarithmic addition and subtraction and weaker dependence on the variations of the standard deviation of the input signal.

The maximum achievable SNR of a logarithmic representation, the number of words required for addition and subtraction, and the dependence of the  $SNR_{log}$  on the variations of  $\sigma_x$ , are all subject to the value of the base  $b$  chosen. The optimization problems presented give a step-by-step methodology that can be followed in order to determine the proper value of  $b$ , that results in a logarithmic representation consistent with the requirements that may be posed in an arithmetic unit that adopts the LNS representation.

## 5 Conclusions

It has been shown that the choice of the base- $b$ , of a  $(k, l, b)$  two's complement logarithmic representation, is of major importance for its SNR performance, the

number of words needed for addition and subtraction, and the dependence on the variations of the standard deviation of the input. Analytic formulas (4)–(13) have been presented that can be used for the numerical calculation of the SNR that a particular representation exhibits, and were employed for the solutions of the optimization problems 1 and 2. The outcomes of this paper make the LNS more attractive for DSP applications since the reduced memory requirements presented have a direct impact on the area of the hardware required for LNS arithmetic and consequently on the power consumption of the circuit. However, if the signal quality is the criterion of the design choices, it is presented the methodology to exploit all the SNR potentials of the logarithmic representation.

## References

1. E. Swartzlander and A. Alexopoulos, "The sign/logarithm number system," *IEEE Transactions on Computers*, vol. 24, pp. 1238–1242, Dec. 1975.
2. M. G. Arnold, T. A. Bailey, J. R. Cowles, and M. D. Winkel, "Applying features of the IEEE 754 to sign/logarithm arithmetic," *IEEE Transactions on Computers*, vol. 41, pp. 1040–1050, Aug. 1992.
3. M. G. Arnold and C. Walter, "Unrestricted faithful rounding is good enough for most LNS applications," in *Proceedings of the 15th IEEE Symposium on Computer Arithmetic (ARITH15)*, (Vail, CO), pp. 237–246, June 2001.
4. V. Paliouras and T. Stouraitis, "Low-power properties of the Logarithmic Number System," in *Proceedings of 15th Symposium on Computer Arithmetic ARITH15*, (Vail, CO), pp. 229–236, June 2001.
5. V. Paliouras, "Optimization of LNS Operations for Embedded Signal Processing Applications," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS'02)*, (Scottsdale, AZ), pp. 744–747, 2002.
6. S. W. Kwa, G. L. Engel, and R. E. Morley, "Quantization Noise Analysis of Sign/Logarithm Data Encoders When Excited by Speech or Sinusoidal Inputs," *IEEE Transactions on Signal Processing*, vol. 48, pp. 3578–3581, December 2000.
7. N. Jayant and P. Noll, *Digital Coding of Waveforms*. Inc. Englewood Cliffs, NJ: Prentice-Hall, 1984.
8. M. G. Arnold, "Iterative Method for Logarithmic Subtraction," in *Proceedings of the Application-Specific Systems, Architectures, and Processors (ASAP'03)*, (The Hague, Netherlands), pp. 315–325, June 24–26, 2003.
9. T. Stouraitis, *Logarithmic Number System: Theory, Analysis and Design*. PhD thesis, University of Florida, 1986.

# A New Logic Transformation Method for Both Low Power and High Testability

Y.S. Son<sup>1</sup> and J.W. Na<sup>2</sup>

<sup>1</sup>Waytotec, Inc. 170-8, Pan Gyo Dong, Bun Dang Gu,  
Sung Nam Si, Gyung Gi Do, South Korea 463-410  
ysson@waytotec.com

<sup>2</sup>Computer Engineering Dept, Hansei University,  
604-5 Dang Jung Dong Kun Po Si,  
Gyung Gi Do, South Korea, 435-742  
jwna@hansei.ac.kr

**Abstract.** An efficient logic transformation method to achieve both low power consumption and high testability is proposed in this paper. The proposed method is based on the redundancy insertion and removal approach. It is also described how redundant connections operate as test points in the test mode. The results of experiments on MCNC benchmark circuits show that the transformed circuit consumes less power in the normal mode and has higher testability in the test mode than the original.

## 1 Introduction

In recent years, the remarkable success and growth of the portable consumer electronics market has driven power consumption to be one of the most important goals in the design of VLSI circuits. The dynamic power dissipation goes up to about 90% of the total power in CMOS circuits because of the charging and discharging operations [1]. The average dynamic power consumption  $E_{avg}$  of a circuit is given by:

$$E_{avg} = \frac{1}{2} \sum_{i=1}^n C_L(i) \cdot V_{dd}^2 \cdot f \cdot P_{TR}(i) \quad (1)$$

$C_L(i)$  and  $P_{TR}(i)$  are the output capacitance and the expected number of transitions of the node  $i$ , respectively,  $n$  is the number of all nodes in the circuit and  $f$  is the clock frequency.  $P_{TR}(i)$  is given by:

$$P_{TR}(i) = 2 \cdot P(i) \cdot (1 - P(i)) \quad (2)$$

$P(i)$  is the signal probability of node  $i$ . Many approaches were proposed to reduce the power consumption of circuits [2,3,4,5,6]. The ever-increasing complexity of the circuit makes the logic testing a harder problem. One approach to solve the problem is DFT (Design for Testability) technique. The signal probability of a node is related to both the dynamic power dissipation and testability. To improve testability of a node, the signal probability must be kept close to 0.5. However, low power design methods make the signal probabilities of internal nodes close to 0 or 1 without considering the testability.

In this paper, we propose a logic design method to achieve both low power consumption and high testability. The rest of this paper is organized as follows. In section 2, the proposed logic design method is explained. In section 3, we describe the negative effects that the low power design has on the testability. A simple solution to improve the testability is presented and the results of experiments on MCNC benchmark circuits are also given. A brief conclusion is given in section 4.

## 2 The Proposed Design Method

### 2.1 Preliminaries

Assume that  $g_i$  denotes the gate whose output node is  $i$ . The set of all fanins of  $g_i$  is denoted by the set  $I_i$ .  $FB_i$  denotes the set of all fanout branches of  $i$ . If there is a path from node  $i$  to node  $j$ ,  $i$  is a transitive fanin of node  $j$  and  $j$  is a transitive fanout of node  $i$ . The set of all transitive fanins of node  $j$  is  $TI_j$  and the set of all transitive fanouts of node  $i$  is denoted by  $TO_i$ .  $v_i$  is the logic value of node  $i$ .  $c_g$  is the control value of gate  $g$ , that is,  $c_g=0(1)$  means that  $g$  is an AND or NAND (OR or NOR) gate. Let  $PO$  and  $FS$  be the sets of all primary outputs and fanout stems of a circuit, respectively.

[Definition 1] A gate is a dominator of  $i$  if all paths from  $i$  to reachable primary outputs must go through it.

Assume that  $D_i$  denotes the set containing all dominators of  $i$ . A fanin of a dominator of  $i$  is off-path if it is the transitive fanout of  $i$ .  $OPI_i$  contains all off-path fanins of all dominators in  $D_i$  and can be determined by the following formula.

A fanout free region (FFR) is a subcircuit that does not contain any fanout stem. All nodes in a FFR must go through the output node of the FFR to reach a primary output.

[Observation 1] Let  $k$  be the output node of a FFR in a circuit. The FFR is unobservable at all primary outputs of the circuit under the following conditions:

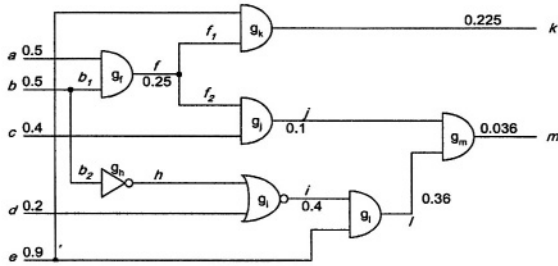
$$\exists i \in OPI_k \text{ such that } v_i = c_{g_j} \text{ where } g_j \in D_k \text{ and } i \in I_j$$

For an example, consider the circuit shown in Fig. 1(a). The signal probability of each node is shown near the name.  $O=\{j,l\}$  and  $FS=\{b,e,f\}$ . A subcircuit  $\{k,h,g\}$  is a FFR and  $D_k=\{l\}$  and  $OPI_k=\{i\}$ . When the logic value of node  $i$  is 0, the subcircuit is unobservable at the primary output  $l$  because  $i$  forces  $l$  to be 0.

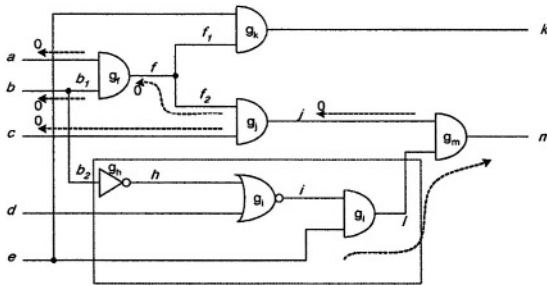
A noninverted line from  $i$  to  $k$  is redundant because  $i=0$  makes  $k$  unobservable and  $i=1$  doesn't make a change in function at the primary output  $l$ . An inverted line from  $i$  to  $h$  is also redundant.

A node, which has very low transition probability and controls one of the dominators of a FFR with very high probability, must be selected as the source of the redundant connection. The sink of the redundant connection is a gate that is in the FFR and whose transition probability is very high.

In Fig. 1(a), all nodes in  $N_j$  are considered as the candidates for the sink. Because  $D_k=\{l\}$ ,  $T_k=\{i\}$ , and  $l$  is an AND gate that is a dominator of the FFR,  $i=0$  means that the  $N_j$  is unobservable at the primary output. We denote the condition of  $i=0$  as  $(i,0)$ . After performing the backward logic implication procedure to find out other



(a) Example circuit



(b) backward logic implication

Fig. 1. An example circuit

conditions satisfying  $i=0$ , we can get  $\{(i,0), (f,0), (c,0), (b,0), (a,0)\}$ . From the Fig. 1(a), we can see that  $(i,0)$  has the highest probability among the conditions.

### 2.2 Proposed Logic Transformation Method

Table 1 shows types of possible redundant connections. If a FFR is unobservable when the value at the source is 0, the non-inverted connections from the source to INV, BUF, AND, or NAND gates in the FFR are redundant. If the value of source is 0 and sinks are OR or NOR gates, the inverted connections are redundant.

If a circuit consists of simple gates and has no reconvergent fanout node, the exact signal probabilities can be computed using the following equation[12] :

$$\begin{aligned}
 P(o) &= 1 - P(i) \quad \text{for INV gate} \\
 P(o) &= \prod_{i \in FIN(o)} P(i) \quad \text{for AND gate} \\
 P(o) &= 1 - \prod_{i \in FIN(o)} (1 - P(i)) \quad \text{for OR gate}
 \end{aligned}
 \tag{3}$$

where  $o$  is the output of the gate. In [13], the authors described a polynomial simulation method to calculate switching activities in a general-delay combinational logic circuit. They show that ignoring spatial correlation for signals that reconverge in a few logic levels introduces negligible error. So, we may estimate signal probability of each node in the FFR by (3) without losing the proper precision.

**Table 1.** Types of redundant connections

Control value of source	Sink	Type of Redundant Connection
0	INV or BUF	non-inverted
	AND or NAND	non-inverted
	OR or NOR	inverted
1	INV or BUF	non-inverted
	AND or NAND	inverted

Let  $r$  be the node that has the highest probability to make a FFR unobservable at all primary outputs. We may assume that  $r=0$  makes the FFR unobservable and a non-inverted line from  $r$  to a node  $k$  of the FFR is redundant. If we let  $P_{bef}(k)$  be the signal probability of  $k$  before redundancy insertion and  $P_{aft}(k)$  that of  $k$  after redundancy insertion,  $P_{aft}(k)$  is given by:

$$P_{aft}(k) = P_{bef}(k) \cdot P(r) \tag{4}$$

From (2), we can compute the transition probability of  $k$  before and after redundancy insertion as follows.

$$P_{bef}^{TR}(k) = 2 \cdot P_{bef}(k) \cdot (1 - P_{bef}(k)) \tag{5}$$

$$P_{aft}^{TR}(k) = 2 \cdot P_{bef}(k) \cdot P(r) \cdot (1 - P_{bef}(k) \cdot P(r))$$

Then, the reduction factor  $\Delta_k$ , which denotes the amount of reduction of transition probability resulted by redundancy insertion, is given by:

$$\Delta_k = P_{bef}^{TR}(k) - P_{aft}^{TR}(k) = 2 \cdot P^* \cdot (\alpha - \beta \cdot P^*) \tag{6}$$

where  $P^*=P_{bef}(k)$ ,  $\alpha=(1-P(r))$ , and  $\beta=(1-P(r)^2)$ .  $P^*$ ,  $\alpha$ , and  $\beta$  for all possible cases are listed in Table 2.  $v(s)$  is the value of source  $s$  that makes sinks unobservable.

**Table 2.**  $P^*$ ,  $\alpha$ , and  $\beta$  according to  $v(s)$  and sink type

Sink	$v(s)$	$P^*$	$\alpha$	$\beta$
AND	0	P(O)	$1-P(r)$	$1-P(r)^2$
	1		$P(r)$	$1-(1-P(r))^2$
NAND	0	$1-P(O)$	$1-P(r)$	$1-P(r)^2$
	1		$P(r)$	$1-(1-P(r))^2$
OR	0	$1-P(O)$	$P(r)$	$1-(1-P(r))^2$
	1		$1-P(r)$	$1-P(r)^2$
NOR	0	P(O)	$P(r)$	$1-(1-P(r))^2$
	1		$1-P(r)$	$1-P(r)^2$

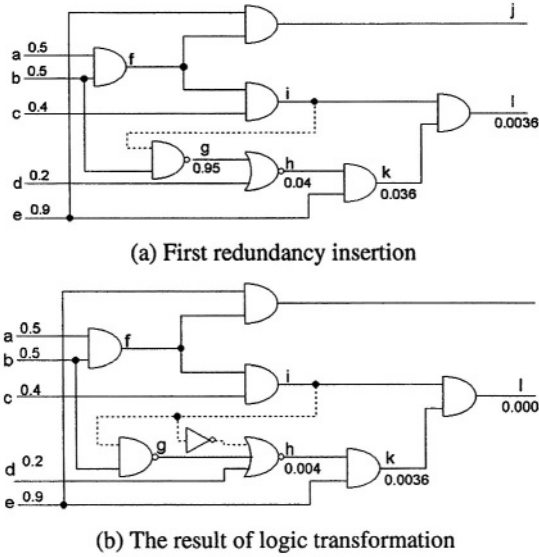


Fig. 2. The result of logic transformation

```

Power_optimize() {
    S_FFR = find_all_FFR()
    logic_simulation()
    while (S_FFR ≠ ∅) {
        R = select_a_FFR(S_FFR)
        D_FFR = find_all_dominator(R)
        control = find_control_fanin_of_dominator(R, D_FFR)
        estimate_signal_activity(R)
        S_candidate = backward_implication(I_control, R)
        S_best = select_best_source(S_candidate)
        check_and_insert_redundancy(S_best, R)
    }
}
    
```

Fig. 3. The proposed algorithm

Fig. 2 shows the two redundant connections that are inserted from *i* to *g* and from *i* to *h*. Compared to Fig. 1(a), inserting a non-inverted line from *i* to *g* makes the signal probabilities of *g*, *h*, and *k* to be 0.95, 0.04, and 0.036, respectively,  $\Delta_g$  is 0.405.

If we let the load capacitance of the gate be the number of fanouts, the total switched capacitance of the circuit reduces from 5.45 to 4.37. We may also insert an inverted line from *i* to *h* as shown in Fig. 2(b). The total switched capacitance of the circuit, however, increases from 4.37 to 4.73 because the second redundant connection has little effect on the signal probabilities of all its transitive fanouts, that is,  $\Delta_h$  is negligible. On the other hand, the amount of switched capacitance due to the inserted INV and the inserted redundant fanout at *i* is larger than that reduced at the FFR. The second redundant connection must not be inserted.

Fig. 3 shows the proposed logic transformation algorithm. First of all, we find the FFRs in the circuit and save them in the set  $S_{FFR}$ . Then, we select a FFR from  $S_{FFR}$ , compute  $D_{FFR}$  which contains the dominators of the output of the selected FFR, and

**Table 3.** Experimental Results on Power Consumption

Circuit	Original			After	
	#PI	#PO	#lit	Area	SA
cm138	6	8	30	1.03	0.87
Decod	5	16	54	1.00	0.88
cm85	11	3	56	1.08	0.95
cu	14	11	72	1.04	0.96
f51m	8	8	320	0.99	0.97
ttt2	24	21	428	1.00	0.97
frg1	28	3	811	1.03	0.96
rot	135	107	1259	1.00	0.99
x2	10	7	62	0.99	0.93
cm42	4	10	27	1.00	0.96
cmb	16	4	65	1.09	0.90
cm163	16	5	57	1.06	0.96

estimate the transition probability of the FFR by equation (2) after logic simulation. During the backward implication procedure, we can get  $S_{\text{candidate}}$  that contains the conditions of making the selected FFR unobservable at all primary outputs. From  $S_{\text{candidate}}$  we choose the condition that has the highest probability and we can get the best source  $S_{\text{best}}$ . The `check_and_insert_redundancy` routine estimates what effect the inserted redundancy has on the total switched capacitance of the circuit and determines whether redundancy is inserted or not. Repeat the above procedures for all FFRs.

Table 3 shows the experimental results on some MCNC benchmark circuits. #PI and #PO columns contain the numbers of primary inputs and primary outputs of original circuits, respectively. Area column shows the ratio of the area of the transformed circuit to that of the original circuit. SA column shows the ratio of power consumption of the transformed circuit to that of the original. The proposed method can reduce the total power consumption of some circuits by 13% with low area overhead.

### 3 Improving the Testability

#### 3.1 Signal Probability and Testability

The stuck-at fault model is the most widely accepted fault model in the industry. In this model it is assumed that the fault causes a line of the circuit to behave as if it is permanently at logic 0 (stuck-at 0 fault) or logic 1 (stuck-at 1 fault). Detection of stuck-at faults needs two conditions, that is, fault activation and fault effect propagation. Driving the faulty node to have the opposite of the faulty value is fault activation. The activated fault effect must be propagated to the primary outputs so as to be detected.

If the pseudo-random sequences generated by linear feedback shift register (LFSR) can't drive a node to be 0 (1), the stuck-at-1 (0) fault at the node is a redundant one.



So, the signal probability of a node is deeply related to its testability. This is because the signal probability of a node indicates the ratio of the number of patterns that drive the node to be 1 to the number of all possible patterns. Lower the signal probability of a node, fewer the patterns that activate the stuck-at-0 fault at the node. Higher the signal probability, fewer the patterns that activate the stuck-at-1 fault at the node.

Consider stuck-at-1 (0) fault of a node  $k$  whose signal probability is  $P(k)$ . Let  $L_0(k)$  denote the order of a test pattern that first drives  $k$  to be 0.  $L_0(k)=i$  means that the node can't be driven to be 0 by the first  $(i-1)$  test patterns and can be driven by the  $i$ -th test pattern. As the signal probability of  $k$  is  $P(k)$ , the probability of  $L_0(k)=i$  is  $P(k)^{i-1} \cdot (1-P(k))$ . Let  $L_1(k)$  denote the order of a test pattern that first drive  $k$  to be 1. Under the assumption that the stuck-at 1 fault and the stuck-at 0 fault are equally likely at a node, we can derive the following theorem on the fault activation length that indicates the number of test patterns needed to activate a stuck-at fault.

[Theorem 1] Let  $P(k)$  is the signal probability of a node  $k$ . If we let  $L(k)$  denote the fault activation length at  $k$ ,  $L(k)$  is given by:

$$L(k) = \lceil \{2 P(k) (1-P(k))\}^{-1} \rceil$$

In case of  $P=0.5$ ,  $P^{TR}$  is the maximum and  $L$  is the minimum. As  $P$  is closer to 1 or 0,  $P^{TR}$  decreases to 0, and  $L$  increases rapidly. When  $P=0$  ( $P=1$ ), the stuck-at-0 (1) fault at the node is a redundant fault. Because logic transformation is the process that makes the signal probabilities of some nodes closer to 1 or 0,  $L$  rapidly increases at those nodes.

After fault activation, the fault effect must be propagated to any primary output in order to detect the activated fault. If the fault effect is arrived at a fanin of a gate whose logic value is controlled by other fanins, it can't be propagated to the output of the gate. This is called fault masking.

The effect on the testability of the circuit that the inserted redundant connections have is well explained in Fig. 4. To reduce the switched capacitance, a redundant connection  $k'$  is inserted from  $k$  to the FFR.  $D$  is one of the dominator of the FFR. When the node  $k$  controls  $D$ , the fault effect activated in the shaded region  $A$  is masked at  $D$  and not propagated to the region  $B$ . The redundancy  $k'$  also makes the activation of the faults of the black region  $C$  very difficult because the signal probabilities of the nodes in the region  $C$  are closer to 0 or 1 due to  $k'$ . The faults occurred at the region  $C$  are likely to be random-pattern-resistant faults.

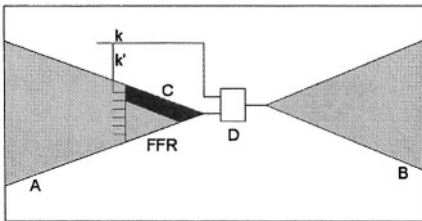


Fig. 4. The effect of redundancy on the testability

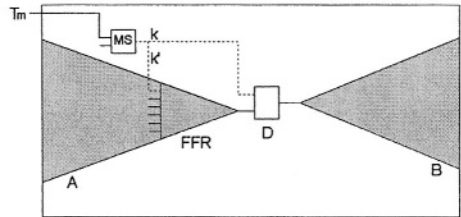


Fig. 5. The redundancy connection with DFT

### 3.2 Proposed DFT Structure

Fig. 5 shows the structure of redundancy to improve the testability of the circuit. MS (Mode Select) is a two-input gate used as a test point and  $T_m$  is a primary input used for selecting the operation mode. In case of the normal  $\text{mode}(T_m=0)$ , the circuit consumes less power. In case of the test  $\text{mode}(T_m=1)$ ,  $k$  and  $k'$  have the non-control values and the faults of the region A are easily propagated to region B without being masked at the dominator D.

In the test mode, the testing strategy is consisted of two phases. In the first phase, a half of test patterns are applied to the circuit with  $T_m=0$ , so faults of all region but A are detected. In the second phase, the rest of test patterns are applied with  $T_m=1$  and faults of region A can be easily propagated to region B through D.

Using fsm, which is a fault simulator developed at Virginia Polytech, we evaluated the fault coverage of the benchmark circuits with 2000 pseudo random patterns. The fault simulation result is shown in Table 4.  $FC_{LT}$  column contains the fault coverage of benchmark circuits processed by logic transformation without DFT. The fault coverage of circuits that are processed by the logic transformation with DFT is shown in  $FC_{LT,DFT}$  column. From Table 4, we can see that the proposed logic transformation method with DFT can improve the fault coverage of some circuits by 5.78% compared with the logic transformation method without DFT.

**Table 4. Experimental Results on Fault Coverage**

Circuit	$FC_{LT}$	$FC_{LT,DFT}$
cm163	91.66	94.31
cm85	95.0	99.25
cmb	20.48	23.03
cu	85.71	86.36
frg1	40.19	42.58
rot	87.23	87.65
x2	94.22	100

Fig. 6 shows the fault coverage versus the number of test patterns for two benchmark circuits, that is, cmb and rot. In case of cmb, 6 redundant connections are inserted and the number of sources is 3. All of the sources are replaced with the proposed DFT structure. Fig. 6 (a) shows the fault coverage for cmb. *org* denotes the fault coverage curve of the original bench circuit. *ndft* and *dft* denote the fault coverage curves of the circuits that are processed by the logic transformation method without DFT and the logic transformation method with DFT respectively. As shown in Fig. 6 (a), *dft* is higher than *org* and *ndft* with fewer test patterns.

Rot is a benchmark circuit that has 26 redundant connections after logic transformation and only 3 among the sources are replaced with the DFT structure. *dft* is higher than *org* and *ndft* by about 1%. Fig. 6 (b) looks like showing little enhancement in the fault coverage. But, in case of *dft*, we can get 97% fault coverage with about 2000 test patterns. But *org* needs more than 2600 test patterns and *ndft* needs more than 3000 test patterns to achieve the same fault coverage. We can reduce the test application time by about 30% or more.

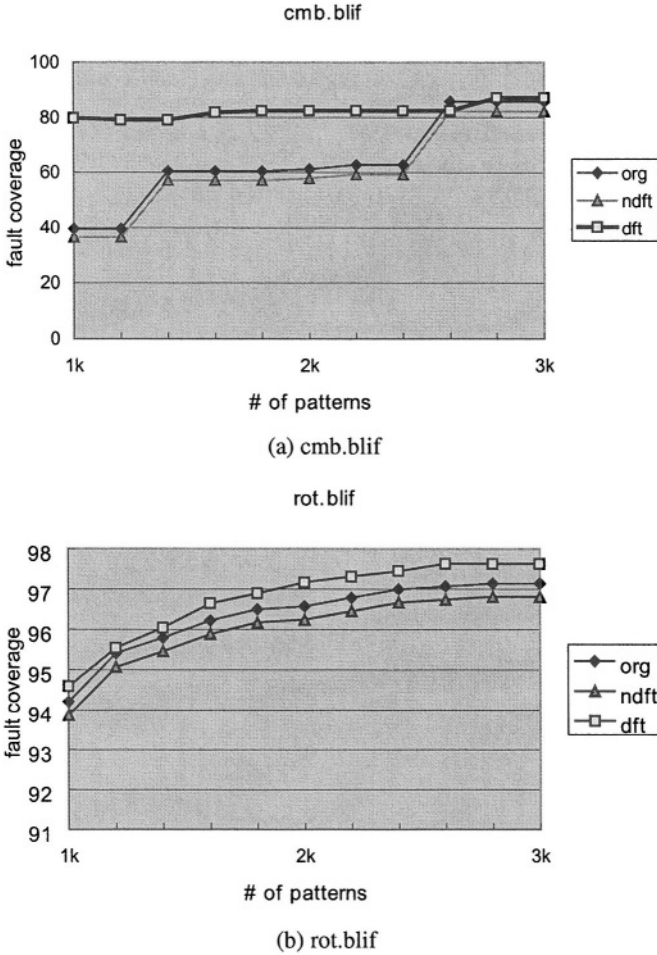


Fig. 6. Fault coverage curve

## 4 Conclusion

In this paper, a novel and efficient logic transformation method targeting both the low power consumption and the high random pattern testability simultaneously were proposed. The proposed method uses the redundancy insertion and removal approaches. Under the condition that a FFR of the circuit is unobservable at all primary outputs, the proposed method reduces the switching activities of the circuit with the low area overhead. The effect of the logic transformation method on the testability was explained. The structure of redundant connections is described to improve the testability of the circuit. From the experiments on the MCNC benchmark circuits, the results demonstrate that the proposed method improves the power consumption and testability of circuits simultaneously.

## References

1. J. M. Rabaey, M. Pedram, 1st ed., *LOW POWER DESIGN METHODOLOGIES*, Kluwer Academic Publishers, 1996.
2. B. Rohfleisch, A. Kolbl, and B. Wruth, Reducing Power Dissipation after Technology Mapping by Structural Transformations, Proc. DAC, pp. 789-794, 1996.
3. S. Chang, et. al, Multilevel Boolean Network Optimizer, IEEE Trans. CAD, vol. 15, no. 12, pp. 1494-1504, Dec. 1996.
4. K. T. Cheng and L. A. Entrena, Multi-level Logic Optimization By Redundancy Addition and Removal, Proc. European DAC. pp. 373-37, 1993.
5. Q. Wang and S. B. Vrudhula, Multi-level Logic Optimization for Low Power using Local Logic Transformations, Proc. ICCAD, pp. 270~277, 1996.
6. Ki-Seok Chung and C. L. Liu, Local Transformation Techniques for Multi-Level Logic Circuits Utilizing Circuit Symmetries for Power Reductions, Proc. ISLPED, pp. 215~220, 1998.
7. J. Savir and R. Berry, At-speed test is not necessarily an AC test, Proc. ITC, pp. 722-728, 1991.
8. Mitrajit Chaterjee and Dhiraj K. Pradhan, A Novel Pattern Generator for Near-Perfect Fault-Coverage, Proc. VLSI Test Symp., pp. 417-425, 1995.
9. J. Savir, Reducing the MISR size, IEEE Trans. Comp., vol. 45, pp. 930-938, 1996.
10. M. F. AlShaibi and C. R. Kime, MFBIST : A BIST Method for Random Pattern Resistant Circuits, Proc. ITC, pp. 176-185, 1996.
11. P. H. Bardell, W. H. McAnney, and J. Savir, 1st ed., *Built-in Test for VLSI : Pseudorandom Techniques*, John Wiley & Sons, 1987.
12. H. Goldstein, Controllability/observability of digital circuits, IEEE Trans. Circuits and Systems, pp. 685-693, 1979.
13. J. C. Costa, J. C. Monteiro, Srinivas Devadas, Switching Activity Estimation using Limited Depth Reconvergent Path Analysis, proc. ISLPED, pp. 184~189, 1997.
14. H. K. Lee and D. S. Ha, Atalanta: an Efficient ATPG for Combinational Circuits, Technical Report, pp. 93-12, Dept. of Electrical Eng., Virginia Polytechnic Institute and State University, 1993.
15. H. K. Lee and D. S. Ha, An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation, Proc. ITC, pp. 946-955, 1991

# Energy-Efficient Hardware Architecture for Variable N-point 1D DCT

Andrew Kinane, Valentin Muresan, Noel O'Connor,  
Noel Murphy, and Sean Marlow

Centre for Digital Video Processing,  
Dublin City University,  
Glasnevin, Dublin 9,  
IRELAND

kinanea@eeng.dcu.ie  
<http://www.cdvp.dcu.ie>

**Abstract.** This paper proposes an energy-efficient hardware acceleration architecture for the variable N-point 1D Discrete Cosine Transform (DCT) that can be leveraged if implementing MPEG-4's Shape Adaptive DCT (SA-DCT) tool. The SA-DCT algorithm was originally formulated in response to the MPEG-4 requirement for object based texture coding, and is one of the most computationally demanding blocks in an MPEG-4 video codec. Therefore energy-efficient implementations are important - especially on battery powered wireless platforms. This N-point 1D DCT architecture employs a re-configurable distributed arithmetic data path and clock gating to reduce power consumption.

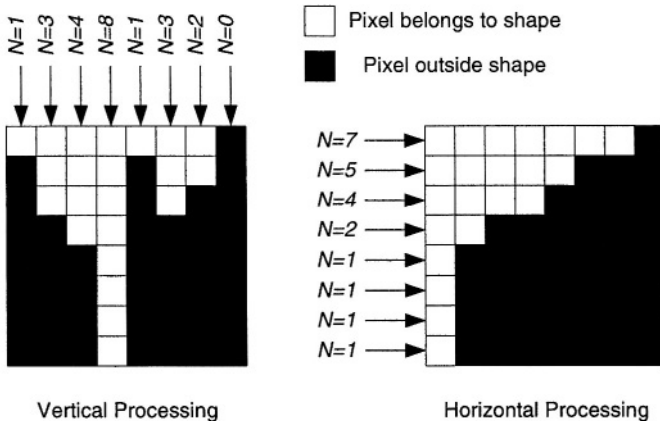
## 1 Introduction

Natural video scenes consist of a stationary background and moving foreground objects that are of arbitrary shape. When encoding texture, a video codec system divides each rectangular video frame into an array of non-overlapping 8x8 texture pixel blocks and processes these sequentially. In previously standardized video coding schemes (e.g. MPEG-1, MPEG-2) the 8x8 Discrete Cosine Transform (DCT) processes all blocks, regardless whether they belong to the background or to a foreground object. The DCT is used because it transforms video data into a format more amenable to efficient compression for transmission or storage. MPEG-4 uses the Shape Adaptive DCT (SA-DCT [1]) to support object-based texture encoding, which in turn allows object manipulation as well as giving improved compression efficiency.

In MPEG-4, the object shape description of a frame is termed the alpha-plane or video object plane (VOP). The alpha-plane of a video object can be provided by (semi-) automatic segmentation of the video sequence. This technique is not covered by the MPEG-4 standardization process and depends on the application. The 8x8 alpha block corresponding to a particular texture block defines which pixels in the texture block are part of the video object (VO). For blocks that are located entirely inside the VOP, the SA-DCT behaves identically

to the 8x8 DCT. Blocks that are located entirely outside the VOP are skipped to save needless processing. Blocks that lie on the VOP boundary are encoded depending on their shape and only the opaque pixels within the boundary blocks are actually coded.

This paper addresses the problem of accelerating the variable N-point 1D DCT function required for the SA-DCT with power efficient hardware. A survey of current state of the art implementations of the DCT and SA-DCT is given in [2]. The SA-DCT is less regular compared to the 8x8 block-based DCT since its processing decisions are entirely dependent on the shape information associated with each individual texture block. The 8x8 DCT requires 16 1D 8-point DCT computations if implemented using the column-row approach<sup>1</sup>. Each 1D transformation has a fixed length of 8, with fixed basis functions. This simplifies hardware implementations since the data path is fixed and all parameters are constant. Depending on the shape, the SA-DCT requires up to 16 1D N-point DCT computations where  $N = 2, 3, \dots, 8$  ( $N = 0, 1$  are trivial cases). Fig. 1 shows an example of how N can vary across a boundary block, where N is defined as the length of VOP pixels in a particular row or column.



**Fig. 1.** Example of boundary block requiring 15 1D DCT processes with various data vector lengths N.

In this case the basis functions vary with N, complicating hardware implementation. However, the variable nature of the N-point DCT load affords the possibility of dynamically clock gating logic that is independent of the shape information.

<sup>1</sup> Each column of the input 8x8 block is processed with an 8-point DCT and results are stored in an intermediate memory. Then the 8 rows of the memory undergo an 8-point DCT giving the final output.

The outline of this paper is as follows. Section 2 discusses the general low power design philosophy adopted and how it relates to the particular properties of the N-point 1D DCT function. Section 3 describes in more detail the energy-efficient architecture that implements the N-point 1D DCT, and the final sections discuss future work and offer some concluding remarks.

## 2 Low Power Design Methodology

### 2.1 Top Down Approach

In general, power consumption in a CMOS circuit has two components - dynamic power and static power. Traditionally the dynamic component was by far the most dominant, but this is changing due to physical phenomena as process technologies shrink below 90nm. It is estimated [3] that when devices are scaled to 45nm (around the year 2007) the static component will be equal in proportion to the dynamic component. Static power is usually tackled using a variety of dynamic power management (DPM) techniques. If a sub-system is not required to process data at all, it may be shutdown in order to avoid unnecessary power consumption. If the sub-system is still required but has a reduced load, a DPM controller can scale the operating frequency and voltage to save power based on the dynamic processing load. Video processing is very non-uniform by nature so there is scope to apply DPM techniques to video processing architectures for energy conservation purposes. Dynamic power consumption is caused by circuit node switching activity. To reduce this component, a system designer primarily attempts to reduce the complexity of the required processing on the premise that if there are less operations to be carried out, there will be less switching and hence less energy dissipation. It is generally accepted that most power savings are achieved at the higher levels of abstraction (algorithmic and architectural levels) since there are wider degrees of design freedom [4,5]. Applying low power techniques at the logic, circuit and physical levels are also important but in general depend on the technology available with which the optimized architecture is to be implemented. Therefore initial work on the N-point 1D DCT has focused on optimizing the number of basic operations involved at a generic architectural/algorithmic level (additions/subtractions, multiplications, shifts, processing stages etc.) while maintaining the required accuracy of the output generated.

### 2.2 N-point 1D DCT Properties

The N-point 1D DCT is described by equation 1, which is essentially N dot products between an N-point data vector  $\mathbf{f}$  and an  $N \times N$  matrix of N cosine basis functions  $\mathbf{A}$ . Each dot product produces a single DCT coefficient  $F(u)$ . In this case, power savings are achieved by optimizing the number of multiplications and additions necessary while preserving the acceptable quality of the DCT coefficients produced as defined by the MPEG-4 standard [6]. Hardware

multipliers are inherently more complex and power consumptive compared to adders so greatest effort is focused on minimizing these.

$$\mathbf{F} = \mathbf{A}\mathbf{f} \quad (1)$$

where

$$\begin{aligned} F(u) &= \sqrt{\frac{2}{N}} C(u) \sum_{x=0}^{N-1} f(x) \cos \left[ \frac{(2x+1)u\pi}{2N} \right] \\ &= \sum_{x=0}^{N-1} A(x) * f(x) \\ C(u) &= \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } u = 0 \\ 1, & \text{for } u \neq 0 \end{cases} \\ &u = 0, \dots, N-1 \end{aligned}$$

### 3 Energy Efficient Architecture

#### 3.1 Re-configurable Distributed Arithmetic Data Path

One commonly employed technique for implementing a dot product of a variable data vector with a vector of constants is Distributed Arithmetic (DA), especially if energy efficiency is paramount to the implementation [7]. In short DA is an efficient architecture for computing a dot product by transforming it to a series of additions, look-ups and shift operations. DA distributes the bits of the input variable vector  $f(x)$  and uses an aggregation of these bits for each bit position to form weights which are linear additive combinations of a constant basis vector  $A(x)$ . These weights are then scale accumulated together to produce a final coefficient  $F(u)$  achieved without the need for any multiplications. Multiplications are inherently more complicated and power consumptive compared to additions (greater area, possibly more clock cycles required, more switching) so eliminating them will certainly result in a more energy-efficient architecture. Since the 1D N-point DCT is itself a series of dot products, DA seems a logical architectural choice.

A variation on DA is New Distributed Arithmetic (NEDA) [8] that distributes the bits of the constant basis vector  $A(x)$  (as opposed to the data vector) and forms weights that are linear additive combinations of input data vector  $f(x)$ . The implication of this alternative approach is that no ROM look-ups are required. This is important for a variable N-point 1D DCT implementation since the values for each NxN matrix where  $N = 2, 3, \dots, 8$  would require storage with conventional DA.

The architecture proposed in this paper leverages the NEDA properties, however, the data path taken to form the weights depends on the value of N. Essentially this implies that the addends from the data vector  $f(x)$  used to form the weights are multiplexed by the value of N. The number of hardware adders required for the NEDA tree is optimized by using the greedy algorithm described in [9].



A conceptual architecture for the 1D N-point 1D DCT is illustrated in Fig. 2. Presented at the input ports are the data vector  $f(x)$  and the value of N for this vector, where N has been previously decoded from the corresponding alpha block. The first stage involves combining the input data vector using 21 two input additions and the results are stored in the first register stage regardless of the value of N. Next, for each of the N coefficients 13 weights are computed using a linear additive combination of a subset of the 23 primary addition values and the primary inputs themselves. The combination taken depends on the value of N. Each coefficient  $F(u)$  is computed differently depending on the value of N, and N is used to multiplex the addends in the weight generation addition stage. The number of weights for each coefficient was chosen to be 13. Experimentation has shown that this is the lowest number possible to comply with the standard in terms of performance [10]. The potential to vary the number of weights used to save power is discussed in the next sub-section. Once these 13 weights for each of the N coefficients have been computed and stored in the second register stage, these weights are then combined in a scaled manner according to the NEDA architecture using a carry save adder tree. The final outputs of each of the carry save adder trees represent the final N coefficient values that can be stored in output registers.

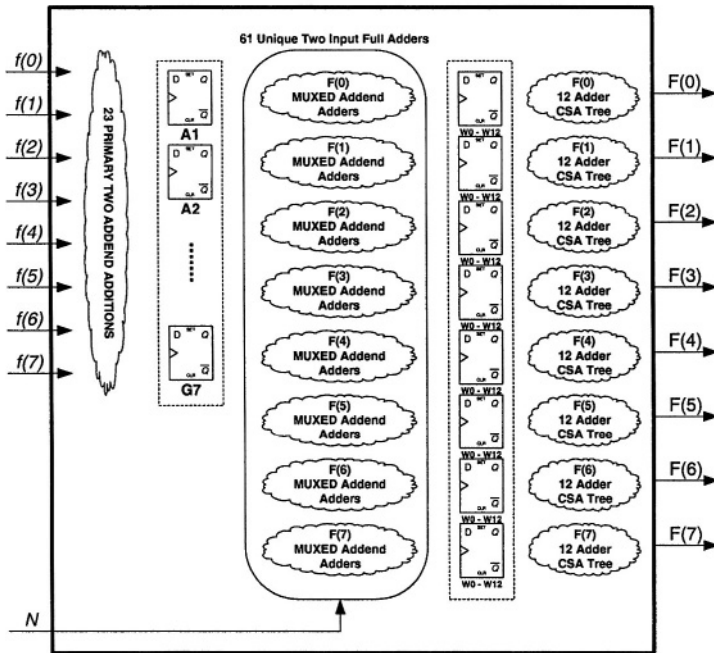


Fig. 2. A conceptual NEDA architecture implementing the 1D N-point 1D DCT, where the value of N is used to multiplex the data path.

### 3.2 Trading Power for Performance

As mentioned in the previous sub-section, 13 weights are required for each coefficient computation (N in total) for strict compliance with the MPEG-4 standard performance requirements [6]. The DCT coefficients are said to be compliant if, when reconstructed to pixels using an inverse transform (IDCT), they are within two grey levels of the same data that has undergone a double precision DCT/IDCT process. However, a user may tolerate some perceptual visual degradation if it means a longer battery life. By reducing the number of weights used, the numbers of adders used to compute the coefficients are reduced as a result thus saving power. Unfortunately, reducing the number of weights affects the accuracy of the coefficients, and it is clear that this represents a power versus performance trade-off. The number of adders required for all values of N using different amount of weights Q to generate DCT coefficients with the proposed architecture are listed in Table 1. For a particular Q value, the total number of unique adders required to implement a variable N-point 1D DCT is an intersection of the adder requirements for each individual N value. The optimal adder requirements are obtained using a combinatorial optimization technique [9].

**Table 1.** Number of adders necessary for each value of N for various numbers of NEDA weights Q

N→	8	7	6	5	4	3	2	1	Total	Saving
Q=13	47	53	28	22	18	14	13	0	<b>180</b>	n/a
Q=12	46	52	27	21	17	13	12	0	<b>172</b>	4.44%
Q=11	45	49	26	20	16	12	11	0	<b>163</b>	9.44%
Q=10	42	47	25	19	15	11	10	0	<b>151</b>	16.11%

Experimentation has been carried out using the MPEG-4 video test sequences using various values of Q for the proposed architecture and the results are presented in Table 2.

**Table 2.** Performance experimentation results for various values of Q

Q	Akiyo			Container			Coastguard		
	PSNR [dB]	Error [%]	Max Pel Error	PSNR [dB]	Error [%]	Max Pel Error	PSNR [dB]	Error [%]	Max Pel Error
13	55.844	0	0	56.095	0	0	55.8	0	0
12	53.519	0.002	3	55.694	0.000	3	53.575	0.005	3
11	49.684	0.872	5	52.953	0.083	4	50.047	0.681	5
10	44.562	10.081	8	44.529	6.586	9	45.109	8.376	8

These results illustrate that using 13 weights ( $Q = 13$ ), there are no errors and the architecture is fully compliant with the MPEG-4 standard. However, as  $Q$  reduces, the pixel errors become more significant as expected but less power is consumed since there is less processing. However, the results show that the pixels do not deviate wildly from the required accuracy. For example, with the “container” test sequence with  $Q = 10$ , 6.586% of the VOP pixels reconstructed violate the standard but the maximum error is only a difference of 9 grey-levels which is imperceptible to the human eye. A user may tolerate such degradation on a mobile platform if it means a longer battery life.

### 3.3 Clock Gating Possibilities

As well as the potential to trade precision for power, this architecture has been designed such that it is possible to clock gate redundant logic based on the value of  $N$ . The adder logic for each coefficient has been partitioned to allow each one to be clock gated individually. To illustrate, consider the possibility that  $N = 5$  for a particular computation. In this case the only valid coefficients are  $F(0)$  to  $F(4)$ . The logic producing coefficients  $F(5)$ ,  $F(6)$  and  $F(7)$  is not needed and so can be clock gated for this particular computation. This can be generalized to say that if  $N$  is less than 8, coefficients  $F(N)$  to  $F(7)$  are irrelevant and can be clock gated. In this way the power consumed is not constant but depends on the shape (and hence the value of  $N$ ).

### 3.4 Summary

The power efficient properties of this architecture may be summarized as follows:

- The 1D SA-DCT computation unit is configurable based on the value of the transform length  $N$ . This is much more efficient than having a separate computation unit for each value of  $N$ .
- The accuracy of the DCT coefficients produced by the architecture can be dynamically adjusted by decreasing the number of adders used in the data path. This can be done to save power if deemed acceptable by the user.
- Within the computation unit unnecessary logic is clock-gated based on the value of  $N$ , such that less power is consumed when the computational load is smaller (smaller  $N$ ).
- The architecture has the general distributed arithmetic property of no multipliers. An  $m$  bit multiplication is much more power consumptive compared to an  $m$  bit addition, especially if the computation must take place in one clock cycle.
- The architecture has the NEDA property of requiring no power-consuming ROM lookups that are necessary with conventional distributed arithmetic. This is especially important since the SA-DCT has numerous cosine basis function possibilities and would require a relatively large ROM.
- Multiplications have been eliminated, but also the number of adders necessary to produce the weights has been optimized using a combinatorial optimization technique [9].

## 4 Conclusions and Future Work

This work has enhanced the NEDA architecture for implementing an 8-point 1D DCT [8] to compute a variable length N-point 1D DCT  $N = 0, 1, \dots, 8$ . This is achieved by implementing a multiplexed coefficient generation data path based on the value of N. In the future, it is intended to leverage this 1D N-point DCT processing element to implement a full energy efficient SA-DCT hardware core. Part 9 of the MPEG-4 standard defines hardware architectures for the most computationally demanding tools in the standard and currently there is no SA-DCT implementation, only a conventional 8x8 DCT [11]. The SA-DCT is required to realize the object-based processing capabilities of the MPEG-4 standard, and it intended that this work would fill the void. The intended design flow is as follows:

- SystemC RTL description of SA-DCT core using Microsoft Visual C++.
- Synthesis check and translation to Verilog using Synopsys SystemC Compiler.
- SystemC/Verilog co-simulation with Synopsys VCS to verify translation.

At this point the design flow diverges into two separate paths. The first path involves targeting an ARM processor prototyping platform (Integrator/CP with Xilinx FPGA) and integration with other DCU MPEG-4 hardware acceleration architectures. This process involves:

- Synthesis of Verilog/VHDL code using Synplicity Pro targeting Xilinx VirtexE XCV2000E FPGA technology.
- Place and route to ARM Integrator/CP platform using Xilinx ISE.
- Integration with other DCU hardware accelerators.

The second design flow involves targeting the Annapolis WildCard-II PCMCIA card to benchmark against other architectures proposed to the MPEG-4 reference hardware forum:

- Synthesis of Verilog/VHDL code using Synplicity Pro targeting Xilinx XC2V3000-4-FG676 FPGA technology (integrated on Annapolis WildCard-II architecture).
- Place and route to Annapolis WildCard-II PCMCIA card using Xilinx ISE.
- Performance evaluation using WildCard-II card features. The WildCard-II allows board level current, voltage and power to be measured dynamically.

It is envisaged that the eventual SA-DCT hardware core developed will be an energy-efficient hardware implementation of the SA-DCT function. This solution could then be integrated onto a wireless platform supporting MPEG-4 where the battery life is limited.

**Acknowledgements.** The support of the Research Innovation Fund of Enterprise Ireland is gratefully acknowledged.

## References

1. Sikora, T., Makai B., Shape-Adaptive DCT for Generic Coding of Video, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 1, February 1995, pp 59–62.
2. Muresan, V., Kinane, A., Larkin, D., Hardware Acceleration Architectures for MPEG-based Mobile Video Platforms: A Brief Overview, Proc. 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), London, April 2003, pp 456–461.
3. Butts, J.A., et al. A Static Power Model for Architects, 33rd International Symposium on Microarchitecture, December 2000.
4. Ruby, W., (Low) Power To The People, EDAVision Magazine, March 2002.
5. Poppen, F., Low Power Design Guide, Low Power Design and Services Group, OFFIS Research Institute, <http://www.lowpower.de> [Accessed 16th Sep. 2003].
6. MPEG-4: Information Technology - Coding of Audio Visual Objects - Part 2: Visual, ISO/IEC 14496-2, Amendment 1, July 15th 2000.
7. Xanthopoulos, T., Chandrakasan, A.P., A Low-Power DCT Core Using Adaptive Bitwidth and Arithmetic Activity Exploiting Signal Correlations and Quantization, IEEE Journal of Solid-State Circuits, Vol. 35, No. 5, May 2000, pp 740 - 750.
8. Shams, A., Pan, W., Chidanandan, A., Bayouni, M.A., A Low Power High Performance Distributed DCT Architecture, Proceedings of the IEEE Computer Society Annual Symposium on VLSI 2002 (ISVLSI'02).
9. Potkonjak, M., Srivastava, M.B., Chandrakasan, A.P., Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15, No. 2, February 1996, pp 151 - 165.
10. Kinane, A., Power-Efficient Hardware Accelerator for MPEG-4 Shape Adaptive Discrete Cosine Transform Tool Software Demo, Internal Technical Report, Visual Media Processing Group, DCU, February 2004.
11. Mattavelli, M., Turney, R., Text of ISO/IEC DTR 14496-9 Information technology - Coding of audio visual objects - Part 9: Reference hardware description, ISO/IEC 14496-9, Pattaya, March 2003.

# Two Level Compact Simulation Methodology for Timing Analysis of Power-Switched Circuits

Stephan Henzler<sup>1</sup>, Georg Georgakos<sup>2</sup>, Jörg Berthold<sup>2</sup>, and Doris Schmitt-Landsiedel<sup>1</sup>

<sup>1</sup> Munich University of Technology, Theresienstr. 90, 80290 Munich, Germany  
henzler@tum.de

<sup>2</sup> Infineon Technologies AG, Coporate Logic, Balanstr. 73, D-81541 Munich, Germany

**Abstract.** Standby-power dissipation in ultra-deep submicron CMOS can be reduced by power switching. As the cut-off device has a strong impact on area consumption, minimum power-down time, signal delay and leakage suppression, a proper sizing of this device is of general importance. Therefore a two level compact simulation methodology is proposed which provides fast and accurate CAD support to the switch design task.

## 1 Introduction

As system complexity and transistor count increase, the standby-power dissipation becomes a serious challenge in ultra-deep submicron CMOS logic. Sub-threshold as well as gate tunneling leakage increase with ongoing technology scaling [3], thus circuit level strategies have to be used to reduce the standby-power dissipation. One of the most promising approaches especially for mobile applications with varying activity is the insertion of cut-off devices between the circuit and the power supply [4,5,8]. The switch transistor can be either a p-channel device between the power supply (vdd) and the circuit or a n-channel device between the circuit and ground (vss). Also combined schemes for special applications have been proposed [6,7,9,10].

By turning off the cut-off switch the total leakage (subthreshold- as well as gate-tunneling-leakage) and therefore the standby-power dissipation can be reduced significantly. When the switch is turned on, the cut-off transistor appears as a parasitic resistance in the power supply line. As the whole active current passes this resistance, it causes a voltage drop across the switch device and therefore the effective supply voltage of the logic is reduced. The dependence of the signal propagation delay on the supply and threshold voltage  $v_{th}$  is described by [1]

$$d \propto \frac{v_{dd}}{(v_{dd} - v_{th})^\alpha} \quad (1)$$

where  $\alpha$  is a technology dependent parameter with values between 1 and 2. Thus the circuit delay depends very sensitively on the supply voltage and even slight voltage drops across the switch device cause severe delay degradation. If the power supply of a static CMOS circuit is not switched it is sufficient for

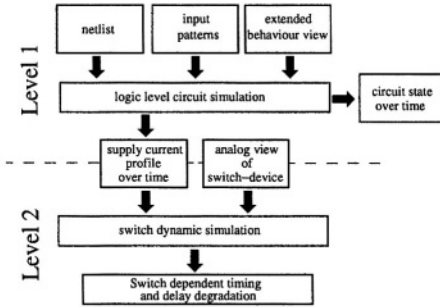
timing analysis to investigate the worst case delay of the critical paths. The switching activity of other paths is not important, as long as the critical paths are not affected. If the power supply is switched by a cut-off switch with a finite on-resistance this statement is not valid any more: As mentioned above the currents caused by the single switching events in the logic constitute the switch current. This current causes a voltage drop across the switch device and therefore reduces the effective supply voltage. Thus each single switching event affects the delay of the whole circuit. This delay is not only determined by the gate delays of the circuit elements within the critical paths but also by the current of the gates switching while the signal propagates through these paths. It is even possible that the critical path is altered if the supply voltage is switched. Consequently it is not a trivial task to dimension the switch device: A large cut-off switch results in unacceptable area consumption and a weak leakage reduction ratio (LRR) of the circuit block switch-off scheme (CBSO). However, a small switch induces a large delay degradation and therefore violation of the timing specification. Additionally the energy overhead due to switching a circuit block on and off strongly depends on the dimensions of the switch device. Thus an unnecessary large switch device increases the minimum time for which it is beneficial to switch-off an unused circuit block. In order to investigate the delay degradation resulting from certain switch dimensions it is necessary to find and investigate the critical data transitions at the inputs [2]. Analog circuit simulation (SPICE) would give the most accurate results, however the amount of input transitions which have to be investigated is typically very large. In order to achieve acceptable simulation time even for many input transitions, we propose a two level compact simulation approach for the evaluation of the switch and data dependent delay degradation. The paper is organized as follows: In section 2 the separation of the simulation task into two separate stages is motivated. Section 3 and 4 describe the theory of these two simulation stages and section 5 shows the application of the methodology to a circuit example.

## 2 Two Level Compact Simulation

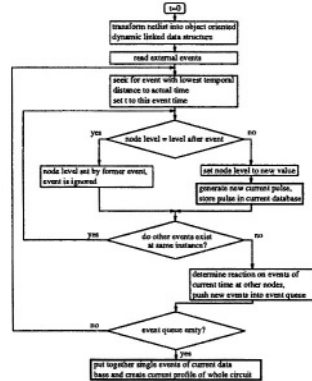
During the design process of power switched circuits, it is important to find the critical input transition. Because of the nonlinear voltage-delay relation in eq. 1 it is also important to investigate the influence of different switch devices and dimensions. Therefore the design process is split up into the following steps:

- find critical transitions
- investigate influence of switch device type and find optimal dimensions

As mentioned in section 1, the critical transition is determined by the activation of a time critical path combined with a maximum switching current. It shall be assumed that the characteristic features of the supply current profile do not depend on the switch device: Of course, the insertion of a switch device causes delay degradation and therefore the current profile is stretched, but the logic function of the circuit and therefore the node transitions are the same. Hence



**Fig. 1.** Principle of the proposed two-level compact simulation methodology. As there is a huge number of simulations to be done during the design of a power switch scheme, the methodology provides reuse of simulation results (extended behaviour view of gates), fast simulation (logic level), result filtering (only the profiles of critical transitions have to be passed to the second level) and fast estimation of the influence of various switches (curve shaping of supply current profile).



**Fig. 2.** Flow chart of the algorithm realizing the first simulation step. After the discrete event-driven logic-level simulation the algorithm creates an estimation of the supply current profile depending on the timing information acquired during the simulation. Deviations with respect to state-of-art logic level simulation are highlighted. These additional steps allow the extraction of physical information (current profile) out of logic level simulation.

it can be assumed that the total charge which is injected over the whole system cycle into the switched power supply line is constant. (Actually this charge depends on the swing of the virtual power rail. This effect is considered in sec. 4.)

In the two level approach presented in this paper, the circuit is analysed without any cut-off switch at first. Due to this analysis possibly time critical transitions can be identified. Candidates are transitions which cause long signal propagation and a large total charge being switched during the transition. By using logic level compact simulation it is possible to check a huge number of input transitions within a limited time. After critical transitions have been found, the supply current profiles of these transitions are passed to the second simulator stage. In this stage the current profiles are reshaped depending on the electrical properties of the switch device. Hence it is possible to investigate the delay degradation caused by many different types of switch devices accurately without analog simulation of the whole (normally very large) logic circuit. The principle of the two level approach is given in Fig. 1. The compact simulation presented here assumes that the considered circuit is large enough so that the current quantisation due to discrete switching events is averaged out in the supply current profile of the ensemble of all gates. Further it is assumed in the second stage that the current which is injected into the virtual power rail during a switching event of a certain gate is linearly dependent of the effective supply voltage, as this current is mainly caused by charging or discharging the load capacitance of the gate



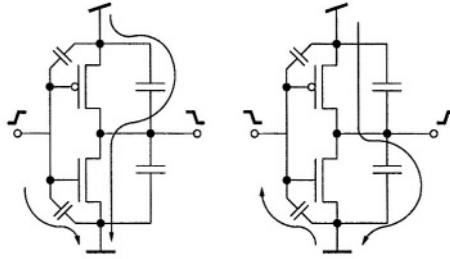
to the effective supply voltage. Based on these assumptions it is possible to separate the circuit simulation into two distinct steps: During the first step the supply current profile of the system without power switch is determined by SPICE simulation, silicon measurement or gate level simulation as preferred in this paper. In the second stage, the resulting current profile is reshaped. Thus the dynamic of the circuit with power switch can be estimated accurately. The two simulation steps and the required input data are described during the next sections.

### 3 Logic Level Circuit Simulation

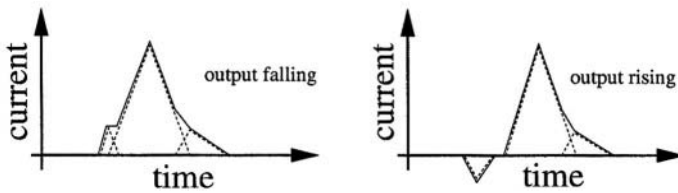
The two step methodology proposed in this paper requires the knowledge of the supply current profile of the circuit without switch transistor as input data of the second simulation stage. Of course, this current profile can be acquired by silicon measurements or analog circuit simulation. However, the target of the methodology discussed here is to investigate a huge number of input transitions and therefore the acquisition time of the current profile of a single transition has to be very small. To achieve minimum simulation time we use an event-driven gate-level simulation. This means that only the logic structure of the circuit is considered. The values of supply currents which are the output of this simulator stage are not relevant for the intrinsic operation of the simulator. The simulation works event-driven, hence there is no fixed time step. Thus both good timing accuracy and fast processing of the simulation task are guaranteed. When the algorithm recognizes an event at a certain node, it checks whether there is a logic reaction on this event at other nodes. If this is the case the event characteristic and event time of this reaction are calculated and the new event is pushed into the event queue. Then the algorithm picks the event with the smallest temporal distance to the current time out of the queue. In order to produce a realistic estimation of the switching events and therefore of the supply current profile, an accurate timing model is important: The delay of a CMOS gate depends on the slope of the input signal, the output loading, but also on which one of the inputs triggers the output transition. By applying look-up tables it is possible to consider the different gate responses depending on which input triggers the switching event, the load capacitance and the slope of the triggering event without wasting much simulation time. Therefore accurate timing can be combined with fast gate-level logic simulation. The gates used in the circuit have to be characterized once by analog simulation, in order to build up the look-up tables:

$$\begin{bmatrix} \text{number of input} \\ \text{input slope} \\ \text{load capacitance} \end{bmatrix} \rightarrow \begin{bmatrix} \text{rising edge delay} \\ \text{falling edge delay} \\ \text{rising edge output slope} \\ \text{falling edge output slope} \end{bmatrix} \quad (2)$$

Finally the operation of the first simulator stage can be described by the flow chart in Fig 2. Besides the triggering of other switching events, each event is combined with a supply current. As Fig. 3 shows, both the rising as well as the falling output event causes a supply current. The calculation of these



**Fig. 3.** Currents injected into the supply line due to different switching events. As one can see, both rising and falling output transitions cause supply currents.



**Fig. 4.** The shape of the current profile caused by an elementary switching event can be modeled by a superposition of triangles. The current profiles are shaped depending on the input and output slopes and the load capacitance of the considered gate.

supply currents is not necessary for the operation of the logic level simulator, but serves as output. In this compact simulation approach, we assume that the charging current of the internal capacities occurs during the output transition of the gate. The times of the single transition events are acquired during the gate level simulations, thus the current profile of the whole circuit can be composed of the single switching currents plus the state dependent leakage currents. With respect to speed and simplicity the current of a single switching event is composed of triangular current profiles as shown in Fig 4. The circuit parameters shaping these current profiles are the input and output slope as well as the load. It should be mentioned that the first simulator stage decomposes complex gates internally into single stage gates in order to investigate transitions of internal nodes within the multi-stage gates.

## 4 Switch Dynamic Simulation

After the supply current profile (vss-current if a switch between the circuit and vss is used or vdd-current if the switch is located between vdd and the circuit) is determined by one of the strategies described above, the second simulator level can estimate the dynamic behaviour of the whole circuit in conjunction with a cut-off switch. Therefore we regard the supply current profile and assume this profile consist of many single switching events i.e. many single charge or discharge processes of small internal capacities. Hence the current profile  $i(t)$  can be decomposed corresponding to

$$i(t) = \frac{dq(t)}{dt} = \zeta(t)vdd(t) \tag{3}$$

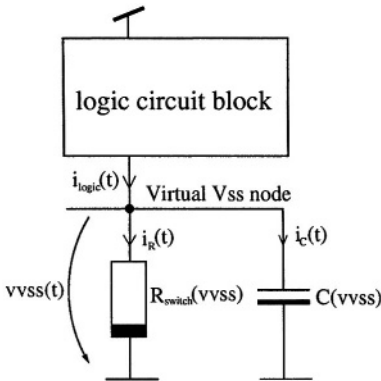
where  $\zeta(t)$  is given by  $\zeta(t) := \frac{i(t)}{vdd(t)}$ .  $\zeta(t)$  is an effective complex conductivity of the logic block at time  $t$  describing charging as well as leakage currents at the considered instance  $t$ . Thus the charge passing through the power supply lines between  $t$  and  $t + dt$  can be expressed by

$$dq = \zeta(t)dt \cdot vdd(t) \tag{4}$$

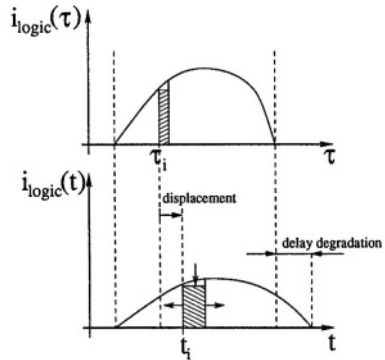
The supply voltage dependence of the gate delay  $d(vdd)$  in the considered technology can be either calculated with simple analytic models (cf. eq. 1) or more accurately by analog simulation. Both approaches result in a function  $d(vdd)$  which can be normalized to the delay at nominal supply voltage  $vdd_{nom}$  resulting in the delay degradation coefficient  $\delta(vdd)$ :

$$\delta(vdd) := \frac{d(vdd)}{d(vdd_{nom})} \tag{5}$$

Both quantities  $\zeta(t)$  and  $\delta(vdd)$  are used during the second simulation step. In this step a cut-off switch is added to the circuit and the current equation of the switched (virtual) power line is investigated. As shown in Fig. 5 for a cut-off device between the logic block and ground (vss) there exist three current components: The current  $i_{logic}(t)$  is the ground current flowing out of the logic block into the vss node.  $i_R(t)$  is the current through the nonlinear on-resistance of the switch device and  $i_c(t)$  is the displacement current through the nonlinear



**Fig. 5.** Only the virtual power supply node is regarded in the second simulator stage. The nonlinear resistance and the nonlinear capacitance are simulated separately and the results are stored in lookup tables.



**Fig. 6.** The second simulator stage reshapes given supply current profiles by stretching the time variable and by damping the amplitude. An infinitesimal element of the profile experiences a displacement and an elongation resulting in signal delay degradation of the whole circuit.

capacitance of the virtual power-rail versus ground. This capacitance consists of the intrinsic capacitance of the switch device, the wiring capacitance of the virtual ground node and the capacitance seen from the virtual power supply into the logic block. Therewith the current equation is

$$\frac{dQ}{d\,vvss}vvss + I(vvss(t)) = i_{Logic}(t) \tag{6}$$

$$q_c = Q(vvss) \tag{7}$$

$$i_R = I(vvss) \tag{8}$$

where  $Q(vvss)$  describes the voltage dependence of the charge on the nonlinear capacitance  $C(vvss)$  and  $I(vvss)$  is the current through the nonlinear switch resistance. Assume that the current profile  $i_{Logic}(\tau)$  for the circuit without power switch is given as input to the second simulator stage where  $\tau$  is the time variable for operation of the system without cut-off switch. Next this current profile will be transformed into the current profile of the circuit with cut-off switch being described by the time variable  $t$ . Therefore the current profile  $i_{Logic}(\tau)$  is partitioned into infinitesimal slices as shown in Fig. 6. The charge that is injected into the ground node within the time interval  $[\tau, \tau + d\tau]$  can be expressed by

$$dq(\tau) = i(\tau)d\tau = \zeta(\tau)vdd\,d\tau \tag{9}$$

In the power switched circuit, this charge is reduced because the internal capacities of the switching gates just charge to the effective supply voltage  $vdd - vvss(t)$  where  $vvss$  is the potential of the virtual power rail. Additionally the time interval  $dt$  during which the charge  $dq$  is injected into the virtual supply line is stretched due to the voltage dependent delay degradation. Both these transformations can be combined to calculate the charge  $dq(t)$  that is injected into the virtual ground node during the corresponding time interval  $[t(\tau); t(\tau) + dt(\tau)]$  in the other time domain.

$$dq(t) = \zeta(\tau(t)) \cdot [vdd - vvss(t)]d\tau(t) \tag{10}$$

$$dt = \delta(vdd - vvss(t))d\tau \tag{11}$$

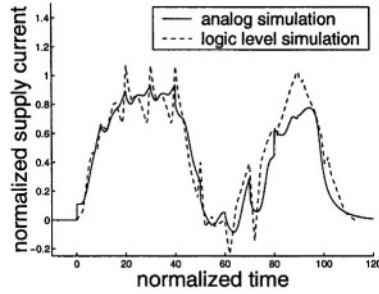
---


$$dq(t) = \zeta(\tau(t)) [vdd - vvss(t)] \frac{d\tau}{\delta(vdd - vvss(t))} \tag{12}$$

Note that this charge is distributed over a larger time interval  $dt$ , thus the considered infinitesimal area element under the current profile is further damped. Using expression 12 the current equation of the virtual power rail is given. Together with the differential relation between the two time domains  $t$  and  $\tau$  (time for system with or without switch) the following system of differential equations has to be solved:

$$\begin{aligned} & \frac{dQ}{d\,vvss}vvss(t) + I(vvss(t)) = \\ & = \zeta(\tau(t))\delta^{-1}(vdd - vvss(t)) [vdd - vvss(t)] \end{aligned} \tag{13}$$

$$dt = \delta(vvss)d\tau \tag{14}$$

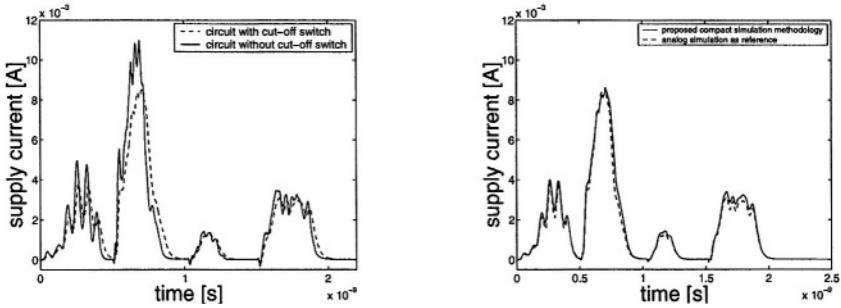


**Fig. 7.** Analog and logic level simulation of a small fraction of six gates of the the Kogge-Stone Adder. As the number of gates is small, the difference of the two approaches can be seen, but if the total charge transfer of a single switching event is modeled accurately this difference vanishes with increasing number of gates due to averaging effects.

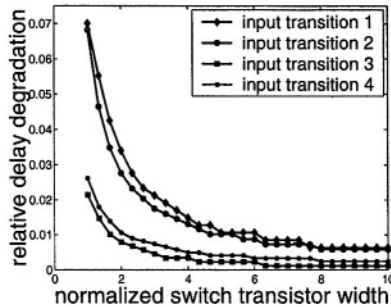
This problem can be solved using a simple numerical integration algorithm. In order to get a good accuracy compared to solving the whole simulation problem with an analog simulator, e.g. SPICE, the nonlinear functions  $i_R = I(vvss)$  and  $q_c = Q(vvss)$  have to be known. However, these functions can easily be acquired from an analog simulation of the switch and then stored in look-up tables. It should be mentioned that these simulations have to be done only once. When the characterisation of the switch and the capacitance is done, many input transitions can be investigated with the compact simulation tool.

## 5 Design Example

The benefits of the two level compact simulation approach are tested and evaluated by the simulation of a 32-bit Kogge Stone adder. The congruence of the logic level simulation with respect to analog circuit simulation depends on the granularity of the look-up tables which describe the gate dynamic. On the other hand, the shape of the current profile modeling a single switching event determines the deviation of the two results even if the timing accuracy of the logic level simulation is perfect. Therefore, if a circuit is analysed which consists of only a few gates, the congruity of the two simulations is poor. However, as the methodology has been developed for the investigation of huge circuits, the current shape of a single elementary switching event is of minor importance, as long as the amount of charge which is transferred due to this switching event is modeled accurately. Reason: If there are many subsequent switching events and many parallel events, eventually with slight timing shifts, the current of a single event is small with respect to the total current. Therefore certain inaccuracies within the shape of a single current pulse will be averaged due to the superposition of all events. Fig. 7 shows a supply current profile of a small subcircuit within a 32-bit Kogge-Stone Adder consisting of 6 gates. As mentioned above, the triangular shape of the elementary pulses causes a deviation with respect to the real current profile, but this error disappears if the whole circuit is observed.



**Fig. 8.** Supply current profile with and without cut-off switch (left). The stretching of the current profile and the resulting delay degradation depend on the total charge which is injected into the virtual power rail during the transition. The analog simulation and the proposed compact simulation methodology show good congruency (right).



**Fig. 9.** Relative degradation of the signal delay of a 32-bit Kogge-Stone Adder for four arbitrary input transitions depending on the width of the cut-off switch. The switch width is normalized to a unit switch.

Next the accuracy of the second simulator stage has been examined by comparing the shaped current profiles of the circuit simulation and our profile shaping approach. Fig. 8 shows the current profile of the Kogge-Stone adder with and without cut-off switch for a couple of input transitions. As predicted, the supply current profile of the power switched circuit is stretched over time. The second graph in Fig. 8 shows the results of the reshaping algorithm of the second simulator stage. The deviation between the current profiles acquired by analog simulation and our fast curve-shaping algorithm is very small, although only a system of two differential equations had to be solved instead of the complete deq-system of the circuit. Finally we used our methodology to analyse the delay degradation of different input transitions for multiple switch dimensions (Fig. 9). The spread of this delay degradation shows the input transition dependent reaction to the insertion of the power-switch. Therefore one can see the importance of the identification of critical transitions: If the switch dimension was based on a subcritical profile, the circuit would fail when critical transitions occur.

## 6 Conclusion

A two level compact simulation approach for analysis of the cut-off switch in power switched circuits is proposed. The simulation task is split into two levels. In the first stage the switch independent dynamic of the circuit block is analysed. In order to reduce the simulation time and therefore enable the examination of many different input transitions, we used an event driven gate level simulation with current profile estimation. This simulation has to be done only once even if several switch scenarios have to be examined. In the second simulator stage, the supply current profiles produced by the first stage are reshaped in order to consider the influence of the cut-off switch. As this stage only has to solve a simple system of two differential equations it is very fast and therefore many different switch devices and dimensions can be investigated. The congruency of our simulation approach and a complete analog simulation has been shown and the methodology has been applied to the switch evaluation of a test circuit.

## References

1. Sakurai, T., Newton, A. R.: Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. *JSSC*, Vol. 25, No. 2, (1990) 584–594
2. Kao, J., Chandrakasan, A., Antoniadis, D.: Transistor Sizing Issues and Tool For Multi-Threshold CMOS Technology. *Design Automation Conference* (1997)
3. Krishnamurthy, R. K., Alvandpour, A., Mathew, S., Anders, M., De, V., Borak, S.: High-performance, Low-power, and Leakage-tolerance Challenges for Sub-70nm Microprocessor Circuits. *ESSCIRC* (2002)
4. Das, K. K., Joshi, R. V., Chuang, A. T., Cook, P. W., Brown, R. B.: New Digital Circuit Techniques for Total Standby Leakage Reduction in Nano-Scale SOI Technology. *ESSCIRC* (2003)
5. Mutoh, S., Douseki, T., Matsuya, Y., Aoki, T., Shigematsu, S., Yamada, J.: 1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS. *JSSC* Vol. 30, No. 8, (1995) 847–854
6. Kawaguchi, H., Nose, K., Sakurai, T.: A Super Cut-Off CMOS (SCCMOS) Scheme for 0.5-V Supply Voltage with Picoampere Stand-By Current. *JSSC* Vol. 35, No. 10, (2000) 1498–1501
7. Min, K., Kawaguchi, H., Sakurai, T.: ZigZag Super Cut-off CMOS (ZSCCMOS) Block Activation with Self-Adaptive Voltage Level Controller: An Alternative to Clock Gating Scheme in Leakage Dominant Era. *ISSCC* (2003)
8. Meer, P. R., Staveren, A.: New standby-current reduction technique for deep sub-micron VLSI CMOS circuits: Smart Series Switch. *ESSCIRC* (2002)
9. Henzler, S., Koban, M., Berthold, J., Georgakos, G., Schmitt-Landsiedel, D.: Design Aspects and Technological Scaling Limits of ZigZag Circuit Block Switch-Off Schemes. *IFIP VLSI-SOC* (2003)
10. Henzler, S., Georgakos, G., Berthold, J., Schmitt-Landsiedel, D.: A Fast Power-Efficient Circuit-Block Switch-Off Scheme. *IEE Electronics Letters* Vol. 40, No. 2, (2004) 103–104

# A Generic Timing Mechanism for Using the APPLES Gate-Level Simulator in a Mixed-Level Simulation Environment

Alexander Maili<sup>1</sup>, Damian Dalton<sup>1</sup>, and Christian Steger<sup>2</sup>

<sup>1</sup> Department of Computer Science, UCD,  
Belfield, Dublin 4, Ireland  
{alexander.maili,damian.dalton}@ucd.ie

<sup>2</sup> Institute for Technical Informatics  
Graz University of Technology  
8010 Graz, Austria  
steger@iti.tugraz.at

**Abstract.** In this paper, we describe a generic timing mechanism, which allows building mixed-level simulation environments that facilitate timing closure for their gate-level modules whilst simulating a whole System-on-Chip (SoC) made of modules at different levels of abstraction. The APPLES gate level accelerator provides fast timing-accurate simulation of gate-level designs. But to enable its use for large SoC-designs, a generic timing mechanism must be developed in order to use the APPLES processor in a mixed-level environment together with higher-level simulation engines. The environment we present here makes use of a universal time mechanism and a flexible client-server implementation to enable a generic and expandable system for mixed-level, mixed-language timing simulation.

## 1 Introduction

As capacities of current ASIC and FPGA technologies enable designs with circuit-sizes of 10 million gates and more, system-on-chip designs are becoming more prevalent nowadays. Since such designs consist of modules at different levels of abstraction, like 3<sup>rd</sup> party IP at behavioral or transaction level or self-designed parts at gate-level with timing information, an efficient and accurate mixed-level timing simulation is mandatory. The APPLES processor presented in [1] allows fast and timing accurate simulation of gate-level designs. A generic timing mechanism that can cope with all levels of abstraction used in SoC-design could enable the use of APPLES in a mixed-level environment for fast and accurate timing simulation of SoC-designs. To date there is no comprehensive timing mechanism that can be used for that purpose.

In this paper we analyze the use of time in simulation at different levels of abstraction and present a generic timing mechanism. We will also show a mixed-level simulation environment that uses the APPLES gate-level accelerator in a case study.



## 2 Related Work

Substantial work has been done in the field of mixed-level simulation and hardware-software cosimulation.

[2] provides an overview of parallel logic simulation in general.

[3] and [4] present a mixed-level cosimulation environment that allows cycle accurate simulation of SoC-designs. They describe how the interfaces between the higher and lower levels of abstraction can be built in order to allow communication between various modules at different abstraction-levels.

In [5], a mechanism for parallel and distributed simulation of VHDL models is presented. [6] shows an approach for optimistic time-based cosimulation.

[7] presents a method of how to integrate SystemC models and hardware based verification into a single environment. A SystemC-simulator is connected to an FPGA based emulator to make use of the acceleration the emulator provides for gate-level designs whilst simulating a mixed-level SoC-design. But since emulators only allow functional verification without timing, such a system lacks in providing appropriate timing information.

Companies like Mentor Graphics with its Seamless CVE environment or Cadence with the Incisive Platform provide solutions for mixed-level simulation. But none of these allow full timing simulation of gate-level modules while simulating a whole mixed-level SoC-design.

None of the existing mixed-level simulation environments support the integration of a gate-level simulation accelerator that enables timing accurate simulation of gate-level parts. We present a timing mechanism that makes use of a so-called universal time to make such systems possible.

In [8], Ghosh et. al. gives an overview on the principles of modeling time in different hardware description languages. Our timing mechanism is sufficiently comprehensive to be applicable to all these levels of timing abstraction.

## 3 The APPLES Gate-Level Accelerator

The APPLES processor is a gate-level logic simulator that can significantly speed up simulation of gate-level designs whilst maintaining full timing closure and SDF back-annotation.[1] As such it is a good option for simulating large gate-level designs. But for SoC-designs only parts may be available at gate-level due to IP security issues. Furthermore, parts of the design may have been previously implemented and tested at gate-level. Consequently it is sufficient to simulate those components at a higher level. The APPLES processor can still be used to accelerate simulation at gate-level. But currently it is not possible to simulate the system across different levels of abstraction.

In this paper we present a generic timing mechanism, which will enable the use of a gate-level accelerator in a heterogeneous mixed-level simulation environment to get fast timing accurate simulation results without losing the ability of simulating the system as a whole.

## 4 A Generic Timing Mechanism

### 4.1 Time at Different Levels of Abstraction

We review how time is used at a dedicated abstraction level starting the analysis at the lowest level and then proceeding towards system and software level.

**Transistor Level.** At this level a circuit is described by means of differential equations. Time is handled as continuous time. The analogue simulators solve these differential equations and update the signal values at infinitesimal delta time steps. The timing interface at this level could be defined as signal value at delta-time.

**Gate Level.** At gate-level circuits are modeled as netlists of gates and interconnects. Delays are annotated to gates and interconnects in order to model the physical behavior of the circuit. Time is no longer a continuous stream of infinitesimal delta time steps. Depending on the actual time-resolution we are talking of discrete time steps at a defined granularity. Simulators at this level and higher are rather event-driven opposed to the solving of differential equations at transistor level. As such the timing interface can be identified as events (meaning change in signal values) at dedicated time-steps.

**Register-Transfer Level.** At this level time can either be modeled directly in terms of delays for signal assignments or indirectly by waiting for upcoming events. An example would be an “@posedge CLK”-statement in Verilog, which would cause an event to happen when the CLK-signal transitions from low to high. But from an interface point of view these two notions of time do not make a difference, since both produce events at dedicated time steps. Thus the timing interface at this abstraction level is also defined as events at discrete time-steps.

**Behavioral or Functional Level.** At functional level we have to distinguish between two types of models: timed and untimed functional models. Untimed functional models describe an algorithm by its behavior only. No timing information is given by the designer. Therefore simulation-time never advances during a simulation run. On the other hand there are timed functional models, which include additional timing information to allow advancing simulation-time whilst the designed algorithm is processed.

As the main focus of this paper is to generate a timing mechanism that allows mixed-level timing simulation of SoCs, untimed functional models are of no importance here. It would not make much sense to simulate a system and retrieve timing information when some modules of the system are described as untimed functional models. These parts must be refined to timed functional models first.

For timed functional models the timing interface can again be defined as events at discrete time-steps, since processing the algorithm during a simulation run causes signals to change their values at defined time-steps.

**Transaction Level.** At this level designs are modeled in terms of modules that interact by making transactions. Such a transaction could either be writing some data to a

bus or loading commands from a queue, to give some example. Time at this level of abstraction is usually just used as an instrument to adhere to the fundamental law of causality. In other words a specific event or change in state shall not occur before a specified transaction is finished. But when we add some timing information about the duration of every transaction, it is possible to get events at specified time stamps as the output of transaction-level simulation. These events at discrete time-stamps can then be defined as the timing interface here.

**Software Level.** At software level instructions or operating system calls executed on a specific processor would be the target of simulation. These instructions have a defined execution time in terms of clock cycles. The timing interface here would be events (finished execution of an instruction) at a number of clock cycles.

## 4.2 The Generic Timing Interface

When comparing the different levels of abstraction we see that there is no real difference for the timing interfaces of gate-level and register-transfer level. They are both defined as signal-value changes at discrete time steps. At transistor level these value changes can happen at infinitesimal time steps. But if we only take snapshots of the values at discrete time-steps, the timing interface is the same again.

When looking at software or transaction level, we see that we have events at time stamps or clock cycles as the timing interface. The clock cycles can easily be converted to time-stamps, given a dedicated clock frequency. Having applied this conversion, the timing interface for all levels of abstraction can be defined as events at discrete time-steps. The only difference is the resolution of discrete time, which will be much finer for gate-level designs than for transaction or software level parts.

## 4.3 Universal Time

These different time resolutions must be synchronized in order to allow simulating a system that is described at different abstraction-levels. The answer to this is called *Universal Time*. It is a time measure that can cover several local discrete times at different resolutions. Figure 1 shows how three logical processes can be synchronized using universal time. As long as the resolutions are multiples of each other the universal time can be defined as the local time with the finest resolution. Where this is not the case, universal time granularity must be much finer. Its resolution is then defined as the greatest common divisor of all local time resolutions. As can be seen in figure 2, this can result in much slower simulation since the universal time must be handled at a much finer resolution and more synchronization steps are necessary. The fact that an event from one logical process cannot be received by another logical process at exactly the same universal time is not a major problem. Time is only used to adhere to the causality constraint. Therefore the only essential requirement is that such an event is always received at the very next possible universal time step that can be modeled by the target's local time.

Nevertheless, the local time resolutions should be multiples of each other, whenever it is possible, in order to significantly enhance simulation performance.

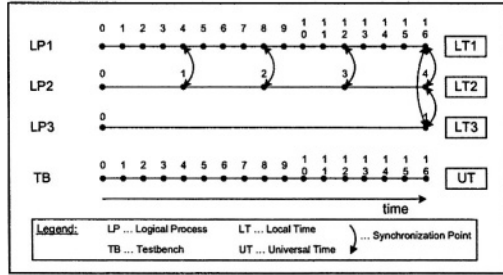


Fig. 1. Synchronization with Universal Time (Resolutions are multiples of each other)

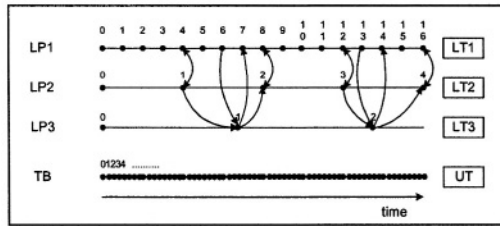


Fig. 2. Synchronization with Universal Time (Resolutions are no multiples of each other)

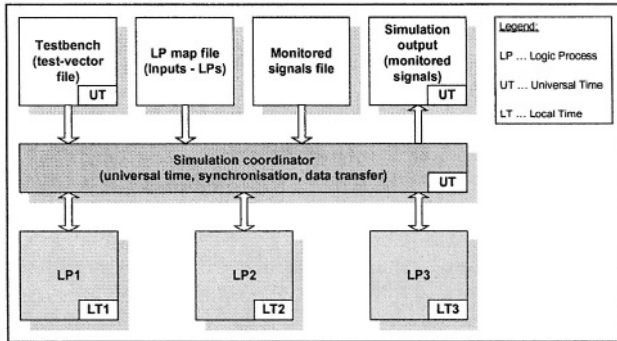


Fig. 3. Mixed-level Simulation Environment

## 5 Case Study – A Mixed-Level Simulation Environment Facilitating Timing Closure

Figure 3 depicts the structure of a possible mixed-level simulation environment. The system consists of three different simulation clients, the logical processes. LP1 is the APPLES processor to simulate the gate-level parts of the design. LP2 is a Verilog simulator (in our case ModelSim) to simulate RTL and behavioral parts. The third LP in the picture is shown to underline the flexibility of the system. It could either be an additional Verilog simulator, a SystemC simulator or any other simulation engine capable of providing an interface to its top-level. The testbench is another logical process, which normally operates at universal time granularity. A central unit, the so-

called simulation coordinator, is responsible for synchronization and handling of the universal time. Some additional files are necessary to provide information about the system structure.

Such a system can be expanded to more logical processes without much effort. The different LPs are ignorant about each other. They just simulate their part of the design. Stimuli are received via a socket connection and value changes on top-level outputs are sent via this socket connection. The central simulation coordinator analyses the messages, converts the local time and forwards the messages to the target LP. A client interface software must be built for each type of simulation engine used as a client. At the moment we have an interface to Verilog simulators via the VPI standard and an interface to the APPLES processor via a special API.

The actual system is implemented using C++ on a Windows platform. Windows sockets are used for communication to allow a distributed simulation over a network.

## 5.1 Central Simulation Coordinator and Universal Time Mechanism

This is the main part of the software. It consists of a socket server that connects and initializes all clients and starts several threads to allow reading and writing of messages via the socket connection. Figure 4 provides an overview of the structure of the simulation coordinator.

The messages that are sent between the clients contain the client-id, signal-name, timestamp and the new value.

Figure 5 shows how universal time is handled in the simulation coordinator.

Every simulation client has a thread in the central simulation coordinator. A client thread is allowed to read and write messages as long as the universal time of a message is equal to the current universal time. If this happens the client sends a request to increment universal time and waits for a notification that universal time has been incremented. After this notification has arrived, the client can continue simulation for the next universal time step.

The request and notification signals are built in a ring-topology to avoid deadlocks and ensure that all clients keep synchronized according to universal time.

## 5.2 Client Interfaces

As mentioned above a client interface module must be built for every type of client. For the current system, three types of clients are necessary, a VPI client to interface to all types of Verilog simulators, an APPLES client to interface to the APPLES processor and a testbench client to apply stimuli. A client class to establish the socket-connection and a message class used to send data are implemented to ensure that the server does not need to make a difference whether a Verilog client, an APPLES-client or a testbench-client is connected.

**Verilog VPI Client.** This client is used to establish connections to Verilog simulators. The VPI interface was chosen because it is a standardised interface included in the Verilog LRM. This allows using the client for any standard Verilog simulator, like ModelSim from Mentor Graphics or NC-Verilog from Cadence Design Systems.

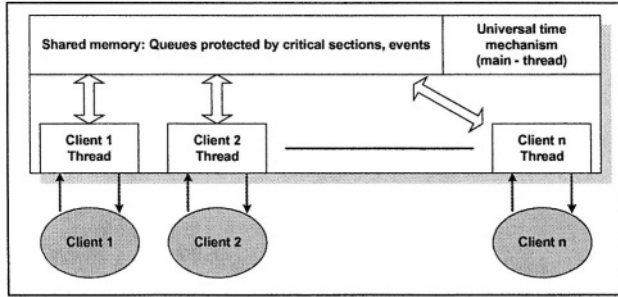


Fig. 4. Central Simulation Coordinator

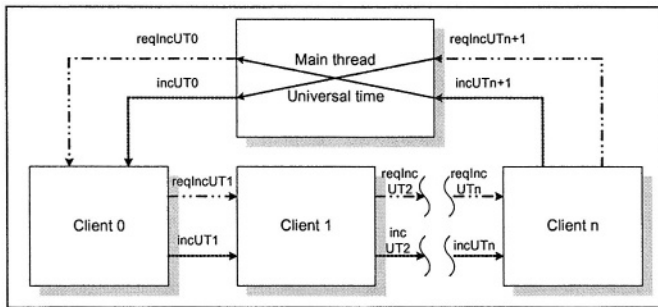


Fig. 5. Universal-Time Mechanism

The VPI client is responsible for connecting to the central simulation coordinator and providing the server with information on the simulation client. After initialisation is complete, it starts sending messages about events on all top-level signals to the server. It also receives messages about value changes on its inputs from the server and applies the new values to the simulator. In order to keep changes to the simulated circuit at a minimum, the VPI client is built so that it only needs to be loaded when invoking the Verilog simulator. No changes in the Verilog source of the simulated circuit are necessary.

The following VPI callback-functions are used here:

*Start of Simulation Callback.* This callback function is invoked at the beginning of a simulation before any event is executed. It initializes the client and connects to the simulation coordinator.

*Send Message Callback.* This function is called for each value change on an output port. It sends a message to the server containing the port name, the timestamp and the new logic value.

*Receive Message Callback.* This function is called every simulation cycle. It reads messages from the socket and applies the new logic values to the according nets. An end of block message read from the socket indicates that there are no more messages for the current time stamp. The callback function re-registers itself for the next time-stamp and returns.

**APPLES Client.** This client is creating an interface between the central simulation coordinator and the APPLES processor, which is an FPGA-based system on a PCI-card. This is achieved by using two parallel threads that send/receive message blocks using the APPLES-API, convert the data format and send/receive the messages to the server via the socket-connection.

**Testbench.** From the server's point of view the testbench is just another logical process like the VPI-client or the APPLES-client. For a behavioral testbench written in Verilog, the VPI-client can be used. Additional clients are needed if SystemC- or "e"-testbenches should be supported. Research is currently ongoing on that topic.

For our case study we take a different approach and use a stimuli-file instead of a behavioral testbench. This file lists all values that shall be applied to the system at the desired universal time-steps. To facilitate such test-vector files, the so-called file-client is implemented. It is used for reading the test-vector files and applying the stimuli to the system as well as for writing an output file that contains information on all monitored signal-value changes during a simulation run. As the other clients, it only talks to the simulation coordinator.

### 5.3 CPU-Time Distribution

In order to find out how the message-load influences the CPU-time distribution between the parts of the software, we created a dummy-system using an Apples-client and a VPI-client, each simulating one of two connected C17G circuits. A File-client is used to fire stimuli at a specified frequency. The system was simulated for 10000 universal time units with different stimuli periods. Figure 6 depicts the measured results for stimuli periods of 5, 100, 1000 and 5000 time-units. The small size of the used circuit does not matter here, since the message-load is generated by the stimuli-file and it is easier to run these experiments using a small circuit.

We see that the File-client consumes only a fraction of the CPU-time compared to the other processes. The simulation coordinator containing the universal time mechanism always consumes a considerable amount of CPU-time. This is due to the fact that the universal time is calculated for every single time-step even if there is no value-update between the clients at a given time. The proportion of time consumed by the Apples-client increases with the number of stimuli, while the VPI-client's part becomes less. This can be explained by the fact that actual simulation time on ModelSim does not change as drastically with the amount of messages than on Apples, because the messages do not need to be sent to the Apples-hardware. This can be time-consuming. If we use another VPI-client instead of the Apples-client, the CPU-time is distributed evenly between them.

In a second experiment, we looked how CPU-time distribution changes with the number of clients. We took the same system as above using 1, 2 or 3 connected clients and the stimuli-file with a constant message load. Simulation time is 10000 universal time units. Figure 8 shows that CPU-time is spread nearly evenly between the simulator-clients. The file-client consumes only a negligible portion. The simulation coordinator, however, is always the dominant process. Thus, the focus must be on enhancing the simulation coordinator.

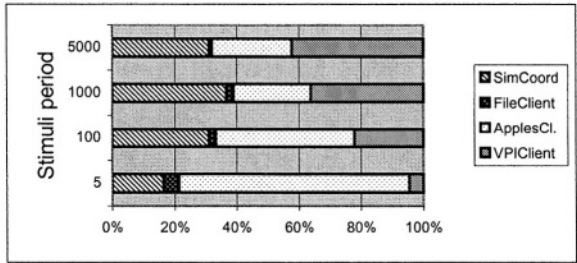


Fig. 6. CPU-Time Distribution for Different Stimuli-Periods

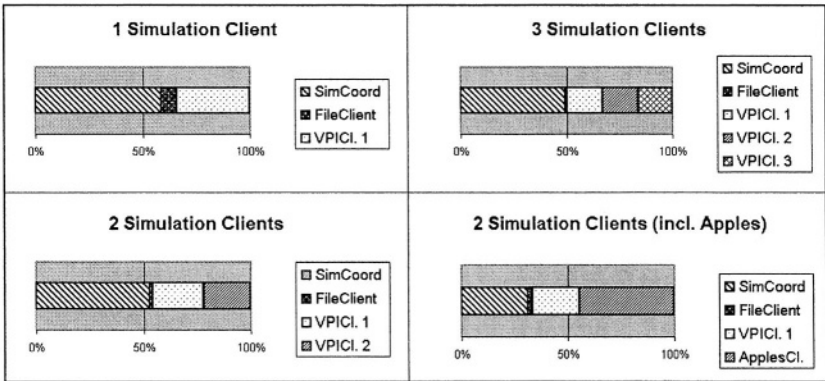


Fig. 7. CPU-Time Distribution for Different Numbers of Simulation-Clients

## 6 Conclusion and Future Work

A generic timing mechanism has been presented that allows timing simulation of a system built of different modules at different levels of abstraction. The Universal-Time mechanism is managed by a central unit, the simulation coordinator. In a case study we presented the implementation of a mixed-level simulation system that makes use of this timing mechanism to generate a mixed-level simulation environment including the APPLES processor for accelerating gate-level simulation. Since the system is built in a generic and modular way, it can be used to link many different simulators together and can be easily expanded by creating new types of client interfaces.

SoC-designers can benefit from such a tool by having the ability to make a full timing accurate simulation of the whole design whilst having only parts of it at a gate-level description. First results with simulation of the ISCAS testbench circuits show us that the mechanism works properly.

The next steps to extend the capabilities of our system are to implement VHDL and SystemC clients to support mixed-language designs. The upcoming VHPI standard should allow a similar standardized interface to all VHDL simulators as the one we now have for Verilog.



Further work will focus on extracting events from behavioral testbenches so that a time ordered list of pending input vectors could be generated in advance. This enables us to speed-up simulation by providing future stimuli at the earliest possible time.

Aspects of the Universal-Time mechanism, which are currently all implemented in software, are being transferred into hardware modules in order to enhance the performance of the system.

## References

1. Dalton, D., *A special purpose hybrid SIMD processor for logic event simulation*, In Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing (PDP'99), 1999, 74-83
2. Vee, V.Y., Hsu, W.J., *Parallel Discrete Event Simulation: A Survey*, Centre for Advanced Information Systems, SAS, Nanyang Technological University, 1999
3. Gerin, P., Yoo, S., Nicolescu, G., Jerraya, A., *Scalable and Flexible Cosimulation of SoC Designs with Heterogeneous Multi-Processor Target Architectures*, In Proceedings of the ASP-DAC, 2001, 63-68
4. Nicolescu, G., Yoo, S., Jerraya, A., *Mixed-level Cosimulation for Fine Gradual Refinement of Communication in SoC Design*, In Proceedings of Design, Automation and Test in Europe (DATE), 2001, 754-759
5. Lungeanu, D., Shi, C.J.R., *Parallel and Distributed VHDL simulation*, In Proceedings of Design, Automation and Test in Europe (DATE), 2000, 658-662
6. Yoo, S., Choi, K., *Optimistic Distributed Timed Cosimulation Based on Thread Simulation Model*, In Proceeding of the Sixth International Workshop on Hardware/Software Codesign, 1998, 71-75
7. Ramaswamy, R., Tessier, R., *The Integration of SystemC and Hardware-assisted Verification*, FPL, 2002
8. Ghosh, S., *Fundamental Principles of Modeling Timing in Hardware Description Languages*, Jour Sys. Arch. 47, 2001, 405-426

# Modeling Temporal and Spatial Power Supply Voltage Variation for Timing Analysis

Howard Chen and Daniel Ostapko

IBM Research Division, Yorktown Heights, New York 10598, USA

**Abstract.** This paper describes the full-chip power supply noise analysis methodology to accurately model the power supply voltage variation for signal integrity and timing analysis. An integrated chip and package power supply RLC model is developed to simultaneously analyze the resistive  $IR$  drop, inductive  $L\Delta I/\Delta t$  noise, and capacitive decoupling on a full-chip scale. Steady-state noise due to maximum average current and transient noise due to power ramp-up, clock gating, and  $V_{DD}$  gating, are modeled with unit-based switching activities. Minimum  $V_{DD}$ , maximum  $V_{DD}$ , and average  $V_{DD}$  are calculated at each location on the chip, and used in timing analysis to provide a better range of  $V_{DD}$  variation than the standard 10% nominal  $V_{DD}$  noise budget. Time-dependent power supply voltage waveforms at various locations, based on a specific switching sequence, are also provided to simulate clock buffers and other timing-critical circuits under common-mode and differential-mode noise.

## 1 Introduction

The advent of nanometer technology and the increasing complexity of system-on-a-chip design present significant challenges for power management and timing analysis. While the increasing power densities have a major impact on thermal reliability and performance, the decreasing power supply voltages worsen the switching current and noise problems. In addition, leakage power, which increases exponentially with the reduction of process parameters such as gate length, oxide thickness  $T_{OX}$ , and threshold voltage  $V_T$ , demands multivariate power optimization that simultaneously exploit multiple- $V_T$ , multiple- $T_{OX}$ , and multiple- $V_{DD}$  configurations coexisting in a single core.

Traditional static  $IR$ -drop sign-off methodology, which only models the average circuit power and a resistive power supply network, not only fails to predict the transient power supply noise induced by various switching activities, but also results in inaccurate timing closure due to dynamic power supply voltage fluctuations. Since the signal swings are extremely sensitive to process variation, voltage fluctuation, and temperature change, it is both important and desirable to perform timing simulation with distributed power supply waveforms that correspond to the specific switching events to accurately model the signal delay and output slew characteristics.

Table 1 shows the signal delay analysis for an 8-inverter chain under various operating conditions, using 90nm technology. The nominal temperature is assumed to be 25°C and the operating temperature is set to 105°C. The process

**Table 1.** Signal delay under different operating conditions

Process	Temp ( $^{\circ}\text{C}$ )	$V_{DD}$ (V)	0 $\rightarrow$ 1 Delay (ps)	% change	1 $\rightarrow$ 0 Delay (ps)	% Change
0.9	105	0.9	64.575	+48.6	69.908	+52.0
0.9	105	1.0	57.412	+32.1	60.548	+31.6
0.9	105	1.1	51.701	+18.9	52.939	+15.1
0.9	25	0.9	61.143	+40.7	63.559	+38.2
0.9	25	1.0	53.572	+23.2	57.687	+25.4
0.9	25	1.1	48.529	+11.6	49.944	+8.6
0.5	105	0.9	52.145	+20.0	54.695	+18.9
0.5	105	1.0	46.304	+6.5	48.868	+6.2
0.5	105	1.1	42.237	-2.8	43.370	-5.7
0.5	25	0.9	48.958	+12.6	53.114	+15.5
0.5	25	1.0	43.469	0.0	45.994	0.0
0.5	25	1.1	39.480	-9.2	40.762	-11.4
0.1	105	0.9	40.654	-6.5	42.139	-8.4
0.1	105	1.0	36.617	-15.8	37.710	-18.0
0.1	105	1.1	34.026	-21.7	34.080	-25.9
0.1	25	0.9	38.430	-11.6	41.284	-10.2
0.1	25	1.0	34.221	-21.3	35.910	-21.9
0.1	25	1.1	31.543	-27.4	31.956	-30.5

variables, which can be statistically defined, are assigned a number between 0 and 1, where 0.5 refers to the median value, 0.1 refers to the top 10 percentile, and 0.9 refers to the bottom 10 percentile. A value of 0 or 1 for this variable yields element and parameter values that are at one end or the other end of their respective ranges, which can be used for best-case or worst-case analysis. With a nominal  $V_{DD}$  of 1V and a  $\pm 10\%$   $V_{DD}$  variation under nominal temperature and nominal process variation, the delay could decrease by 9.2% in the best case or increase by 12.6% in the worst case when the signal switches from 0 to 1. Similarly, with a  $\pm 10\%$   $V_{DD}$  variation under nominal temperature and nominal process variation, the delay could decrease by 11.4% in the best case or increase by 15.5% in the worst case when the signal switches from 1 to 0. Excluding interconnect delay, it is estimated that a 10%  $V_{DD}$  drop will increase the delay by 10% and a temperature increase from  $25^{\circ}\text{C}$  to  $105^{\circ}\text{C}$  may increase the delay by 5%. In addition, process variations such as across-chip linewidth variation (ACLV) could significantly affect circuit speed and change the signal delay by more than 20%.

In this paper, we describe the methodology to model the spatial power supply voltage variation for static timing analysis, and the temporal power supply voltage variation for dynamic timing analysis. The power supply voltage waveforms at various locations on the chip can be used to simulate and measure the noise-induced clock jitters and duty cycles. In our variability-aware design methodology, the clock jitter is defined as the difference between the irregular clock period under power supply noise and the normal clock period without power supply noise. The duty cycle is defined as the percentage of time when the clock waveform is above the average value of  $V_{DD}$  and  $V_{SS}$ .

## 2 Power Supply Noise

The power supply noise ( $\Delta V$ ) is caused by the impedance ( $Z = R + j\omega L$ ) of the power supply network and the current ( $I$ ) that flows through the power supply network. In order to accurately simulate the power supply noise, we need to consider not only the resistive  $IR$  drop, but also the inductive  $L\Delta I/\Delta t$  noise.

In traditional VLSI design, the resistive  $IR$  drop occurs mostly on the chip, and the inductive  $\Delta I$  noise only occurs on the package. They are often analyzed separately and designed with their respective noise limits. Since the maximum  $\Delta I$  noise occurs during switching when the current change  $\Delta I$  is maximum, and the maximum  $IR$  drop occurs when the current  $I$  is at its peak, the worst-case  $\Delta I$  noise and worst-case  $IR$  drop do not occur at the same time. It is therefore too pessimistic to analyze the  $\Delta I$  noise and  $IR$  drop separately, and then add the two worst cases together. An integrated package-level and chip-level power bus model with detailed switching and timing information is needed to accurately analyze the  $V_{DD}$  variation over time, and properly calculate the total power supply noise  $\Delta V = IR + L\Delta I/\Delta t$ .

Fig. 1 illustrates the transient effect of  $L\Delta I/\Delta t$  noise on power supply voltage when circuits are in transition from idle power to maximum average power. Since the power supply voltage not only drops more than 10%, but also lasts for an extended period of time, signals that are connected to this power supply could experience significant additional delay due to the low-frequency noise. A static  $IR$  drop analysis that does not consider the dynamic effect of switching activities cannot properly diagnose how timing and functional errors are caused by power supply voltage variations.

The power supply voltage drop may result in false logic switching if the noise exceeds the threshold voltage during steady state. It will also affect timing closure if the noise introduces additional delay during transient state. Since the

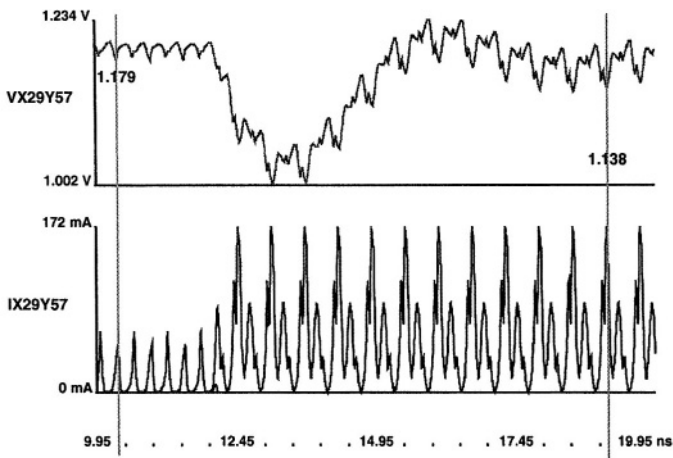


Fig. 1. Transient  $V_{DD}$  and current waveform

device current is proportional to  $(V_{DD} - V_T)^k$ , where  $V_T$  is the threshold voltage and  $k$  is a superlinear parameter between 1 and 2, a 10% noise may have a 15% impact on circuit performance if  $k$  is equal to 1.5. As the power supply voltage continues to scale down in future technologies, the power supply noise will have an increasingly significant impact on device current and circuit performance.

### 3 Power Supply Distribution Model

The design of a robust power supply distribution network is subject to many constraints [6]. First, the average DC voltage drop for the chip has to be limited to less than 30 mV for power densities up to  $1 W/mm^2$ . Second, transient power supply fluctuation should be less than 10% of the nominal power supply, when the total chip power changes abruptly over a few cycles, causing the power supply to oscillate at the resonant frequency of the chip and package. Finally, the common-mode noise on global interconnects has to be less than 200 mV for the worst-case wide-bus activity.

To address the deficiency of a static  $IR$  drop analysis and prevent any potential chip failure due to the collapse of power rails, we have developed a complete power supply noise model (Fig. 2) that includes both the package model and the on-chip power bus model to simultaneously simulate the resistive  $IR$  drop and the inductive  $L\Delta I/\Delta t$  noise.

In order to reduce the complexity of a full-chip power supply noise analysis, a hierarchical approach is used to build the chip and package power distribution model. At the package level, a coarse-grid birthday-cake model [2] is generated to represent the equivalent inductance between adjacent regions on a single-chip or multi-chip module package. At the chip level, a fine-grid model with C4 pitch [3] is used to represent the multilayer RLC power bus network.

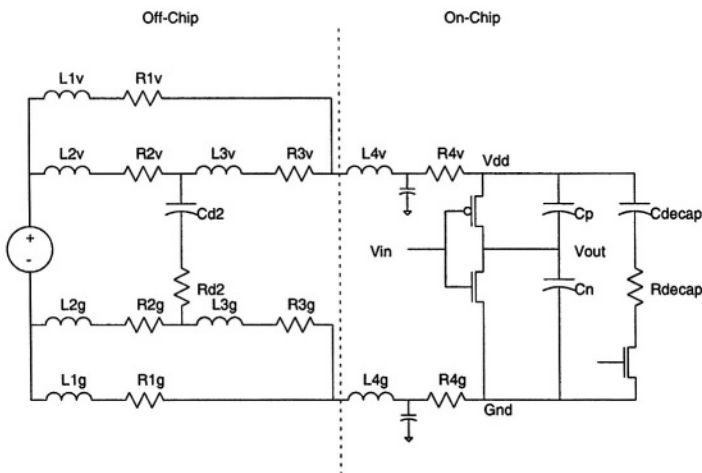


Fig. 2. Power supply distribution model

More importantly, in order to ensure the accuracy of a full-chip power supply noise analysis, we employ a sophisticated switching-circuit model [1] that truly captures the dynamic effect of transient current. Based on the circuit simulation results of our common power analysis methodology [4], we model the switching activities of each functional unit with a piecewise linear current source that mimics the switching pattern and current signature of the real circuits. For example, if the circuits operate at a certain power level such as hold power, maximum average power, clock-gated hold power, or clock-gated maximum average power, within a given cycle, then the waveform that best represents the current switching condition in one of the several possible states will be selected. As the circuits switch from one state to another state, the composite waveform will change accordingly from cycle to cycle to facilitate a vector-based dynamic power supply noise analysis.

Alternatively, if circuit simulation results are not available in the early design stage, a simplified triangular or trapezoidal current waveform can be derived from the average current  $I_{ave}$  and peak current  $I_{peak}$  of each macro during the hold state when only the clocks are running, and the functional state when the circuits are switching with maximum power.

According to the size of the power grid, the switching circuit model for each functional unit will be either partitioned into smaller units or lumped with other units, and connected to the corresponding points on the power bus. Fig. 3 illustrates the current waveforms of 6 circuits that are connected to the same power bus at one local hot spot. Signals can be switched simultaneously or with their respective delay patterns. The corresponding  $V_{DD}$  waveform of the noisy power supply is shown at the bottom of Fig. 3.

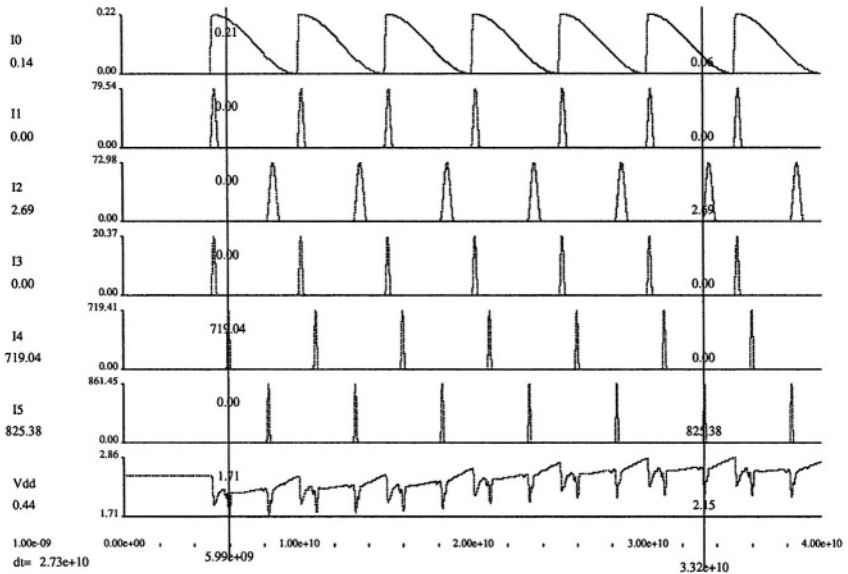


Fig. 3.  $V_{DD}$  variation due to delayed switching

It is worth noting the important role that timing plays in the noise analysis, as the noise doubles if two identical drivers switch at the same time, and the noise can be reduced by half if the same two drivers switch at different times. Therefore, in areas where hundred of drivers are located, it is critical to properly model the switching factor and signal delay of each circuit, to minimize the compounding effect of noises that may be erroneously superimposed.

## 4 Decoupling Capacitor Optimization

To reduce the power supply fluctuation, decoupling capacitors are often used to support the large current transients generated by the simultaneous switching of on-chip circuits and off-chip drivers. By charging up during the steady state, the decoupling capacitors can assume the role of power supply and provide the current needed during switching.

In a simplified circuit model, the electric charge before switching can be represented by  $C_D \times V_{DD}$ , where  $C_D$  is the decoupling capacitance and  $V_{DD}$  is the nominal power supply voltage. The electric charge after switching can be represented by  $(C_D + C_S) \times (V_{DD} + \Delta V)$ , where  $C_S$  is the switching capacitance, and  $\Delta V$  is the power supply noise. From the conservation of charge, where  $C_D \times V_{DD} = (C_D + C_S) \times (V_{DD} + \Delta V)$ , we can easily derive the upper bound on transient power supply voltage fluctuation  $\Delta V = -V_{DD} \times C_S / (C_D + C_S)$ . To limit  $\Delta V$  within 10% of  $V_{DD}$ , decoupling capacitance  $C_D$  should be at least 5 to 9 times the switching capacitance  $C_S$  to prevent the decoupling capacitors from being significantly discharged during circuit switching. For silicon-on-insulator (SOI) technology, which offers faster device switching speed, but lower device junction capacitance, more decoupling capacitance in close proximity of the hot spots may be needed. For today's multi-GHz microprocessor design, as much as 500 nF on-chip decoupling capacitance may be used to control the power-supply noise. If the thick-oxide gate provides  $10 \text{ fF}/\mu\text{m}^2$ , as much as  $50 \text{ mm}^2$  area may be needed for the placement of decoupling capacitors. Therefore, it is important to estimate and allocate the area needed for on-chip decoupling capacitors during floor planning and the early design stage.

The proper amount of decoupling capacitance should also be carefully selected, so as not to generate a resonant frequency near the operating frequency, which will significantly increase the impedance and power supply noise. Ironically, the parasitic resistance that causes  $IR$  drop and latch-up problems can help to resolve the resonance problem by introducing a damping effect and reducing the resonance impedance  $Z = L/(RC)$ . Depending on the locations of the decoupling capacitors, on-chip decoupling capacitors are effective in reducing the high-frequency noise, while off-chip decoupling capacitors are effective in reducing the low-frequency noise.

The on-chip decoupling capacitors include the intrinsic device and junction capacitors that are connected between  $V_{DD}$  and  $G_{ND}$ , such as the non-switching circuit capacitors, the parasitic metal wiring capacitors, and the n-well capacitors for bulk CMOS devices, which are connected between  $V_{DD}$  and  $G_{ND}$ . Additional

decoupling capacitors such as the gate-oxide capacitors and trench capacitors can also be added to minimize the power supply noise. The n-well capacitor is the reverse-biased  $pn$  junction capacitor between the n-well and p-substrate. The circuit capacitor represents the built-in capacitor between  $V_{DD}$  and  $G_{ND}$  in non-switching circuits. For a simple inverter buffer, about 1/2 of the gate capacitance, 1/2 of the diffusion capacitance, and 3/4 of the gate-to-diffusion capacitance contribute to the intrinsic decoupling capacitance. Since only the circuits that are not switching can provide decoupling capacitance, the non-switching device capacitance is calculated by subtracting the switching capacitance from the total device capacitance. The switching capacitance from clock circuits which charge and discharge at the frequency of  $f$  cycles per second can be calculated from  $C = P/V^2f$ , where  $P$  is the switching power and  $V$  is the power supply voltage. The switching capacitance for logic circuits which usually charges and discharges in alternating cycles can be calculated from  $C = 2P/V^2f$ . The total capacitance from non-switching circuits can be estimated from  $(P/(V^2f)) * (1 - SF)/SF$ , where  $SF$  is the switching factor. In addition, the thin-oxide capacitors  $C_{ox}$  are usually added near the drivers, high-power macros, or any available empty space on the chip to alleviate the switching noise problem.

The optimization of on-chip decoupling capacitors involves an iteration process between circuit simulation and floor planning. Given the specifications and location of each functional block, the circuit simulator will analyze the switching noise on the power bus, identify the hot spots, and determine the amount of decoupling capacitance  $C_i$  needed for each region  $i$ . The floor planner then translates the amount of decoupling capacitance into physical area  $A_i$ , and determines the locations and dimensions of decoupling capacitors. It may be possible to allocate a certain portion of the decoupling capacitor inside the macro, if space is available. The positions of neighboring blocks may also be affected, if there is not enough room to embed the decoupling capacitors. The added decoupling capacitors will be modeled with proper time constants and simulated with the new floor plan during the next iteration until  $\Delta V$  is contained.

## 5 Timing and Signal Integrity Analysis

We have used the full-chip power supply noise analysis methodology to generate the power supply voltage distribution map for many high-performance microprocessors. Fig. 4. illustrates the minimum transient  $V_{DD}$  distribution (0.791V - 0.982V), average transient  $V_{DD}$  distribution (0.955V - 0.997V), and maximum transient  $V_{DD}$  distribution (1.015V - 1.136V) of a 30W chip under the nominal supply voltage of 1V. Depending on the placement locations of functional units and the overlaying power grid, each circuit will be assigned a corresponding minimum  $V_{DD}$  for worst-case delay simulation, a maximum  $V_{DD}$  for best-case delay simulation, and an average  $V_{DD}$  for nominal-case delay simulation. Transient and steady-state  $V_{DD}$  and  $G_{ND}$  waveforms (Fig. 1) at each location are also provided for the common-mode and differential-mode noise analysis of clock buffers and other timing-critical circuits.



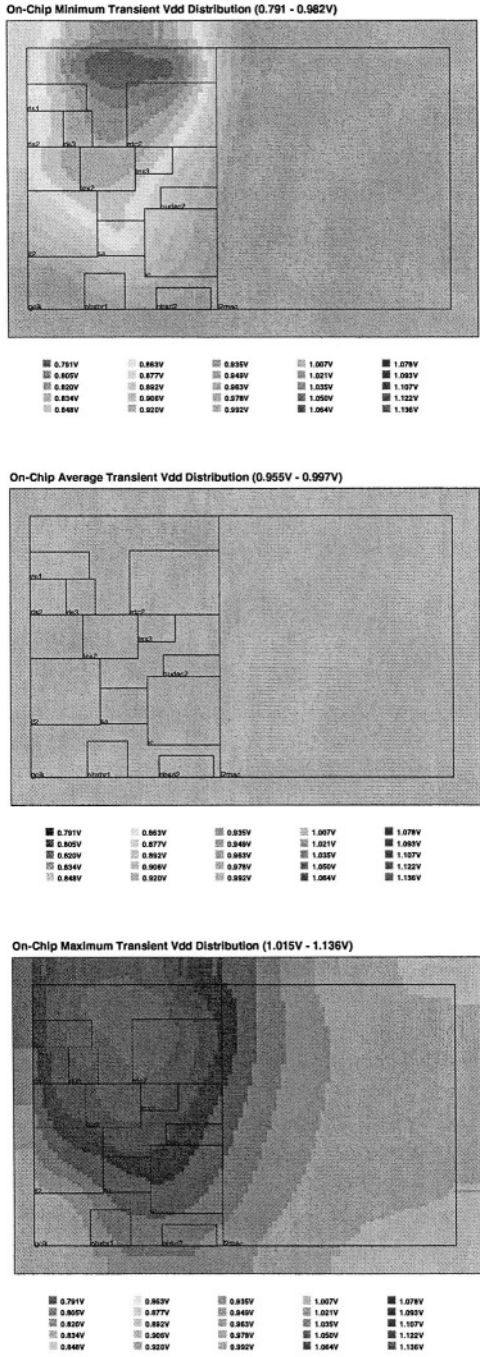


Fig. 4. Minimum, average, and maximum Transient  $V_{DD}$  distribution

The full-chip power supply noise analysis not only performs a complete  $IR$  drop and  $\Delta I$  noise analysis, but also allows designers to easily identify the hot spots and fix potential problems by redesigning the circuit or adding decoupling capacitors. For signal integrity analysis, the noise margin for each signal net,  $N_{margin}(i)$ , is calculated by subtracting the coupling noise  $N_{stalk}(i)$  and the power supply noise  $N_{pwr}(i)$  from each signal's total noise limit  $N_{limit}(i)$ . The worst-case coupling noise can be calculated by assuming that all the near-end and far-end noises from adjacent signals occur at the same time, and  $N_{stalk}(i) = \sum_j C_{ij} L_{ij}$ , where  $L_{ij}$  is the coupled length between adjacent nets  $i$  and  $j$ , and the coupling coefficient  $C_{ij}$  is a function of the interconnect layers, wire spacing, coupled length, and driver/receiver types. Similarly, the worst-case power supply noise  $N_{pwr}(i)$  is assumed to be the maximum  $V_{DD}$  drop in the region where net  $i$  is located,  $N_{pwr}(i) = \max_{(x,y) \in i} (V_0 - V_{xy})$ , where  $V_0$  is nominal power supply voltage and  $V_{xy}$  is the power supply voltage at location  $(x,y)$ .

To identify the potential noise violators whose noise margin  $N_{margin}$  is less than 0, we can perform a net-based full-chip noise analysis by calculating the noise margin for each net. This preliminary analysis does not consider noise propagation from the primary input to the primary output, and only checks the noises that are associated with each net. Since the noise margin calculation is based on the worst-case assumption that all near-end and far-end coupling noises from adjacent wires occur at the same time, it may identify a large number of potential noise violators. A more comprehensive noise study based on timing and statistical models is therefore needed to reexamine the nets with potential problems. The timing of adjacent signals will determine the delay and effective arrival time of each near-end noise and far-end noise. Statistical information such as signal start-time probability and noise magnitude variation is also used to calculate the probability of noise exceeding noise limit. If this probability is above the pre-determined probability limit, it is considered a *true* noise violation which must be fixed by rerouting the nets, resizing the power bus, or redesigning the circuits. In our experiment, among all the *potential* noise violators, approximately 10% will fail the statistical analysis and be considered *true* violators.

Finally, the coupling noise and power supply noise can be combined with the leakage noise and charge-sharing noise to construct the noise graph [5], and be propagated through the network for a full-chip static noise analysis.

## 6 Conclusions

The increasing amount of low-frequency and mid-frequency transient power supply noise due to clock gating and other power saving techniques, and the super-linear dependence of signal delay on supply voltage, have prompted us to develop a full-chip power supply noise analysis methodology to more accurately model the power supply voltage variation during timing simulation and analysis. Based on an integrated chip and package power supply distribution model that simultaneously analyze the resistive  $IR$  drop and the inductive  $L\Delta I/\Delta t$  noise on a

full-chip scale, our analysis provides not only space-dependent across chip voltage variation for static timing analysis, but also time-dependent power supply voltage waveforms for dynamic timing simulation. As the power supply voltage and threshold voltage continue to scale down in the nanometer technology era, we can ill afford to ignore the potential impact of power supply noise on timing, and vice versa. The full-chip power supply noise analysis presented in this paper will allow designers to preserve signal integrity and achieve the timing targets with a tighter bound and better model of power supply voltage variations.

## References

1. H. Chen and J. Neely, "Interconnect and circuit modeling techniques for full-chip power supply noise analysis," *IEEE Transactions on Components, Packaging and Manufacturing Technology - Part B: Advanced Packaging*, Vol. 21, No. 3, August 1998, pp. 209-215.
2. B. McCredie and W. Becker, "Modeling, measurement, and simulation of simultaneous switching noise," *IEEE Transactions on Components, Packaging and Manufacturing Technology - Part B: Advanced Packaging*, Vol. 19, No. 3, August 1996, pp. 461-472.
3. L. Miller, "Controlled collapse reflow chip joining," *IBM Journal of Research and Development*, Vol. 13, No. 3, May 1969, pp. 239-250.
4. J. Neely, *et al.*, "CPAM: A common power analysis methodology for high-performance VLSI design," in *Proceedings of the IEEE 9th Topical Meeting on Electrical Performance of Electronic Packaging*, October 2000, pp. 303-306.
5. K. Shepard and V. Narayanan, "Noise in deep submicron digital design," in *Digest of Technical Papers, International Conference on Computer-Aided Design*, November 1996, pp. 524-531.
6. J. Warnock, *et al.*, "The circuit and physical design of the POWER4 microprocessor," *IBM Journal of Research and Development*, Vol. 46, No. 1, January 2002, pp. 27-51.

# On Timing and Power Consumption in Inductively Coupled On-Chip Interconnects

T. Murgan<sup>1</sup>, A. García Ortiz<sup>2</sup>, C. Schlachta<sup>1</sup>, H. Zimmer<sup>1</sup>,  
M. Petrov<sup>1</sup>, and M. Glesner<sup>1</sup>

<sup>1</sup> Institute of Microelectronic Systems,  
Darmstadt University of Technology  
Karlstr. 15, D-64283 Darmstadt, Germany  
murgan@mes.tu-darmstadt.de

<sup>2</sup> IBM Deutschland Entwicklung GmbH  
Schoenaicher 220, D-71032 Boeblingen, Germany  
agarcia@de.ibm.com

**Abstract.** This work analyses the effects on timing and power consumption of the inductive coupling in long high-frequency on-chip interconnects. By means of extensive simulations it is shown that the common assumptions used until now when considering only line inductance effects do not hold. In fact, signal integrity, voltage glitches and cross-talk, signal delay, rise and fall times, as well as power dissipation strongly depend on the mutual inductances and the input data toggling pattern.

## 1 Introduction

With the advent of Very Deep Sub-Micron (VDSM) technologies, the influence of the on-chip interconnect structure on performance, area, and power dissipation steadily increases. The need for performance pushes operating frequencies in the Gigahertz domain, inductive effects becoming therefore of utmost importance. With such a technological drift, delay slow-down, overshoots, and inductive cross-talk become critical bottlenecks in integrated circuit performance. Inductance emerged thus as a primary factor to be taken into consideration by design engineers.

In addition to the extensive research done in inductance extraction and modelling, the effects of on-chip inductance on various aspects of integrated circuits performance – especially on the interconnects – have also been often highlighted [9,4]. However, the effects of the inductive coupling on timing and power dissipation in long on-chip interconnects have not been yet tackled from an explicit point of view. In this work, we show that timing and power dissipation related issues, like signal integrity, cross-talk, rise times, and optimal repeater insertion among others do not obey general rules and strongly depend on the input data toggling pattern. The main contribution of this work is the outcome that general conclusions drawn in the case of modelling only the line inductance do not hold when taking into account the inductive coupling. Furthermore, by means

of extensive simulations it is pointed out that the aforementioned parameters strongly depend on the input data pattern.

This work is organised as follows: Section 2 briefly discusses the distributed interconnect models employed in literature and presents the model used in this work. The core of the work consists of sections 3 and 4. First, the effects of inductively coupled lines on signal integrity, cross talk, and delay are analysed. Second, the influence on signal rise and fall times, short-circuit, switching and overall power consumption, as well as on optimal repeater insertion are discussed. Some concluding remarks finalise the work in section 5.

## 2 Distributed Models for Interconnects

Each interconnect line exhibits an associated self-inductance and a corresponding mutual inductance to all neighbouring lines. Interconnect models evolved throughout the last decades from a trivial capacitive model to the widely used distributed Resistance-Capacitance (RC) model and later to the distributed Resistance-Capacitance-Self Inductance (RCL) models. Some of the latter models neglect the mutual inductances between the lines. In any case, RCL circuits generate more complex signals than simple RC circuits due to overshoots, faster rise times, and longer delays. We consider the mutual inductances between all elementary segments of the distributed interconnect model, as the main focus of this work is to precisely analyse the effects of the inductive coupling on timing and power dissipation in on-chip interconnects. In the sequel, we will refer to a complete RCL interconnect model as an RCLM (*Resistance-Capacitance-Inductance-Mutual Inductance*) model.

In Figure 1, a 3-segment RCLM model for a 4-bit wide interconnect is represented. For simplicity, only the mutual inductances between the second segment of the third line and all the second and third segments of the other lines are represented. To be noticed that  $m_{j-k}$  stands for the mutual coupling coefficient between the  $j$ -th segment of line  $i$  and the  $l$ -th segment of line  $k$ . An  $S$ -segments distributed model for an  $N$ -lines interconnect has thus  $N \cdot S$  line inductances, and  $N \cdot S(N \cdot S - 1)/2$  mutual inductance coefficients. In order to reduce the order of the RCLM model, several methods for making the inductance matrix sparse have been proposed [4]. Nevertheless, in order to avoid stability problems, one has to make sure that the sparsified matrix is positive definite. Since the focus of this paper is only to work with a very precise model, we employ the full RCLM model.

There are two main factors in determining the required RC or RCL interconnect model: the damping of the interconnect line and the signal rise time to time of flight ratio [6]. The damping factor of an RCL line is given by:

$$\xi = \frac{Rl}{2} \sqrt{\frac{C}{L}} = \frac{R_t C_t}{2\sqrt{C_t L_t}} = \frac{\tau_{RC}}{2\tau_{LC}},$$

where  $R$ ,  $L$ , and  $C$  are the resistance, inductance, and capacitance per unit length respectively,  $l$  is the length of the line,  $R_t$ ,  $L_t$ , and  $C_t$  are the total resistance,

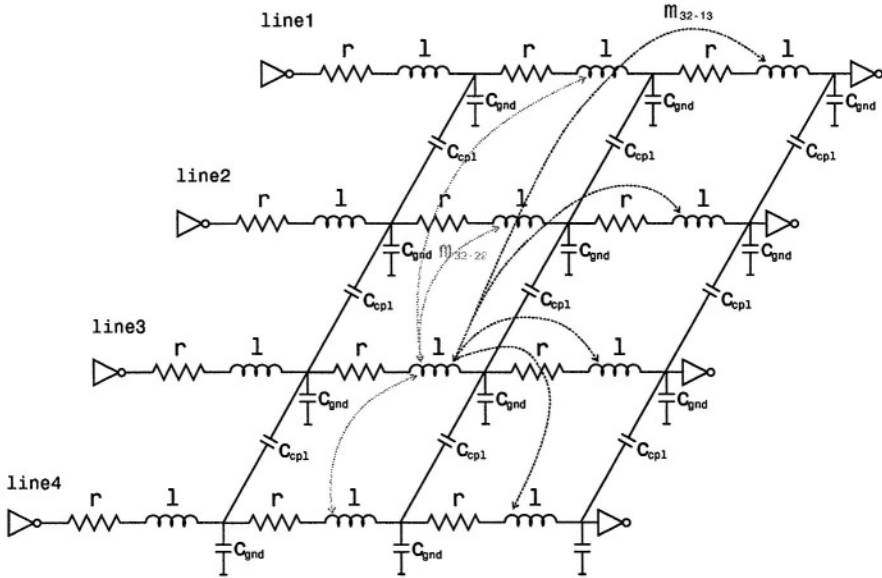


Fig. 1. Distributed full RCLM interconnect model.

inductance, and capacitance of the line respectively, and  $\tau_{RC}$  and  $\tau_{LC}$  are the RC and LC time constants of the line. When  $\xi$  decreases, which actually means that the effects of the reflections increase, the RC model becomes inaccurate.

The inductance of on-chip wires is not scalable with length and no good approximation formula exists for mutual inductances of two parallel lines of unequal length. Therefore, it is indicated to use a field solver for determining the inductive coupling more accurate than with closed formulas [8]. As an illustrative example, in this work, we chose to model a 1mm long 5-bit wide bus by splitting every line into 10 segments. The thickness, width, and spacing of the lines are all  $1\mu\text{m}$  while the distance to the lower metal layer is considered also  $1\mu\text{m}$ . The values have been chosen according to the upper layers characteristics of a typical 130nm 1.8V technology. The employed transistor models used for implementing the buffers also correspond to such a technological node.

The total ground and coupling capacitances have been extracted with Fast-Cap [10]. Afterwards, the values have been divided by the number of segments to obtain the distributed values. FastHenry [7] was used to extract the total resistances and inductances. In order to obtain the required distributed values, the input file for FastHenry has been adapted by splitting a 1mm line into 10 serial segments closely spaced to one another. In this way, the 50 line inductances and 1225 mutual coupling coefficients were derived which have been dumped into a SPICE file. The lines are driven by inverters with  $75\Omega$  output resistance and the input signals have a 5ps rise-time.

The capacitive coupling effect is a “short-range” effect as only the mutual capacitances between adjacent bus lines have an important influence on cross-

**Table 1.** Inductive coupling.

Lines	$K_{12}$	$K_{13}$	$K_{14}$	$K_{15}$
Coupling	0.709576	0.577800	0.502238	0.449695

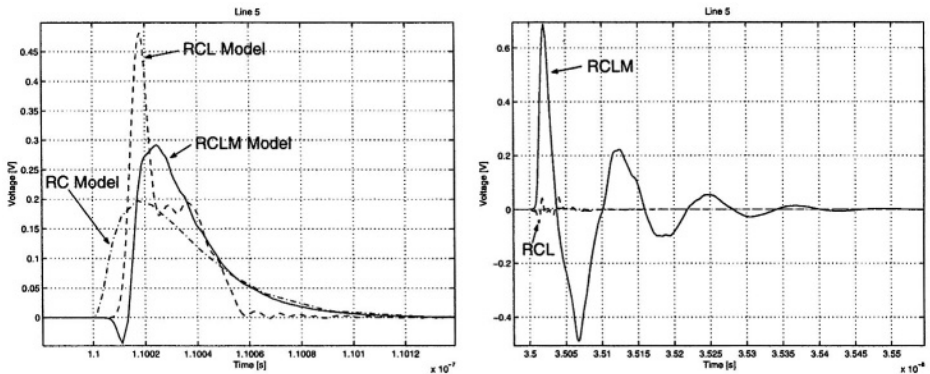
talk. On the contrary, the mutual inductance decays only slowly with bus-line spacing making the inductive effect a “long-range” one [5]. The inductive far-coupling effect can be observed in Table 1, which lists the mutual inductance coefficients between a segment of the first line and all other segments belonging to the same row in the distributed RCLM model.

### 3 Timing Issues in Inductively Coupled Lines

In previous works [9], it has been shown that line inductances play a major role in the timing characteristics of a bus line. If they are not taken into consideration, many undesired timing effects could appear, e.g. cross-talk glitches, wrong switching, false delay, signal integrity issues. In this section, we extend those previous results by considering explicitly the effects of the mutual inductances which appear in very deep sub-micron technologies.

The performed SPICE simulations allow to compare the three cases of modelling an interconnect line, namely: a distributed RC line, a distributed RCL line without mutual inductances, and a distributed full RCLM line including inductive coupling.

Our simulations have shown that there exist significant differences among the three models. The RC model cannot predict the behaviour of an RCL circuit, while the RCL model cannot handle the increased variability of RCLM circuits. Those discrepancies are present in both simulation results shown in figure 2. Two complementary effects of the coupling inductances on quiet lines can be



(a) Transition from “01000” to “00010”. (b) Transition from “11000” to “00000”.

**Fig. 2.** Voltage glitches at the far end of the quiet fifth line.

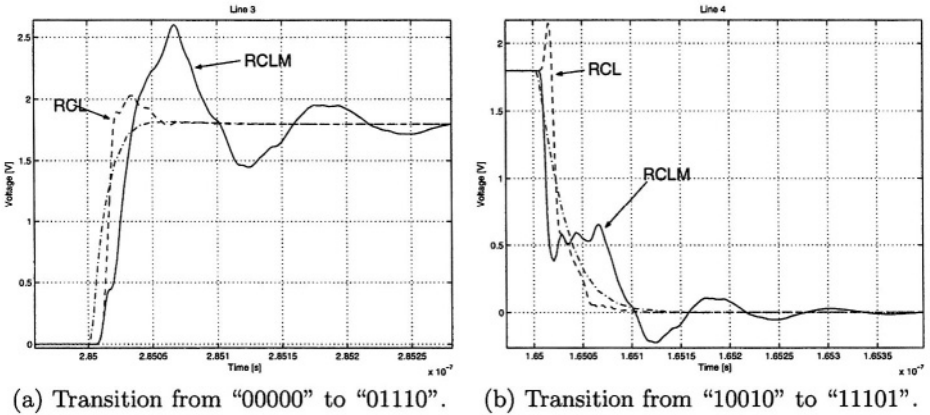


Fig. 3. Voltage glitches and cross-talk on toggling lines.

observed, depending on the input data toggling pattern. First, for a data toggling on the bus from "01000" to "00010" the voltage glitch in line 5 is reduced from 0.48V to 0.29V. On the other hand, in the case of a "11000" to "00000" toggle the voltage glitch in the same line 5 increases from 0.05V to 0.69V. Thus, an acceptable glitch could be treated like an unacceptable one and vice-versa if mutual inductances are not to be taken into account. In the latter situation the settling time is also clearly increasing. For lines toggling together with some of their aggressors, voltage glitches of more than 0.8V may appear (see Figure 3a). On the contrary, it is also possible that voltage glitches predicted by the RCL model almost disappear when employing the full RCLM model (see Figure 3b).

As mentioned in section 2, inductive coupling is in contrast to capacitive coupling a "long-range" effect. Thus, the sandwiched lines do not have the same shielding effect as for mutual capacitances. In Figure 4a it can be seen that the

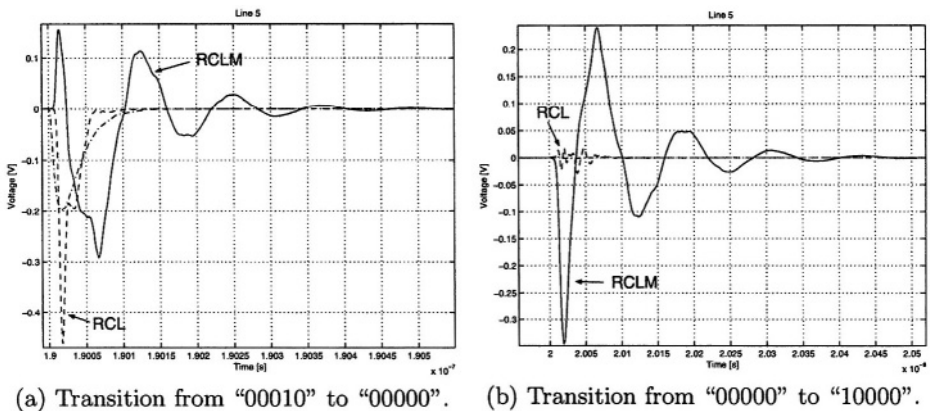


Fig. 4. Cross-talk noise at the end of the fifth line.



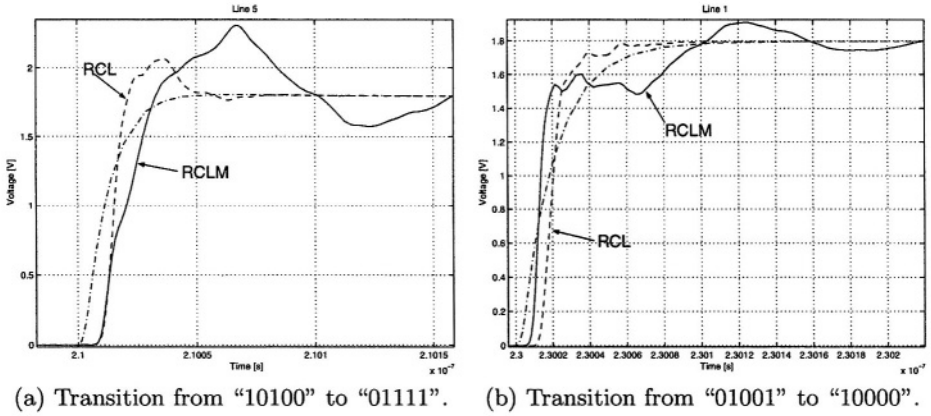


Fig. 5. Effects of RC, RCL, and RCLM modelling on signal delay.

influence of capacitive coupling is comparable to that of the inductive one when the first-order aggressor toggles. However, for some transition activity patterns, in the case of a toggle in the fourth-order neighbour (see Figure 4b), the voltage glitch predicted by inductive coupling may be approximately one order of magnitude larger than the one predicted when modelling only the line inductance and even two orders of magnitude that the one given by the RC model. Moreover, in Figure 4b, it can be observed that also the settling time may increase dramatically, which can have some influence on the switching power consumption as discussed in section 4.

The inclusion of line inductance not only shows the existence of overshoots, which introduce large cross-talk noise on neighbouring lines, but also increase the signal delay [2,9]. Figure 5 shows that the delay when considering line inductances can vary significantly. In the sequel we compare the signal delay predicted by the full RCLM model with respect to the RCL one.

Neglecting the line inductance in a RCL model and using a simple RC delay model “always results in underestimating the propagation delay” [9]. However, in a full RCLM scenario, the signal delay difference can be positive or negative (in our example in Figure 5: 33,16% or -23,84% respectively), depending on the data toggling pattern. Consequently, the latter statement does not hold for full RCLM models, which is a very important result of the present work.

In brief, it can be observed that the inductive coupling from an aggressor bus line seriously modifies noise and cross-talk, can generates high voltage glitches

Table 2. Signal delay in RC, RCL, and RCL modelled lines.

Line	Transition	RC Delay	RCL Delay	RCLM Delay
5	“10100” → “01111”	9.3ps	15.1ps	18.7ps
1	“01001” → “10000”	16.1ps	19.9ps	13.3ps

as well as reducing them, or alters delays. Furthermore, the above-mentioned variations strongly depend on the data toggling pattern.

### 4 Power Consumption Analysis

According to the dependency with the temporal variation of the input signals, power consumption in digital CMOS circuits can be classified in two main categories, namely the dynamic power consumption and the static power consumption. The dynamic power consumption arises from the transient behaviour of the input signals and is associated with two main phenomena: the capacitive switching current caused by the loading and unloading of internal capacitances, and the short-circuit current generated when a direct path between the supply voltage and ground appears. Moreover, VDSM CMOS technologies exhibit a steady leakage current even when transitions do not occur [1]. Nonetheless, this component of power consumption is still small in the case of interconnects in a 130nm technological node, so it is not taken into consideration in this work.

The switching component of power dissipation corresponds to the amount of energy needed to completely charge the parasitic capacitors and is given by  $CV^2\alpha$ . However, this is true only when the transient state is over. Otherwise, there is still a current flowing through the inductances when the next transition occurs and the voltage on the capacitors is not necessarily settled, and therefore some energy is stored in these elements. Under this circumstances, as it can be seen in the following equations, the total dissipated energy may change:

$$E = \int_{t_i}^{t_f} i u dt \rightarrow E_C = \frac{1}{2} C [u(t_f)^2 - u(t_i)^2], \quad E_L = \frac{1}{2} L [i(t_f)^2 - i(t_i)^2]$$

where  $t_i$  and  $t_f$  are the initial and final transition time instance respectively.

Simulations showed, that in the case of an RCLM interconnect model, some differences in the switching power dissipated exclusively in the lines appear. Nevertheless, those discrepancies are getting important only when switching occurs very fast. In those situations, our simulations showed that the signals are degraded anyway to such an extent which makes them unacceptable.

Additionally, in the case of the RCLM model, cross-talk may cause voltage glitches at the inverter output. Consequently, the driven capacitance may be partly loaded or unloaded for a very short time, therefore slightly increasing the dynamic power consumption. Yet, such glitches should be avoided as they appear together with significant cross-talk.

**Table 3.** Rise and fall times for lines 3 & 4 at the transition “10101”→“11011”.

Line		RC	RCL	RCLM
4	Rise Time	54.5ps	40.8ps	36.2ps
3	Fall Time	57.5ps	37.5ps	74.9ps

The switching component of power consumption is independent of the rise and fall times of the input waveforms. However, direct current paths between  $V_{dd}$  and  $GND$  appear exactly during the rise and the fall of input signals. The short-circuit power can be approximated in first term by:

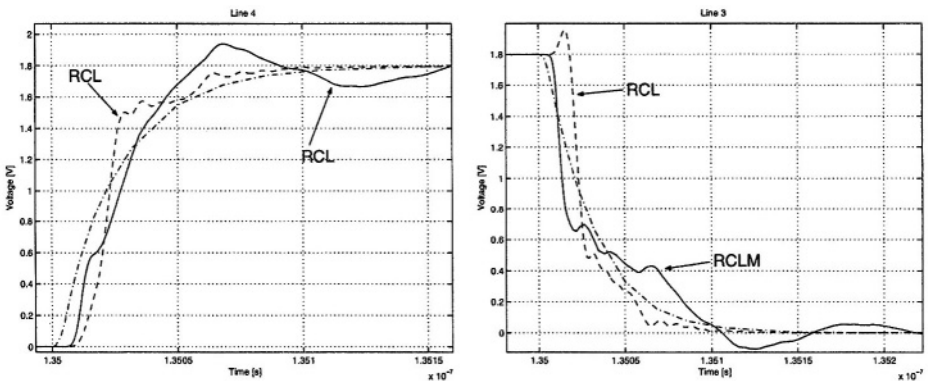
$$P_{short-circuit} = \frac{\beta}{12} (V_{dd} - V_{THn} - |V_{THp}|)^3 \tau \frac{t_i}{T_{clock}},$$

where  $\tau$  is the transition time of the input signal at the load gate, and  $\beta$  is a technological parameter which has a very small dependence on  $\tau$  [3].

For interconnects modelled only by line inductances, previous works showed that the rise and fall times of signal waveforms improve as the inductance effects increase [9]. Nevertheless, when taking into account the coupling inductance, the statement does not remain true. Table 3 and figure 6 show opposite effects with respect to different lines in the case of the same data data toggling context. In the first case, the rise time in line 4 improves by 11.27% with respect to the RCL model and by 33.57% with respect to the RC model. As for the second case, in line 3, the RCLM-predicted fall time deteriorates with 99.73% and 30.26% compared to the RCL and RC prediction respectively. Again, we notice a substantial dependence on the data toggling pattern.

Because of important overshoots and undershoots, simulations proved that the overall short-circuit power consumption does not necessarily have to decrease. Spurious short-circuit currents for an inverter with  $V_{THn} = |V_{THp}| = 0.4V$  can be identified in Figure 7 by means of a simple analysis. In case (a), a supplementary spurious short-circuit current appears in the RCLM model. Case (b) presents a reverse situation, where a spurious short-circuit current shows up in the RCL model but not in the RCLM one.

In order to assure high-performance, long resistive interconnects are driven by repeaters. Those repeaters are large gates and are responsible for a notable part of the total power consumption. When not taking into account the mutual



(a) Transition from "10101" to "11011". (b) Transition from "10101" to "11011".

Fig. 6. Effects of RCLM modelling on rise and fall times.

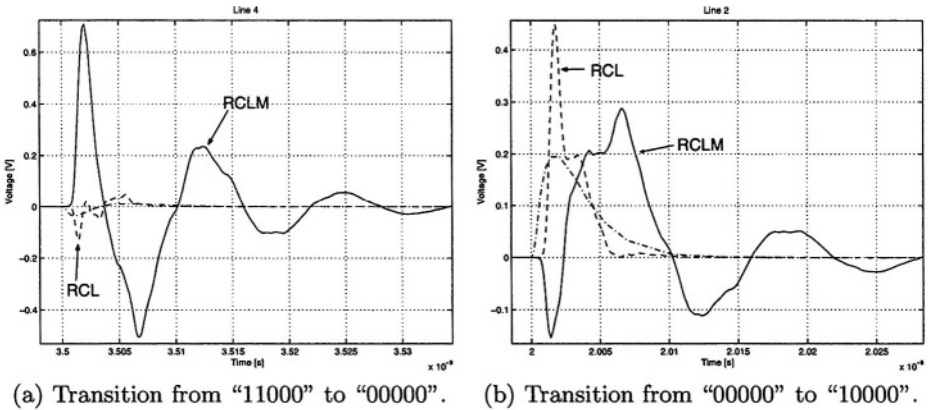


Fig. 7. Appearance of spurious short-circuit currents.

inductances, as line inductance effects increase, the optimum number of repeaters for minimum propagation delay decreases [9]. Moreover, fewer and smaller repeaters result in a significant reduction of the dynamic power consumption in the repeaters. Nonetheless, as seen in section 3, the worst case signal delay predicted by an RLCM interconnect model degrades dramatically compared to the signal delay resulted in simulations of an RCL model. Thus, in inductively coupled on-chip interconnects, the optimum number of repeaters is higher than the number predicted by an RCL model and the expected savings in area and power consumption are too optimistically.

In brief, the power dissipation in inductively coupled lines differs to the one projected by models including only line inductances. As for timing, power consumption strongly depends on the input data toggling pattern.

## 5 Concluding Remarks

This paper analyses the effects on timing and power consumption in on-chip interconnects when taking into account the inductive coupling between parallel lines of a bus. We compare the results of three sets of simulations for three different distributed bus models: RC, RCL, and RCLM. For the third set of simulations, a full RCLM model was used, allowing thus to run very accurate simulations.

One set of our simulations confirms the results obtained in previous works where only RCL models have been employed [9,6]. Nevertheless, the effects of modelling mutual inductances on timing and power consumption are significant and the assumptions attained when modelling only the line inductance do not hold in the more general RCLM case. For parameters like cross-talk, signal delay, rise and fall times serious errors of the RCL model with respect to the RCLM one have been reported. Besides the conclusions drawn regarding the severe limitations of the RCL models in predicting the most important timing and

power consumption related parameters, another essential contribution of this work is the outcome that those parameters strongly depend on the transition activity pattern.

## References

1. A. P. Chandrakasan and R. W. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.
2. C.-K. Cheng, J. Lillis, S. Lin, and N. Chang. *Interconnect Analysis and Synthesis*. John Wiley & Sons, 2000.
3. M. A. El-Moursy and E. G. Friedman. Inductive Interconnect Width Optimization for Low Power. In *Intl. Symposium on Circuits and Systems (ISCAS)*, May 2003.
4. K. Gala, D. Blaauw, J. Wang, V. Zolotov, and M. Zhao. Inductance 101: Analysis and Design Issues. In *Design Automation Conference (DAC)*, June 2001.
5. L. He, N. Chang, S. Lin, and O. S. Nakagawa. An Efficient Inductance Modeling for On-Chip Interconnects. In *IEEE Custom Integrated Circuits Conference*, pages 457–460, 1999.
6. Y. Ismail, E. G. Friedmann, and J. L. Neves. Figures of Merit to Characterize the Importance of On-Chip Inductance. *IEEE Trans. on VLSI Systems*, 7:442–449, December 1999.
7. M. Kamon, M. Tsuk, and J. White. FastHenry: A Multipole-Accelerated 3D Inductance Extraction Program. *IEEE Trans. on Microwave Theory and Techniques*, 1994.
8. S. Lin, N. Chang, and S. Nakagawa. Quick On-Chip Self- and Mutual-Inductance Screen. In *Intl. Symposium on Quality Electronic Design*, pages 513–520, March 2000.
9. Y. Massoud and Y. Ismail. Grasping the Impact of On-Chip Inductance. *IEEE Circuits & Devices Magazine*, pages 14–21, July 2001.
10. K. Nabors and J. White. FastCap: A Multipole-Accelerated 3D Capacitance Extraction Program. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 1991.

# Signal Sampling Based Transition Modeling for Digital Gates Characterization\*

Alejandro Millán<sup>1,2</sup>, Jorge Juan<sup>1,2</sup>, Manuel J. Bellido<sup>1,2</sup>,  
Paulino Ruiz-de-Clavijo<sup>1,2</sup>, David Guerrero<sup>1,2</sup>, and Enrique Ostúa<sup>1,2</sup>

<sup>1</sup> Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica  
Av. Reina Mercedes, s/n (Edificio CICA) - 41012 Sevilla (Spain)  
Tel.: +34 955056666 - Fax: +34 955056686  
<http://www.imse.cnm.es>

{amillan, jjchico, bellido, paulino, guerre, ostua}@imse.cnm.es  
<sup>2</sup> Departamento de Tecnología Electrónica - Universidad de Sevilla  
Av. Reina Mercedes, s/n (E. T. S. Ingeniería Informática) - 41012 Sevilla (Spain)  
Tel: +34 954556161 - Fax: +34 954552764  
<http://www.dte.us.es>

**Abstract.** Current characterization methods introduce an important error in the measurement process. In this paper, we present a novel method to drive the timing characterization of logic gates under variable input transition times. The method is based on sampling and scaling realistic transition waveforms and it is easy to implement and introduces negligible computational overhead in the characterization process. We show how models characterized using the proposed method may improve accuracy from 5% to 8%.

## 1 Introduction

During the design stages of a digital circuit, simulation is applied to validate a design before fabrication. Typically, only critical parts of the circuit are simulated using low-level simulators working at the electric level (SPICE, HSPICE [1], etc.). This kind of simulation is accurate but very time and resources consuming and is usually not applicable to the whole circuit. On the other hand, logic-level simulators are able to handle big circuits and even whole systems by using simplified, logic-level models and very fast event-driven algorithms [2]. This is done at the expense of accuracy, due to the simplified nature of logic-level models and algorithms. The accuracy of the logic-level simulation is typically measured by comparing to electrical-level simulation results which are much more accurate.

The scientific community has spent a great effort in developing better logic-level models, commonly referred to as delay models. These models try to calculate the propagation time of an input transition in a logic gate to the output. The propagation delay depends on multiple factors: the output load of the gate,

\* This work has been partially supported by the MCYT VERDI project TIC 2002-2283 and the MEC/SEEU/DGU project PHB2002-0018-PC of the Spanish Government.

the waveform of the input transition, the supply voltage and the internal design characteristics of the gate itself. New generation delay models try to take into account all these effects [3,4,5,6,7,8,9].

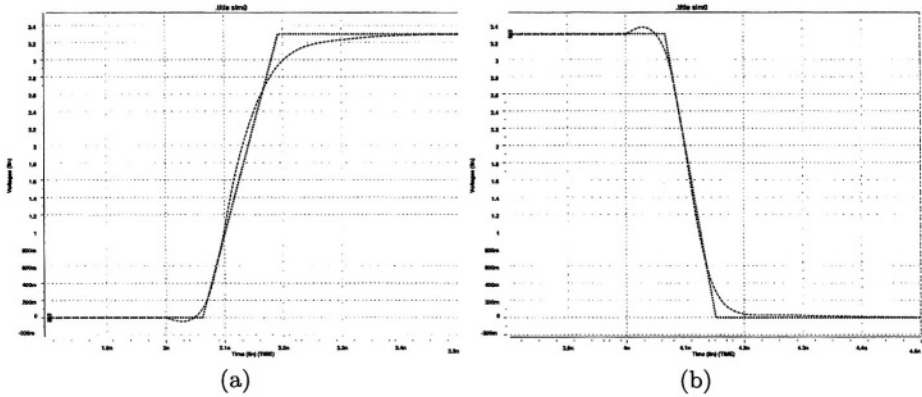
In order to build new delay models and/or characterize the model parameters of a logic block, accurate electrical simulation of the block is carried out under controlled external conditions that may be altered as necessary. These variable conditions are the supply voltage, the output load and the input waveform. Providing a given supply voltage is not a problem and the output load can be easily and accurately modelled by a capacitor, at least in a CMOS technology. The input waveform, however, presents some problems: on the one hand, we need to be able to apply different types of input transitions, ranging from slow to fast ones to cover all the possible operations of the gate but, on the other hand, the generated input transitions need to be similar to the actual waveforms in a real circuit. The most commonly used approach is to build linear input transitions and use the transition slope or the transition time as the variable representing the waveform shape. The delay model, then, will include the dependence on this transition time. This approach assumes that the linearized input transition is equivalent to a real transition and a mapping between linear and real transitions is provided. Some of these mapping approaches are:

- A ramp with the same slope than the real transition at  $V_{DD}/2$  [6].
- Similar to the previous one but applying a correcting factor [10].
- A ramp crossing the real transition at given voltage levels, like 10%-90%, 20%-80%, or 30%-70% [11].

Among many proposed approaches for input transition linearization not a commonly criteria has been accepted because none of them have proved to be fully satisfactory. In our opinion, using linear input transitions to drive a model characterization is a source of inaccuracy since real transitions often diverge from a linear ramp, specially at the beginning and end of the transition as shown in Fig. 1.

In this paper we propose a better method to synthesize input transitions for gate characterization. The method consists of using a PWL (piecewise linear) curve instead of a ramp. A reference PWL curve is obtained by sampling a single “real” transition obtained from accurate electric-level simulation. The number of points in the PWL curve is high enough to make it very close in slope to the real input transition. The slope of the curve measured at the 20% and 80% of the supply rail is taken as the characteristic parameter of the transition. During a characterization process, this reference curve is used so that arbitrary values of the slope are obtained by scaling the time axis of the reference curve as necessary to obtain a modified transition of the required slope. This is a simple and fast calculation. The modified curve can then be applied to the gate under test to collect new timing data.

It is important to remark that only one reference curve is necessary for a single technology process or even for several technology processes, and that it is easy to obtain by simulating a single transition in a reference circuit with an



**Fig. 1.** Ramp approach method using 20%–80% criterion.

accurate electrical simulator like HSPICE. Also, like in the conventional ramp approach, the only characteristic transition parameter used is the slope.

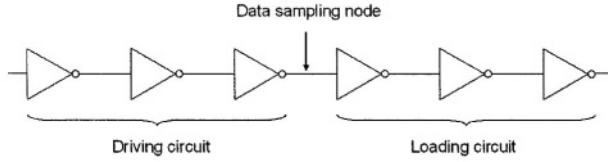
By using this approach we can drive any characterization process where input transition time needs to be controlled by using more realistic waveforms that will provide us with more accurate data. To show this, after presenting the method to obtain the reference transition in Sect. 2, we will set up a simple look-up table based model in Sect. 3. The model will be characterized using the conventional approach with input ramps and using the proposed sampled input transitions. In Sect. 4, we will use a test bench that will be simulated using HSPICE under different load and input conditions. Electrical simulation results will be compared to calculations from the previous characterized models. Finally, in Sect. 5, we will summarize some conclusions.

## 2 Obtention and Usage of the Sampled Input Signals

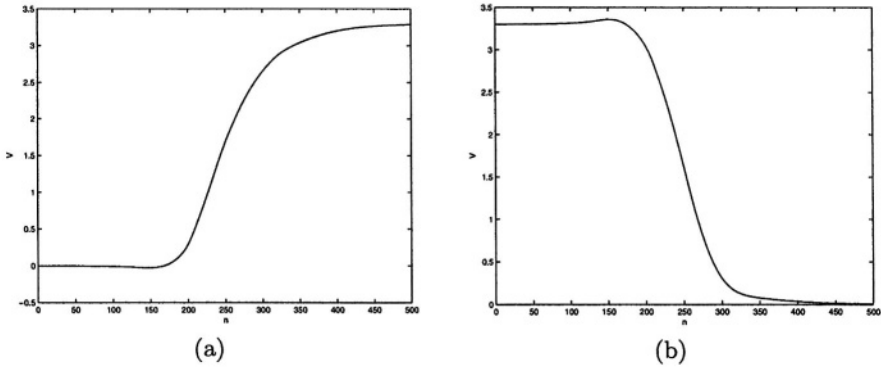
To obtain the sampled signal, we have simulated an inverter chain with six gates (Fig. 2). The analysis has been carried out in a  $0.35\ \mu\text{m}$  CMOS technology using the standard cell library provided by the foundry, and all simulations have been performed using HSPICE. We have supplied a step input to the first inverter and sampled the output of the third one allowing this inverter to be supplied with a real input signal (provided by the second gate) and to be charged with a real gate at its output (the fourth gate). This process has been performed for both raising and falling output cases obtaining two sets of 500 samples that conform the two sampled input signals (Fig. 3). The sample period has been established to be 1 ps.

In order to use the sampled data, we only have to scale the set of samples depending on the input slope we need. For example, if we need an input transition time of 100 ps, we must firstly establish a criterion for the choice of these points (we have used the 20%-80% criterion) and scan the set of samples until we find the two samples that better approximate the values  $0.20V_{DD}$  and  $0.80V_{DD}$ .





**Fig. 2.** Obtention of sampled reference curve.



**Fig. 3.** Sampled signals for both raising (a) and falling (b) output cases (each one compounded of 500 samples).

Once we have the indexes of these samples, we must calculate the new period time as the ratio between the needed input transition time and the amount of samples:

$$t_{step} = \frac{0.60\tau_{in}}{i_{80\%} - i_{20\%}} \tag{1}$$

Now, we can supply these new data to the gate and perform the simulation. We have carried out our tests using HSPICE and have automated the process of generating this scaled input according to this simulator. The developed function defines a PWL input signal and supplies it to the gate under analysis. The mentioned function has been implemented in C language.

### 3 Application to Simple Look-Up Table Delay Model

In this section we will apply the proposed method to the characterization of a simple look-up table based delay model. A CMOS inverter is taken as a sample gate and timing information (propagation delay and output transition time) is collected for a range of input transition time values and two typical loading conditions using accurate circuit-level simulation (HSPICE). All these measures are stored in a table so that timing information for arbitrary input transition time and load can be calculated by interpolating the stored data.

We want to point out that, in our opinion, the use of look-up table models in general is not a good choice because of their high requirements in characterization time and data storage. Equation-based models are much more efficient. However, using a table model in this paper allow us to analyze the behaviour of the proposed method isolated from additional effects.

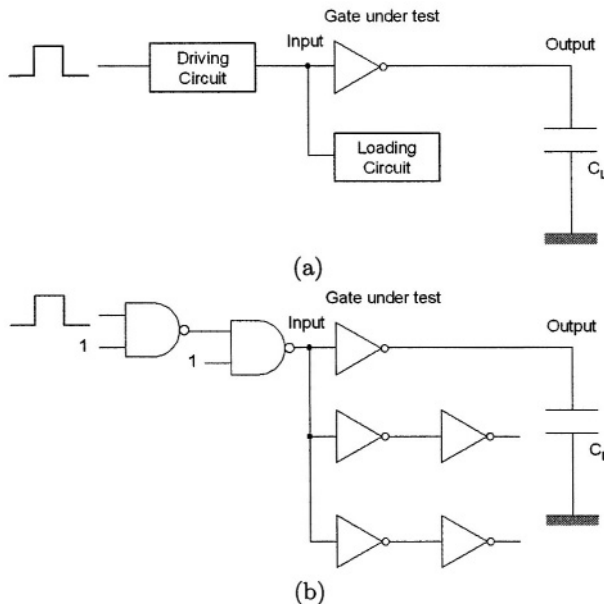
To build the model table, we need to control the input waveform to provide different transition times. The characterization process is done twice, once using traditional straight input ramps method (IR-method), and once again using scaled sampled inputs (SI-method) as described in the previous section. In the next section we will compare the accuracy of both approaches.

## 4 Simulation Results and Analysis

To compare the IR-method with the proposed SI-method, we will use the test circuit in Fig. 4.a.

The test circuit provides various realistic input transitions to the gate under test. Since the transition waveform will depend on the nature of the driving circuit and the loading conditions at the input node to the gate under test, different cases have been simulated:

- Case 1: The driving circuit is a chain of two CMOS inverters.
- Case 2: The driving circuit is a chain of two CMOS NAND gates with one of their inputs set to logic ‘1’.



**Fig. 4.** Test circuit: (a) Generic circuit and (b) Driving NANDs circuit.

**Table 1.** HSPICE results for case 1

Input behav.	Load circ.	$C_L/C_{in}$	$\tau_{in}$ (ps)	$t_{p0}$ (ps)	$\tau_{out}$ (ps)
Rise	0	2	136.3	83.7	126.7
		4	135.2	112.8	182.8
	1	2	188.3	90.0	137.1
		4	187.0	121.3	194.0
	2	2	245.0	95.2	147.9
		4	243.0	128.8	205.8
Fall	0	2	101.0	86.2	167.8
		4	100.6	119.6	248.3
	1	2	133.5	93.5	174.8
		4	133.1	128.1	255.0
	2	2	166.3	100.4	182.3
		4	165.7	136.3	262.5

**Table 2.** HSPICE results for case 2

Input behav.	Load circ.	$C_L/C_{in}$	$\tau_{in}$ (ps)	$t_{p0}$ (ps)	$\tau_{out}$ (ps)
Rise	0	2	200.7	90.6	138.9
		4	199.3	122.3	196.0
	1	2	255.2	95.5	149.1
		4	253.2	129.2	207.2
	2	2	315.3	99.6	159.5
		4	312.7	135.3	218.8
Fall	0	2	137.9	95.1	177.5
		4	137.5	130.2	256.7
	1	2	167.2	101.1	183.8
		4	166.6	137.4	263.7
	2	2	196.3	106.8	190.3
		4	195.7	144.2	270.8

The loading circuit is a chain of two inverters. In order to obtain several transition times and different input waveforms, the circuit has been simulated in three ways: (a) without loading circuit, (b) with one loading circuit, and (c) with two loading circuits (Fig. 4.b).

For each case and for every loading circuit configuration, the gate under test is analyzed for two values of the loading capacitance:  $C_L = 2C_{in}$  and  $C_L = 4C_{in}$ ; where  $C_{in}$  is the equivalent input capacitance of the gate under test.

HSPICE simulations are carried out on all these cases, both for rising and falling input transitions, so that extensive timing information about the operation of the gate under a range of realistic conditions is obtained. HSPICE results are shown in Table 1 and Table 2 for cases 1 and 2 respectively.

In order to analyze the accuracy of the proposed characterization method, the input transition times obtained from HSPICE simulations are used to compute the calculated delay and output transition time from the look-up table models characterized in Sect. 2. Calculations are done by linear interpolation. Propaga-

**Table 3.** IR-method and SI-method relative error for case 1

Input behav.	Load circ.	$C_L/C_{in}$	$\tau_{in}$ (ps)	IR-method		SI-method	
				$t_{p0}$ (%)	$\tau_{out}$ (%)	$t_{p0}$ (%)	$\tau_{out}$ (%)
Rise	0	2	136.3	8.58	9.20	0.90	0.12
		4	135.2	8.74	6.88	0.41	0.56
	1	2	188.3	7.41	8.43	1.53	0.49
		4	187.0	9.06	9.32	0.98	0.04
	2	2	245.0	5.27	7.24	0.90	0.78
		4	243.0	8.42	9.02	1.77	0.52
Fall	0	2	101.0	3.88	3.73	0.25	0.42
		4	100.6	3.54	1.97	0.41	0.56
	1	2	133.5	4.54	6.44	1.20	1.02
		4	133.1	4.89	3.90	1.09	0.38
	2	2	166.3	4.50	7.39	2.27	2.06
		4	165.7	5.84	6.04	2.17	1.57

**Table 4.** IR-method and SI-method relative error for case 2

Input behav.	Load circ.	$C_L/C_{in}$	$\tau_{in}$ (ps)	IR-method		SI-method	
				$t_{p0}$ (%)	$\tau_{out}$ (%)	$t_{p0}$ (%)	$\tau_{out}$ (%)
Rise	0	2	136.3	6.26	7.85	0.72	0.17
		4	135.2	8.43	8.97	0.63	0.16
	1	2	188.3	4.68	6.41	0.41	0.32
		4	187.0	7.63	8.61	1.17	0.21
	2	2	245.0	3.79	3.85	0.15	0.07
		4	243.0	5.80	7.97	0.59	0.26
Fall	0	2	101.0	5.25	7.45	2.10	2.09
		4	100.6	5.75	4.44	2.00	0.80
	1	2	133.5	4.98	8.07	2.79	2.77
		4	133.1	6.47	6.44	2.82	1.96
	2	2	166.3	4.51	8.73	3.17	3.51
		4	165.7	6.87	8.39	3.31	3.13

tion delay and output transition times results are shown in Table 3 and Table 4. For the sake of clarity, only deviation percentages with respect to HSPICE are shown.

Both in Table 3 and Table 4, we can see that the proposed approach gives much better results than the traditional method, in every of the 24 cases simulated. In order to get the big picture, summarized results are presented in Table 5, where minimum, maximum, and average deviations with respect to HSPICE have been calculated for the six different simulated configurations of input and output loading conditions.

In most cases, the proposed approach introduce an average relative error around 1% or below, while the traditional method average error is around 6%. Another important result is that the “maximum” error observed with the pro-

**Table 5.** Summarized results

		Minimum		Average		Maximum	
		IR	SI	IR	SI	IR	SI
$t_{p0}$	Case 1 Rise	5.27%	0.41%	7.91%	1.08%	9.06%	1.77%
	Fall	3.54%	0.25%	4.53%	1.23%	5.84%	2.27%
	Case 2 Rise	3.79%	0.15%	6.10%	0.61%	8.43%	1.17%
	Fall	4.51%	2.00%	5.64%	2.70%	6.87%	3.31%
$\tau_{out}$	Case 1 Rise	6.88%	0.04%	8.35%	0.42%	9.32%	0.78%
	Fall	1.97%	0.38%	4.91%	1.00%	7.39%	2.06%
	Case 2 Rise	3.85%	0.07%	7.28%	0.20%	8.97%	0.32%
	Fall	4.44%	0.80%	7.25%	2.38%	8.73%	3.51%

posed method is 3.5%, while the “minimum” error introduced by the ramp approach is always above 3.5% except for one case, that is 2%. In some cases, the relative error is reduced up to 8%. Timing estimation is then clearly improved by using the sampled input based method.

## 5 Conclusions

A novel method to drive the timing characterization of logic gates under variable input transition times has been presented. The method is based on sampling and scaling realistic transition waveforms and it is easy to implement and introduces negligible computational overhead in the characterization process.

It has been shown that models characterized using the proposed method may improve accuracy from 5% to 8%, just by using the proposed method.

At the sight of these results, we see that using simple input ramps may be an important source of error during model characterization (5%-8%). The proposed method greatly improves this situation by reducing the error due to the use of synthetic input transitions below 2% in most cases.

## References

1. HSPICE User's Manual. Meta-Software (1999)
2. Mizco, A.: Digital logic testing and simulation. Ed. Harper and Row (1986)
3. Bisdounis, L., Nikolaidis, S., Koufopavlou, O.: Analytical transient response and propagation delay evaluation of the CMOS inverter for short-channel devices. *IEEE Journal of Solid-State Circuits* **33(2)** (February 1998) 302–306
4. Kayssi, A. I., Sakallah, K. A., Mudge, T. N.: The impact of signal transition time on path delay computation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* **40(5)** (May 1993) 302–309
5. Auvergne, D., Azemard, N., Deschacht, D., Robert, M.: Input waveform slope effects in CMOS delays. *IEEE Journal of Solid-State Circuits* **25(6)** (December 1990) 1588–1590

6. Daga, J. M., Auvergne, D.: A comprehensive delay macro modeling for submicrometer CMOS logics. *IEEE Journal of Solid-State Circuits* **34(1)** (January 1999)
7. Bellido-Diaz, M. J., Juan-Chico, J., Acosta, A. J., Valencia, M., Huertas, J. L.: Logical modelling of delay degradation effect in static CMOS gates. *IEE Proc. Circuits Devices and Systems* **147(2)** (April 2000) 107–117
8. Juan-Chico, J., Ruiz-de-Clavijo, P., Bellido, M. J., Acosta, A. J., Valencia, M.: Inertial and degradation delay model for CMOS logic gates. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Geneva, (May 2000) 1–459–462
9. Rezzoug, M., Maurine, P., Auvergne, D.: Second Generation Delay Model for Submicron CMOS Process. In *Proc. 10th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Göttingen (Germany), (2000) 159–167
10. Chatzigeorgiou, A., Nikolaidis, S., Tsoukalas, I.: A modeling technique for CMOS gates. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **18(5)** (May 1999) 557–575
11. Jun, Y., Jun, K., Park, S.: An accurate and efficient delay time modeling for MOS logic circuits using polynomial approximation. *IEEE Transactions on CAD* **8(9)** (September 1989) 1027–1032

# Physical Extension of the Logical Effort Model

B. Lasbouygues<sup>1</sup>, R. Wilson<sup>1</sup>, P. Maurine<sup>2</sup>, N. Azémard<sup>2</sup>, and D. Auvergne<sup>2</sup>

<sup>1</sup> STMicroelectronics Design Department 850 rue J. Monnet, 38926, Crolles, France  
{benoit.lasbouygues, robin.wilson}@st.com

<sup>2</sup> LIRMM, UMR CNRS/Université de Montpellier II, (C5506),  
161 rue Ada, 34392 Montpellier, France  
{pmaurine, azemard, auvergne}@lirmm.fr

**Abstract.** The logical effort method has appeared very convenient for fast estimation and optimization of single paths. However it necessitates a calibration of all the gates of the library and appears to be sub-optimal for a complex implementation. This is due to the inability of this model in capturing I/O coupling and input ramp effects. In this paper, we introduce a physically based extension of the logical effort model, considering I/O coupling capacitance and input ramp effects. This extension of the logical effort model is deduced from an analysis of the supply gate switching process. Validation of this model is performed on 0.18 $\mu\text{m}$  and 0.13 $\mu\text{m}$  STM technologies. Application is given to the definition of a compact representation of CMOS library timing performance.

## 1 Introduction

In their seminal book [1], I. E. Sutherland and al introduced a simple and practical delay model. They developed a logical effort method allowing designers to get a good insight of the optimization mechanisms, using easy hand calculations. However this model suffers of a limited accuracy when strong timing constraints have to be imposed on combinatorial paths. An empirical extension of the logical effort model has been proposed in [2] for considering the input ramp effect. However the I/O coupling capacitance effect, first introduced by Jeppson [3], remains neglected in this method.

Moreover, with the dramatic increase of the leakage current, designers may be willing to design with multiple threshold voltage CMOS (MTCMOS) and several supply voltage values. As a consequence, it is necessary to develop a delay model allowing the designers to deal with multiple  $V_{\text{TH}}$  and  $V_{\text{DD}}$  values, with reduced calibration time penalty.

In this paper we introduce a physical extension of the logical effort. Both the I/O coupling and input ramp effects are considered together with the timing performance sensitivities to the  $V_{\text{TH}}$  and  $V_{\text{DD}}$  values.

The rest of the paper is organized as followed. In section 2, starting from the Sakurai's alpha power law [4], we first physically justify and then extend the logical effort model. Section 3 is devoted to the validation of the extended model and to its application to the definition of a new timing performance representation. Section 4 discusses briefly how to increase the accuracy in choosing the sampling points to be reported in traditional look up tables before to conclude in section 5.

## 2 Timing Performance Model

In the method of logical effort [1] the delay of a gate is defined by

$$d = \tau \cdot (p + gh) \quad (1)$$

$\tau$  is a constant that characterizes the process to be used,  $p$  is the gate parasitic delay. It mainly depends on the source/drain diffusion capacitance of the transistors.  $g$ , the logical effort of the gate, depends on the topology of the gate.  $h$ , the electrical effort or gain, corresponds to the ratio of the gate output loading capacitance to its input capacitance value. It characterizes the driving possibility of the gate.

Using the inverter as a reference these different parameters can be determined from electrical simulation of each library cell. However, if this simple expression can be of great use for optimization purpose no consideration is given to the input slew effect on the delay and to the input-to-output coupling [3]. As a result, the different parameter values cannot be assumed constant over the design space.

As it can be shown from [5], eq.1 characterizes the gate output transition time. However, timing performance must be specified in terms of cell input (output) transition time and input-to-output propagation delay. A realistic delay model must be input slope dependent and must distinguish between falling and rising signals. In order to get a more complete expression for the gate timing performance, let us develop eq.1, starting from a physical analysis of the switching process of an inverter. We first consider a model for the transition time.

### 2.1 Inverter Transition Time Modeling

Following [5], the elementary switching process of a CMOS structure, and thus of an inverter, can be considered as an exchange of charge between the structure and its output loading capacitance. The output transition time (defining the input transition time of the following cell) can then be directly obtained from the modeling of the charging (discharging) current that flows during the switching process of the structure and from the amount of charge ( $C_{L-TOT} \cdot V_{DD}$ ) to be exchanged with the output node as

$$\tau_{outHL} = \frac{C_{L-TOT} \cdot V_{DD}}{I_{NMax}} \quad \tau_{outLH} = \frac{C_{L-TOT} \cdot V_{DD}}{I_{PMax}} \quad (2)$$

$V_{DD}$  is the supply voltage value and  $C_{L-TOT}$  the total output load which is the sum of two contributions:  $C_{L-TOT} = C_{INT} + C_{EXT}$ .  $C_{INT}$  is the internal load proportional to the gate input capacitance, constituted of the coupling capacitance,  $C_M$ , between the input and output nodes [3] and of the transistor diffusion parasitic capacitance,  $C_{DIFF}$ . Note that  $C_M$  can be evaluated as one half the input capacitance of the P(N) transistor for input rising (falling) edge [6], or directly calibrated on Hspice simulation.  $C_{EXT}$  is the load of the output node, including the interconnect capacitive component.

In eq.2 the output voltage variation has been supposed linear and the driving element considered as a constant current generator delivering the maximum current available in the structure. Thus the key point here is to determine a realistic value of this maximum current. Two controlling conditions must be considered.



*Fast input control conditions*

In the Fast input range, the input signal reaches its maximum value before the output begins to vary, in this case the switching current exhibits a constant and maximum value:

$$I_{MAX}^{Fast} = K_{N,P} \cdot W_{N,P} \cdot (V_{DD} - V_{TN,P})^{\alpha_{N,P}} \tag{3}$$

This is directly deduced from the Sakurai’s representation [4],  $\alpha_{N,P}$  being the velocity saturation indexes of N, and P transistors,  $K_{N,P}$  is an equivalent conduction coefficient to be calibrated on the process.

From (2, 3) we obtain the expression of the transition time for a Fast input control condition as

$$\begin{aligned} \tau_{outHL}^{Fast} &= \tau \cdot (1+k) \cdot \frac{C_{L-TOT}}{C_{IN}} \equiv \tau \cdot (p_{HL} + g_{HL} \cdot h) \\ \tau_{outLH}^{Fast} &= \tau \cdot \frac{(1+k)}{k} \cdot R \cdot \frac{C_{L-TOT}}{C_{IN}} \equiv \tau \cdot (p_{LH} + g_{LH} \cdot h) \end{aligned} \tag{4}$$

$\tau$  and R, defined below, are respectively a unit delay characterizing the process, and the dissymmetry between N and P transistors while k is the transistor P/N width ratio.  $C_{IN}$  is the gate input capacitance.

$$\tau = \frac{C_{OX} \cdot L_{GEO} \cdot V_{DD}}{K_N \cdot (V_{DD} - V_{TN})^{\alpha_N}} \quad R = \frac{K_N \cdot (V_{DD} - V_{TN})^{\alpha_N}}{K_P \cdot (V_{DD} - V_{TP})^{\alpha_P}} \tag{5}$$

Finally  $p_{HL,LH}$  and  $g_{HL,LH}$  are the parameters characterizing the topology of the inverter under consideration. They are defined by:

$$\begin{aligned} p_{HL} &= \frac{(1+k) \cdot (C_{MHL} + C_{DIFF})}{C_{IN}} & g_{HL} &= (1+k) \\ p_{LH} &= R \cdot \frac{(1+k) \cdot (C_{MLH} + C_{DIFF})}{k \cdot C_{IN}} & g_{LH} &= R \cdot \frac{(1+k)}{k} \end{aligned} \tag{6}$$

As clearly shown the eq.4-6 constitute an explicit representation of the logical effort model [1] for non-symmetrical inverters.

*Slow input control conditions*

In the slow input range, the transistor is still in saturation when its current reaches the maximum value but its gate source voltage is smaller. This results in a smaller value of the maximum switching current. Let us evaluate this value. From the alpha power law model we can write

$$I_N(t) = K_N \cdot W_N \cdot \left( \frac{V_{DD} \cdot t}{\tau_{IN}} - V_{TN} \right)^{\alpha_N} \tag{7}$$

where  $\tau_{IN}$  is the transition time of the signal applied to the gate input. This leads to

$$\frac{dI}{dt} = \frac{\alpha_N \cdot K_N \cdot W_N \cdot V_{DD}}{\tau_{IN}} \cdot \left( \frac{V_{DD} \cdot t}{\tau_{IN}} - V_{TN} \right)^{\alpha_N - 1} \tag{8}$$

Defining  $\Delta t$  as the time spent by the transistor to deliver its maximum current, (8) becomes:

$$\frac{\Delta I}{\Delta t} = \frac{\alpha_N \cdot K_N \cdot W_N \cdot V_{DD}}{\tau_{IN}} \cdot \left( \frac{V_{DD} \cdot \Delta t}{\tau_{IN}} \right)^{\alpha_N - 1} \equiv \frac{I_{NMAX}}{\Delta t} \quad (9)$$

Then under the approximation that the current variation is symmetric with respect to its maximum value, we can evaluate the total charge removed at the output node as:

$$\frac{C_{L-TOT} \cdot V_{DD}}{2} = \frac{I_{MAX} \cdot \Delta t}{2} \quad (10)$$

Combining eq.9 and 10, we obtain the value of the maximum current for a slow input ramp applied to the input as

$$I_{N-MAX}^{Slow} = \left\{ \frac{\frac{1}{\alpha_N} \cdot \frac{1}{K_N} \cdot \frac{1}{W_N} \cdot C_{L-TOT} \cdot V_{DD}^2}{\tau_{IN}} \right\}^{\frac{\alpha_N}{1+\alpha_N}} \quad (11)$$

Finally, by combining eq.2 and 11, the expression of the transition time for slow input control condition is obtained as

$$\tau_{outHL,LH}^{Slow} = \left\{ \left( \frac{V_{DD} - V_{TN,P}}{\frac{1}{\alpha_{N,P}} \cdot V_{DD}} \right)^{\alpha_{N,P}} \cdot \tau_{outHL,LH}^{Fast} \cdot \tau_{IN}^{\alpha_{N,P}} \right\}^{\frac{1}{1+\alpha_{N,P}}} \quad (12)$$

Defining the supply voltage effort as

$$U_{N,P} = \frac{V_{DD} - V_{TN,P}}{\frac{1}{\alpha_{N,P}} \cdot V_{DD}} \quad (13)$$

results in a logical effort like expression of the output transition time for a slow input ramp condition

$$\tau_{outHL,LH}^{Slow} = \left\{ U_{N,P}^{\alpha_{N,P}} \cdot \tau_{IN}^{\alpha_{N,P}} \cdot \tau \cdot (p_{HL,LH} + g_{HL,LH} \cdot h) \right\}^{\frac{1}{1+\alpha_{N,P}}} \quad (14)$$

Note here that for advanced processes, in which the carrier speed saturation dominates ( $\alpha_{N,P}=1$ ), this expression can be simplified

$$\tau_{outHL,LH}^{Slow} = \sqrt{U_{N,P} \cdot \tau_{IN} \cdot \tau \cdot (p_{HL,LH} + g_{HL,LH} \cdot h)} \quad (15)$$

with

$$U_{N,P} = \frac{V_{DD} - V_{TN,P}}{V_{DD}} \quad (16)$$

*Unifying Fast and Slow domain representation*

Let us now consider the case of a rising edge applied to a gate input. As it can be deduced from (4) and (14), the logical effort delay model

$$\tau_{outHL,LH}^{Fast} = \tau \cdot (p_{HL,LH} + g_{HL,LH} \cdot h)$$

can be used as a metric for both Fast and Slow input ramp domains. Indeed, considering the sensitivity of the different expressions to the input slope, we can express the normalized inverter output transition time as

$$\frac{\tau_{outHL,LH}}{\tau_{outHL,LH}^{Fast}} = \text{Max} \left[ \begin{matrix} 1 \\ \left\{ U_{N,P} \cdot \sigma_{HL,LH} \right\}^{\frac{\alpha_{N,P}}{1+\alpha_{N,P}}} \end{matrix} \right] \tag{17}$$

where  $\sigma_{HL,LH}$  is equivalent to an input slew effort

$$\sigma_{HL,LH} = \frac{\tau_{IN}}{\tau_{outHL,LH}^{Fast}} \tag{18}$$

Eq.17 is of great interest, since it clearly shows that the output transition time of any inverter of a given library can be represented by a single expression. As shown, the right part of eq. 17 is design parameter independent. As a result this gives the opportunity, in a library, to characterize the complete set of drive of an inverter by one look up table of one line. This is obtained by using a representation relative to the slew effort parameter  $\sigma$ . As a validation of these results we represent in Fig.1, the output transition time variation (with respect to  $\sigma_{HL}$ ) of 7 inverters of a 0.13 $\mu\text{m}$  process. As expected all the curves pile up on the same one, representing the output transition time sensitivity to the slew effort  $\sigma_{HL}$ .

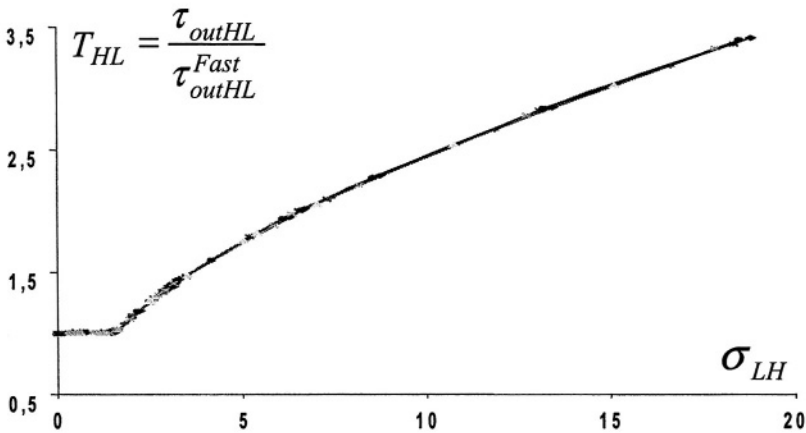


Fig. 1. Output transition time of the 7 inverters of a 0.13 $\mu\text{m}$  process

## 2.2 Extension to Gates

Following [7], the extension to gates is obtained by reducing each gate to an equivalent inverter. For that we consider the worst-case situation. The current capability of the N (P) parallel array of  $m$  transistors is evaluated as the maximum current of an inverter with identically sized transistors. The array of N (P) series-connected transistors is modeled as a voltage controlled current generator with a current capability reduced by a factor  $DW_{HL,LH}$ . This reduction factor (DW) is defined as the ratio of the current available in an inverter to that of a series-connected array of transistors of identical size.

$$DW_{HL,LH} = \frac{I_{N,P}(Inv)}{I_{N,P}(Gate)} \cdot \frac{W_{N,P}(Inv)}{W_{N,P}(Gate)} \quad (19)$$

DW corresponds to the explicit form of the logical effort [3]. Let us evaluate the expression of  $DW_{HL,LH}$ . In the Fast input range, the maximum current that can provide an array of  $n$  serially connected transistors is defined by:

$$\frac{I_R}{K_{N,P} \cdot W_{N,P}} = \{V_{DD} - V_{TN,P} - (n-1)R_{N,P} \cdot I_{ARRAY}\}^{\alpha_{N,P}} \quad (20)$$

where  $R_{N,P}$  is the resistance of the bottom transistors working in linear mode. Using a first order binomial decomposition of (20), the reduction factor for Fast input controlling conditions is obtained as

$$DW_{HL,LH} = 1 + \alpha_{N,P} \cdot K_{N,P} \cdot W_{N,P} \cdot (n-1) \cdot R_{N,P} \cdot (V_{DD} - V_{TN,P})^{\alpha_{N,P}-1} \quad (21)$$

which is a generalization of the result introduced in [7] for a DSM process, with full carrier speed saturation ( $\alpha=1$ ), where the value of the reduction factor reduces to

$$DW_{HL,LH} = 1 + K_{N,P} \cdot W_{N,P} \cdot (n-1) \cdot R_{N,P} \quad (22)$$

Using this reduction factor of the maximum current (21) to evaluate the output transition time (2) results in the normalized gate output transition time expression

$$\frac{\tau_{outHL,LH}}{\tau_{outHL,LH}^{Fast}} = Max \left[ \begin{array}{c} 1 \\ \left\{ U_{N,P} \cdot \sigma_{HL,LH} \right\}^{\frac{\alpha_{N,P}}{1+\alpha_{N,P}}} \end{array} \right] \quad (23)$$

where

$$\tau_{outHL,LH}^{Fast} = \tau \cdot (p_{HL,LH} + g_{HL,LH} \cdot h) \quad (24a)$$

and

$$\begin{aligned} p_{HL} &= DW_{HL} \cdot \frac{(1+k) \cdot (C_{MHL} + C_{DIFF})}{C_{IN}} & g_{HL} &= DW_{HL} \cdot (1+k) \\ p_{LH} &= R \cdot DW_{LH} \cdot \frac{(1+k) \cdot (C_{MLH} + C_{DIFF})}{k \cdot C_{IN}} & g_{LH} &= R \cdot DW_{LH} \cdot \frac{(1+k)}{k} \end{aligned} \quad (24b)$$

As shown p and g, the parasitic contribution to the gate delay and the logical effort, strongly depend on the topology of the considered cell through the parameters k and DW. Expression (23) constitutes an extension of the logical effort model considering the input ramp effect.

### 2.3 Gate Propagation Delay Model

A realistic delay model must be input slope dependent and must distinguish between falling and rising signals. As developed by Jeppson in [3], considering the input-to-output coupling effect, the input slope effect can be introduced in the propagation delay  $\theta_{HL,LH}$  as

$$\begin{aligned}
 t_{HL} &= \frac{v_{TN}}{2} \tau_{IN} + \left(1 + \frac{2C_M}{C_M + C_L + C_{DIFF}}\right) \frac{\tau_{outHL}}{2} \\
 t_{LH} &= \frac{v_{TP}}{2} \tau_{IN} + \left(1 + \frac{2C_M}{C_M + C_L + C_{DIFF}}\right) \frac{\tau_{outLH}}{2}
 \end{aligned}
 \tag{25}$$

$\tau_{IN}$  ( $\tau_{out,HL,LH}$ ) is the transition time of the input (output) signal, generated by the controlling gate. As shown, for each edge, the delay expression is a linear combination of the output transition time of the controlling and the switching gate.

Normalizing (25) with respect to the metric  $\tau_{out}^{Fast}$ , gives a generic gate propagation delay expression for all the drives of a cell

$$\Theta_{HL,LH} = \frac{t_{HL,LH}}{\tau_{outHL,LH}^{Fast}} = \frac{v_{TN,P}}{2} \sigma_{HL,LH} + \frac{2 \cdot \alpha_{HL,LH}}{(\alpha_{HL,LH} + A) + h} + \frac{\tau_{outHL,LH}}{2 \cdot \tau_{outHL,LH}^{Fast}}
 \tag{26}$$

In this equation  $\alpha_{HL,LH}$  are the Meyer coefficients [6] for falling and rising edges that can be calibrated on the process the average value is  $\alpha = 0.5$ ). The parameter A is related to the drain diffusion capacitance ( $C_{DIFF} = A \cdot C_{IN}$ ).

For a typical cell, the second term of eq.26 could be neglected for value of the electrical effort h greater than 3 or equivalently for important value of the input slew effort  $\sigma_{HL,LH}$ .

Although this term becomes quickly negligible, we have to note here that it can have a significant effect when imposing a small value of the electrical effort. It directly shows the minimum limit of the electrical effort value to be reasonably imposed on a path. Indeed, for small values of h (<2), the first and third terms of (25) become smaller than the second one. Any further increase of the transistor width along the path is inefficient in improving the corresponding gate speed that is limited by the I/O coupling effect and the parasitic content of the gate.

Combining finally (25) and (23) gives a general expression of the normalized propagation delay of any combinational gate as:

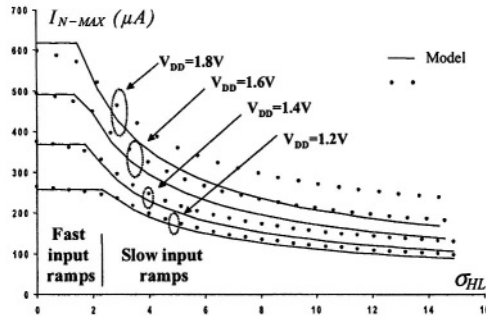
$$\Theta_{HL,LH} = \left[ \frac{1 + v_{TN,P} \cdot \sigma_{HL,LH}}{2} \right] + \frac{2 \cdot \alpha_{HL,LH}}{(\alpha_{HL,LH} + A) + h}
 \tag{27}$$

$$\left[ \frac{v_{TN,P} \cdot \sigma_{HL,LH} + \left\{ U_{N,P} \cdot \sigma_{HL,LH} \right\}^{iv\alpha_{N,P}}}{2} \right]$$

This expression is of great interest. A part the last term, that is negligible in a typical design range (h between 3 and 6), eq.27 clearly indicates that the propagation delay of any gate of a library can be represented with only one equation.

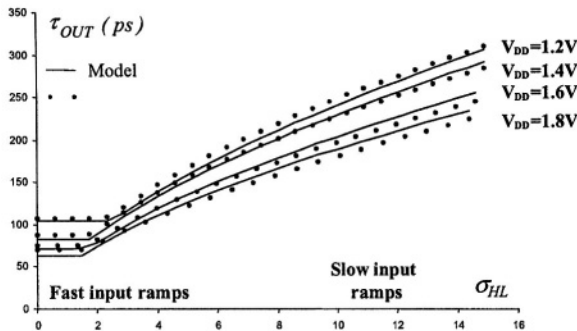
### 3 Validation

In order to validate this model we first compare the estimation of performance of an inverter, designed in a  $0.18\mu\text{m}$  process, calculated from eq.3, 11 and 17, to values obtained from Hspice simulations. Different supply voltage conditions have been considered. The value of the index saturation index has been obtained from direct calibration on the transistor current simulation. Fig.2-3 give some example of the calculated and simulated evolutions of both the inverter maximum switching current and output transition time with respect to  $\sigma_{HL}$ .



**Fig. 2.** Comparison between simulated and calculated values of the maximum switching current provided by an inverter ( $W_N=1\mu$   $k=2$ ,  $l=0.18\mu\text{m}$ ,  $F_o=C_L/C_{IN}=5$ ) for different supply voltage values.

As shown the accuracy of the model is satisfactory. Note that the underestimation of the switching current for high supply voltage is mainly due to the short circuit current. This has a minor impact on the evaluation of the output transition time that is determined from the 40% and 60% points of the signal voltage swing.



**Fig. 3.** Comparison between simulated and calculated values of the output transition time of an inverter ( $W_N=1\mu$   $k=2$ ,  $l=0.18\mu\text{m}$ ,  $F_o=C_L/C_{IN}=5$ ) for different supply voltage values.

A more global validation has also been performed. Following eq.23 and 26, and using the appropriate representation, it is easy to deduce from the model that the transition time and the propagation delay variations of the different drive possibilities of a gate can be represented by a set of four laws, one by edge of each performance parameter. We validate this representation by representing in Fig.1, 4-6 the simulated input transition time sensitivities of the output transition time and delay of inverter, Nand and Nor gates, from a 0.13 $\mu\text{m}$  library (at least five drive each). All the values are normalized with respect to  $\tau_{out}^{Fast}$ . As shown, all the simulated sampling points extracted from the Timing Library Format, corresponding to a particular gate, pile up on one curve. This clearly demonstrates that, using  $\tau_{out}^{Fast}$  as a metric of performance, it is possible not only to simplify the library timing performance representation [8], but also to minimize the number of data to be simulated for the complete library characterization.

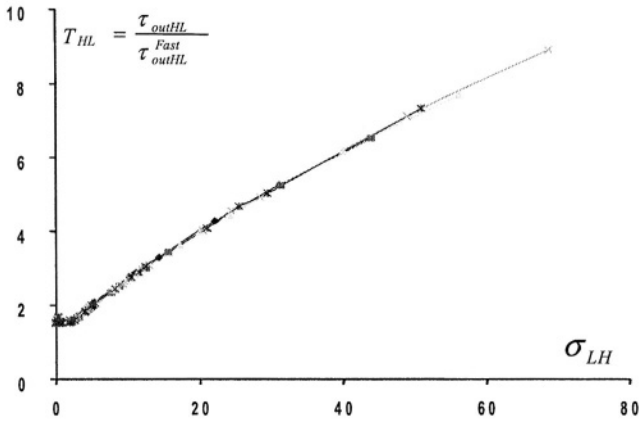


Fig. 4. Output transition time representation of the 5 Nand2 of a 0.13 $\mu\text{m}$  process

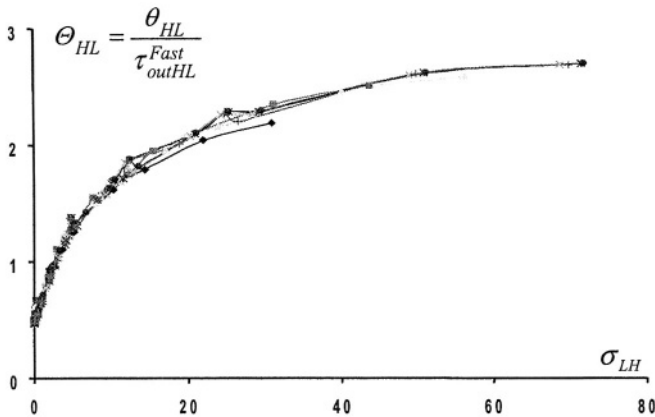


Fig. 5. Propagation delay representation of the 7 inverters of a 0.13 $\mu\text{m}$  process.

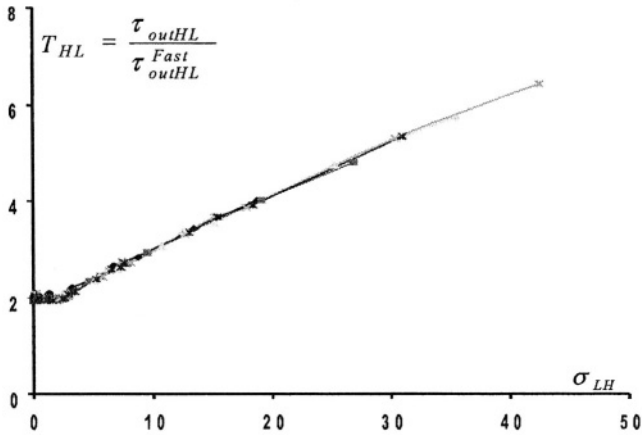


Fig. 6. Output transition time representation of the 5 Nor2 of a 0.13 $\mu\text{m}$  process

## 4 Discussion

In submicron process the transition time and the propagation delay exhibit a non-linear variation with respect to the controlling and loading conditions (Fig. 1,4-6). This non-linear range must clearly be determined with closest simulation steps because interpolating in this range may induce significant errors. It is obvious that the relative accuracy, obtained with a tabular method, is strongly dependent on the granularity of the table that is not necessarily constant but must cover a significant part of the design range. For that, indications for defining the granularity and the coverage of the design space must be available. As illustrated in Fig. 1,4-6, the non-linearity corresponds to the Fast /Slow input ramp domain boundary. Thus the evaluation of this boundary is of great interest to determine the data points to be sampled and reported in the look up tables. The proposed model offers an easy way to determine this boundary. It is just necessary to find the controlling and loading conditions for which eq.1 and 17 have the same value. This gives

$$\tau_{IN}^{lim} = \frac{\tau}{U_{N,P}^{\alpha_{N,P}}} \left\{ P_{HL,LH} + g_{HL,LH} \cdot \frac{C_L}{C_{IN}} \right\} \quad (28)$$

Then to increase the accuracy of the tabular approach it is necessary to fix the granularity of the table in such a way that the data points belonging to the diagonal of the table respect the condition defined by eq.28.

## 5 Conclusion

We have introduced an explicit extension of the logical effort model in order to consider both the I/O coupling and the input ramp effects and defined a simple but accu-



rate representation of the timing performance of the simple CMOS structures. Validation of the model has been done on  $0.18\mu\text{m}$  and  $0.13\mu\text{m}$  processes. Application has been given to the definition of a compact representation of CMOS library timing performance. As discussed this representation must be of great interest in defining the granularity and the evolution of the look up tables used to represent the timing performance of CMOS library.

## References

- [1] I. Sutherland, B. Sproull, D. Harris, "Logical Effort: Designing Fast CMOS Circuits", Morgan Kaufmann Publi., California, 1999.
- [2] X. Yu, V. G. Oklobdzija, W/ Walker, "Application of the logical effort on design arithmetic blocks" 35<sup>th</sup> Asilomar Conf. On Signals, Systems and Computers, California, Nov. 2001,pp 872-874
- [3] K. O. Jeppson, "Modeling the influence of the transistor gain ratio and the input-to-output coupling capacitance on the CMOS inverter delay", IEEE J. SSC, vol.29, pp.646-654, 1994.
- [4] T. Sakurai and A.R. Newton, "Alpha-power model, and its application to CMOS inverter delay and other formulas", J. of Solid State Circuits vol.25, pp.584-594, April 1990.
- [5] C. Mead and L. Conway, "Introduction to VLSI systems", Reading MA: Addison Wesley 1980.
- [6] J. Meyer "Semiconductor Device Modeling for CAD" Ch. 5, Herskowitz and Schilling ed. Mc Graw Hill, 1972.
- [7] "Not given for anonymous review" IEEE trans. on Computer Aided Design, 2002.
- [8] Cadence open book "Timing Library Format References" User guide, v. IC445, 2000.

# An Extended Transition Energy Cost Model for Buses in Deep Submicron Technologies

Peter Caputa, Henrik Fredriksson, Martin Hansson,  
Stefan Andersson, Atila Alvandpour, and Christer Svensson

Electronic Devices, Dept. of E.E., Linköping University,  
SE-581 83 Linköping, Sweden  
Tel. +46 13 28{2379, 2254, 2859, 2671, 5818, 1223}  
Fax. +46 13 13 92 82

{petca,henfr,martinh,steana,atila,christer}@isy.liu.se

**Abstract.** In this paper we present and carefully analyze a transition energy cost model aimed for efficient power estimation of performance critical deep submicron buses. We derive an accurate transition energy cost matrix, scalable to buses of arbitrary bit width, which includes properties that closer capture effects present in high-performance VLSI buses. The proposed energy model is verified against Spectre simulations of an implementable bus, including drivers. The average discrepancy between results from Spectre and the suggested model is limited to 4.5% when fringing effects of edge wires is neglected. The proposed energy model can account for effects that limit potential energy savings from bus transition coding.

## 1 Introduction

In today's era of advanced deep submicron (DSM) semiconductor technologies, the power consumption due to pure computation tends to decrease with scaling, while the cost of on-chip and inter-chip communication has an increasing impact on total power consumption. This is mainly due to the relative scaling of cell capacitances, and the increase of inter-wire capacitance due to an enlarged wire aspect ratio aimed at reducing the wire resistance. Interconnects tend to dominate the chip power consumption, due to their large total capacitance [[1],[2]]. Excessive power consumption heats the die and causes on-chip leakage currents to increase, which considerably reduces battery life time of mobile products and requires efficient and bulky cooling devices. Since global interconnects tend to consume an increasing amount of power, it is highly motivated to study and improve this undesirable situation [[3]].

A simple power estimator is based on transition activity. The transition activity,  $\alpha$ , of a circuit node is an accurate power measure when the node is decoupled (i.e. no cross-talk) from any other active node. If this is true, the power consumption can be calculated using the well known formula  $P = \frac{1}{2}\alpha f C V_{dd}^2$ , where  $C$  is the capacitance between the node and ground and  $f$  is the operating frequency. Unfortunately, as technology scales, not many interconnects can be

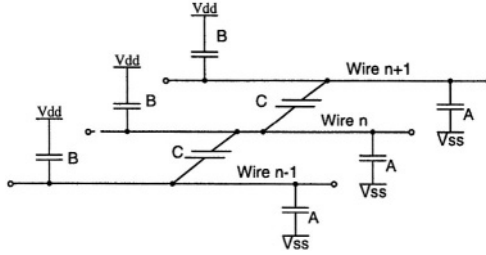
treated as isolated. In a DSM process, the inter-wire capacitance between adjacent wires is significantly larger than the capacitance between a wire and the substrate (self capacitance) [[4],[5]]. However, in most cases there is a non-zero density of metal between the global interconnect layer and the substrate. In a bus energy model, one should therefore not neglect the inter-layer capacitance and only include inter-wire capacitance between wires routed in the same layer as done in [[6]].

Many different schemes to decrease energy dissipation have been presented, such as low-swing signaling [[7]], usage of repeaters [[8]], charge recycling [[9]], and data coding [[10]]. Lately, bus coding has become a very hot topic. The idea is to code the data on the bus to avoid transitions that are expensive from an energy consumption point of view [[3]]. To determine which transitions are expensive, we need an accurate bus energy model. The model should include all relevant parameters but still be simple enough so that it can be used in practice. It is also important to know when to, or not to, code a bus, i.e. when the encoder and decoder together with the coded bus consumes more energy than the uncoded bus. One should also keep in mind that the data rate of a coded bus is lower than that of an uncoded bus of the same width. By making the assumption to omit the transient behavior of the bus signals, as in [[6]], and only analyze the energy difference between two consecutive bus states, we present a DSM bus model that more accurately accounts for effects that appear in VLSI on-chip bus designs. In this work we propose a bus energy model that includes both inter-wire and inter-layer capacitances for buses. We also include the internal capacitances of the bus drivers to obtain an energy model, which provides a better analysis of possible energy savings from bus transition coding. The proposed high-level bus energy model can be used as an essential tool to determine the best type of coding and to estimate the energy consumption of interconnects.

## 2 Proposed DSM Bus Model

### 2.1 Wire Model

Our main interest is aimed at transition energy consumption. Thus, we have chosen a lumped representation of all relevant capacitances in our bus model. Fig. 1 depicts the bus model for three parallel wires. Wire capacitance A (B) is the accumulated inter-layer capacitance between bus wire  $n$  and  $V_{ss}$  ( $V_{dd}$ ), including drain capacitance for the nMOS (pMOS) transistor in the last driver stage. During a switching event, short circuit current will flow through the driver causing charge to be drawn from  $V_{dd}$ . We model this charge flow by an equivalent compensation capacitance included in A and B. By adding the capacitance between  $V_{dd}$  and the wire, we add a cost of discharging the wires. In high-performance VLSI systems, one cannot afford the area penalty involved in assigning channels solely for buses. Interconnects, in standard logic circuits, are driven by transistors to either  $V_{dd}$  or  $V_{ss}$ . Therefore, capacitive coupling between a bus and any surrounding interconnect can statistically be modeled as equally sized capacitances to  $V_{dd}$  and  $V_{ss}$ . Thus, our model will handle situations when we have a

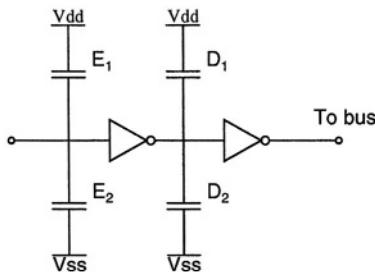


**Fig. 1.** Capacitance model of a 3-bit bus.

non-zero metal density in layers above and beneath the bus.  $C$  is the inter-wire capacitance between adjacent wires. In DSM buses this capacitive component tends to be dominating.

## 2.2 Driver Model

In most previous work the bus driver has been a single CMOS inverter and for transition energy considerations, only the total inverter output capacitance to ground has been modeled. In reality, a long bus wire needs a strong driver to provide high-speed signal transitions. Unfortunately, the logic prior to the driver cannot drive a large inverter directly without considerable delay penalty. Thus, a chain of inverters, progressively up-scaled by a tapering factor, is commonly inserted. We propose a driver model that not only considers the input node capacitance to both  $V_{ss}$  and  $V_{dd}$ , but also uses two cascaded ideal inverters to drive the bus wire. When multiple inverters are used, the discharging of a wire causes charging of intermediate nodes in the driver chain, thus adding to the total cost. Our proposed driver model is shown in Fig. 2. For the first inverter stage,  $E_1$  and  $E_2$  is the total input node capacitance to  $V_{ss}$  and  $V_{dd}$ , respectively. Similarly for the second stage,  $D_1$  and  $D_2$  represent the total input node capacitance to  $V_{dd}$  and  $V_{ss}$ , respectively. The capacitances associated with any buffers inserted prior to the two main driver inverters can be included in  $D$  and  $E$ .



**Fig. 2.** Proposed driver model.

### 2.3 Model Parameters

From the capacitance parameters in the wire and driver models, we are able to create a cost matrix that gives the total energy transition cost when going from one bus state to another. For a more convenient analysis we derive model parameters that are ratios between these capacitance values. These ratios are similar to the ones presented in [[6]]. The following parameters are derived from Fig. 1 and Fig. 2:

$$\lambda = \frac{C}{A + B} \quad (1)$$

$$\beta = \frac{A}{A + B} \quad (2)$$

$$\alpha = \frac{A + B + 2C}{D_1 + D_2} \quad (3)$$

$$\gamma = \frac{E_1}{E_2} = \frac{D_1}{D_2} \quad (4)$$

$$\phi = \frac{D_1}{E_1} \quad (5)$$

The cross coupling factor  $\lambda$  is the ratio between total inter-wire capacitance and inter-layer capacitance.  $\lambda$  is typically zero in bus models of older silicon technologies where only inter-layer capacitance is considered.  $\beta$  is the ratio between inter-layer capacitance to  $V_{ss}$  and total inter-layer capacitance. The model used in [[6]] has  $\beta=1$ . For a bus routed over synthesized logic, the value of  $\beta$  will decrease, which redistributes the amount of inter-layer capacitance between  $V_{dd}$  and  $V_{ss}$ , respectively.  $\gamma$  is the pMOS/nMOS scaling ratio, which is typically around 2.5 [[11]].  $\phi$  is the tapering factor, typically set to 3 [[11]], of the two inverters in the driver chain.  $\alpha$  states how large the total bus capacitance (excluding miller effect) is compared to the input node capacitance of the last inverter driver. The drivers are sized to comfortably drive the middle bus wires. The capacitive load on the two edge wires is smaller as only one inter-wire capacitance is present.

### 3 Transition Table Derivation

In order to calculate the amount of charge drawn from the supply rail at a specific bus transition, we need to know the sum of all capacitances charged with current from the supply. To do this we first define a bit-transition-cost function  $R_n(i_{n+1}, i_n, i_{n-1}; f_{n+1}, f_n, f_{n-1})$ , which is the capacitance charged with current from the supply rail and the driver connected to bit  $n$  when going from initial bus state  $i$  to final state  $f$ . We also define the function  $S_n(i_{n+1}, i_n; f_{n+1}, f_n)$  for the case when bit  $n$  has only one adjacent line. Using the lumped capacitances from our previously defined DSM bus model, the functions  $R_n$  and  $S_n$  are shown in Table 1 and Table 2, respectively. In Table 1 and Table 2, *up* is the effective capacitance being charged when bus line  $n$  changes state from 0 to 1 as shown in eq.(6), while *down* is the effective capacitance being charged when bus line  $n$

**Table 1.** Bit-transition-cost function  $R_n(i_{n+1}, i_n, i_{n-1}; f_{n+1}, f_n, f_{n-1})$ , for wires having two neighbors.

$i_{\downarrow} f^{\rightarrow}$	000	001	010	011	100	101	110	111
000	0	0	$up + 2C$	$up + C$	0	0	$up + C$	$up$
001	0	0	$up + 3C$	$up + 2C$	0	0	$up + 2C$	$up + C$
010	$down$	$down$	0	$-C$	$down$	$down$	$-C$	$-2C$
011	$down$	$down$	$C$	0	$down$	$down$	0	$-C$
100	0	0	$up + 3C$	$up + 2C$	0	0	$up + 2C$	$up + C$
101	0	0	$up + 4C$	$up + 3C$	0	0	$up + 3C$	$up + 2C$
110	$down$	$down$	$C$	0	$down$	$down$	0	$-C$
111	$down$	$down$	$2C$	$C$	$down$	$down$	$C$	0

**Table 2.** Bit-transition-cost function  $S_n(i_{n+1}, i_n; f_{n+1}, f_n)$ , for wires having one neighbor.

$i_{\downarrow} f^{\rightarrow}$	00	01	10	11
00	0	$up + C$	0	$up$
01	$down$	0	$down$	$-C$
10	0	$up + 2C$	0	$up + C$
11	$down$	$C$	$down$	0

changes state from 1 to 0 as shown in eq.(7). To account for the extra inter-layer capacitance present on edge wires, due to fringing effects, one could modify the  $up$  and  $down$  costs for  $S_n$  so that they contain an additional capacitance factor. For wide buses, the end wire fringing effect will have minor impact on overall transition cost, so we choose not to include it here.

$$up = A + D_1 + E_2 = C \left( \frac{\beta}{\lambda} + \left( 1 + \frac{1}{\gamma\phi} \right) \left( \frac{\frac{1}{\lambda} + 2}{\alpha \left( 1 + \frac{1}{\gamma} \right)} \right) \right) \tag{6}$$

$$down = B + D_2 + E_1 = C \left( \frac{1-\beta}{\lambda} + \left( \frac{1}{\gamma} + \frac{1}{\phi} \right) \left( \frac{\frac{1}{\lambda} + 2}{\alpha \left( 1 + \frac{1}{\gamma} \right)} \right) \right) \tag{7}$$

One can utilize the bit-transition-cost functions  $R_n$  and  $S_n$  to derive transition cost tables for buses of arbitrary bit width. The capacitance charged in a  $N$  bit wide bus is the sum of the bit transition costs for each bit. Thus, the total effective capacitance when changing bus state from  $(i_N, i_{N-1}, \dots, i_2, i_1)$  to  $(f_N, f_{N-1}, \dots, f_2, f_1)$  is:

$$C_{tot}(i_N, \dots, i_1; f_N, \dots, f_1) = S_1(i_2, i_1; f_2, f_1) + \sum_{n=2}^{N-1} R_n(i_{n+1}, i_n, i_{n-1}; f_{n+1}, f_n, f_{n-1}) + S_N(i_{N-1}, i_N; f_{N-1}, f_N) \tag{8}$$

As an example, the total effective capacitance being charged when transitioning from state 111010 to 101101 is calculated as  $S_1(10; 01) + R_2(010; 101) + R_3(101; 110) + R_4(110; 011) + R_5(111; 101) + S_6(11; 01) = 2up + 2down + 6C$ .

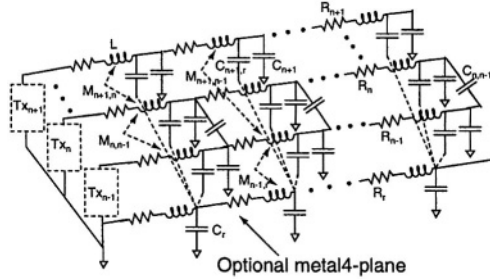


Fig. 3. Distributed RLC-interconnect model.

## 4 Simulation Versus Proposed Model

### 4.1 Spectre Simulation Model

To evaluate the accuracy of the proposed transition energy cost model, we compare it to simulations of a realistic and implementable 4-bit  $3000\ \mu\text{m}$  long global bus, including drivers. All transistor and wire parameters refer to a 6-metal layer  $0.18\ \mu\text{m}$  process (supply voltage  $V_{dd}=1.8\ \text{V}$ ) available in industry. The bus is routed in top metal6 having a thickness of  $1.04\ \mu\text{m}$ . Each signal wire has a square cross section and the wire separation is set to  $1.04\ \mu\text{m}$ . To emulate routing over a mesh of synthesized logic, 20% of the bus length is drawn over a metal4 plane tied to  $V_{dd}$ , while another 20% of the bus length runs over a metal4 plane connected to  $V_{ss}$ . The bus geometry was entered into the 2D electromagnetical field solver available in the circuit simulator HSPICE (version 2003.3). The field solver extracts matrices for parasitic resistance (R), inductance (L), capacitance (C), and skin-effect resistance ( $R_s$ ). Neglecting skin-effect resistance, we create a distributed bus model of discrete RLC-sections (one per  $50\ \mu\text{m}$  interconnect) as shown in Fig. 3. Each conductor has not only a capacitance to the substrate, but also an inter-wire capacitance and mutual inductance to each of the other conductors present in the structure. The values of R, L, and C were obtained from the field solver. Each signal wire in the bus is driven by a chain of 4 inverters, progressively up-scaled by a tapering factor of 3. The last inverter stage has a pMOS to nMOS width ratio of  $W_p/W_n = 60\ \mu\text{m}/24\ \mu\text{m}$ , and an output impedance matched to the line characteristic impedance to achieve signal rise times around  $65\text{ps}$ . All possible transitions on the distributed RLC-bus model, driven by the 4-stage inverter chain, were simulated in the Spectre circuit simulator from Cadence. The total integrated current drawn from the supply was measured for each transition.

## 5 Proposed Model Versus Previous Work

To further evaluate the performance of the proposed energy cost model, we compare its transition cost distribution to models previously presented in literature.

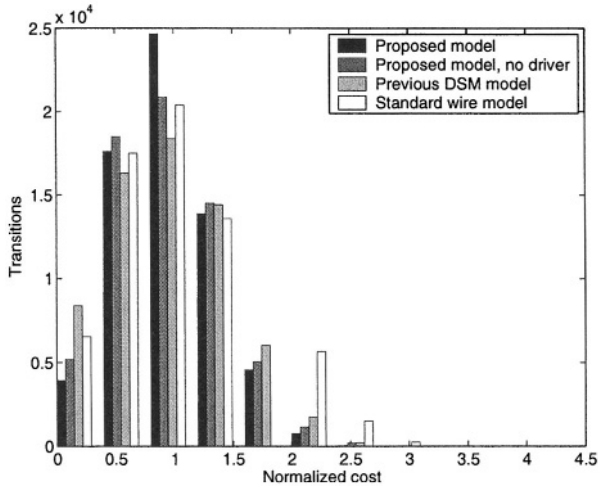
**Table 3.** Relative errors in proposed transition energy cost model compared to Spectre simulated bus structure.

	max [%]	avg [%]	avg(abs) [%]
Full Spectre model	42.38	21.11	21.72
No dedicated end cap	19.54	4.54	5.74
No dedicated end cap, no long cap, no mut ind	9.39	1.27	3.81

**Table 4.** Transition energy cost [pC] for a 4-bit bus. Upper row is Spectre simulation with equalized end capacitances. Lower row is calculated energy using proposed transition energy model where  $\alpha = 3.86$ ,  $\beta = 0.43$ ,  $\lambda = 0.85$ ,  $\gamma = 2.29$ ,  $\phi = 3.08$  and  $C = 196.5 fF$ .

$i_{\downarrow} f^{-1}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0.00	0.87	1.19	1.34	1.19	1.99	1.70	1.77	0.87	1.70	1.99	2.09	1.34	2.09	1.77	1.80
	0.00	0.76	1.12	1.17	1.12	1.88	1.53	1.58	0.76	1.53	1.88	1.93	1.17	1.93	1.58	1.64
0001	0.37	0.00	1.92	0.83	1.60	1.16	2.47	1.30	1.26	0.85	2.74	1.61	1.76	1.28	2.56	1.35
	0.39	0.00	1.86	0.76	1.51	1.12	2.27	1.17	1.15	0.76	2.62	1.53	1.56	1.17	2.33	1.23
0010	0.37	1.60	0.00	0.51	1.91	3.07	0.85	1.29	1.27	2.47	0.84	1.30	2.09	3.21	0.96	1.35
	0.39	1.51	0.00	0.41	1.86	2.98	0.76	1.17	1.15	2.27	0.76	1.17	1.92	3.03	0.82	1.23
0011	0.73	0.73	0.73	0.00	2.31	2.23	1.61	0.81	1.66	1.62	1.59	0.81	2.51	2.39	1.75	0.90
	0.78	0.75	0.75	0.00	2.25	2.22	1.51	0.76	1.55	1.51	1.51	0.76	2.31	2.27	1.56	0.82
0100	0.37	1.27	1.91	2.09	0.00	0.84	0.85	0.96	1.60	2.47	3.07	3.21	0.51	1.30	1.29	1.35
	0.39	1.15	1.86	1.92	0.00	0.76	0.76	0.82	1.51	2.27	2.98	3.03	0.41	1.17	1.17	1.23
0101	0.73	0.40	2.64	1.58	0.40	0.00	1.62	0.49	1.99	1.62	3.82	2.72	0.93	0.49	2.07	0.90
	0.78	0.39	2.61	1.51	0.39	0.00	1.51	0.41	1.90	1.51	3.72	2.62	0.80	0.41	1.92	0.82
0110	0.73	2.00	0.71	1.26	0.71	1.91	0.00	0.47	2.00	3.23	1.91	2.41	1.26	2.41	0.47	0.90
	0.78	1.90	0.75	1.15	0.75	1.86	0.00	0.41	1.90	3.02	1.86	2.27	1.15	2.27	0.41	0.82
0111	1.10	1.13	1.44	0.75	1.11	1.07	0.77	0.00	2.39	2.38	2.66	1.92	1.68	1.60	1.26	0.45
	1.17	1.14	1.49	0.75	1.14	1.10	0.75	0.00	2.29	2.25	2.61	1.86	1.55	1.51	1.15	0.41
1000	0.37	1.26	1.60	1.76	1.92	2.74	2.47	2.56	0.00	0.85	1.16	1.28	0.83	1.61	1.30	1.35
	0.39	1.15	1.51	1.56	1.86	2.62	2.27	2.33	0.00	0.76	1.12	1.17	0.76	1.53	1.17	1.23
1001	0.73	0.39	2.33	1.25	2.33	1.91	3.23	2.09	0.39	0.00	1.91	0.80	1.25	0.80	2.09	0.90
	0.78	0.39	2.25	1.15	2.25	1.86	3.02	1.92	0.39	0.00	1.86	0.76	1.15	0.76	1.92	0.82
1010	0.73	1.99	0.40	0.93	2.64	3.82	1.62	2.07	0.40	1.62	0.00	0.49	1.58	2.72	0.49	0.90
	0.78	1.90	0.39	0.80	2.61	3.72	1.51	1.92	0.39	1.51	0.00	0.41	1.51	2.62	0.41	0.82
1011	1.10	1.12	1.13	0.42	3.04	2.98	2.38	1.60	0.79	0.77	0.75	0.00	2.00	1.91	1.27	0.45
	1.17	1.14	1.14	0.39	3.00	2.96	2.25	1.51	0.78	0.75	0.75	0.00	1.90	1.86	1.15	0.41
1100	0.73	1.66	2.31	2.51	0.73	1.59	1.61	1.75	0.73	1.62	2.23	2.39	0.00	0.81	0.81	0.90
	0.78	1.55	2.25	2.31	0.75	1.51	1.51	1.56	0.75	1.51	2.22	2.27	0.00	0.76	0.76	0.82
1101	1.10	0.79	3.04	2.00	1.13	0.75	2.38	1.27	1.12	0.77	2.98	1.91	0.42	0.00	1.60	0.45
	1.17	0.78	3.00	1.90	1.14	0.75	2.25	1.15	1.14	0.75	2.96	1.86	0.39	0.00	1.51	0.41
1110	1.10	2.39	1.11	1.68	1.44	2.66	0.77	1.26	1.13	2.38	1.07	1.60	0.75	1.92	0.00	0.45
	1.17	2.29	1.14	1.55	1.49	2.61	0.75	1.15	1.14	2.25	1.10	1.51	0.75	1.86	0.00	0.41
1111	1.46	1.52	1.84	1.17	1.84	1.82	1.53	0.79	1.52	1.53	1.82	1.11	1.17	1.11	0.79	0.00
	1.57	1.53	1.88	1.14	1.88	1.84	1.49	0.75	1.53	1.49	1.84	1.10	1.14	1.10	0.75	0.00





**Fig. 4.** Transition cost distribution for proposed model, proposed model excluding driver, previous DSM model [[6]], and standard wire model.

Fig. 4 plots the transition cost distribution histogram for our proposed model (including and excluding the driver), the model in [[6]] (including capacitance to ground and inter-wire capacitance), and a standard bus model (only including capacitance to ground) using the same subset of extracted bus parameters. The costs are based on calculations for an extended 8-bit version of the 4-bit bus discussed in section 5.1. Each transition distribution is normalized to the average cost of each model. The average cost for our proposed model is 1.58 pC (1.15 pC) including (excluding) the driver. The model in [[6]] and the standard wire model have an average cost of 1.15 pC and 0.39 pC, respectively. We make two observations in the histogram of Fig. 4. Firstly, our energy model has a higher concentration around the average cost. This is due to the fact that a substantial part of the inter-layer capacitance consists of drain capacitance from the rather up-sized drivers, making  $\lambda$  small. Secondly, both low-cost and costly transitions occur more infrequently in our proposed model compared to all other models. Any transition causing the bus only to discharge lines (e.g. the first column in Table 4) has a non-zero energy cost, as has not been the case in previous models. This non-zero cost effect is captured by driver and inter-layer capacitances to both  $V_{dd}$  and  $V_{ss}$ , included in our proposed energy model.

## 5.1 Proposed Model Performance

From the bus-driver model in section 4.1 we extract the following parameter values for our proposed transition energy model:  $\alpha = 3.86$ ,  $\beta = 0.43$ ,  $\lambda = 0.85$ ,  $\gamma = 2.29$ ,  $\phi = 3.08$  and  $C = 196.5$  fF. We use the field solver extracted line capacitance to  $V_{dd}$ ,  $V_{ss}$  and nearest neighbor. Moreover, capacitances related to internal nodes in the 4-stage inverter driver chain were derived. Of all driver

related capacitances derived for the energy model, 41% is gate capacitance, 30% is drain capacitance, while the remaining 29% is modeled as an equivalent extra capacitance due to transition short circuit current.

We calculate the energy cost for every possible transition on the 4-bit bus using our proposed energy model. When we compare the results with values obtained from the Spectre simulation, we note an average and maximum deviation of 21.1% and 42.4%, respectively. This difference is mainly caused by two effects not included in our proposed transition energy model. Firstly, due to edge effects, the inter-layer capacitance for the two edge wires having only one neighbor is larger compared to wires having two neighbors. If the inter-layer capacitance for all wires in the bus is equalized in the Spectre simulated structure, the average and maximum deviation between the models decreases to 4.5% and 19.5%, respectively. Secondly, if we discard capacitance between non-neighboring wires as well as all mutual inductances in the Spectre simulated structure, the average and maximum model discrepancy decreases to 1.3% and 9.4%, respectively. These results are summarized in Table 3. The second row of summary Table 3 is based on results explicitly presented in Table 4. For each bus transition in Table 4, the upper row is the Spectre simulated energy cost, while the lower row represents the energy cost calculated using our proposed transition energy model.

## 6 Conclusion

We have proposed and analyzed a transition energy cost model aimed for efficient power estimation of performance critical DSM buses. We further derived an energy transition cost matrix scalable to any bus geometry of arbitrary bit width. The matrix can be utilized to more accurately determine energy benefits of applying transition coding to bus topologies. Our proposed energy model includes properties that closer capture effects present in high-performance VLSI buses, as verified against Spectre simulations of an implementable bus. The histogram plotted in Fig. 4 shows a concentration of transition costs around the average value for the suggested energy model compared to the other models. Thus, the included multi-buffer driver stage in our proposed energy model can account for effects that limit potential energy savings from bus transition coding.

**Acknowledgment.** We thank Dr. Oscar Gustafsson, Dr. Tina Lindqvist, and Dr. Jacob Löfvenberg for helpful discussions regarding problems associated with transition energy cost modeling.

## References

- [1] D. Liu, C. Svensson: Power Consumption Estimation in CMOS VLSI Chips, IEEE J. of Solid-State Circuits, vol. 29, pp. 663-670, June 1994.
- [2] G. Chandra, P. Kapur, K.C. Saraswat: Scaling Trends for the On Chip Power Dissipation, Proc. of the IEEE 2002 International Interconnect Technology Conference, pp. 154-156, 2002.

- [3] P.P. Sotiriadis: Interconnect Modeling and Optimization in Deep Sub-Micron Technologies. Ph.D. Thesis, Massachusetts Institute of Technology May 2002.
- [4] E. Macii, M. Poncino, S. Salerno: Combining wire swapping and spacing for low-power deep-submicron buses, Proc. 13th ACM Great Lakes Symposium on VLSI, 2003, pp. 198-202.
- [5] T. Sakurai: Design challenges for  $0.1 \mu\text{m}$  and beyond, Proc. Asia and South Pacific Design Automation Conf., 2001, pp. 293-296.
- [6] P.P. Sotiriadis, A.P. Chandrakasan: A bus energy model for deep submicron technology, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10 , Issue: 3 , June 2002, pp. 341-350.
- [7] M. Hiraki, H. Kojima, H. Misawa, T. Akazawa, and Y. Hatano: Data-dependent logic swing internal bus architecture for ultralow-power LSIs, IEEE J. Solid-State Circuits, vol. 30, Apr. 1995, pp. 397-401.
- [8] P. Kapur, G. Chandra, K.C. Saraswat: Power Estimation in Global Interconnects and its Reduction Using a Novel Repeater Optimization Methodology, Proc. 39th Design Automation Conference, 2002, pp. 461-466.
- [9] P.P. Sotiriadis, T. Konstantakopoulos, A.P. Chandrakasan: Analysis and implementation of charge recycling for deep sub-micron buses, Low Power Electronics and Design, International Symposium on, 6-7 Aug., 2001, pp. 364-369.
- [10] P.P. Sotiriadis, A.P. Chandrakasan: Bus energy reduction by transition pattern coding using a detailed deep submicrometer bus model, IEEE Transactions on Circuits and Systems I, vol. 50 , Issue: 10, Oct. 2003, pp. 1280-1295.
- [11] J. Rabaey, A.P. Chandrakasan, B. Nikolic: Digital Integrated Circuits, a design perspective, 2nd edition, ISBN: 0-13-597444-5.

# Moment-Based Estimation of Switching Activity for Correlated Distributions

Alberto García-Ortiz<sup>1</sup>, Tudor Murgan<sup>2</sup>, and Manfred Glesner<sup>2</sup>

<sup>1</sup> IBM Deutschland Entwicklung GmbH  
Schoenaicher Str. 220, D-71032 Boblingen (Germany)  
agarcia@de.ibm.com

<sup>2</sup> Institute of Microelectronic Systems, Darmstadt University of Technology  
Karlstraße 15, D-64283 Darmstadt (Germany)

**Abstract.** Energy estimation at high-levels of abstraction is getting a mandatory step in current design flows. With that goal, a moment-based stochastic technique has been recently introduced [5,4]. It provides fast power estimations using only word-level parameters; however, in its current formulation it is limited to un-correlated scenarios. Therefore, it cannot be broadly applied. In this paper an extension is proposed to suppress the aforementioned limitation. The technique employs the characterization of the probability density function (PDF) with a projection in a two-dimensional base constructed upon a novel set of orthogonal polynomials. The approach has been validated with practical experiments. Comparisons with reference bit level simulations and previous works are reported to assess the accuracy of the technique.

## 1 Introduction

Power estimation and optimization at high-levels of abstraction is getting a compelling need for the design community, in order to tackle the increasing design complexity and aggressive power budgets of current products.

Already in the early works concerning high-level power estimation a powerful framework was created [6] and afterwards refined [1,10,7] to estimate the power consumption in DSP architectures. It is based on the analysis (at high levels of abstraction) of the switching probability of the signals in a design. Word-level statistics such as standard deviation, mean value and temporal correlation are propagated through the architecture, and used to perform bit level estimations of the switching probability. The differences among the several techniques relay on the concrete formulas employed to perform the estimation, while the principal assumptions and restrictions are almost the same. Specifically, a major limitation that should be mentioned is the underlying assumption that the signals have a distribution close to be Gaussian. Although for linear architectures containing only adders and constant multipliers that simplification can be usually taken, non-linear architectures require more powerful techniques.

The problem of switching activity estimation in general non-linear architectures has been considered in [2,3,5,4] among others. The work of [2] provides a solid theoretical framework for power estimation. However, it requires expensive numerical integration procedures, which dissuade the use of this methodology for practical industrial designs.

In [3] a very accurate formulation is presented for un-correlated signals. However, since this technique requires the knowledge of the complete probability density function, its use is restricted in real applications. Finally, the results of [5,4] describe a technique that alleviates the previous limitation, by requiring only the moments of the signal. Since these moments can be more easily propagated through the design, the approach has a higher practical significance. The main problem in this case is that it is limited to un-correlated signals; i.e., scenarios where the current values of the signals in the design are independent from the previous ones. A technique to palliate this severe assumption is developed in this work.

Classic estimation methodologies have focused on transition probability, also referred as switching or transition activity. Although this is definitely a relevant metric for estimating dynamic power consumption in digital blocks, Very Deep Sub-Micron (VDSM) technologies require to take into account some other effects. Among them, coupling capacitance and leakage current are the most relevant. Since previous reported moment-based approaches [5] can handle static power consumption, we do not consider leakage effects further on in this work.

Due to the increase in the aspect ratio of deep sub-micron wires [11], the coupled capacitance between adjacent bus lines tends to increase and even to dominate the total capacitance of the wire. Thus, the power consumption associated with the drivers of a bus does not depend only on the toggling of the individual lines, but also on the simultaneous transition of adjacent bus lines. Using a model similar to [8], which assumes a full-rail swing of the bit lines equals to  $V_{dd}$ , the energy dissipated in each line driver can be estimated as:

$$E_{bus}(i) = \frac{V_{dd}^2}{2} (C_t t_i + C_e te_i) \quad (1)$$

where  $C_t$  is the total ground capacitances and  $C_e$  the coupling capacitance. The factor  $t_i$  (called switching or *transition activity*) measures the probability of a toggling in the  $i$ th bus line. The equivalent factor for the coupling capacitance is the *equivalent spatial transition activity*,  $te_i$ . For accurate power estimations in interconnect structures both  $t_i$  and  $te_i$  must be estimated.

The goal of this work is to provide an efficient estimation procedure for  $t_i$  and  $te_i$  at high-levels of abstraction, using only word-level parameters of the signal (in particular only the raw moments), and without any limitation concerning the temporal correlation or linearity of the design.

The paper is organized as follows: next section introduces the general estimation methodology and the novel orthogonal base proposed in the work. Afterwards, extensive experimental results are reported to assess the advantages of the technique. The paper finishes with some concluding remarks.

## 2 Estimation Approach

The early evaluation of the power consumption in VDSM technologies requires to estimate for each relevant signal in the design the transition activity ( $t_i$ ) and the spatial transition activity ( $te_i$ ). From a probabilistic point of view, the signals can be modeled by stochastic random processes  $X[t]$ . As it is usual in the literature, we assume that these

processes are either stationary, or can be divided in quasi-stationary segments. Thus, they can be characterized by a probability density function  $f(x)$ . If that PDF is analytically known,  $t_i$  and  $te_i$  can be exactly determined.

Since the transition activity and equivalent spatial transition activity depend on the previous and actual value of the signal, they require the joint PDF of  $\{X[n], X[n-1]\}$ . Let  $f(x, y)$  denote that PDF and let  $Tran_i(x, y)$  be a function which returns a one if the  $i$ th bit of  $x$  and  $y$  differ (i.e., an XOR). Then, for a  $B$ -bits signal:

$$t_i = \sum_{x=0}^{2^B-1} \sum_{y=0}^{2^B-1} f(x, y) Tran_i(x, y) \tag{2}$$

In the same way,  $te_i$  can be obtained by substituting  $Tran_i(x, y)$  with the cost function associated with  $te_i$ . The main drawback of this theoretical approach is that its complexity increases exponentially with the number of bits. Moreover, it requires the analytical knowledge of a two-dimensional PDF, which represents a major restriction in real scenarios. Thus, a more practical approach is needed.

In this work, the estimation of a two-dimensional PDF as a projection in a polynomial orthogonal base is used as a vehicle for calculating  $t_i$  and  $te_i$ . The projection can be formally derived using a scalar product [9].

Let us consider a positive two-dimensional weight function  $\varphi(x, y)$ , which defines a valid scalar product  $\langle \cdot, \cdot \rangle$  as:

$$\langle f_1(x, y), f_2(x, y) \rangle = \int_x \int_y f_1(x, y) f_2(x, y) \varphi(x, y) dx dy \tag{3}$$

Using this function and a similar approach to that of [4,5], the best approximation for the given PDF can be found as an orthogonal projection in the hyper-plane defined by that base. The main difference with previous works is that now the polynomial base should contain two variables. Explicitly:

$$f(x, y) \approx \sum_{k=0}^{M_n-1} c_k q_k(x, y) \varphi(x, y) \tag{4}$$

with the coefficients  $c_k$  given by:

$$c_k = \frac{\langle f(x, y) \varphi(x, y)^{-1}, q_k(x, y) \rangle}{\langle q_k(x, y), q_k(x, y) \rangle} \tag{5}$$

and  $\{q_k(x, y) : 0 \leq k < M_n\}$  a base  $M_n$  of orthogonal polynomials.

Furthermore, the numerator of Eq. (5) can be expressed using the concept of two-dimensional raw moment of a signal,  $\gamma_{r,s}$ . They are defined as [9]:

$$\gamma_{r,s} \stackrel{\text{def}}{=} \int_x \int_y x^r y^s f(x, y) dx dy \tag{6}$$

Then:

$$\langle f(x, y) \varphi(x, y)^{-1}, q_k(x, y) \rangle = \sum_{r,s} a_k(r, s) \gamma_{r,s} \tag{7}$$

where  $a_k(r, s)$  are the coefficient of the  $k$ th polynomial related with the monomial  $x^r y^s$ .

If now the estimation of  $f(x, y)$  is plugged into Eq. (2) and the summands are reordered, it is obtained that:

$$t_i = \sum_{r,s} \gamma_{r,s} \text{TransPol}_i(r, s) \quad (8)$$

where  $\text{TransPol}_i(r, s)$  is a constant term which can be pre-calculated in advance as:

$$\text{TransPol}_i(r, s) = \sum_{k=0}^{M_n-1} \sum_{x=0}^{2^B-1} \sum_{y=0}^{2^B-1} q_k(x, y) \varphi(x, y) \text{Tran}_i(x, y) \quad (9)$$

For  $te_i$  a similar formula can be obtained just using the const function asociated with  $te_i$  insted of  $\text{Tran}_i(x, y)$ . In any case, the estimation is calculated as a linear combination of the moments, where the number of terms is much less that those in Eq. (2). For example, if  $B = 10$  more than one million additions are required by the unoptimized approach, while the present formulation requires typically less than 20 terms (see sec. 3). To use the raw moments of the signal is a key issue in the approach. Since they can be propagated through the design [4], they can be very efficiently estimated at high-levels of abstraction, reducing the requirements for time-consuming simulations.

## 2.1 Selection of the Polynomial Base

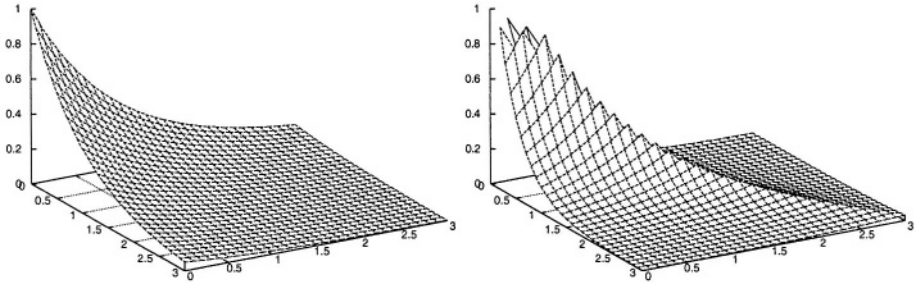
As in the case of uncorrelated distributions, a key factor is the selection of an appropriate weight function, and associated polynomial base. However, in this case the selection is even more critical because of the intrinsic higher complexity of a two-dimensional approximation.

The simpler possibility for defining a two-dimensional weight function consists of combining two one-dimensional functions with a multiplication. Thus,  $\varphi(x, y) = \varphi(x)\varphi(y)$ , and  $q_{i,j}(x, y) = q_i(x)q_j(y)$ , which can be applied for any of the classical polynomial bases (Laguerre, Legendre, and Hermite) well known in the literature [9]. The main limitation of this approach is associated with an overall lower accuracy when estimating highly correlated distributions. The intuitive rationale is that the multiplication of two one-dimensional functions produces an un-correlated two-dimensional PDF. If that function is used for estimating highly-correlated distributions, a large number of approximating terms would be required. However the order of the approximation is limited in the praxis to 4 or 5, since higher moments are very sensitive.

In order to provide a more accurate (and flexible) approach, it is desirable to develop a family of polynomial bases which can be adapted to the particular characteristics of the PDF. A suitable alternative, that mimics the properties of the classical Laguerre polynomials [9], and can be parameterized is:

$$\varphi_a(x, y) = \frac{a+1}{s^2} \exp\left(-\frac{X+Y+a|X-Y|}{s}\right) \quad (10)$$

where the parameter  $a$  selects the temporal correlation of the signal, while  $s \stackrel{\text{def}}{=} \frac{a+1}{2(a+2)}$  is related with the standard deviation of the PDF. If  $a$  equals zero, the orthogonal polynomials become the product of Laguerre polynomials. However, as  $a$  increases, the shape



**Fig. 1.** Weight function for  $a=0$  (left) and  $a=2.6$  (right).

of the approximating functions gets concentrated to the axis  $X_t = X_{t,1}$ , which in turn increases the correlation (see Fig. 1).

For this weight function, the Gram-Schmidt orthogonalization algorithm [9] is employed to generate an orthonormal base; i.e., a base where the “vectors” (in this case functions) are not only orthogonal, but also of norm one. Given an existing orthogonal base  $\{v_0, v_1, \dots, v_{n-1}\}$  with  $n$  elements, and a new candidate vector  $\hat{v}_n$ , the algorithm finds a new orthogonal vector (related with  $\hat{v}_n$ ) which can be included in the previous initial base to provide a set of  $n + 1$  orthogonal vectors.  $v_n$  is calculated by subtracting to  $\hat{v}_n$  its projection in the original base. Thus,

$$v_n = \hat{v}_n - \sum_{i=0}^{n-1} \frac{\langle \hat{v}_n, v_i \rangle}{\langle v_i, v_i \rangle} v_i$$

If the process is incrementally applied to a set of non-orthogonal polynomials vectors, the result is an orthogonal base which can be employed for our moment-based estimation procedure. Further on, the base can be normalized to be orthonormal, which makes the denominator of Eq. (5) equal to one.

Since the transition activity is symmetric around the axis  $X_t = X_{t-1}$ , it is advantageous to derive an orthogonal base which maintains this characteristic. It can be achieved by selecting a proper initial input vector set for the Gram-Schmidt algorithm. A suitable alternative (all are equivalent) is:

$$B = \{1, x + y, x - y, x^2 + y^2, x \cdot y, x^2 - y^2, x^3 + y^3, \dots\}$$

The cardinality of this set equals  $M_n$ , the number of linear terms used for the estimation. For example, if all the moments of order 2 are known, the base can contain 6 vectors (from 1 until  $x^2 - y^2$ ). In general when the moments until order  $M$  are known,  $M_n = (M + 1)(M + 2)/2$  terms can be employed.

It is worth noting that with the selection of aforementioned base  $B$ , the polynomials with odd symmetry do not contribute to the total transition activity and they do not need to be considered. Since the orthogonal base, and the contribution of each polynomial can be calculated off-line, only Eq. (5) and the final additions must be performed on-line to produce the estimation.



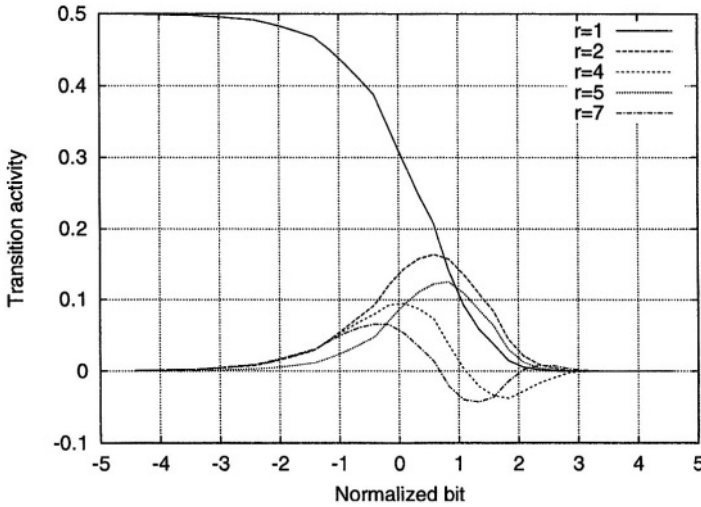


Fig. 2. Transition activity associated with several orthogonal vectors.

As an example, Fig. 2 depicts the contribution to the transition activity associated with the first eighth polynomials. The figure refers to a base with correlation parameter  $\alpha = \frac{8}{5} = 1.6$ , which corresponds to an correlation factor  $\rho = \frac{36}{61} \approx 0.59$ . The contribution of the polynomials 3, 6, and 8 is zero, and hence, they are not represented. The first polynomial exhibits the classical transition profile: 0.5 for the LSBs, 0 in the MSBs, and a monotonic variation in between. The activity of the other polynomials is characterized by non-zero values only inside the interval  $[-3, 3]$ . Consequently, the transition activity outside this interval is practically constant and independent of the moments of the PDF. This means that signals with variations of the transition activity outside  $[-3, 3]$  cannot be accurately modeled with the current base. However, as the parameter  $\alpha$  increases, the abovementioned interval moves to the left, and PDFs with a higher correlation can be properly modeled.

### 3 Experimental Validation and Discussion

In order to determine the validity of the approach, and the most appropriate value for that parameter  $\alpha$ , an extensive set of experiments have been performed. Two families of signals have been analyzed. The first one refers to the absolute value of a correlated Gaussian distribution, while the second corresponds to the addition of several  $\chi^2$  distributions of degree one.

The first family has been selected because its transition activity characteristics have been explicitly investigated in [7], where an explicit estimation approach has been proposed. Since that study has been tailored for only one family of signals, the approach represents a stringent reference in terms of accuracy for our more flexible approach. The family has the additional advantage that the two-dimensional PDF can be expressed in

explicit form; which allows a more accurate comparison of the different estimation approaches. Let  $\phi(x, y; \rho, \sigma)$  denote the probability function of a two-dimensional Gaussian signal with correlation factor  $\rho$  and standard deviation  $\sigma$ . Then, the PDF after performing the absolute value is:

$$f(x, y; \rho, \sigma) = \phi(x, y; \rho, \sigma) + \phi(-x, y; \rho, \sigma) + \phi(x, -y; \rho, \sigma) + \phi(-x, -y; \rho, \sigma) \quad (11)$$

for the positive values of  $x$  and  $y$ , and zero elsewhere.

The second family is constructed by adding  $L$  consecutive samples from a  $\chi^2(1)$  distribution (which can be obtained by squaring the magnitude of a Gaussian distributed signal). Hence,  $z[n] = \sum_{i=0}^{L-1} g^2[n-i]$ , with  $g[n]$  the uncorrelated Gaussian distributed signal. Although the samples of  $g[n]$  are temporally uncorrelated, the  $L$  additions introduce a given correlation for  $z[n]$ . It is straightforward to find that  $\rho = \frac{L-1}{L}$ . It is worth to notice that this family of functions appear very commonly in real applications; e.g., when calculating the norm of a complex signal.

For each of these two families of functions, the bit-level transition activity and equivalent spatial transition activity has been measured and compared with the approximation provided by the correlated moment-based approach previously proposed. The technique has been evaluated for five different orthogonal bases, associated with the weight function given by Eq. (10) and a correlation parameter  $a$  equals to 0, 1.6, 2.6, 3.6, and 4.6. Furthermore, the analysis has been performed varying the standard deviation of the signals in four consecutive runs. The mean squared error is employed as a metric for characterize the accuracy of the different approaches. In all the cases, the error is represented as a percentage of the total transition activity in a signal of 9 bits. The simulation results, reported in Tab. 1 and Tab. 2, also include a comparison with [7]. It is worth noting that this technique cannot estimate  $te_i$ , which represent a main drawback in comparison with our technique. In the following, we concentrate our analysis on  $t_i$  where a comparison with previous works can be done. (The behavioral or  $te_i$  is similar to that of  $t_i$ , but the error is slightly higher)

The first conclusion that can be drawn from the experimental results is that the parameter  $a$  plays a significant role for determining the accuracy of the estimation. If the PDF has a small correlation, a lower value of the parameter  $a$  reduces the error, while as the correlation increases, higher values of  $a$  are preferred. In the case of Tab. 1, the error associated with a uncorrelated signal (i.e.,  $\rho = 0$ ) is 4% for  $a = 0$ , but it increases until 9% when  $a = 4.6$ . However, in a highly correlated signal (for example  $\rho = 0.97$ ) the error decreases from 24% to 7% instead of increasing. Similar results can be observed for Tab. 1, but slightly more accentuated. An adequate selection of the parameter  $a$  can reduce the error from 25% to 7%.

Respect to the number of linear terms required for the approximation, it is observed that typical values of  $N = 4$  or  $N = 5$  (corresponding to 15 and 21 linear terms respectively) are adequate. In general, bases with a higher correlation parameter require more terms to produce sensible approximations. If the base corresponding to  $a = 2.6$  is used,  $N = 3$  can be employed without degrading significantly the accuracy. To analyze in more detail these observations, Fig. 3 represent the transition activity in each bit of a signal with correlation  $\rho = 0.25$ . As the approximation order increases, the estimation get closer, where the main improvement occurs in the region of where the transition activity moves from 0.5 to 0.

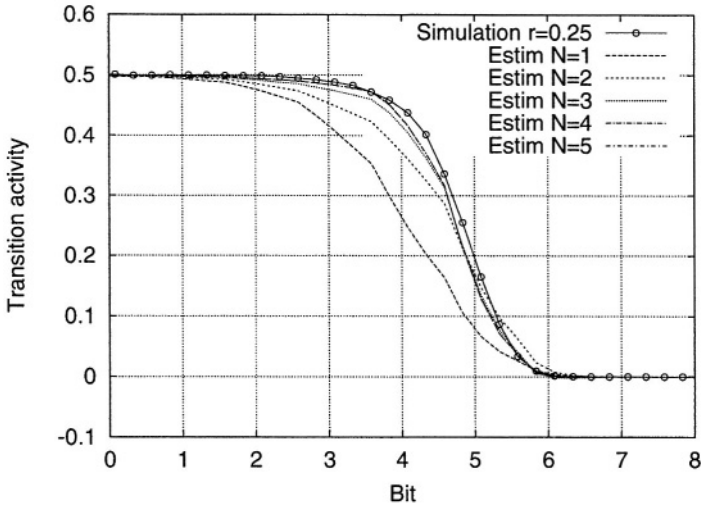


Fig. 3. Effect of the approximation order  $N$  when estimating the transition activity at the bit-level.

The last but one column of both tables reports the estimation error when the best of the five polynomial bases is selected for each single experiment. The results show that an excellent accuracy around 5% can be achieved. To assess the quality of the approach, it is worth noting that the approach of [7] exhibits a mean accuracy of 7% in the best possible scenario (i.e., the first family of signals). If the estimation procedure wants to be simplified, a single base can be selected. A good compromise is provided by  $a = 2.6$ , that allows a mean error of around 8% in the two cases cases.

The second result provided by Tab. 1 and Tab. 2 is related with the lost of accuracy incurred when considering a single model for the signal. In this sense, if the technique of [7] is employed for signals that are not the magnitude of a Gaussian signal, the approximation error increases notably to 21%. With the classical DBT model the error would be even greater. In our approach however, an stable mean error value around 8% can be always achieved with the base  $a = 2.6$ . Moreover, if the best base is selected, the error is as low as as 5%.

### 4 Conclusion

This paper presents a technique for the efficient estimation of those parameters as transition activity and equivalent spatial transition activity required for power estimation in VDSM technologies. Beside its simplicity and efficiency (it consist of a linear combination of the raw moments of the signal), it is not restricted to un-correlated or linear systems as previous works. The foundation of the technique is based on the theory of orthogonal projections in a polynomial base, which ensures a solid framework.

A key contribution of the work is the development of a parameterizable set of two-dimensional orthogonal polynomials which provides accurate estimations. For typical

**Table 1.** Mean squared error during the estimation of the bit-level transition activity for Gaussian signals with a given correlation  $\rho$ . Comparison between different polynomial bases and previous work.

$\rho$	Pol. base ( $a = 0$ )					Pol. base ( $a = 1.6$ )					Pol. base ( $a = 2.6$ )					Pol. base ( $a = 3.6$ )					Pol. base ( $a = 4.6$ )					Best base	Prev. work	
	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5			
00	11	6	3	4	4	23	13	8	6	6	31	11	5	5	5	37	4	6	6	5	42	14	18	7	9	3	5	
25	10	6	3	4	4	22	14	9	7	6	30	11	6	5	5	36	4	5	5	5	42	11	16	7	8	3	5	
50	9	5	3	3	3	19	13	9	8	7	27	13	8	6	6	33	8	4	4	4	39	4	8	5	4	3	5	
75	10	8	5	4	3	12	10	9	8	8	19	14	11	9	8	26	14	11	8	7	32	10	8	5	5	3	4	
80	12	9	7	5	4	10	9	8	8	8	17	13	11	10	9	23	14	12	9	8	29	13	10	7	6	4	4	
85	15	12	9	7	6	7	7	7	7	7	13	12	10	10	9	20	14	12	10	10	25	14	12	9	8	6	5	
90	19	16	13	11	10	6	4	4	5	5	8	8	8	8	9	14	12	11	11	10	20	15	13	11	10	4	5	
93	24	21	17	15	13	10	7	5	4	4	6	5	6	6	7	10	10	9	9	10	15	13	12	11	11	4	7	
95	29	26	21	19	17	14	11	8	7	6	8	5	5	4	5	6	7	7	7	8	10	10	10	10	10	4	8	
97	36	33	28	25	24	22	19	16	13	11	15	11	9	7	6	9	6	5	5	5	6	6	6	6	7	5	10	
99	53	50	45	42	40	41	37	34	31	29	33	29	27	25	23	28	23	22	19	17	22	16	16	13	12	12	15	
<i>mean value =</i>					13	12				9					9					8						8	5	7

**Table 2.** Mean squared error during the estimation of the bit-level transition activity for non-Gaussian signals. Comparison between different polynomial bases and previous work.

M	Base ( $a = 0$ )					Base ( $a = 1.6$ )					Base ( $a = 2.6$ )					Base ( $a = 3.6$ )					Base ( $a = 4.6$ )					Best base	Prev. work		
	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5				
2	15	9	9	7	7	8	5	5	5	4	11	6	4	8	4	15	13	9	22	11	20	26	19	55	28	4	21		
3	23	20	18	17	16	10	10	10	10	9	7	7	7	7	7	9	6	6	6	6	13	6	5	7	6	5	20		
4	30	29	28	26	25	17	16	16	16	16	11	11	12	12	12	7	8	9	9	9	8	7	7	7	7	7	22		
<i>mean value =</i>					17	16				10	10				9	8				12	9					23	14	5	21

applications, a constant value of that parameter could be used with errors around 8%. If the PDF of the signals is expected to have a significant variability, or higher accuracy is required, a more flexible approach based on different bases can be selected. Currently ongoing effort in an automatic selection procedure based on signal correlation is very promising.

## References

1. S. Bobba, I. N. Hajj, and N. R. Shanbhag. Analytical expressions for average bit statistics of signal lines in DSP architectures. In *ISCAS*, pages 33 – 36, 1998.
2. Jui-Ming Chang and Massoud Pedram. *Power optimization and synthesis at behavioural and system levels using formal methods*. Kluwer Academic Publishers, 1999.
3. Alberto García-Ortiz, Lukusa D. Kabulepa, and Manfred Glesner. Transition activity estimation for generic data distributions. In *Int. Symp. on Circuits and Systems (ISCAS)*, pages 468 –471, May 2002.
4. Alberto García-Ortiz, Lukusa D. Kabulepa, and Manfred Glesner. Switching activity estimation in non-linear architectures. In *Int. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 269–278, September 2003.
5. Alberto García-Ortiz, Lukusa D. Kabulepa, Tudor Murgan, and Manfred Glesner. Moment-based power estimation in very deep sub-micron technologies. In *Int. Conf. on CAD (ICCAD)*, November 2003.
6. P.E. Landman and J.M. Rabaey. Architectural power analysis: the dual bit type method. *IEEE Trans. on VLSI Systems*, 3:173 – 187, June 1995.
7. M. Lundberg, K. Muhammad, K. Roy, and S. K. Wilson. A novel approach to high-level switching activity modeling with applications to low-power DSP system synthesis. *IEEE Trans. on Signal Processing*, 49:3157 – 3167, December 2001.
8. C.-G. Lyuh, Taewhan Kim, and Ki-Wook Kim. Coupling-aware high-level interconnect synthesis for low power. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 609 –613, 2002.
9. L. Rade and B. Westergren. *Mathematische Formeln*. Springer, second edition, 1995.
10. J. H. Satyanarayana and K. K. Parhi. Theoretical analysis of word-level switching activity in the presence of glitching and correlation. *IEEE Trans. on VLSI Systems*, pages 148 – 159, April 2000.
11. D. Sylvester and C. Hu. Analytical modeling and characterization of deep-submicrometer interconnect. *Proc. of the IEEE*, 89(5):634–664, May 2001.

# Table-Based Total Power Consumption Estimation of Memory Arrays for Architects

Minh Q. Do, Per Larsson-Edefors, and Lars Bengtsson

Department of Computer Engineering, Chalmers University of Technology,  
SE 41296 Gothenburg, Sweden  
{minh,perla,labe}@ce.Chalmers.se

**Abstract.** In this paper, we propose the White-box Table-based Total Power Consumption (WTTPC) estimation approach that offers both rapid and accurate architecture-level power estimation models for some processor components with regular structures, such as SRAM arrays, based on WTTPC-tables of power values. A comparison of power estimates obtained from the proposed approach against circuit-level HSPICE power values for a 64-b conventional 6T-SRAM memory array implemented in a commercial 0.13-um CMOS technology process shows a 98% accuracy of the WTTPC approach.

## 1 Introduction

Nowadays, as a result of CMOS technology scaling, leakage power dissipation has become a significant portion of the total power consumption in deep-submicron VLSI chip [1]. The 2001 International Technology Roadmap for Semiconductors (ITRS) predicts that by the 70-nm generation, leakage may constitute as much as 50% of total power dissipation [2]. Thus, it is really important for architects to have the ability to rapidly estimate, with sufficient accuracy, both dynamic and static power consumption of the architectural design while exploring several alternatives searching for the optimal one.

Leakage power consumption due to subthreshold leakage currents is closely related to the physical behavior of MOS transistors, the type of circuitry involved and the process technology parameters. Therefore, it can only be accurately estimated by using circuit-level power estimation-simulation tools like SPICE, PowerMill, etc. However, due to the large number of circuit-level nodes, the complexity and time required to perform power estimation at this level are prohibitively large making rapid analysis of large designs impractical. In contrast, architecture-level tools provide faster results as compared with circuit-level ones while sacrificing accuracy.

Recently, some research have been directed towards developing models for estimating dynamic and static power consumption at architectural design level. Wattch is a collection of architecture-level power models that are based on the Simple-Scalar toolset (together with CACTI 2.0) which is commonly used to model micro-architectures in educational and some research environments [3]. Wattch divides the main microprocessor units into four categories, i.e. array structures, CAMs, combinational logic, and clocking, and then uses activity-based statistical power estimation models to produce the power estimates. Nevertheless, since Wattch uses a simple

technology scaling mechanism to scale down from 0.8- $\mu\text{m}$  to 0.1- $\mu\text{m}$  technology and since Watch does not have any model to efficiently estimate static power consumption, it has relatively poor accuracy compared to circuit-level power estimation tools, especially for deep-submicron technologies. Butts and Sohi [4] proposed a genetic, high-level model for micro-architecture components. The model is based on a key design parameter,  $K_{design}$ , capturing device type, device geometry and stacking factors that can be obtained based on simulations. This model of subthreshold leakage accurately addresses some different issues affecting static power in such a way that it makes it easy to reason about leakage effects at the micro-architectural level. However, it turns out not to be well suited for some types of SRAM circuits with power-saving and leakage-reduction techniques like MT-CMOS, Gated-Vdd, and Drowsy Cache. Also, it was never released as publicly available software. Parikh *et al.* [5] developed an architectural model for subthreshold and gate leakage that explicitly captures temperature, voltage, and parameter variations. This model was implemented in the micro-architectural HotLeakage simulation tool based on Watch and the Cache-decay simulator. This was an attempt to develop the methodology of Butts and Sohi to address the effect of temperature on leakage power consumption. However, the accuracy of the leakage power estimation for any complex circuit structures like memory arrays, caches, etc., is unknown. Another effort to develop further the methodology of Butts and Sohi is the work by Mahesh *et al.* [6]. In this work, the authors developed analytical models parameterized in terms of high-level design parameters to estimate leakage power in SRAM arrays. An error margin of “less than 23.9%” compared to HSPICE power values is achieved by this method.

Schmidt *et al.* [7] developed an automatic black box memory-modeling approach based on nonlinear regression, which intends to combine good *model properties* (i.e. accuracy, speed, etc.) with good *modeling properties* (i.e. automatism, fit to design flow, low overhead and IP protection). Nevertheless, this approach offers its advantages at the price of a complex and computationally expensive model characterization phase. For typical memory arrays whose regular *internal structures* are known and can easily be analyzed, a white box modeling approach (e.g. our WTTTPC approach) can be a good alternative to the black box one, offering a simpler and faster model characterization phase.

In [8] Eckerbert *et al.* presented a methodology to accurately estimate total power consumption (including static power) at the RT-level using simulation-based power estimation models. The methodology takes into account the changes in the component environment, which occur between characterization and estimation. By separating the different power dissipation mechanisms this methodology achieves high degrees of accuracy in estimating power consumption. Although it is a complex and accurate, RT-level simulation approach and it mainly focuses on estimating total power consumption of complex components, such as arithmetic-logic circuits, it still serves as a good hint for us. Furthermore, this methodology can be used together with our proposed approach, where necessary, providing an architecture-level solution to the problem of estimating total power consumption of all processor components.

In this paper, we propose the White-box Table-based Total Power Consumption estimation approach (WTTTPC) that offers accurate architecture-level power estimation models for some processor components with regular structures, such as SRAM arrays, based on tables of power values. The structure of these tables is created before the characterization based on the structure of the component (since this is a white-box approach), while the actual simulation values are entered during the characterization.

Those tables, therefore, are referred to as *pre-characterized* tables during characterization, and as *WTTPC-tables* after the characterization has been done. The independent inputs to the pre-characterized tables are *Type of component* (i.e. type of SRAM arrays), *Component State* (S), *Temperature* (T), *Frequency* (F), *Threshold Voltage* ( $V_T$ ), *Supply Voltage* ( $V_{dd}$ ) and *Process Corner* (PV). At a close look, our WTTPC approach consists of two underlying phases: *Characterization* and *Power Estimation*.

1. **Characterization:** takes as input the netlist of a typical component, its states and memory-organization parameters, generates power values using HSPICE simulations and tabulates those values into pre-characterized tables. The power values in those tables are the per-cycle, per-hardware-access dynamic power consumption, and the per-cycle leakage power consumption of that component.
2. **Power Estimation:** takes as inputs the WTTPC-tables, states of the component, input traces, and produces power consumption estimates in a cycle-by-cycle manner. Total power consumption of the component is an averaged per-cycle value of these cycle-by-cycle dynamic and static power consumption values.

A more detailed description about the WTTPC approach and its implementation is given in section 3 of this paper.

## 2 Structure of a Power-Performance Simulator

The Power-Performance simulator for parallel DSP architectures (DSP-PP) [9] is a SimpleScalar-Wattch like cycle-accurate power-performance simulator that was designed and is currently being implemented by us. DSP-PP consists of two components: the *Cycle-by-Cycle Performance Simulator (CPS)* and the *Power-Dissipation Estimator (PDE)*. The CPS is an execution-driven cycle-accurate performance simulator. The main functions of CPS are as follows:

- Accepts as input, an executable program obtained by compilation of input benchmarks as well as the PE/DSP configurations. (PE denotes a processor element).
- Simulates, cycle-by-cycle, instruction execution and dataflows between PE components as well as between parallel DSP architecture components.
- Generates output performance statistics (i.e. program cycle counts) and cycle-by-cycle traces.

The PDE consists of power consumption models for DSP components and a *total-power-estimation-engine* module used to calculate the overall power dissipation of the entire DSP parallel architecture in a cycle-by-cycle manner. These power models include WTTPC-tables of power values for memory arrays and similar types of components; parameter sets for other types of DSP components such as arithmetic-logic circuits, etc. The main functions of PDE are as follows:

- Accepts as input, cycle-by-cycle traces from CPS for different hardware components involved in the DSP parallel architecture, as well as the PE/DSP configuration and the configuration of the entire DSP parallel architecture.
- Generates power estimation values in a cycle-by-cycle manner for the given configuration.



In order to reduce the number of WTTPC-tables created for each component, similar to what was done in [8], the PDE is designed so that it can interpolate (using curve-fitting interpolation functions) between component characterization points covering the entire possible design range of that component.

### 3 The White-Box Table-Based Total Power Consumption Approach

In general, architecture-level power consumption estimation methods can be classified into two groups: *Analytical* (statistical) and *Simulation-based*. Analytical power estimation models have been used in several projects, e.g. [3][4][5] and [6]. The advantage of the analytical model is the simplicity of the formulas used to calculate the dynamic and leakage power consumption estimates. These simple formulas allow architects to rapidly obtain the power estimates and consider power characteristics of alternative designs. However, analytical models usually offer low accuracy compared to the estimates from circuit-level power estimation tools like SPICE and its clones. Moreover, due to the simplicity of the formulas, analytical models may not cover the complete deep-submicron behavior of MOS transistors and wiring, causing unacceptable decrease in accuracy [10]. In contrast to the analytical, the simulation-based power estimation methods offer very accurate power estimates at the price of long estimation run-time. The simulation-based power estimation methods can be implemented by *table-based* or *equation-based* power models. The difference between equation-based and table-based models is that the latter ones are “discrete” tabulated power consumption values while the former ones are mathematical equations resulted from “generalization” of those “discrete” power estimates using curve-fitting techniques, e.g. linear and non-linear regressions.

Our approach is table-based; its *Characterization* and *Power Estimation* phases are shown in Figs 1a and 1b. Component *Characterization* and *Power Estimation* processes were introduced in RT-level simulation-based power estimation quite a long time ago [11][12]. However, these phases are different from our *Characterization* and *Power Estimation* ones. Characterization in the RTL power estimation takes as input the *trace* (i.e. input vectors), the component netlist, and produces as output a set of model parameters. Instead of taking a trace as input, the characterization phase in our approach takes the *component states* defined by fixed component input combinations. The complexity of the component states is much lower than that of the trace making the characterization phase become fast and simple. The third input is the *memory-organization parameters* used to obtain power estimates (using re-organization formulas) for those components that are larger and differently organized than the regular, characterized ones. Using HSPICE simulation for each component, WTTPC-tables of power values are created. These tables together with the *component states* and the *input trace* serve as inputs to the power estimation phase of our WTTPC approach. Furthermore, it is important to note that our approach assumes that the component environment remains fixed from characterization to power estimation phases. This assumption is the major difference between the WTTPC approach and the Eckert's RTL power estimation methodology given in [8].

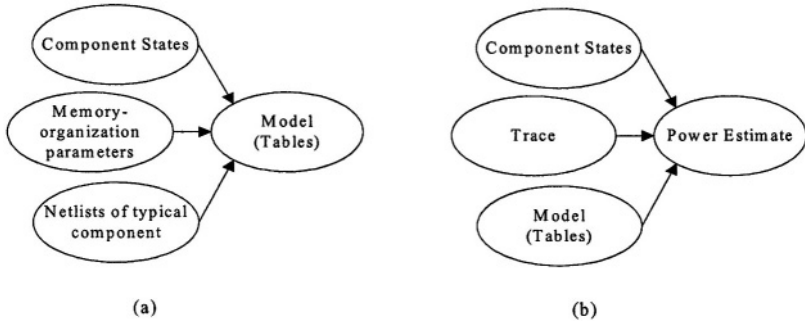


Fig. 1. (a) Characterization phase, and (b) Power Estimation phase

The number of possible input combinations and the number of  $S$  determine the complexity of the table-based power models and the required estimation time of the power consumption simulator, hence it must be very carefully chosen. The component states are determined for each type of component based on its operation. States should include both “active” and “stand-by” ones. The number of  $S$  also needs to be chosen carefully so that the power models can give the highest possible accuracy with a reasonable number of entries in the table. For each set of input parameters (i.e.  $T$ ,  $F$ ,  $V_T$ ,  $V_{dd}$  and  $PV$ ) the number of characterized points needs to be selected so that it not only can cover the entire possible variation range of that parameter, but also reduce the complexity of the power models. In the framework of DSP-PP three points, which are maximal, minimal and nominal values, are selected for each parameter. For example, three temperatures  $T = 40, 70$  and  $110^\circ\text{C}$  are selected to represent the entire operational temperature range of the processor memory arrays. Beside these, the three inputs of  $V_T$ ,  $V_{dd}$  and  $PV$  are technology-dependent parameters that can be used to provide the WTTPC approach with the ability to handle technology parameter variation and some power-saving and leakage-reduction techniques, like MT-CMOS, Gated-Vdd, and Drowsy Cache. For any component the characterization phase is typically performed only once by a cell-library designer. Computer architects having access to the netlist of new components also can perform the characterization of their components and create new tables for later use. The power estimation phase is performed by the power consumption simulator (i.e. by the *total-power-estimation-engine* module of the PDE in DSP-PP) each time when it is used.

## 4 Validation

In modern microprocessors, the SRAM is extensively used for caches, register files, branch predictors, etc. For instance, researchers have shown that 40% of the total power in the Alpha 21264 [13] and 60% of the total power in the Strong-ARM [14] are due to memory structures. The SRAM obviously is one of the main power consumers to the total design power. Besides, due to its operational characteristics, a large portion of the SRAM array is often idle or in standby mode, hence giving rise to leakage power. Thus, an SRAM array was selected for our validation.

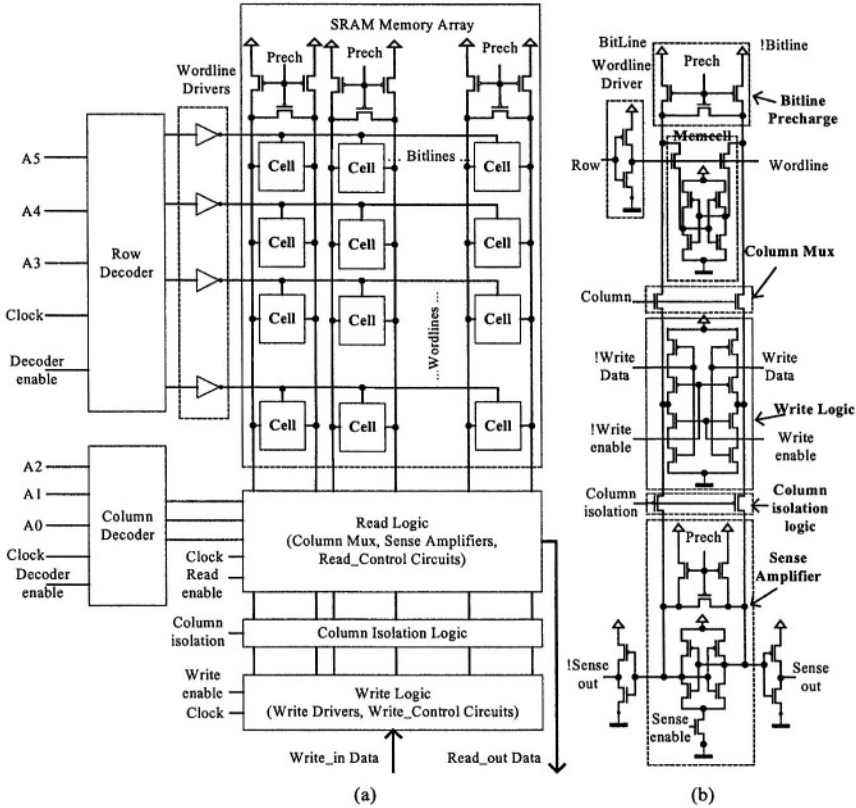


Fig. 2. (a) Block diagram of the 64-b SRAM array, (b) circuit diagram of a column of the 64-b SRAM array.

### 4.1 Validation Methodology

We compare the power values obtained from our approach against the HSPICE power values for a conventional 64-b 6T-SRAM memory array. Figs 2a and 2b show the block diagram and the circuit diagram of a column, respectively, of the SRAM array used in this validation. During the characterization of this memory array, a set of input parameters needs to be defined.

### Technology Parameters

For conventional memory arrays in typical applications, the operational temperatures (T) are ranging from 40°C to 110°C, thus it is reasonable to choose the three temperature points 40, 70 and 110°C. The frequency of memory accesses (F) is also an important parameter defining the characteristics of memory array. In this validation, F = 25 MHz was selected. This relatively low frequency of memory accesses provides a well-functioning SRAM array for validation purposes without losing any generality. The supply voltage ( $V_{dd} = 1.2$  V), the normal threshold voltages (i.e.  $V_T = V_{TH0}$ : NMOS  $V_{TH0}$  is slightly lower than 0.25 V, whereas PMOS  $|V_{TH0}|$  is slightly higher than 0.25

V), and the typical process corner value (PV = typical) for a commercial 0.13-um CMOS process are also selected.

### Test Input Traces

There are seven different test input read/write sequences (traces) used for this validation, which includes 10 continuous reads (10 Reads only), 10 continuous writes (10 Writes only), 5 continuous reads followed by 5 continuous writes (10 In-order R\_then\_W), 5 continuous writes followed by 5 continuous reads (10 In-order W\_then\_R), 10 interleaved read\_then\_write (10 interleaved R\_then\_W), 10 interleaved write\_then\_read (10 interleaved W\_then\_R), and 10 random Reads/Writes to the memory array (10 Random R/W). Using HSPICE, the power estimation phase takes as input trace, WTTTC-tables of power values, states of the memory array and produces average total power consumption values for each trace.

**Table 1.** States of the memory array with its input signal definition

Memory array states ( $S_{mem}$ )	Input signal definition (according to the input signals shown in Fig. 2b)
Reading	Row=0, Column=1, Prech=1, Write_Data=0, Write_enable=0, Column_isolation=0, Sense_enable=1
Writing 0/1	Row=0, Column=1, Prech=1, Write_Data=0/1, Write_enable=1, Column_isolation=1, Sense_enable=0
Precharging	Row=1, Column=0, Prech=0, Write_Data=0, Write_enable=0, Column_isolation=1, Sense_enable=0
Leakage	Row=1, Column=0, Prech=1, Write_Data=0, Write_enable=0, Column_isolation=1, Sense_enable=0

### 4.2 HSPICE Simulation Setup

In order to provide a simulation platform for validation, a netlist of a 64-b 6T-SRAM memory array has been created and sized properly in a commercial 0.13-um CMOS process. There are two different types of power values that need to be obtained using HSPICE, the average per-cycle power consumption values for different states of the array, and the average of the total power consumption values for different number of accesses (read/write) with different access order (traces). The former values are obtained during the characterization phase by running HSPICE simulation for several short transient analyses. These obtained values then are tabulated as the average per-cycle power estimates for different states of the array. The latter power values are used for comparison against the power values obtained from the WTTTC approach for different traces.

### 4.3 Validation of the WTTTC Approach Using an SRAM Array

The implementation of a conventional 64-b 6T-SRAM array using the WTTTC approach consists of two phases:

### Characterization

Memory array states ( $S_{mem}$ ): {Reading-1, Reading-0, Writing-1-to-1,  
Writing1-to-0, Writing0-to-1, Writing0-to-0), Leak}

Temperature:  $T = 70^{\circ}\text{C}$ ; Frequency of memory access: 25 MHz;

Threshold Voltage:  $V_T = V_{TH0}$ ; Supply Voltage:  $V_{dd} = 1.2\text{ V}$ ;

Process Corner: PV = TT (typical, typical)

Taking as input these parameters and the netlist of the SRAM array, HSPICE produces the average per-cycle power values for each state  $S_{mem}$ . Table 1 shows the states of the memory array with its input signal definitions.

### Power Estimation

$$\text{Total Power (per-cycle)} = P_{read} + P_{write} + P_{leak} \quad (1)$$

$$P_{read} = N_{read\_1} \times P_{read\_1} + N_{read\_0} \times P_{read\_0};$$

$$P_{write} = P_{write\_0} + P_{write\_1};$$

$$P_{write\_0} = N_{write(0,0)} \times P_{write(0,0)} + N_{write(0,1)} \times P_{write(0,1)}; \quad (2)$$

$$P_{write\_1} = N_{write(1,0)} \times P_{write(1,0)} + N_{write(1,1)} \times P_{write(1,1)};$$

$$P_{leak} = P_{leak\_1bit} \times N_{mem};$$

Here,  $P_{read}$  and  $P_{write}$  are per-cycle reading-from-a-cell, writing-to-a-cell average power consumptions, respectively, and  $P_{leak}$  is per-cycle average leakage power consumption of the memory array. The values  $P_{read\_1}$ ,  $P_{read\_0}$ ,  $P_{write\_0}$ , and  $P_{write\_1}$  are per-cycle reading-1, reading-0, writing-to-0 and writing-to-1 average power consumptions, respectively. The values  $P_{write(0,0)}$ ,  $P_{write(0,1)}$ ,  $P_{write(1,0)}$  and  $P_{write(1,1)}$  are detailed per-cycle writing0-to-0, writing0-to-1, writing 1-to-0, and writing1-to-1 average power consumptions, respectively, and  $P_{leak\_1bit}$  is per-cycle average leakage power consumption of a memory cell. Other values  $N_{read\_1}$ ,  $N_{read\_0}$ ,  $N_{write(0,0)}$ ,  $N_{write(0,1)}$ ,  $N_{write(1,0)}$ ,  $N_{write(1,1)}$  and  $N_{mem}$  are total numbers of reading-1, reading-0, writing0-to-0, writing0-to-1, writing 1-to-0 and writing1-to-1 accesses to the memory array, respectively, and  $N_{mem}$  is total number of memory cells in the array. These numbers are extracted from the input traces. Using equation 1 the total per-cycle power consumption values of the 64-b SRAM array for different traces are calculated. These power values are then compared with HSPICE simulated ones.

## 4.4 Results and Discussions

The per-cycle power consumption values for every state of the memory array obtained from characterization phase are showed in Table 2. It is obvious from Table 2 that reading consumes less power than writing, while writing0-to-1, writing 1-to-0 to the memory array consumes most power. The first result can be explained by the fact that write drivers need to be sized twice as large as the cross-coupled inverters of the differential sense amplifier to provide well-functioning writes to memory cells, hence consuming more power. The latter result is due to large write currents needed to pull up the cell voltage above (or pull down the cell voltage below) the toggling points of the cross-coupled inverters to flip the cell. Leakage power consumption of a memory

cell is small, however, for memory arrays of realistic size (i.e. Mbytes), the leakage value will be of the same order of the reading or the writing dynamic ones. Table 2 also shows that leakage power of a memory cell does not depend on its content. This is due to the symmetric property of the 6T-SRAM memory cell used in the validation.

Table 3 shows the power estimates obtained from the power estimation phase of the WTPPC approach, and the HSPICE simulated total per-cycle power values for each sequence of memory accesses. Here, we also can see that the accuracy of power estimates obtained from the WTPPC approach is larger than 98% compared to the HSPICE values. These results show the validity of our proposed WTPPC approach in term of accuracy in estimating power consumption for a conventional SRAM array.

**Table 2.** Per-cycle power consumption estimates for the 64-b SRAM array

	Memory array states (Smem)	Per-cycle power consumption estimates
Reading	Reading-1	4.63 (mW)
	Reading-0	4.63 (mW)
Writing	Writing0-to-0	5.86 (mW)
	Writing0-to-1	6.26 (mW)
	Writing1-to-0	6.26 (mW)
	Writing1-to-1	5.86 (mW)
Leak	Leak (memcell=0)	2.80 (nW)
	Leak (memcell=1)	2.80 (nW)

**Table 3.** Total power consumption estimates for the 64-b SRAM array and accuracy values

Test sequence of memory accesses	HSPICE power values (mW)	WTPPC approach	
		Power value (mW)	Accuracy (%)
10 Reads only	4.68	4.63	98.93
10 Writes only	5.87	5.86	99.75
10 In-order R_then_W	5.27	5.32	99.05
10 In-order W_then_R	5.28	5.32	99.24
10 Interleaved R_then_W	5.27	5.36	98.29
10 Interleaved W_then_R	5.28	5.36	98.48
10 Random R/W	5.27	5.36	98.29

## 5 Conclusions

In this paper the White-box Table-based Total Power Consumption estimation approach has been presented. This approach offers accurate architecture-level power estimation models for some processor components, such as SRAM arrays, based on WTPPC-tables of power values. Validation has been done against circuit-level HSPICE simulated power values for a 64-b conventional 6T-SRAM array implemented in a commercial 0.13-um CMOS technology process achieving an accuracy of more than 98%.

The proposed WTPPC approach, as far as we can tell, is the first one that can offer relatively simple high-accuracy architecture-level power estimation models accounting for both dynamic and static power consumption. However, the size of the ana-

lyzed SRAM array shown in this paper is still small compared to a realistic one. Therefore, our on-going research works are targeting methods to provide power consumption estimates for larger memory arrays.

**Acknowledgements.** The authors would like to thank Dr. Daniel Eckerbert for his comments and critical discussions on power consumption estimation methodologies and its classification.

## References

1. Ferre A. and Figueras J., "Characterization of Leakage Power in CMOS Technologies", Proc. of the IEEE Intl. Conf. on Electronics, Circuits and Systems, Vol. 2, Lisbon, Portugal, Sept. 1998, pp. 185–8.
2. SIA, International Roadmap for Semiconductors, 2001.
3. Brooks D. et al., "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", Proc. of ISCA, Vancouver, BC Canada, June 2000, pp. 83–94.
4. Butts J. A. and Sohi G. S., "A Static Power Model for Architects", Proc. of the Intl. Symp. on Micro-architectures, Monterey, CA, USA, Dec. 2000, pp. 191–201.
5. Parikh D. et al., "Comparison of State-Preserving vs. Non-State-Preserving Leakage Control in Caches", Proc. of the Workshop on Duplicating, Deconstructing and Debunking (held in conjunction with ISCA), San Diego, CA, USA, June 2003, pp. 14–25.
6. Mamidipaka M. et al. "Leakage Power Estimation in SRAMs", Technical Report No. 03-32, Center for Embedded Computer Systems, University of California, Irvine, USA, Sept. 2003.
7. Schmidt E. et al., "Memory Power Models for Multilevel Power Estimation and Optimization", IEEE Transaction on VLSI Systems, Vol. 10, April 2002, pp. 106–9
8. Eckerbert D. and Larsson-Edefors P., "A Deep Submicron Power Estimation Methodology Adaptable to Variations Between Power Characterization and Estimation", Proc. of the 2003 Asia-South Pacific Design Automation Conf., Kitakyushu, Japan, Jan. 2003, pp. 716–9.
9. Do Q. M., Bengtsson L. and Larsson-Edefors P., "DSP-PP: A Simulator/Estimator of Power Consumption and Performance for Parallel DSP Architecture", Proc. of Parallel and Distributed Computing and Networks Symp. (PDCN), Innsbruck, Austria, Feb. 2003, pp.767–72
10. Do Q. M., and Bengtsson L., "Analytical Models for Power Consumption Estimation in the DSP-PP Simulator: Problems and Solutions", Technical Report No. 03-22, Chalmers University of Technology, Göteborg, Sweden, Aug. 2003.
11. Eckerbert D., "Power Estimation and Multi-Phase Clock Generation for the Deep Submicron Era", Ph.D dissertation, Chalmers University of Technology, Göteborg, Sweden, 2003.
12. Gupta S. and Najm F. N., "Power modeling for high level power estimation", IEEE Transactions on VLSI Systems, Vol. 8, Feb. 2000, pp. 18–29.
13. Gowan M. K. et al. "Power Considerations in the Design of the Alpha 21264 Microprocessor", Proc. of Design Automation Conf., San Francisco, CA, USA, June 1998, pp. 726–31.
14. Montanaro J. et al., "A 160-Mhz, 32-b, 0.5-W CMOS RISC Microprocessor", IEEE Journal of Solid-state Circuits, Vol. 31, Nov. 1996, pp. 1703–14.

# A Physically Oriented Model to Quantify the Noise-on-Delay Effect

Tobias Gemmeke and Tobias G. Noll

RWTH-Aachen, EECS, Schinkelstr. 2,  
52062 Aachen, Germany  
gemmeke@ieee.org, tgn@eecs.rwth-aachen.de,  
<http://www.eecs.rwth-aachen.de>

**Abstract.** The increase in dynamic on-chip noise has led to severe signal integrity problems in modern chip design. With respect to capacitive cross-talk there are two major fault mechanisms: false switching and the noise-on-delay effect. In a new approach an analytical, physically motivated model is proposed which quantifies the noise-on-delay effect aiming at quick timing verification. The accuracy of the model is validated in a comparison with the results of a circuit simulator. Moreover, its application in a standard design flow is demonstrated. As the model is based on physical circuit parameters it is also well suited for what-if analysis.

## 1 Introduction

In recent years the progress in technology and the increase in clock frequency has led to a growing importance of noise. Main drivers are the increased ratio of coupling to ground capacitances, faster on-chip slew rates together with unscaled wires, and reduced threshold voltages combined with high-speed circuit techniques.

The noise effects can mainly be classified into two groups: On one hand, logical information can be destroyed, i.e. noise causes a signal distortion beyond the specific logic threshold. On the other hand, there is the noise-on-delay effect, which causes a change in delay and reduces slew rate, respectively. This paper focuses on the latter.

To assure the functionality of a design, noise avoidance rules can be applied (e.g. [11,2]). However, a post-layout verification step is still necessary. When performing timing verification of a circuit one faces the problem of finding the right degree of decoupling. A fully coupled, complex netlist covers all effects at high simulation time. Still there is the problem of covering the worst-case stimulus. In contrast, a simplified netlist might not yield all timing violations.

In this paper, a physically motivated, analytical model is presented, that quantifies the noise-on-delay effect and allows to detect portions of a circuit, that require in-depth investigation (e.g. by circuit simulation) and possibly modifications in the circuit to work properly.

In the following section the noise-on-delay effect is briefly introduced using a bus-like circuit. Section 3 describes the classical approach to include cross-talk in



quick timing verification. The new model is derived in Sec. 4. Its use in a standard design flow is described in the last section. Within this flow, nets not covered by the model are flagged to indicate the need for higher order modeling. Such limits are for example global interconnect lines of significant interconnect resistance.

## 2 Noise-on-Delay Effect

The test circuit used for the following analysis is shown in Fig. 1. Simultaneous switching on the capacitively coupled nodes can increase (odd mode) as well as decrease (even mode) propagation delay. In the following, only the noise induced increase in delay (e. g. the propagation delay at 50 % of  $V_{dd}$ ) is covered. Hence, the aggressors always switch synchronously in the opposite direction as compared to the victim (odd mode).

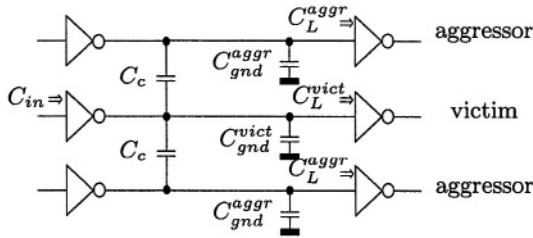


Fig. 1. Test circuit.

Figure 2 shows the signal transition of the victim line for different numbers of simultaneously switching aggressors  $na \in [0, 1, 2]$ . The time axes is normalized to the propagation delay with quiet aggressors  $t_{p0}$  ( $na = 0$ ).

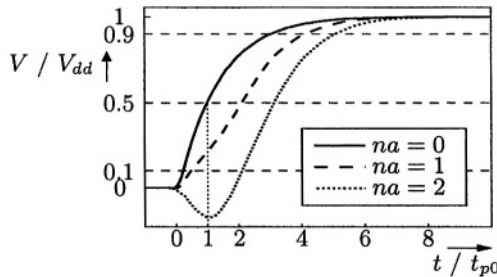


Fig. 2. Simulated waveforms on victim of the test circuit.

In this example a fan-out of  $FO = C_T / C_{in} = 32$  is applied to victim driver with the total capacitance  $C_T$  defined as

$$C_T = C_L + C_{wire} \tag{1}$$

with the total wiring capacitance of the victim and aggressor

$$C_{wire} = C_c + C_{gnd}^{aggr} = 2C_c + C_{gnd}^{vict}. \tag{2}$$

In Fig. 2 the coupling capacitance  $C_c$  is assumed to be half of the total wiring capacitance hence  $C_{gnd}^{vict} = 0$ , in this case. As can be observed from Fig. 2 coupling leads in this case to a delay penalty of more than 200 %.

### 3 Miller-Factor

A classical approach to cover the effect of cross-talk in a decoupled netlist is to add multiples of the coupling capacitances to the ground capacitances  $C_{gnd}$  of both affected nets. For this purpose each coupling capacitance is multiplied by the so-called Miller-factor  $m$ . Originally this factor is limited to the interval of 0 (even mode) to 2 (odd mode), which is based on the physical charge variation regarding large signal behavior.

To investigate this model the test circuit of Fig. 1 was simulated for 36 combinations of the circuit parameters fan-out  $FO$ , ratio of current drive of the aggressors to that of the victim  $r$  (measured in terms of saturation currents  $r = I_{sat}^{aggr} / I_{sat}^{vict}$ ), number of simultaneously switching aggressors  $na$ , and share of coupling capacitance per neighbor to wiring capacitance  $C_c / C_{wire}$ . The applied figures of the parameters are listed in Table 1.

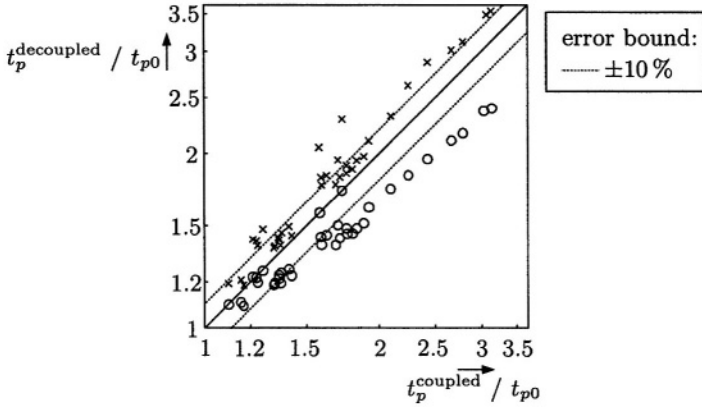
**Table 1.** Independent circuit parameters

$FO$	$r = I_{sat}^{aggr} / I_{sat}^{vict}$	$na$	$C_c / C_{wire}$
4	1.4:1 ( $2W_n = W_p$ )	1	0.25
16	1:1	2	0.5
32	4:1		

The propagation delays  $t_p$  for the 50%-levels ( $V(t_p) = 50\%V_{dd}$ ) of the decoupled netlist are plotted in Fig. 3 versus those of the coupled netlist. For each configuration the coupling capacitance to an active aggressor (odd mode) is decoupled with a Miller-factor of  $m=2$  (Fig. 3, marker: o). The coupling capacitances of quiet neighbors are not directly grounded to respect resistive shielding through the driver of the quiet aggressor line [4].

In some cases the delay times match, but overall the relative error varies with the circuit parameters with a maximum of more than 20 % in this case.

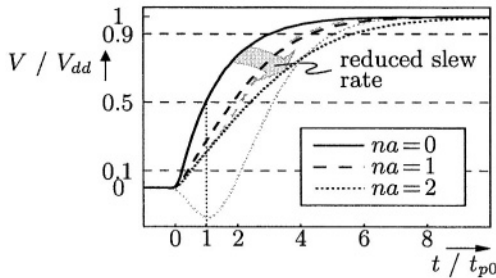
In [5,3] it was proposed to tackle this problem by adapting the Miller-factor for each circuit configuration. With this approach the Miller-factor lies within the interval of -1 (even mode) to 3 (odd mode). This methodology seems hard to integrate in a standard design flow for full chip verification. However, for the purpose of a worst-case analysis (odd mode) a Miller-factor of  $m = 3$  can be selected. The corresponding results show an relative error of more than 30% (Fig. 3, marker: x).



**Fig. 3.** Delay time analysis applying decoupling with a Miller-factor of  $\circ$ :  $m = 2$  and  $\times$ :  $m = 3$

### 4 $\Delta t$ -Model

The reason for the relatively low accuracy of the Miller-factor approach can be explained on the waveforms shown in Fig. 4. Decoupling with a Miller-factor effectively reduces the slew rate. In the case of one active aggressor  $na = 1$  the decoupled netlist matches the actual one quite well. In contrast, the actual waveform (apart from the bootstrap noise) appears to be shifted in time for two active aggressors ( $na = 2$ ). Apparently, this can not be modeled with a slowdown of the signal transition by modifications of the load capacitance.



**Fig. 4.** Signal transitions applying decoupling with a Miller-factor of  $m = 2$  (black), original transitions (grey) ( $r = 1.4:1$ ,  $FO = 32$ ,  $C_c / C_{wire} = 0.5$ ).

In order to overcome the limitations of the Miller-factor approach a new model is developed. For the verification of large designs it is essential to have a model of low complexity to reduce the computational effort. At the same time, sufficient accuracy is required as to avoid a design based on overly pessimistic analysis results.

The following model is based on the consideration of the circuit as a capacitive voltage divider. The estimated delay times simply add to the results of any timing analysis that works on a decoupled netlist.

### 4.1 Derivation

It is momentarily assumed that the aggregated driving capability of the active aggressors  $\sum^{na} I_{sat}^{aggr} / C_T^{aggr}$  is at least twice as large as the victim's  $I_{sat}^{vict} / C_T^{vict}$ . In this case, the aggressors are able to transfer an additional charge of  $\Delta Q = C_c \cdot \Delta V$  onto the victim node, which causes its voltage to raise above or drop below the supply voltage  $V_{dd}$  and  $Gnd$ , respectively (s. Fig. 2,  $na = 2$ ). Then, the victim's driver has to compensate for this bootstrap noise before it starts the intended transition. In this sense, the waveforms with and without simultaneous switching aggressors feature almost identical transitions but shifted in time. This shift in time is actually observed in Fig. 2. Hence, the waveform  $V(t, na)$  with  $na$  switching aggressors can be approximated by the transition with quiet neighbors  $V(t, na=0)$

$$V(t, na) \approx V(t - \Delta t(na), na = 0); \tag{3}$$

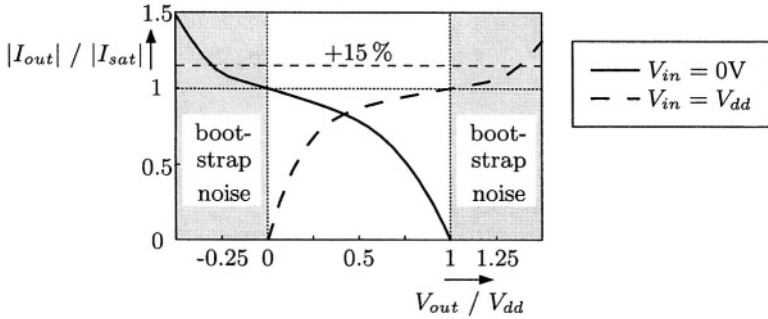
where the shift in time  $\Delta t$  is modeled assuming an average output current of the victim  $I_{avg}$

$$\Delta t(na) = \frac{C_c(na) \cdot \Delta V}{I_{avg}}. \tag{4}$$

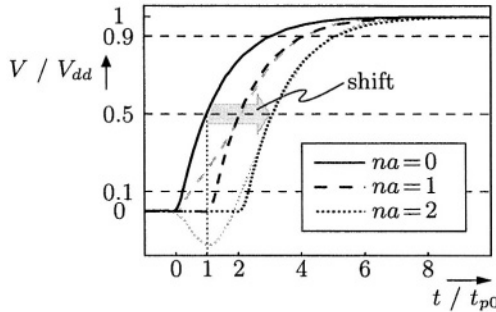
Typical applications feature a voltage swing  $\Delta V$  equal to the supply voltage  $V_{dd}$ . In a worst-case analysis the coupling capacitance  $C_c$  is equal to the sum of coupling capacitances of all active aggressors (odd mode). In general, the output current depends on the output voltage as shown in Fig. 5. If the victim's transition is delayed until the input signals of the gate have settled, the transistors of the victim are stimulated in the saturation region, i. e. (neglecting the channel length modulation) the output current is constant and equal to the saturation current  $I_{sat}^{vict}$ . So, for moderate bootstrap noise the saturation current gives a good estimated for the compensation current  $I_{avg} \approx I_{sat}^{vict}$  flowing during the time  $\Delta t$ .

The shifted waveforms as derived by this model are shown in Fig. 6. For  $no=2$  the  $\Delta t$ -model matches the simulated one not only at one specific voltage level (cf. Sec. 3) but for the whole transition. In the case of only one active aggressor ( $na = 1, r = 1.4 : 1$ ) the predicted output signal fails in the beginning. However the waveform is properly approximated well before the 50 % level, although the simulation of the coupled circuit actually features no bootstrap noise at all. Hence, the starting assumption of this section assuming a largely higher drive of the aggressors is relieved, as it is sufficient that their aggregated driving capability is slightly larger than the victim's.

Moreover, the  $\Delta t$ -model can be used to compute probability distributions of delay times. For this purpose the set of quiet and active neighbors is permuted.



**Fig. 5.** Output current of an inverter as a function of applied voltages (transistors sized for equal saturation currents:  $|I_{sat}^{NMOS}| = |I_{sat}^{PMOS}|$ ).



**Fig. 6.** Signal transitions using the  $\Delta t$ -model to shift simulated waveform with quiet neighbors  $na = 0$  ( $r = 1.4 : 1$ ,  $FO = 32$ ,  $C_c / C_{wire} = 0.5$ ).

For each subset only coupling capacitances of active neighbors are included in the parameter  $C_c$  of the model. The probability distribution is derived by relating the probabilities of the subsets to the corresponding delay times.

### 4.2 Limitations

To verify the model derived above, the test circuit was simulated with various values of the circuit parameters as described in Sec. 3.

In Fig. 7 the delays estimated by the  $\Delta t$ -Model are plotted versus those of the coupled circuit simulation. It can be seen that almost all figures of the  $\Delta t$ -model lie within a 10% error bound.

Also indicated are the limits of the model as derived so far. The marked examples in the upper region feature a current drive ratio of  $r = 4 : 1$ , and both aggressors are switching. Here, excessive bootstrap noise occurs, i. e. the circuit is not properly designed. In the case of the lower outliers, a single aggressor is switching, that features identical drive as the victim. As these situations can be easily detected from the netlist, an approach to extend the model for the use in a standard design flow is described in the next section.

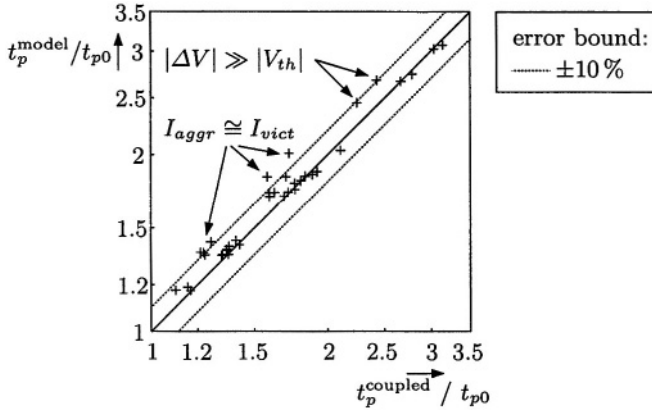


Fig. 7. Delay time analysis applying the  $\Delta t$ -model.

## 5 Application

Figure 8 shows an example of a standard design flow, that is extended to include the noise-on-delay effect in timing analysis. Any delay time  $t_{p0}$  is simply incremented (odd mode) by  $\Delta t$  as determined by the model. The original delays  $t_{p0}$  are easily derived from a decoupled netlist.

In the following, the  $\Delta t$ -model is extended for its use in the standard design flow of fig. 8. The critical cases are covered as follows:

### 5.1 Equal Drive $m=2$

The situation of equally driven lines is rare in worst-case delay analysis. However, the waveforms of the decoupled and coupled circuit simulation match, if the flagged net is decoupled to  $Gnd$  with the traditional Miller-factor of  $m = 2$ . Then, the timing analysis of the appropriately decoupled netlist covers the equal drive case. Hence, there is no additional delay in these cases ( $\Delta t = 0$ ).

### 5.2 Large Noise $\Delta V$

It is important to note, that excessive voltage pulses on quiet victim lines can lead to logic failures. Simple estimation of the induced voltage spike allows to flag the cases, when the predicted noise voltage  $\hat{v}_{out}$  is above or below a specific threshold. For this purpose, the amplitude of the pulse can be modeled with the transition time  $t_t^{aggr}$  of the aggressor and the victim's time constant  $\tau$  [7]

$$t_t^{aggr} \approx 2.75 \cdot C_L^{aggr} \cdot V_{dd} / I_{sat}^{aggr} \tag{5}$$

$$\tau = C_L^{vict} \cdot V_{dd} / I_{sat}^{vict} \tag{6}$$

$$\hat{v}_{out} \approx \begin{cases} C_c / C_L^{vict} & \tau \gg t_t^{aggr} \\ C_c / C_L^{vict} \cdot \tau / t_t^{aggr} & \tau \ll t_t^{aggr} \end{cases} \tag{7}$$

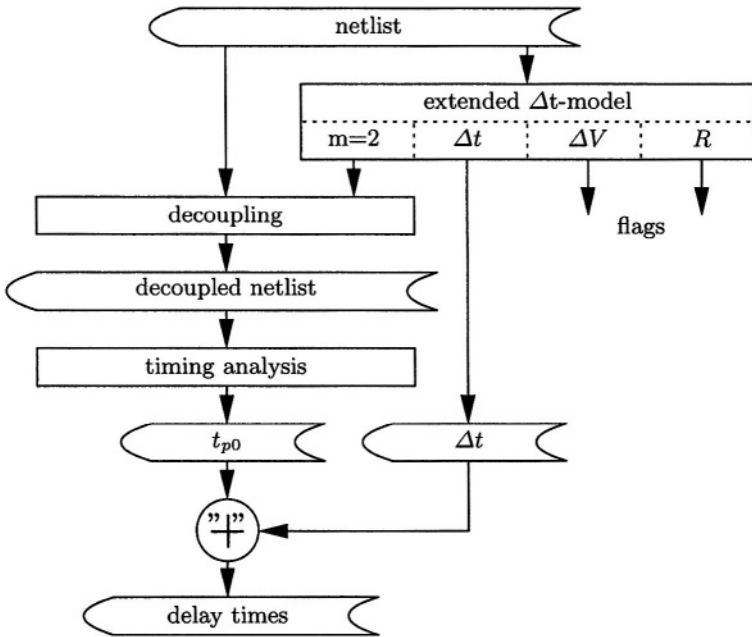


Fig. 8. Standard design flow with extended  $\Delta t$ -model.

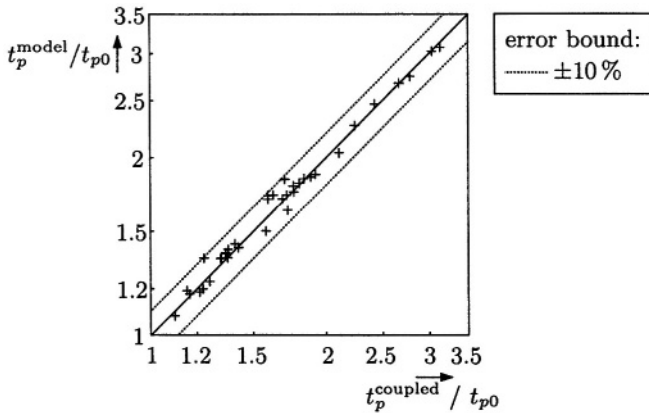


Fig. 9. Delay times applying the extended  $\Delta t$ -model  $t_p^{\text{model}}$ .

Voltage levels largely above  $V_{dd}$  or below  $Gnd$  lead to a significant increase of the output current as can be seen in Fig. 5. The additional currents in the transistor that are normally in the cut-off region, and the forward-biased pn-junction of the drain region add to the saturation current of the driving transistor. A simple fix to the  $\Delta t$ -model for these cases is to add an offset to  $I_{avg}$ . Here, this increase is set to 15 %, which is derived empirically from the simulations.

The results of this extended  $\Delta t$ -model are compared to those of the circuit simulation of the coupled netlist in Fig. 9. Apparently, the extensions successfully reduce the error of the model.

### 5.3 Long Line $R$

Long lines of significant resistance  $R$  are flagged to indicate the need for higher order modeling as their effects are not covered by the  $\Delta t$ -model. The net needs to be modeled as  $RC$  circuit if  $R_{eq}/R \leq 20$  [8].

### 5.4 Windowing

Additionally, the calculation of the coupling capacitance  $C_c$  in (3) can include knowledge of switching activities based on activity windows of logic dependencies [6] or relative arrival times [1,10], the latter being challenging due to the inherent interactions [9].

### 5.5 Experimental Results

An implementation in PERL of the  $\Delta t$ -model was used to analyze a datapath for high-performance digital signal processing. The design features about 1.3 mill. transistors and about 1 mill. wiring capacitances of which 90 % are coupling capacitances. The analysis of the coupled netlist runs approximately 15 minutes on a 750 MHz UltraSPARC-III. For 1.1 % of all nets the  $\Delta t$ -model determines an increase in delay due to coupling of more than 800 ps ( $f_{clk}=125$  MHz), showing the importance of covering cross-talk in timing analysis.

## 6 Conclusion

In a new approach, a physically motivated model was derived which allows to quantify the noise-on-delay effect due capacitive cross-talk. It relies on the results of any existing approach to model delays of an appropriately decoupled netlist. The additional timing, as determined by this  $\Delta t$ -model, is simply added to the delay time of the corresponding path. The accuracy of the model was confirmed in a comparison with circuit simulation results of a fully coupled netlist. With the presented analytical expression of reasonable small complexity the timing of a design can be quickly verified. Moreover, the model relies on physical circuit parameters enabling what-if analysis.



## References

1. Agarwal, K. et al.: Efficient Generation of Delay Change Curves for Noise-Aware Static Timing Analysis. *Int. Conf. on VLSI Design (VLSID)* (2002) 77–84
2. Becer, M. R., Blaauw, D., Zolotov, V., Panda, R., Hajj, I. N., Analysis of Noise Avoidance Techniques in DSM Interconnects using a Complete Crosstalk Noise Model. *Design, Automation and Test in Europe Conf.* (2002) 456–63
3. Chen, P., Kirkpatrick, D. A., Keutzer, K.: Miller Factor for Gate-Level Coupling Delay Calculation. *Int. Conf. on Computer Aided Design (ICCAD)* (2000) 68–74
4. Dartu, F., Pileggi, L. T., Calculating Worst-Case Gate Delays Due to Dominant Capacitance Coupling. *Design Automation Conf. (DAC)* (1997)
5. Kahng, A., Muddu, S., Sarto, E. On Switch Factor Based Analysis of Coupled RC Interconnects. *Design Automation Conf. (DAC)* (2000) 79–84
6. Kirkpatrick, D. A., Sangiovanni-Vincentelli, A. L.: Digital Sensitivity: Predicting Signal Interaction using Functional Analysis. *Int. Conf. on Computer-Aided Design* (1996) 536–41
7. Roca, M., Moll, F., Rubio, A., Electric design rules for avoiding crosstalk in microelectronic circuits. *14th Int. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)* (1994) 76–83
8. Sakurai, T., Approximation of Wiring Delay in MOSFET LSI. *IEEE Journal of Solid-State Circuits (JSSC)* **SC-18(4)** (1983) 418–26
9. Sapatnekar, S. S.: On the Chicken-and-Egg Problem of Determining the Effect of Crosstalk on Delay in Integrated Circuits. *Proc. IEEE 8th topical meeting on Electrical Performance Packaging* (1999) 245–8
10. Sasaki, Y. et al.: Crosstalk Delay Analysis of a 0.13- $\mu\text{m}$  Node Test Chip and Precise Gate-Level Simulation Technology. *IEEE Journal of Solid-State Circuits (JSSC)* **38(5)** (2003) 702–8
11. Shepard, K., Narayanan, V.: Conquering Noise in Deep-Submicron Digital ICs. *IEEE Design and Test of Computers* (1998) 51–62

# Noise Margin in Low Power SRAM Cells

S. Cserveny, J.-M. Masgonty, and C. Piguet

CSEM SA, Neuchâtel, CH  
Stefan.cserveny@csem.ch

**Abstract.** Noise margin at read, at write and in stand-by is analyzed for the 6 transistor SRAM cell in a  $0.18\ \mu\text{m}$  process considering specific low power conditions such as low supply voltage and source-body biasing. These conditions reduce the noise margin. By using an asymmetrical cell design in which read is performed only on one of the two complementary bit lines, the noise margin can be improved and the bias limits extended for a reduced power consumption.

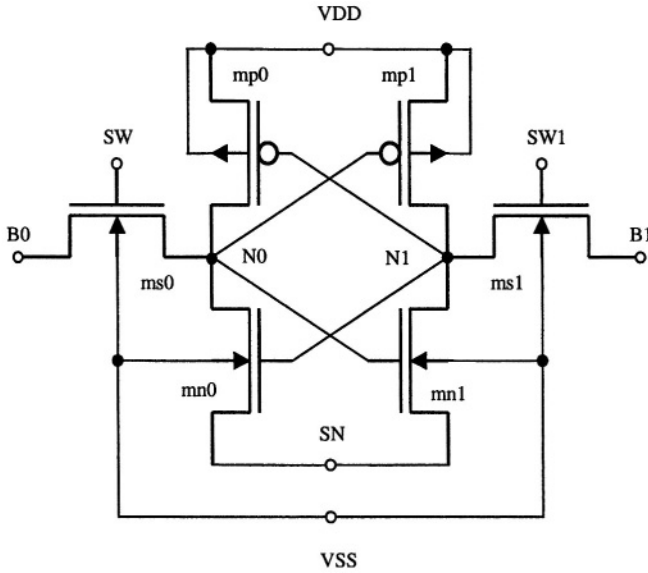
## 1 Introduction

The increasing complexity of integrated circuits results in a general power awareness [1], especially for the increasing number of portable battery (or accumulator) powered applications. In processor-based systems-on-chip the memories occupy an increasing part of the area, limit most of the time the speed and are the main part of the power consumption. The trend in the scaled down deep sub-micron technologies is toward an increased contribution of the static consumption [2], a major problem for the most frequent SRAM application, the cache memories [3, 4]. This increased contribution becomes even more important in the extreme case of applications with very long idle modes as in wireless micro sensor systems [5], in which the standby period is much longer than the active mode.

The dynamic power consumption is directly related to the VDD supply value, therefore there is a tendency to reduce this value with the process scaling, and even more if allowed by the speed requirements. Several techniques were proposed to further reduce the dynamic power; among them the physically divided read bit line [6, 7] using an asymmetrical cell, which is also interesting as a low leakage approach [8].

The static consumption is due to the different leakage currents described in [9]. There are digital blocks that can be switched off in long stand-by modes, however flip-flop based storage circuits have to be biased properly to maintain their state. The source-body biasing allows important leakage reduction in such SRAM cells, while the supply voltage is maintained to guarantee safe information storage [8, 10].

In this paper the safety of the 6-transistor SRAM cell operation at read, write and stand-by is analyzed by evaluating the effects on its noise margin of a reduced supply voltage and the use of the source-body bias leakage reduction technique. A reference minimum size symmetrical SRAM cell is considered first and the improvement possibilities offered by the asymmetrical cell are explored next. Even if this analysis is done considering a  $0.18\ \mu\text{m}$  process, for which a very low power SRAM has been realized, the identified tendencies are to be considered also in deeper scaled processes.



**Fig. 1.** The 6T SRAM cell with the possibility to separate the select gate signals (asymmetrical cell) and to apply a source-body bias to the NMOS transistors for leakage reduction

## 2 The SRAM Cell

Fig. 1 presents the generic 6-transistor SRAM cell considered in this paper.

In the basic symmetrical cell there is a unique word select signal applied to both  $ms_0$  and  $ms_1$  select NMOS gates (SW1 connected to SW) and the common source SN of the two pull-down NMOS  $mn_0$  and  $mn_1$  is connected to the ground VSS. The W/L ratios of the transistors on the two sides of the cell are the same. The reference cell in this  $0.18\ \mu\text{m}$  process uses minimum size transistors  $0.22\ \mu\text{m} / 0.18\ \mu\text{m}$  for all inverter transistors ( $mp_0$ ,  $mp_1$ ,  $mn_0$  and  $mn_1$ ) and longer  $0.22\ \mu\text{m} / 0.295\ \mu\text{m}$   $ms_0$  and  $ms_1$  select transistors.

In Fig. 1 the possibility for separate select gate signals SW and SW1 has been shown, corresponding to the asymmetrical cell described in [6, 7]. With this cell read is performed only on the bit line B0 as only SW goes high. The dynamic consumption at read can be further reduced by physically splitting these single read bit lines into several sub-bit lines; as the sub-bit lines are connected to a fraction of the cells in the column, the capacitive load to be discharged is significantly reduced. Out of all pre-charged sub-bit lines only those for which a “1” is read will be discharged. On the other hand both bit lines are needed to write safely at any supply value without voltage boosters, therefore this asymmetrical cell needs separated SW and SW1 select signals, with a small area penalty. As only one select transistor is activated at read, the W/L ratios for the select and pull down NMOS transistors on the two sides of the cell can be optimized differently resulting in an asymmetrical design.

In the considered  $0.18\ \mu\text{m}$  process the static leakage is strongly dominated by the subthreshold current of the turned-off NMOS transistors, the minimum size PMOS

leaking about 100 times less. With present process scaling tendencies towards much deeper nanometric devices, special care is needed to deal with important tunneling gate currents and gate induced drain leakage [9], even if the subthreshold leakage remains dominant [10], and it seems that for all these leakage components the NMOS will leak more than the PMOS. A source-body bias increasing the threshold is the best way to reduce the subthreshold leakage, however, it is interesting to notice that the source-body biasing is beneficial also for the gate current as it reduces the gate fields.

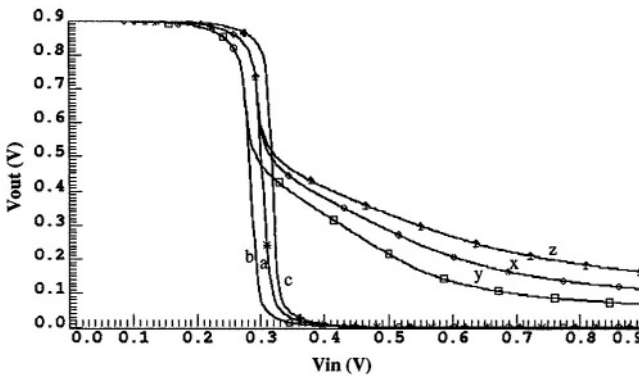
In order to allow NMOS source-body biasing in the 6 transistors SRAM cell, the common source of the cross-coupled inverter NMOS (SN in Fig. 1) is not connected to the body. Body pick-ups can be provided in each cell or for a group of cells and they are connected to the VSS ground. A positive VSB bias between SN and VSS will reduce the subthreshold leakage of a non-selected cell (SW and SW1 at VSS). It is important to notice that, as both select transistors ms0 and ms1 have their gates at ground, their leakage current will decrease rapidly with this bias because, besides the body effect on the threshold voltage, there is the very strong exponential effect related to the increasing negative gate-source voltage; unfortunately, no such effect occurs for the cut-off pull-down transistor in the flip-flop mn0 or mn1, only the body effect reduces its leakage because it has the same SN node potential on its gate and source.

### 3 Read and Stand-By Noise Margin

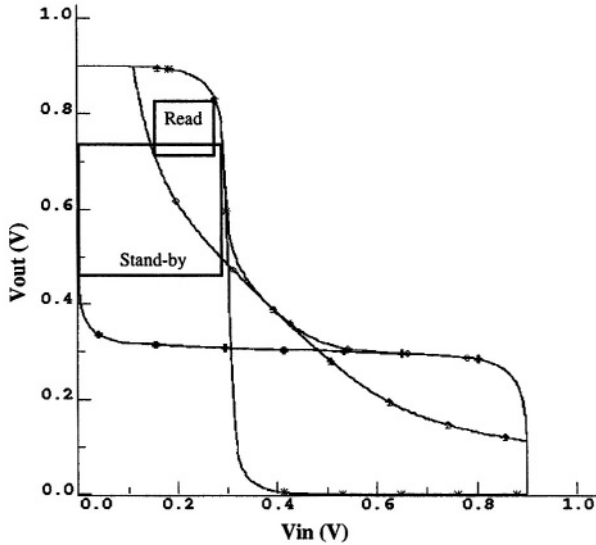
In order to estimate the safety of the data retention, the noise margin is extracted with the method described in the literature [11, 12] as the maximum size square that can be nested into the cross-coupled voltage transfer characteristics. Examples of transfer characteristics to be considered in stand-by and at read are shown in Fig.2.

The central characteristics (a and x) in Fig. 2 are those of the reference cell.

In stand-by the flip-flop inverter (mp0 and mn0 or mp1 and mn1) transfer characteristic is considered. In the three examples in Fig. 2 (a, b and c) the effect of using a stronger (b) or a weaker (c) inverter NMOS is compared with the reference (a).



**Fig. 2.** Stand-by and read transfer characteristics at 125°C VDD = 0.9 V and V(SN) = 0 for the N-fast P-slow corner; the MOS W/L ratios in  $\mu\text{m}/\mu\text{m}$  are 0.22/0.18 (a, x, z), 0.42/0.18 (b, y) and 0.22/0.295 (c) for mn, 0.22/0.295 (x, y) and 0.22/0.18 (z) for ms and 0.22/0.18 for mp



**Fig. 3.** Worst case (N-fast P-slow 125°C) stand-by and read static noise margin analysis for the reference symmetrical cell at  $V(\text{SN}) = 0$  and  $V_{\text{DD}} = 0.9 \text{ V}$

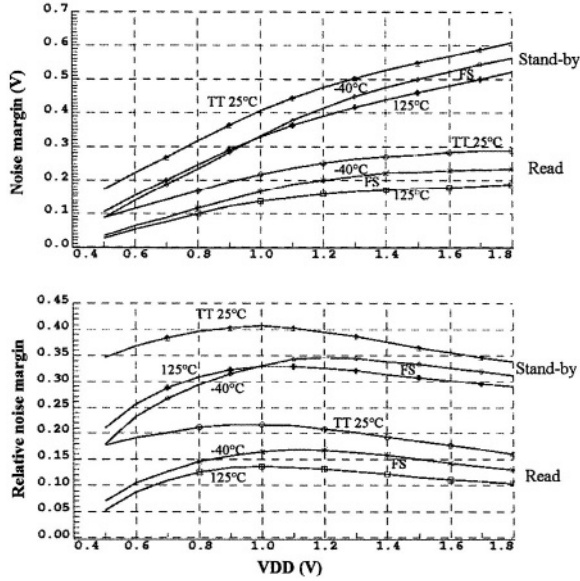
At read (x, y and z in Fig. 2), the loading effect of the select MOS (ms0 respectively ms1), with the bit line precharged at  $V_{\text{DD}}$  when the read starts, shifts the low output voltage part of the characteristic upwards; this shift is more important when the inverter NMOS is weaker (x above y) and the select NMOS stronger (z above x).

As a result, the noise margin is much smaller at read than in stand-by (Fig. 3).

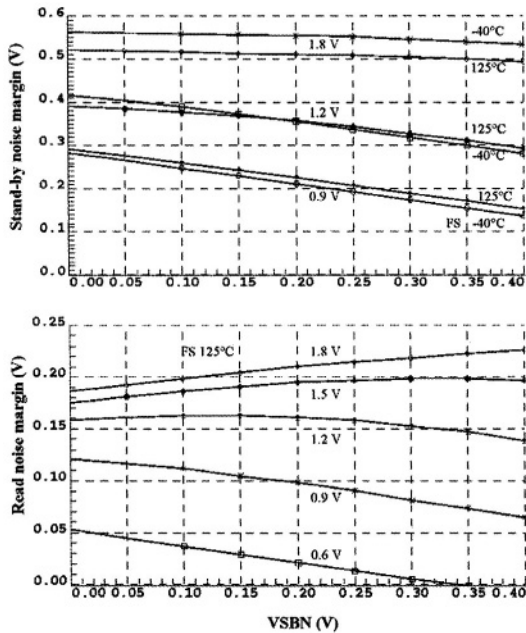
Such stand-by and read noise margins of the reference symmetrical cell extracted by simulation are analyzed in the following results.

As the  $V_{\text{DD}}$  supply voltage decreases from its nominal 1.8 V value, both stand-by and read noise margins decrease (Fig. 4). As many perturbation signals are related to the  $V_{\text{DD}}$  value, it is interesting to analyze also the noise margin to  $V_{\text{DD}}$  ratio; this relative noise margin first improves at moderate  $V_{\text{DD}}$  reduction before taking down at the lower values. As expected, the N-fast P-slow process corner is the worst case. Also the highest temperature is worst, except for the stand-by margin at low  $V_{\text{DD}}$ .

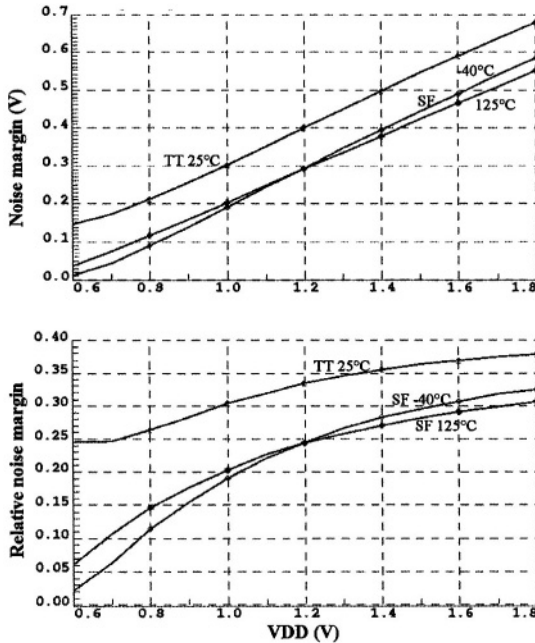
If a NMOS source-body bias  $V_{\text{SBN}}$  is applied between the SN node and the VSS ground (Fig. 5), the worst-case stand-by noise margin decreases, mainly at low  $V_{\text{DD}}$ ; notice the temperature effect change between high and low  $V_{\text{DD}}$ . There is no such temperature effect change for the read noise margin (always N-fast P-slow highest temperature worst case), however it increases with  $V_{\text{SBN}}$  at high  $V_{\text{DD}}$  and decreases only at low  $V_{\text{DD}}$ . Consequently, at low  $V_{\text{DD}}$ , where the  $V_{\text{SBN}}$  source-body bias dangerously reduces the read noise margin while the stand-by margin, even if reduced, remains acceptable, it is recommended to switch this bias off at read [8, 10].



**Fig. 4.** VDD supply voltage dependence of the stand-by and read noise margin (top) and noise margin over VDD ratio (bottom) for the symmetrical reference cell without source-body bias for different process corners (TT typical, FS N-fast P-slow worst case) and temperatures



**Fig. 5.** VSBN (SN node source-body bias) dependence of the stand-by (top) and read (bottom) noise margin for the symmetrical reference cell for the worst-case N-fast P-slow process corner at different VDD supply voltages and temperatures



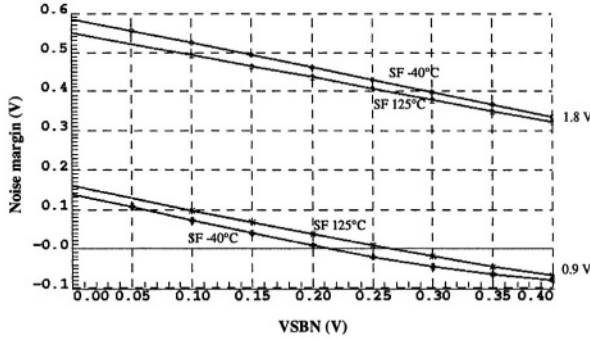
**Fig. 6.** VDD supply voltage dependence of the write noise margin (top) and noise margin over VDD ratio (bottom) for the symmetrical reference cell without source-body bias for different process corners (TT typical, SF N-slow P-fast worst case) and temperatures

## 4 Noise Margin at Write

In order to modify the data stored in the cell the critical task is for the bit line that has to pull down the internal node N0 or N1 previously high for the selected cell (SW and SW1 high). The write noise margin is defined as the maximum voltage on this bit line for which the cell flips; in the presence of a VSBN bias, its value is subtracted from the bit line flipping value.

Both the write noise margin and its VDD relative value decrease with the VDD supply (Fig. 6). There is a severe limitation for the minimum acceptable VDD value that gives the bottom requirement of at least a 10% relative margin. As expected, the N-slow P-fast process corner is the worst case, while the temperature effect changes between high and low VDD values.

In the presence of a NMOS source-body bias VSBN, the write noise margin further decreases (Fig. 7), becoming totally unacceptable at low VDD. In fact, for the noise margin requirements, the write is far more demanding than the read. At 0.9 V with a VSBN = 0.3 V the worst case stand-by margin of 173 mV is very good, the 80 mV read margin is just under a 10% limit, however at write there is no margin left at all, it is negative; it is even more important to connect the SN node to ground at write than at read, however both are welcome as done in [8, 10].

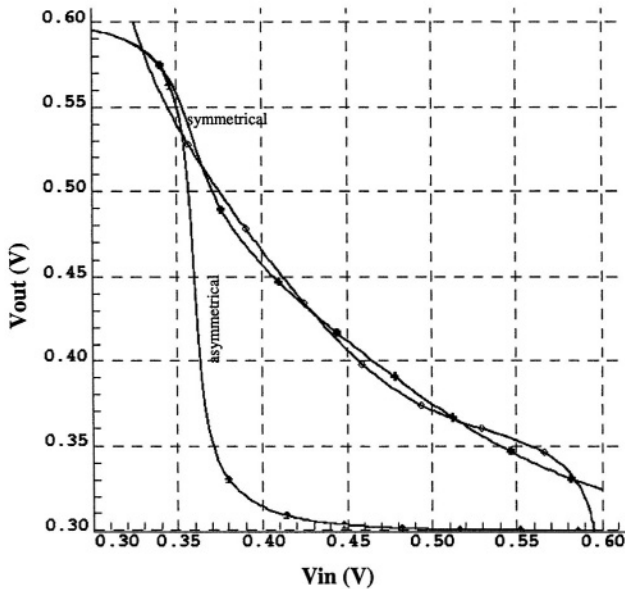


**Fig. 7.** VSBN (SN node source-body bias) dependence of the write noise margin for the symmetrical reference cell in the worst-case N-slow P-fast process corner at different VDD supply voltages and temperatures

### 5 Noise Margin Improvement with the Asymmetrical Cell

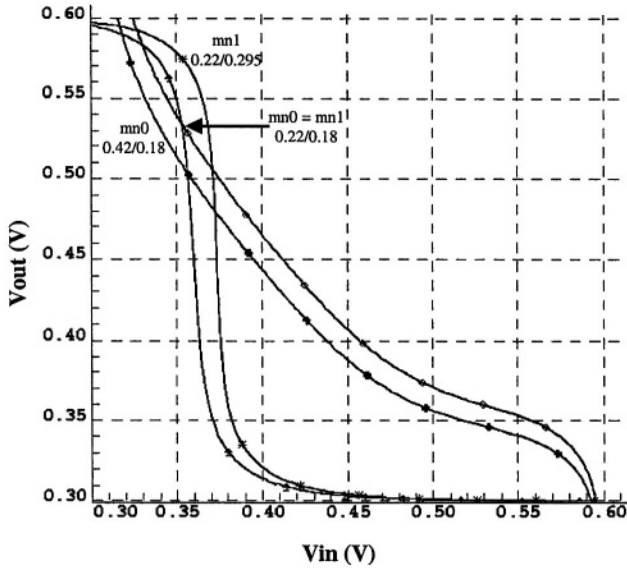
The asymmetrical cell has been proposed in [6, 7] to reduce the dynamic power. Here the capability of this approach to improve the noise margin at low VDD and in the presence of a VSBN bias is analyzed.

In the following example (Fig. 8 and Fig. 9) an extreme bias case is used to better demonstrate what can be achieved by making the cell asymmetrical.



**Fig. 8.** Worst case (N-fast P-slow 125°C) read noise margin analysis for the symmetrical and asymmetrical cells with the reference MOS W/L ratios at VDD = 0.6 V and VSBN = 0.3 V





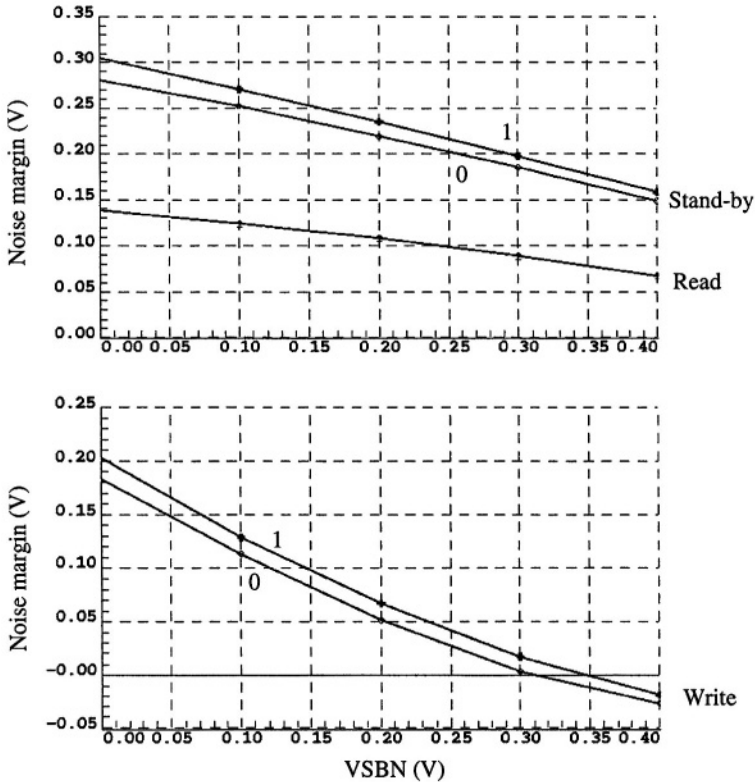
**Fig. 9.** Worst case (N-fast P-slow at 125°C) read noise margin analysis comparison for the asymmetrical cell using the reference W/L ratios ( $mn0=mn1$   $0.22\mu\text{m}/0.18\mu\text{m}$ ) with a longer  $mn1$  ( $0.295\ \mu\text{m}$ ) and with a larger  $mn0$  ( $0.42\ \mu\text{m}$ ) at  $V_{DD} = 0.6\ \text{V}$  and  $V_{SBN} = 0.3\ \text{V}$

In this example  $V_{DD} = 0.6\ \text{V}$  is extremely low especially because a  $V_{SBN}$  bias is also used. The cross-coupled transfer characteristics of the symmetrical cell not only show a very small noise margin, they also display 5 crossings instead of 3 (Fig. 8); such 5 crossings are observed also at  $V_{DD} = 0.9\ \text{V}$  (Fig. 3 in [8]), however with the 2 extra crossings not so far from the middle one. As in the asymmetrical cell read activates only SW, the transfer characteristic of the other side is that of the inverter  $mn1$  and  $mp1$  that is no more loaded by the cut-off  $ms1$  select NMOS. Using still the same W/L values the noise margin is not improved, however the 5 crossings are avoided.

Optimizing differently the W/L ratios on the two sides in a further step, the noise margin can be substantially increased with a stronger  $mn0$  and a weaker  $mn1$  (Fig. 9).

In the asymmetrical cell the requirements are very different on the two sides. On the read side  $mn0$  and  $ms0$  should be strong enough for the required driving capability and the ratio of their W/L (called transfer ratio) should be large enough for the desired read disturb noise margin. More relaxed requirements apply to the other side used only to write and keep the stored data. It has been shown in Fig. 2 that a weaker  $mn$  shifts the inverter characteristic to the right and a stronger  $mn$  and a weaker  $ms$  (i.e. a larger transfer ratio) shifts the  $ms$  loaded inverter characteristic downwards closer to the axis, therefore to the left in the cross-coupled representation in Fig. 9. Notice that in the symmetrical cell increasing the  $mn$  width (i.e. the transfer ratio) with the same value as in Fig. 9, the cross-coupled characteristics are as bad as those in Fig. 8 just shifted to the left, with the same 5 crossings shape and small margin. The freedom to shift separately the two characteristics in the asymmetrical cell allows for a better noise margin, or a smaller transfer ratio constraint for a required margin.

The reduced leakage cell proposed in [8], besides the source-body bias technique, uses less-leaking non-minimal length transistors whenever effective; the asymmetrical



**Fig. 10.** VSNB (SN node source-body bias) dependence of the worst case noise margin for the designed asymmetrical cell at  $V_{DD} = 0.9$  V for stand-by and read (N-fast P-slow  $75^{\circ}\text{C}$ ) and for write (N-slow P-fast  $-25^{\circ}\text{C}$ ) in the states 1 (node N1 high) and 0 (node N0 high)

cell is favorable to implement this. As shown in Fig. 9, a longer  $mn1$  improves the noise margin, and a relaxed transfer ratio requirement allows also for a longer  $mn0$  if its width and the  $ms0$  W/L are adapted for the desired speed; the VSNB bias is very effective to reduce the  $ms0$  and  $ms1$  leakage (they have a negative gate-source voltage in a non-selected cell with SW and SW1 at ground), therefore they can be minimum length. These arguments have been used to design an asymmetrical cell that has much less leakage, more driving capability and a little better noise margin (Fig. 10) than the reference cell; the area penalty is about 25%, however also a larger number of metal pitches takes advantage of this area increase.

Notice that the noise margin (as the stand-by leakage) can be better for the state 1 than for the state 0 that has more stringent design requirements. Locally switched source-body bias is used [8] to keep the active mode read and write noise margins and the speed as without such a bias. First measurements of the SRAM using this cell in an industrial application show it is fully functional.

## 6 Conclusions

Low-power SRAM use reduced VDD supply for less dynamic power consumption and source-body biasing for static leakage reduction. The present analysis shows that such conditions reduce the noise margin of the SRAM cell, and therefore the range of their applicability is limited by the noise margin requirements for a safe operation.

An asymmetrical cell approach has been proposed before to reduce the dynamic power [6, 7] and the leakage [8]. It is shown here that the noise margin performance at low VDD and in the presence of a source-body bias can be improved by taking advantage in the design of the difference in requirements on the side used for read and the side used only to write and keep the stored data.

Simulated results are presented for the asymmetrical cell designed in a standard single threshold voltage 0.18  $\mu\text{m}$  digital process. The low-power SRAM using this cell has been integrated in a RF transceiver circuit for an industrial wireless system and the measurements show that it is fully functional with the desired performance.

## References

- [1] T. Sakurai, "Perspectives on Power-Aware Electronics", Plenary Talk 1.2, *Proc. ISSCC 2003*, San-Francisco, CA, Feb. 9-13, 2003, pp. 26-29
- [2] C. Pigué, S. Cserveny, J.-F. Perotto, J.-M. Masgonty: Techniques de circuits et méthodes de conception pour réduire la consommation statique dans les technologies profondément submicroniques, *Proc. FTFC'03*, pp. 21-29
- [3] H. Hanson, M.S. Hrishikesh, V. Agarwal, S.W. Keckler, D. Burger, "Static Energy Reduction Techniques for Microprocessor Caches", *IEEE Trans. VLSI Systems*, vol. 11, pp. 303-313, June 2003
- [4] A. Agarwal, H. Li, K. Roy, "A Single-Vt Low-Leakage Gated-Ground Cache for Deep Submicron", *IEEE J. Solid-State Circuits*, vol. 38, pp. 319-328, Feb. 2003
- [5] A. Chandrakasan, R. Min, M. Bhardwaj, S. -H. Cho, A. Wang, "Power Aware Wireless Microsensor Systems", *Proc. ESSCIRC 2002*, pp. 47-54
- [6] J.-M. Masgonty, S. Cserveny, C. Pigué "Low Power SRAM and ROM Memories", *Proc. PATMOS 2001*, paper 7.4
- [7] S. Cserveny, J.-M. Masgonty, C. Pigué, F. Robin, "Random Access Memory", *US Patent US 6'366'504 B1*, April 2, 2002
- [8] S. Cserveny, J.-M. Masgonty, C. Pigué, "Stand-by Power Reduction for Storage Circuits", in *J. J. Chico and E. Macii (Eds.): PATMOS 2003, LNCS 2799*, pp. 229-238, Springer 2003
- [9] K. Roy, S. Mukhopadhyay, H. Mahmoodi-Meimand, "Leakage Current Mechanism and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuit", *Proc. IEEE*, vol. 91, pp. 305-327, Feb. 2003
- [10] A. Agarwal, K. Roy: "A Noise Tolerant Cache Design to Reduce Gate and Sub-threshold Leakage in the Nanometer Regime", *Proc. ISLPED'03*, pp. 18-21
- [11] E. Seevinck, F.J. List, J. Lohstroh "Static-Noise Margin Analysis of MOS SRAM Cells", *IEEE J. Solid-State Circuits*, vol. 22, pp. 748-754, Oct. 1987
- [12] A. J. Bhavnagarwala, X. Tang, J. D. Meindl "The Impact of Intrinsic Device Fluctuations on CMOS SRAM Cell Stability", *IEEE J. Solid-State Circuits*, vol. 36, pp. 658-665, April 2001

# Delay Evaluation of High Speed Data-Path Circuits Based on Threshold Logic

Peter Celinski<sup>1</sup>, Derek Abbott<sup>1</sup>, and Sorin D. Cotofana<sup>2</sup>

<sup>1</sup> Centre for High Performance Integrated Technologies & Systems (CHiPTec)  
The Department of Electrical and Electronic Engineering,  
The University of Adelaide, SA 5005, Australia.  
celinski@eleceng.adelaide.edu.au

<sup>2</sup> Computer Engineering Group, Electrical Engineering Department,  
Delft University of Technology,  
Mekelweg 4, 2628 CD Delft, The Netherlands.  
sorin@ce.et.tudelft.nl

**Abstract.** The main result is the development, and delay comparison based on Logical Effort, of a number of high speed circuits for common arithmetic and related operations using threshold logic. The designs include 8 to 64-input AND, 4-bit carry generate, and the carry-out of a (7,3) parallel (population) counter. The circuits are designed using both domino gates and the recently proposed CMOS Charge Recycling Threshold Logic (CRTL). It is shown that compared to domino, the CRTL design examples are typically between 1.3 and 2.7 times faster over a wide range of load values, while presenting the same input capacitance to the driver.

## 1 Introduction

As the demand for higher performance very large scale integration processors with increased sophistication grows, continuing research is focused on improving the performance and power dissipation of the arithmetic and other units.

The aim of this paper is firstly to propose a Logical Effort (LE) [3] based delay model for CRTL gates. Secondly, this model is used as the basis for demonstrating the performance advantage of Charge Recycling Threshold Logic [2] over conventional CMOS dynamic logic for a range of circuits used in processor datapaths. The proposed circuits include very wide AND, 4-bit carry, and (7,3) counter critical path. The main motivation for using an LE based delay comparison is the desire to avoid the common and largely unsatisfactory presentation of circuit performance results commonly found in the literature in the form of delay numbers with insufficient information to allow comparison across processes and loading conditions.

We begin in Section 2 by giving a brief overview of threshold logic, including a description of CRTL. Section 3 briefly reviews Logical Effort and presents the delay model for CRTL gates. The circuit design examples are presented and evaluated in Section 4. Finally a conclusion and suggestions for future work are given in Section 6.

## 2 Threshold Logic AND CRTL

Threshold logic (TL) was introduced over four decades ago, and over the years has promised much in terms of reduced logic depth and gate count compared to conventional logic-gate based design. Efficient CMOS TL gate implementations have recently become available, and a small number of applications based on TL gates have demonstrated its ability to achieve high operating speed, low power dissipation and significantly reduced area [1].

A threshold logic gate is functionally similar to a hard limiting neuron. The gate takes  $n$  binary inputs  $x_1, x_2, \dots, x_n$  and produces a single binary output  $y$ . A linear weighted sum of the binary inputs is computed followed by a thresholding operation. The Boolean function computed by such a gate is specified by the gate threshold,  $T$ , and the weights  $w_1, w_2, \dots, w_n$ , where  $w_i$  is the weight corresponding to the  $i^{\text{th}}$  input variable  $x_i$ . The binary output  $y$  is given by

$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^n w_i x_i \geq T \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

A TL gate can be programmed to realize many distinct Boolean functions by adjusting the threshold  $T$ . For example, an  $n$ -input TL gate with  $T = n$  will realize an  $n$ -input AND gate and by setting  $T = n/2$ , the gate computes a majority function. This versatility means that TL offers a significantly increased computational capability over conventional NAND-NOR-NOT logic.

We now briefly describe the realization for CMOS threshold gates presented in [2]. Fig. 1 shows the circuit structure. The sense amplifier (cross coupled transistors M1-M4) generates output  $y$  and its complement  $y_i$ . Precharge and evaluate is specified by the enable clock signal  $E$  and its complement  $E_i$ . The inputs  $x_i$  are capacitively coupled onto the floating gate  $\phi$  of M5, and the threshold is set by the gate voltage  $t$  of M6. The potential  $\phi$  is given by  $\phi = \sum_{i=1}^n C_i x_i / C_{tot}$ , where  $C_{tot}$  is the sum of all capacitances, including parasitics, at the floating node. Weight values are thus realized by setting capacitors  $C_i$  to appropriate values. These capacitors may be implemented between the polysilicon-1 and polysilicon-2 layers.

The enable signal,  $E$ , controls the precharge and activation of the sense circuit. The gate has two phases of operation, the evaluate phase and the equalize phase. When  $E_i$  is high the output voltages are equalized. When  $E$  is high, the outputs are disconnected and the differential circuit (M5-M7) draws different currents from the formerly equalized nodes  $y$  and  $y_i$ . The sense amplifier is activated after the delay of the enable inverters and amplifies the difference in potential now present between  $y$  and  $y_i$ , accelerating the transition to full swing. In this way the circuit structure determines whether the weighted sum of the inputs,  $\phi$ , is greater or less than the threshold,  $t$ , and a TL gate is realized. For full details, please refer to [2].

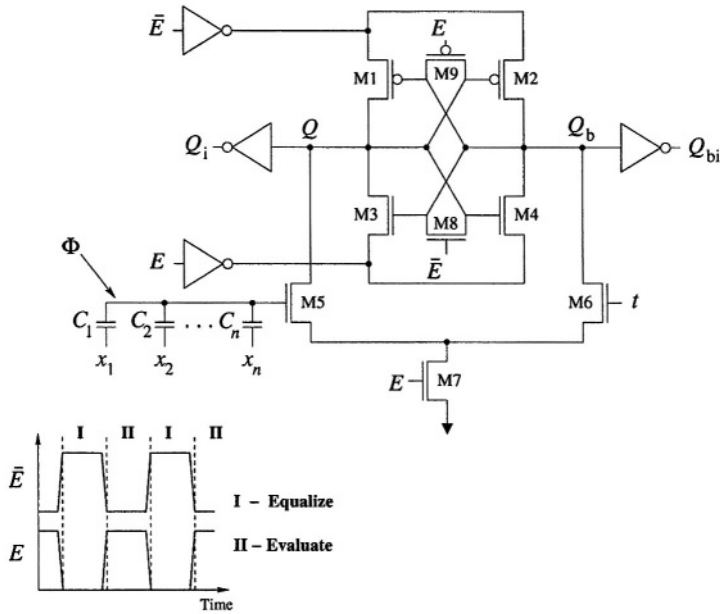


Fig. 1. The CRTL gate circuit and Enable signals.

### 3 Logical Effort

Logical effort (LE) is a design methodology for estimating the delay of CMOS logic circuits [3]. It provides a means to determine the best number of logic stages, including buffers, required to implement a given logic function, and to size the transistors to minimize the delay.

The total delay of a gate,  $d$ , is comprised of two parts, an intrinsic parasitic delay  $p$ , and an effort delay,  $f = gh$ :

$$d = (gh + p)\tau. \tag{2}$$

The delay unit  $\tau$  is the delay of an inverter driving an identical copy of itself, without parasitics. This normalization enables the comparison of delay across different technologies. The parasitic delay is largely independent of the transistor sizes in the gate. The effort delay,  $gh$ , depends on the ratio of the sizes of the transistors in the gate to the load capacitance and the complexity of the gate. The former term is called *electrical effort*,  $h$ , and the latter is called *logical effort*,  $g$ . The logical effort,  $g$ , characterizes the gate complexity, and is defined as the ratio of the input capacitance of the gate to the input capacitance of an inverter that can produce equal output current. By definition an inverter has a logical effort of 1. These considerations may be extended to the treatment of delay through a chain of  $N$  gates, to minimize the total path delay for a given load and input capacitance.

The path effort,  $F$ , is given by the product of the path logical effort, the path branching effort and the path electrical effort,  $F = GBH$ . The path delay,  $D$ , is minimized when each stage in the path bears the same stage effort and the minimum delay is achieved when the stage effort is  $f_{min} = g_i h_i = F^{1/N}$ . This leads to the main result of logical effort, which is the expression for minimum path delay  $D_{min} = NF^{1/N} + \sum p_i$ .

The accuracy of the delay predicted by LE for any gate can be improved by calibrating the model by simulating the delay as a function of load (electrical effort) and fitting a straight line to extract parasitic delay,  $p$ , and the logical effort,  $g$ . We will use this technique to develop a calibrated logical effort based model for the delay of the CRTL gates. For a full understanding of this paper the reader should be familiar with logical effort, for details refer to [3].

## 4 The CRTL Delay Model

We begin by providing a set of assumptions which will simplify the analysis, a proposed expression for the worst case delay of the CRTL gate and a derivation of the model's parameters. The TL gate is assumed to have  $n$  logic inputs (fanin) and  $T$  is the threshold of the gate. The potential of the gate of transistor M6,  $t$ , in Fig. 1 is given by  $t = (T/n) \times V_{dd}$ . In the worst case, the voltage  $\phi$  takes the values  $\phi = t \pm \delta/2$ , where  $\delta$  is given by  $\delta = V_{dd}/n$ .

The worst case (greatest delay) condition occurs when the difference between  $\phi$  and  $t$  is minimal. The value of  $\phi = t - \delta/2$  corresponds to the rising and falling edges of the nodes  $Q$  and  $Q_b$ , respectively, in Fig. 1, and conversely for  $\phi = t + \delta/2$ .

The gate inputs are assumed to have unit weights, ie.  $w_i = 1$ , since the delay depends only on the value of  $\phi$  and  $t$ . Also, without loss of generality, we will assume positive weights and threshold. Since the gate is clocked, we will measure average 50% transition delay from the clock  $E$  to  $Q_i$  and  $Q_{bi}$ . Generally, the delay will depend on the threshold voltage,  $t$ , the step size,  $\delta$ , and the capacitive output load on  $Q_i$  and  $Q_{bi}$ . To simplify the analysis, we will fix the value of  $t$  at 1.5 V. This value is close to the required gate threshold voltage in typical circuit applications. The delay is not strongly dependent on the actual gate threshold, so the subsequent results are valid over a wide range of threshold voltages,  $t$ . Therefore, for modeling purposes, the worst case delay is assumed to depend only on the fan-in and gate loading, and allows us to propose a model based on expressions similar to those for conventional logic based on the theory of logical effort.

The delay of the CRTL gate may be expressed as Equation (3). This delay is the total delay of the sense amplifier and the buffer inverters connected to  $Q$  and  $Q_b$ , and depends on the load,  $h$ , and the fanin,  $n$ , as follows

$$d_{E \rightarrow Q_i} = \{g(n)h + p(n)\}\tau. \quad (3)$$

The load,  $h$ , is defined as the ratio of load capacitance on  $Q_i$  (we assume the loads on  $Q_i$  and  $Q_{bi}$  are equal) and the CRTL gate unit weight capacitance. Both logical effort and parasitic delay in Equation (3) are a function of the fanin.

**Table 1.** Extracted CRTL gate logical effort,  $g$ , parasitic delay,  $p$ , parameters for  $n = 2$  to 60 and  $h = 0$  to 20 for the 0.35  $\mu\text{m}$ , 3.3 V, 4M/2P process at 75°C and the gate delay normalized to FO4 for  $h = 1, 5$  and 10.

$n$	$g$	$p$	$d_{E \rightarrow Qi, h=1}$	$d_{E \rightarrow Qi, h=5}$	$d_{E \rightarrow Qi, h=10}$
2	0.346	2.5	0.55	0.82	1.15
5	0.357	3.3	0.71	0.98	1.33
10	0.365	4.0	0.84	1.13	1.48
15	0.376	4.3	0.90	1.19	1.56
20	0.375	4.7	0.98	1.27	1.63
30	0.400	5.0	1.04	1.35	1.74
40	0.424	5.1	1.07	1.40	1.80
50	0.439	5.2	1.09	1.43	1.85
60	0.460	5.2	1.09	1.45	1.90

The gates used in this work are modeled using an industrial 0.35  $\mu\text{m}$ , 3.3V, 4M/2P process technology at 75°C. The delay unit  $\tau$  is 40 ps,  $p_{inv}$  is 1.18 and simulated FO4 (fan out of four inverter) delay is 204 ps.

The values of  $g$  and  $p$  in Equation (3) were extracted by linear regression from simulation results for a range of fanin from  $n = 2$  to 60 while the electrical effort was swept from  $h = 0$  to 20 as shown in Table 1. The Table also gives the absolute gate delay for three values of electrical effort,  $h = 1, 5$  and 10, where  $h$  is the ratio of the load capacitance to the unit input capacitance.

By fitting a curve to the parameters  $g$  and  $p$ , CRTL gate delay may be approximated in closed form by

$$d_{E \rightarrow Qi} = \{(0.002n + 0.34)h + \ln(n) + 1.6\}\tau. \quad (4)$$

In order to use the parameters in Table 1 and Equation (4), it is necessary to compensate for the parasitic capacitance at the floating gate of M5. The parasitic capacitance,  $C_p$ , contributes to a reduced voltage step,  $\delta$ , on the gate of M5 in Fig. 1 with respect to the threshold voltage,  $t$ . This reduction in  $\delta$  is equivalent to an increased value for the fanin. This effective fanin,  $n_{eff}$ , is given by

$$n_{eff} = \left\{ \frac{\sum_{i=1}^n C_i + C_p}{\sum_{i=1}^n C_i} \right\} n_0, \quad (5)$$

where  $n_0$  is the number of inputs to the gate and  $n_{eff}$  is the value used to calculate the delay. Typically, for a large fanin CRTL gate, by far the major contribution to the parasitic capacitance will be from the bottom plate of the floating capacitors used to implement the weights. In the process used in this work, this corresponds to the poly1 plate capacitance to the underlying n-well used to reduce substrate noise coupling to the floating node. For a 32-input CRTL gate with 3.37 fF poly1-poly2 unit capacitors ( $4\mu\text{m}^2$ ), the extracted layout parasitic capacitance of poly1 to substrate is 29 fF, and the  $\sum_{i=1}^n C_i = 32 \times 3.37 = 108$  fF. From Equation (5) the effective fanin to be used in the delay calculation is  $((108+29)/108) \times 32 \approx 41$ .



## 5 Design Examples and Comparison

In order to illustrate the application of the model presented in the previous Section, the delay of wide AND gates used in ALUs, the 4-bit carry generate function used in adders, and the carry-out of a (7,3) parallel counter, designed using both domino and CRTL are evaluated and compared. In all examples the transistors of the domino circuits (first stage only for multi stage circuits) are sized to present the same input capacitance as the minimum sized inverter (1.8  $\mu\text{m}$  of gate width, minimum length), which is approximately equal to the CRTL unit weight input capacitance, to ensure all designs affect the delay of the driver equally.

### 5.1 Wide AND

As the first example, we consider the design of wide AND gates used for example in ALUs for zero detection. The minimum delay domino trees for 8 to 64 inputs are listed in Table 2, e.g. 4, 4, 2 denotes a 3 layer tree design of the 32-input AND consisting of four 4-input AND gates in the input layer, four 4-input gates in the second layer and a 2-input gate in the third layer. These minimum delay trees were obtained by extracting the logical effort and parasitic delay of 2-, 4- and 8-input domino AND gates from simulations (see Chap. 5 of [3] for details), and finding the tree which minimizes the sum of effort and parasitic delay. The calibrated electrical effort and parasitic delay were used in the domino delay calculation.

Table 2 shows the FO4 delay for domino and CRTL AND gates with fanin from 8 to 64, for path electrical effort  $H=1$  and  $H=10$ , corresponding to the values of  $h=1$  and  $h=10$  in Table 1. Note that increased values of  $n_{eff}$  are used to obtain the correct CRTL delay values from Table 1.

Comparing Tables 1 and 2, the CRTL gate design is on average approximately 1.8 to 2.7 times faster than domino-CMOS for a path electrical effort of 10 and 1, respectively.

**Table 2.** Minimum delay domino-CMOS AND tree designs with fanin  $n=8, 16, 32$  and FO4 delay comparison with CRTL for path electrical effort  $H=1$  and 10

$n$	Domino tree	$H=1$		$H=10$	
		Domino	CRTL	Domino	CRTL
8	4, 2	1.91	0.84	2.34	1.48
16	4, 4	2.33	0.98	2.8	1.63
32	4, 4, 2	3.22	1.07	3.47	1.80
64	4, 4, 4	3.5	1.1	3.93	1.95
Average		2.74	1.0	3.14	1.72

## 5.2 4-Bit Carry Generate

The carry generate signal,  $\alpha$ , of a 4-bit block may be calculated using a single TL gate as follows [4]:

$$\alpha = \text{sgn} \left\{ \sum_{i=0}^3 2^i (a_i + b_i) - 2^4 \right\}. \quad (6)$$

The sum of weights  $N = 30$ , so the worst case delay of this gate will correspond to the delay of a gate of effective fanin,  $n_{eff}$ , of approximately 40. For  $H=10$ , corresponding to a 33.7 fF load capacitance, Table 1 gives the expected delay of 1.8 FO4, or 372 ps. Using Equation (4), the calculated delay is 379 ps.

The domino gate used to compute the same function for comparison is the well known Manchester-carry circuit. The electrical effort and parasitic delay for the slowest input,  $g_{j-3}$ , were extracted from simulation [3] and used to calculate the worst case delay for  $h=1$  and  $h=10$ . The results are shown in Table 3. It should be noted that the domino gate delay numbers exclude the delay of generating the bitwise  $p_i$  and  $g_i$  signals.

Under the conditions of equal input capacitance and load, the CRTL gate is 1.3 to 1.6 times faster. This is a significant delay improvement even in this case of a function with a small number of logic inputs.

**Table 3.** 4-bit carry generate,  $G_j^{j-3}$ , and (7,3) counter  $c_{out}$  FO4 delay comparison with CRTL for path  $H=1$  and 10

Function	$H=1$		$H=10$	
	Domino	CRTL	Domino	CRTL
$G_j^{j-3}$	1.7	1.07	2.42	1.8
(7,3) $c_{out}$	1.5	0.84	1.9	1.48

## 5.3 (7,3) Counter Critical Path

As the final design example, we consider the critical path of a (7,3) parallel counter, commonly used in multipliers. The domino critical path for  $c_{out}$  consists of two full adders. The CRTL implementation computes the majority function using a single gate, where the output is logic 1 if 4 or more inputs are 1. The delay results are shown in Table 3. The CRTL implementation is between 1.3 and 1.8 times faster.

## 6 Conclusions

A logical effort based delay model for CRTL gates was introduced and applied to the evaluation of a number of common datapath circuit elements. It was shown that compared to domino, the CRTL design examples are between 1.3 and 2.7

times faster over a wide range of loads, while presenting a significantly reduced input capacitance to the driver from the previous stage. The design examples used in the comparison were chosen to illustrate the typical performance gains of CRTL over domino which may be expected and this will be strongly application dependent. The important consideration of relative power dissipation is the subject of ongoing work.

## References

1. P. Celinski, S. Cotofana, J. F. López, S. Al-Sarawi, and D. Abbott. State-of-the-art in CMOS threshold logic VLSI gate implementations and applications. In *Proceedings of the VLSI Circuits and Systems Conference*, volume 5117, pages 53–64, Spain, 2003. SPIE.
2. P. Celinski, J. F. López, S. Al-Sarawi, and D. Abbott. Low power, high speed, charge recycling CMOS threshold logic gate. *IEE Electronics Letters*, 37(17):1067–1069, August 2001.
3. I. E. Sutherland, R. F. Sproull, and D. L. Harris. *Logical Effort, Designing Fast CMOS Circuits*. Morgan Kaufmann, 1999.
4. S. Vassiliadis, S. Cotofana, and K. Bertels. 2-1 addition and related arithmetic operations with threshold logic. *IEEE Trans. Computers*, 45(9):1062–1067, September 1996.

# Author Index

- Abbo, Anteneh A. 532  
Abbott, Derek 899  
Acquaviva, Andrea 352  
Al-Hashimi, Bashir M. 198  
Alvandpour, Atila 849  
Amirante, Ettore 413  
Andersson, Daniel A. 463  
Andersson, Stefan 849  
Arnold, Mark G. 208, 675  
Arslan, Tughrul 585  
Atienza, David 510  
Auguin, Michel 603  
Auvergne, D. 100, 110, 722, 838  
Azémar, N. 100, 110, 722, 838
- Barat, Francisco 311  
Bargagli-Stoffi, Agnese 413  
Barke, Erich 453  
Barreto, Raimundo S. 362  
Bastian, Fabricio B. 732  
Belleudy, Cécile 603  
Bellido, Manuel J. 829  
Ben Fradj, Hanene 603  
Bengtsson, Lars 869  
Berthold, Jörg 392, 789  
Bertoletti, Massimo 129  
Bioul, Gery 574  
Bisdounis, L. 332  
Bitterlich, Torsten 90  
Bjerregaard, T. 301  
Blionas, S. 332, 613  
Boemo, Eduardo 574  
Bogliolo, Alessandro 352  
Bona, Andrea 541  
Borodenkov, Alexei 564  
Boselli, Giorgio 138  
Brandolese, C. 238  
Bucci, Marco 481  
Büttner, Finn 322  
Burns, F. 471  
Bystrov, A. 471
- Cai, Yici 433  
Caputa, Peter 849  
Carvalho, Fernando F. 362
- Catthoor, Francky 311, 510  
Celinski, Peter 899  
Chae, Soo-Ik 159  
Chakrabarti, Chaitali 711  
Chan, Cheong-fat 701  
Chang, Hongliang 36  
Chen, Howard 809  
Chormoviti, A. 593  
Choudhary, Vishal 532  
Choy, Chiu-sing 701  
Colmenar, J.M. 623  
Combe, Marylene 381  
Corporaal, Henk 311  
Cotofana, Sorin D. 899  
Cserveny, S. 889
- Daga, Jean-Michel 381  
Dallavalle, Carlo 16  
Dalton, Damian 799  
De Man, Hugo 1  
Deconinck, Geert 311  
Deschamps, Jean-Pierre 574  
Dimitrakopoulos, G. 248  
Do, Minh Q. 869  
Dragone, Nicola 129  
Drosos, C. 332
- Erdogan, Ahmet T. 585
- Farine, Pierre-André 169  
Fischer, Jürgen 413  
Flandre, Denis 189  
Fornaciari, W. 238  
Foster, A. 613  
Frank, Uri 402  
Fredriksson, Henrik 849  
Friedman, Eby G. 750  
Fu, Jingjing 433  
Fukuoka, Kazuki 423  
Furber, Steve 289
- Galanis, M.D. 652  
García-Ortiz, Alberto 819, 859  
Garnica, O. 623  
Gemmeke, Tobias 879

- Georgakos, Georg 392, 789  
 Ginosar, Ran 402  
 Glesner, Manfred 819, 859  
 Goutis, Costas E. 501, 652, 742  
 Grass, Eckhard 258  
 Greenstreet, Mark R. 48  
 Gruber, Dominik 413  
 Guardiani, Carlo 129  
 Guerrero, David 829  
 Guglielmo, Michele 481  
 Guitton-Ouhamou, Patricia 603  
 Guntzel, Jose Luis 732  
 Gustafsson, Oscar 662  
  
 Hamada, Kenji 423  
 Hansson, Martin 849  
 Har, Dong-Soo 691  
 Hassoune, Ilham 189  
 Helms, D. 17  
 Henzler, Stephan 392, 789  
 Hofmann, Richard 643  
 Hong, Xianlong 433, 442  
 Hu, Xiaobo Sharon 553  
 Hu, Xiaodong 442  
  
 Iijima, Masaaki 423  
 Ismail, Mohammed 564  
 Itoh, Kiyoo 3  
  
 Jayapala, Murali 311  
 Jing, Tong 442  
 Johansson, Kenny 662  
 Juan, Jorge 829  
 Julien, Nathalie 342  
  
 Kakarountas, Athanasios P. 501, 742  
 Kalogerakis, P. 248  
 Kanopoulos, Nick 2  
 Karatasos, Dimitris 742  
 Katoch, Atul 179  
 Kavvadias, Nikolaos 593, 633  
 Kawahara, Takayuki 3  
 Kaya, Idris 453  
 Khan, Zahid 585  
 Kinane Andrew 780  
 Kinniment, D.J. 278  
 Kleihorst, Richard P. 532  
 Koch, Peter 322  
 Koelmans, A. 471  
 Kolovos, P. 248  
  
 Kretzschmar, Claudia 90  
 Krstić, Miloš 258  
 Kwon, Soon 159  
  
 Landrault, A. 722  
 Larsson-Edefors, Per 463, 869  
 Lasbouygues, B. 110, 838  
 Lattanzi, Emanuele 352  
 Laurent, Johann 342  
 Lazzari, Cristiano 732  
 Legat, Jean-Didier 189  
 Leung, Lap-Fai 553  
 Liberali, Valentino 138  
 Liu, Yijun 289  
 López, S. 623  
 Luo, Zuying 433  
 Luzzi, Raimondo 481  
  
 Macii, Enrico 58  
 Mader, Dominik 228  
 Mahadevan, S. 301  
 Maili, Alexander 799  
 Mamagkakis, Stylianos 510  
 Manzak, Ali 711  
 Marlow, Sean 780  
 Martin, Eric 342  
 Martins Maciel, Paulo R. 362  
 Masgonty, J.-M. 889  
 Masselos, K. 613  
 Maurine, P. 100, 110, 722, 838  
 Meijer, Maurice 179  
 Meijer, R.I.M.P. 372  
 Mendias, J.M. 510  
 Merrett, Geoff 198  
 Metafas, D. 332  
 Michel, X. 100  
 Mignolet, J-Y. 613  
 Millán, Alejandro 829  
 Mohsen, Amjad 643  
 Mooney III, Vincent J. 148  
 Muehlberger, Andreas 491  
 Müller, Dietmar 90  
 Müller, Matthias 228  
 Muresan, Valentin 780  
 Murgan, Tudor 819, 859  
 Murphy, Noel 780  
  
 Na, J.W. 770  
 Nagel, Jean-Luc 169  
 Nebel, W. 17

- Neffe, Ulrich 491  
 Neve, Amaury 189  
 Nieuwland, André K. 179  
 Nikolaidis, Spiridon 593, 633  
 Nikolaidis, Spiros 501, 613  
 Nikolos, D. 248  
 Noll, Tobias G. 879  
 Numa, Masahiro 423
- O'Connor, Noel 780  
 Oh, Myeong-Hoon 691  
 Olbrich, Markus 453  
 Oliveira Júnior, Meuse N. 362  
 Olivieri, Mauro 119  
 Olsen, Anders Brødløs 322  
 Osada, Kenichi 3  
 Ostapko, Daniel 809  
 Ostúa, Enrique 829
- Palermo, Gianluca 521  
 Paliouras, V. 760  
 Pan, Zhu 433  
 Papaix, Caroline 381  
 Park, Jun Cheol 148  
 Park, Young-Kil 159  
 Pessolano, F. 372  
 Petrov, M. 819  
 Pfeiffenberger, Philipp 148  
 Pfleiderer, Hans-Jörg 69  
 Piguet, Christian 169, 889  
 Poncino, Massimo 58  
 Pun, Kong-pong 701
- Qian, Haifeng 36  
 Quarantelli, Michele 129
- Racape, Emmanuel 381  
 Reis, Ricardo 732  
 Rémond, Yann 268  
 Ren, Jihong 48  
 Renaudin, Marc 268  
 Rieger, Edgar 491  
 Robert, M. 722  
 Rothbart, Klaus 491  
 Ruan, Jie 208  
 Ruiz-de-Clavijo, Paulino 829  
 Rusu, Ana 564
- Salerno, Sabino 58  
 Salewski, Silke 453
- Salice, F. 238  
 Sapatnekar, Sachin S. 36  
 Scarana, Mirko 119  
 Schlachta, C. 819  
 Schmidt, E. 17  
 Schmitt-Landsiedel, Doris 392, 413, 789  
 Schuster, Christian 169  
 Scotti, Giuseppe 119  
 Senn, Eric 342  
 Sevat, Leo 532  
 Shang, D. 471  
 Sialelli, Vincent 381  
 Sicard, Gilles 268  
 Silvano, Cristina 521  
 Simon, Sven 228  
 Slimani, Kamel 268  
 Sokolov, D. 471  
 Son, Y.S. 770  
 Song, Eunseok 159  
 Soudris, D. 510, 613, 652  
 Sparsø, J. 301  
 Spiliotopoulos, Vassilis 501  
 Stan, Mircea R. 79  
 Steger, Christian 491, 799  
 Sutter, Gustavo 574  
 Svensson, Christer 849  
 Svensson, Lars "J" 463
- Tada, Akira 423  
 Tahedl, Markus 69  
 Tan, Sheldon X.-D. 433  
 Tatsaki, A. 332  
 Teichmann, Philip 413  
 Tenhunen, Hannu 564  
 Theodoridis, George 652, 742  
 Tragoudas, S. 652  
 Trifiletti, Alessandro 119, 481  
 Trucco, Gabriella 138  
 Tsui, Chi-Ying 553
- Valencia, Leonardo 685  
 Vander Aa, Tom 311  
 Vassiliadis, N. 593  
 Velenis, Dimitrios 750  
 Verle, A. 100  
 Vouzis, P. 760
- Wang, Yin 442  
 Wanhammar, Lars 662  
 Wehn, Norbert 218

Weiss, Reinhold 491

Wellig, Armin 218

Wilson, R. 110, 838

Wortmann, Andreas 228

Yakovlev, A.V. 278, 471

Yan, Guiying 442

Yang, Jing-ling 701

Yang, Yang 442

Zaccaria, Vittorio 541

Zafalon, Roberto 541

Zhang, Yan 79

Zimmer, H. 819

Zory, Julien 218

# Lecture Notes in Computer Science

For information about Vols. 1–3124

please contact your bookseller or Springer

- Vol. 3263: M. Weske, P. Liggesmeyer (Eds.), *Object-Oriented and Internet-Based Technologies*. XII, 239 pages. 2004.
- Vol. 3260: I. Niemegeers, S.H. de Groot (Eds.), *Personal Wireless Communications*. XIV, 478 pages. 2004.
- Vol. 3255: A. Benzúr, J. Demetrovics, G. Gottlob (Eds.), *Advances in Databases and Information Systems*. XI, 423 pages. 2004.
- Vol. 3254: E. Macii, V. Paliouras, O. Koufopavlou (Eds.), *Integrated Circuit and System Design*. XVI, 910 pages. 2004.
- Vol. 3253: Y. Lakhnech, S. Yovine (Eds.), *Formal Techniques in Timed, Real-Time, and Fault-Tolerant Systems*. X, 397 pages. 2004.
- Vol. 3249: B. Buchberger, J.A. Campbell (Eds.), *Artificial Intelligence and Symbolic Computation*. X, 285 pages. 2004. (Subseries LNAI).
- Vol. 3246: A. Apostolico, M. Melucci (Eds.), *String Processing and Information Retrieval*. XIV, 316 pages. 2004.
- Vol. 3241: D. Kranzlmüller, P. Kacsuk, J.J. Dongarra (Eds.), *Recent Advances in Parallel Virtual Machine and Message Passing Interface*. XIII, 452 pages. 2004.
- Vol. 3240: I.Jonassen, J. Kim (Eds.), *Algorithms in Bioinformatics*. IX, 476 pages. 2004. (Subseries LNBI).
- Vol. 3239: G. Nicosia, V. Cutello, P.J. Bentley, J. Timmis (Eds.), *Artificial Immune Systems*. XII, 444 pages. 2004.
- Vol. 3238: S. Biundo, T. Frühwirth, G. Palm (Eds.), *KI 2004: Advances in Artificial Intelligence*. XI, 467 pages. 2004. (Subseries LNAI).
- Vol. 3232: R. Heery, L. Lyon (Eds.), *Research and Advanced Technology for Digital Libraries*. XV, 528 pages. 2004.
- Vol. 3224: E. Jonsson, A. Valdes, M. Almgren (Eds.), *Recent Advances in Intrusion Detection*. XII, 315 pages. 2004.
- Vol. 3223: K. Slind, A. Bunker, G. Gopalakrishnan (Eds.), *Theorem Proving in Higher Order Logics*. VIII, 337 pages. 2004.
- Vol. 3221: S. Albers, T. Radzik (Eds.), *Algorithms – ESA 2004*. XVIII, 836 pages. 2004.
- Vol. 3220: J.C. Lester, R.M. Vicari, F. Paraguaçu (Eds.), *Intelligent Tutoring Systems*. XXI, 920 pages. 2004.
- Vol. 3217: C. Barillot, D.R. Haynor, P. Hellier (Eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004*. XXXVIII, 1114 pages. 2004.
- Vol. 3216: C. Barillot, D.R. Haynor, P. Hellier (Eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004*. XXXVIII, 930 pages. 2004.
- Vol. 3210: J. Marcinkowski, A. Tarlecki (Eds.), *Computer Science Logic*. XI, 520 pages. 2004.
- Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), *Principles and Practice of Semantic Web Reasoning*. VII, 165 pages. 2004.
- Vol. 3207: L.T. Yang, M. Guo, G.R. Gao, N.K. Jha (Eds.), *Embedded and Ubiquitous Computing*. XX, 1116 pages. 2004.
- Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), *Text, Speech and Dialogue*. XIII, 667 pages. 2004. (Subseries LNAI).
- Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), *UbiComp 2004: Ubiquitous Computing*. XVI, 452 pages. 2004.
- Vol. 3203: J. Becker, M. Platzner, S. Vernalde (Eds.), *Field Programmable Logic and Application*. XXX, 1198 pages. 2004.
- Vol. 3202: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), *Knowledge Discovery in Databases: PKDD 2004*. XIX, 560 pages. 2004. (Subseries LNAI).
- Vol. 3201: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), *Machine Learning: ECML 2004*. XVIII, 580 pages. 2004. (Subseries LNAI).
- Vol. 3199: H. Schepers (Ed.), *Software and Compilers for Embedded Systems*. X, 259 pages. 2004.
- Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), *Groupware: Design, Implementation and Use*. XI, 378 pages. 2004.
- Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), *Inductive Logic Programming*. XI, 361 pages. 2004. (Subseries LNAI).
- Vol. 3193: P. Samarati, P. Ryan, D. Gollmann, R. Molva (Eds.), *Computer Security – ESORICS 2004*. X, 457 pages. 2004.
- Vol. 3192: C. Bussler, D. Fensel (Eds.), *Artificial Intelligence: Methodology, Systems, and Applications*. XIII, 522 pages. 2004. (Subseries LNAI).
- Vol. 3190: Y. Luo (Ed.), *Cooperative Design, Visualization, and Engineering*. IX, 248 pages. 2004.
- Vol. 3189: P.-C. Yew, J. Xue (Eds.), *Advances in Computer Systems Architecture*. XVII, 598 pages. 2004.
- Vol. 3186: Z. Bellahsene, T. Milo, M. Rys, D. Suciu, R. Unland (Eds.), *Database and XML Technologies*. X, 235 pages. 2004.
- Vol. 3185: M. Bernardo, F. Corradini (Eds.), *Formal Methods for the Design of Real-Time Systems*. VII, 295 pages. 2004.
- Vol. 3184: S. Katsikas, J. Lopez, G. Pernul (Eds.), *Trust and Privacy in Digital Business*. XI, 299 pages. 2004.
- Vol. 3183: R. Traunmüller (Ed.), *Electronic Government*. XIX, 583 pages. 2004.
- Vol. 3182: K. Bauknecht, M. Bichler, B. Pröll (Eds.), *E-Commerce and Web Technologies*. XI, 370 pages. 2004.



- Vol. 3181: Y. Kambayashi, M. Mohania, W. Wöß (Eds.), *Data Warehousing and Knowledge Discovery*. XIV, 412 pages. 2004.
- Vol. 3180: F. Galindo, M. Takizawa, R. Traummüller (Eds.), *Database and Expert Systems Applications*. XXI, 972 pages. 2004.
- Vol. 3179: F.J. Perales, B.A. Draper (Eds.), *Articulated Motion and Deformable Objects*. XI, 270 pages. 2004.
- Vol. 3178: W. Jonker, M. Petkovic (Eds.), *Secure Data Management*. VIII, 219 pages. 2004.
- Vol. 3177: Z.R. Yang, H. Yin, R. Everson (Eds.), *Intelligent Data Engineering and Automated Learning – IDEAL 2004*. XVIII, 852 pages. 2004.
- Vol. 3176: O. Bousquet, U. von Luxburg, G. Rätsch (Eds.), *Advanced Lectures on Machine Learning*. IX, 241 pages. 2004. (Subseries LNAI).
- Vol. 3175: C.E. Rasmussen, H.H. Bühlhoff, B. Schölkopf, M.A. Giese (Eds.), *Pattern Recognition*. XVIII, 581 pages. 2004.
- Vol. 3174: F. Yin, J. Wang, C. Guo (Eds.), *Advances in Neural Networks - ISNN 2004*. XXXV, 1021 pages. 2004.
- Vol. 3173: F. Yin, J. Wang, C. Guo (Eds.), *Advances in Neural Networks-ISNN 2004*. XXXV, 1041 pages. 2004.
- Vol. 3172: M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, T. Stützle (Eds.), *Ant Colony, Optimization and Swarm Intelligence*. XII, 434 pages. 2004.
- Vol. 3170: P. Gardner, N. Yoshida (Eds.), *CONCUR 2004 - Concurrency Theory*. XIII, 529 pages. 2004.
- Vol. 3166: M. Rauterberg (Ed.), *Entertainment Computing – ICEC 2004*. XXIII, 617 pages. 2004.
- Vol. 3163: S. Marinai, A. Dengel (Eds.), *Document Analysis Systems VI*. XI, 564 pages. 2004.
- Vol. 3162: R. Downey, M. Fellows, F. Dehne (Eds.), *Parameterized and Exact Computation*. X, 293 pages. 2004.
- Vol. 3160: S. Brewster, M. Dunlop (Eds.), *Mobile Human-Computer Interaction – MobileHCI 2004*. XVII, 541 pages. 2004.
- Vol. 3159: U. Visser, *Intelligent Information Integration for the Semantic Web*. XIV, 150 pages. 2004. (Subseries LNAI).
- Vol. 3158: I. Nikolaidis, M. Barbeau, E. Kranakis (Eds.), *Ad-Hoc, Mobile, and Wireless Networks*. IX, 344 pages. 2004.
- Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), *PRICAI 2004: Trends in Artificial Intelligence*. XX, 1023 pages. 2004. (Subseries LNAI).
- Vol. 3156: M. Joye, J.-J. Quisquater (Eds.), *Cryptographic Hardware and Embedded Systems - CHES 2004*. XIII, 455 pages. 2004.
- Vol. 3155: P. Funk, PA. González Calero (Eds.), *Advances in Case-Based Reasoning*. XIII, 822 pages. 2004. (Subseries LNAI).
- Vol. 3154: R.L. Nord (Ed.), *Software Product Lines*. XIV, 334 pages. 2004.
- Vol. 3153: J. Fiala, V. Koubek, J. Kratochvíl (Eds.), *Mathematical Foundations of Computer Science 2004*. XIV, 902 pages. 2004.
- Vol. 3152: M. Franklin (Ed.), *Advances in Cryptology – CRYPTO 2004*. XI, 579 pages. 2004.
- Vol. 3150: G.-Z. Yang, T. Jiang (Eds.), *Medical Imaging and Augmented Reality*. XII, 378 pages. 2004.
- Vol. 3149: M. Danelutto, M. Vanneschi, D. Laforenza (Eds.), *Euro-Par 2004 Parallel Processing*. XXXIV, 1081 pages. 2004.
- Vol. 3148: R. Giacobazzi (Ed.), *Static Analysis*. XI, 393 pages. 2004.
- Vol. 3146: P. Erdi, A. Esposito, M. Marinaro, S. Scarpetta (Eds.), *Computational Neuroscience: Cortical Dynamics*. XI, 161 pages. 2004.
- Vol. 3144: M. Papatriantafilou, P. Hunel (Eds.), *Principles of Distributed Systems*. XI, 246 pages. 2004.
- Vol. 3143: W. Liu, Y. Shi, Q. Li (Eds.), *Advances in Web-Based Learning – ICWL 2004*. XIV, 459 pages. 2004.
- Vol. 3142: J. Diaz, J. Karhumäki, A. Lepistö, D. Sannella (Eds.), *Automata, Languages and Programming*. XIX, 1253 pages. 2004.
- Vol. 3140: N. Koch, P. Fraternali, M. Wirsing (Eds.), *Web Engineering*. XXI, 623 pages. 2004.
- Vol. 3139: F. Iida, R. Pfeiffer, L. Steels, Y. Kuniyoshi (Eds.), *Embodied Artificial Intelligence*. IX, 331 pages. 2004. (Subseries LNAI).
- Vol. 3138: A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, D.d. Ridder (Eds.), *Structural, Syntactic, and Statistical Pattern Recognition*. XXII, 1168 pages. 2004.
- Vol. 3137: P. De Bra, W. Nejdl (Eds.), *Adaptive Hypermedia and Adaptive Web-Based Systems*. XIV, 442 pages. 2004.
- Vol. 3136: F. Meziane, E. Métails (Eds.), *Natural Language Processing and Information Systems*. XII, 436 pages. 2004.
- Vol. 3134: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), *Extreme Programming and Agile Methods - XP/Agile Universe 2004*. XIV, 233 pages. 2004.
- Vol. 3133: A.D. Pimentel, S. Vassiliadis (Eds.), *Computer Systems: Architectures, Modeling, and Simulation*. XIII, 562 pages. 2004.
- Vol. 3132: B. Demoen, V. Lifschitz (Eds.), *Logic Programming*. XII, 480 pages. 2004.
- Vol. 3131: V. Torra, Y. Narukawa (Eds.), *Modeling Decisions for Artificial Intelligence*. XI, 327 pages. 2004. (Subseries LNAI).
- Vol. 3130: A. Syropoulos, K. Berry, Y. Haralambous, B. Hughes, S. Peter, J. Plaice (Eds.), *TeX, XML, and Digital Typography*. VIII, 265 pages. 2004.
- Vol. 3129: Q. Li, G. Wang, L. Feng (Eds.), *Advances in Web-Age Information Management*. XVII, 753 pages. 2004.
- Vol. 3128: D. Asonov (Ed.), *Querying Databases Privately*. IX, 115 pages. 2004.
- Vol. 3127: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.), *Conceptual Structures at Work*. XI, 403 pages. 2004. (Subseries LNAI).
- Vol. 3126: P. Dini, P. Lorenz, J.N.d. Souza (Eds.), *Service Assurance with Partial and Intermittent Resources*. XI, 312 pages. 2004.
- Vol. 3125: D. Kozen (Ed.), *Mathematics of Program Construction*. X, 401 pages. 2004.