Introduction to Advanced System-on-Chip Test Design and Optimization

Erik Larsson

Springer

INTRODUCTION TO ADVANCED SYSTEM-ON-CHIP TEST DESIGN AND OPTIMIZATION

INTRODUCTION TO ADVANCED SYSTEM-ON-CHIP TEST DESIGN AND OPTIMIZATION

by

ERIK LARSSON
Linköpings University, Sweden

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN-10 1-4020-3207-2 (HB) Springer Dordrecht, Berlin, Heidelberg, New York
ISBN-10 0-387-25624-5 (e-book) Springer Dordrecht, Berlin, Heidelberg, New York
ISBN-13 978-1-4020-3207-3 (HB) Springer Dordrecht, Berlin, Heidelberg, New York
ISBN-13 978-0-387-25624-5 (e-book) Springer Dordrecht, Berlin, Heidelberg, New York

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

Printed on acid-free paper

Contents

Preface

Advances in both semiconductor processing technologies as well as integrated circuit (IC) design techniques and tools have enabled the creation of micro-electronic products that contain a huge amount of complex functionality on a single die. This allows the integration into a single IC of the components that traditionally resided on one or multiple printed circuit boards. As the latter traditional product was commonly referred to as "system", a modern IC that contains the same functionality goes by the name "System-On-Chip" or SOC. The benefits of this ongoing integration are obvious: (1) a smaller form factor, as we want small products that fit our pockets, (2) higher performance, as our needs to do, hear, see more with micro-electronics seem insatiable, and (3) lower power, as these portable, mobile products work of batteries with a limited energy budget.

Designing SOCs is a challenge. The functional and layout designs need to be modular and hierarchical, as flat design is no longer an option for such "monster chips". Increasingly, also the on-chip interconnect becomes hierarchical; we refer to this as Network-On-Chip or NOC. In order to reduce time-to-market and leverage external expertise, we see large modules being imported and reused. The latter design practice has divided the IC design community into 'core providers' and 'SOC integrators'.

Testing SOCs brings forward new challenges as well. The only viable way to contain the growing complexity of SOCs is to apply a modular test approach. Modular testing is required (1) for heterogeneous SOCs, which contain non-logic modules, such as embedded memories, analog and RF modules, e-FPGAs, etc., and (2) for black-boxed third-party cores, for which the test is developed by the core provider, but applied by the SOC integrator. However, modular testing also has precious benefits in terms of (3) reduced test pattern generation efforts due to "divide-n-conquer", and (4) test reuse over multiple generations of SOCs. The research challenges related to a modular test approach are the following.

- Design and test development is distributed over multiple parties, companies, geographical locations, and time. This brings with it challenges with respect to the transfer of "test knowledge" from core provider to SOC integrator.
- Cores and other modules are typically deeply embedded within the SOC, without direct access to SOC pins or other test resources. Hence, we need to add an on-chip test access infrastructure, that enables proper testing of the SOC, but that is as much as possible transparent when not in test mode.
- The fact that there is no longer one monolithic chip test, but many smaller tests instead, brings with it many optimisation issues with respect to test coverage, test application time, power dissipation during test, silicon area used by the on-chip infrastructure, etc. Designers and test engineers need support to make the right choices in this complex matter, where everything seems to be related with everything.

This book by my fellow researcher Erik Larsson tries to shed light on the many issues that come to play in the arena of modular SOC testing. It was written from the need to have a book that could serve as text book for university students. On the one hand, several text books are available on the fundamentals of classical test approaches. On the other hand, available books on modular SOC testing have too much a research nature and too little of introductory material to serve the purpose of student text book. And indeed, Larsson's book exactly fits the void in between these two categories.

This book consists of three parts. In Part 1, the book starts out with a brief overview of classical test approaches. This provides the type of introduction someone new in the field will need, with many references to the well-known text books for further, in-depth coverage. Part 2 describes the challenges and problems specific to testing large, modular, and heterogeneous SOCs. The experienced reader is suggested to right away jump to this part of the book. Subsequently, Part 3 provides details of the research work Larsson and his team have done to address these research challenges. The central key word in all these solutions is "integration"; integration of design and test, and integration of solutions to tackle the many multi-faced optimisation problems at hand. The entire book is written in the very clear style that also marks Larsson's research papers and presentations.

I strongly believe that a modular and hence scalable approach is the only feasible way forward to address the manufacturing test challenges of ICs that are every-increasing in size and complexity. Hence, I do recommend this book; to researchers, to test methodology developers at EDA companies, to design and test engineers, and of course to university students.

ERIK JAN MARINISSEN
Principal Scientist
Philips Research Laboratories
Eindhoven, The Netherlands

Acknowledgements

This book would not have been written without the help of many. I would like to thank Zebo Peng, Petru Eles, and the rest of the group at Embedded Systems Laboratory at the Department of Computer and Information Science at Linköpings Universitet, Sweden. I would like to thank Julien Pouget, Stina Edbom, Klas Arvidsson, and Per Beijer for their contributions to the book. And I would like to thank Krishendu Chakrabarty, Erik Jan Marinissen, and Vishwani D. Agrawal for comments on the draft. I would like to thank Hideo Fujiwara for the time in Japan, Mark de Jongh for inspiration and Cindy Zitter for encouragements, and finally friends and familly.

ERIK LARSSON
Linköpings Universitet
Linköping, Sweden

PART 1
TESTING CONCEPTS

Chapter 1

Introduction

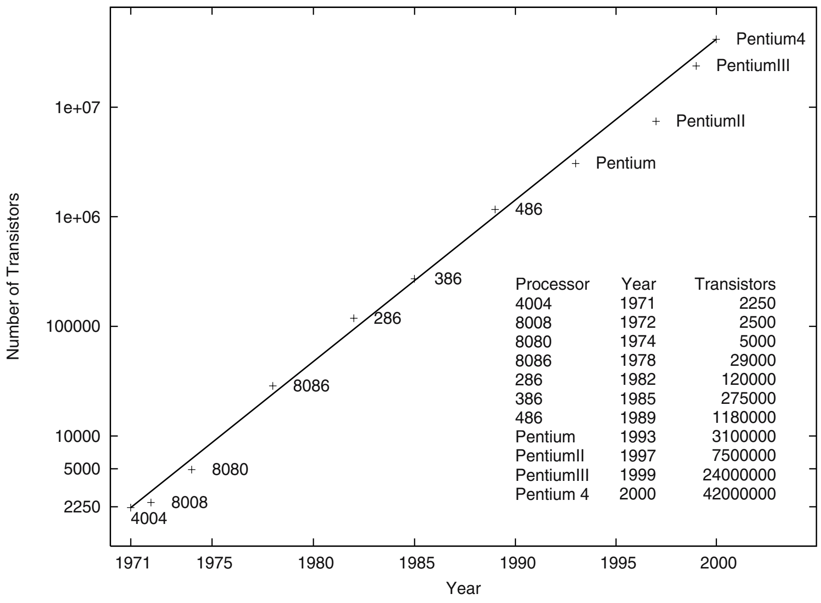
The aim of this book is to discuss production test, including related problems, their modeling, and the design and optimization of System-on-Chip (SOC) test solutions. The emphasis is on test scheduling, how to organize the testing, which is becoming important since the amount of test data is increasing due to more complex systems to test, and the presence of new fault types because of device size miniaturization. The focus of the discussion in the book is on the system perspective since the increasing complexity of SOC designs makes it harder for test designers to grasp the impact of each design decisions on the system's test solution, and also because locally optimized test solution for each testable unit do rarely lead to globally optimized solutions.

An important aspect when taking the system perspective is the computational cost versus modeling and optimization granularity. A fine grain model taking a high number of details into account is obviously to be preferred compared to a model that is considering only a few details. However, a fine grain model leads to high computational cost, and the computational cost is important to keep under control since the SOC designs are increasing in size, and also since the nature of the design process of test solutions is an iterative process. Furthermore, as the number of design parameters increases, which makes the design space large, it is of most import to decide on which problems to spend the computational effort.

The number of transistors per chip has increased enormously over the years. Moore predicted that the number of transistors per chip should roughly double every eighteen months [198]. In Figure 1, the prediction (Moore's law) versus the actual number of transistor per processor is plotted for some Intel processors. The prediction has been amazingly precise.

Chip design is a challenge driven by the increasing number of transistors, device size miniaturization, and shorter development times. An increasing number of transistors can be handled by modeling the system at higher abstraction levels; however, device size miniaturization requires early knowledge of the design at transistor-level. The design times can be shortened by reusing previous designs, or parts (modules or cores) of previous designs. A core-based design methodology is an alternative since it allows pre-defined and pre-verified blocks of logic, so called cores, to be combined with glue logic to become the designed system. The cores can be reused from previous designs or bought from core vendors.

The SOC integrator is the person that designs the system. The design work includes decision on which cores to be used in the design and how the cores



Processor	Year	Transistors
4004	1971	2250
8008	1972	2500
8080	1974	5000
8086	1978	29000
286	1982	120000
386	1985	275000
486	1989	1180000
Pentium	1993	3100000
PentiumII	1997	7500000
PentiumIII	1999	24000000
Pentium 4	2000	42000000

*Figure 1.*The number of transistors for some Intel processors (points) and the Moore's law prediction (straight line) [107, 198].

should be combined to a system. Each of the cores in the design may origin from previous designs within the company, or can be bought from some other companies (core vendors), or can be completely newly designed. The SOC test integrator is the person that designs the test solution for the system. Figure 2(b) shows a simplified SOC design flow. In Figure 2(a) the work by the SOC integrator and the SOC test integrator is organized in a sequence. The SOC test integrator starts working on the test solution when the SOC integrator has fixed and finished the design. An alternative to a sequential design flow is shown in Figure 2(c) where the SOC test integrator takes part as early as in the SOC design. The advantage of the latter approach is that the SOC test integrator can have an impact on design decisions; the selections of cores. In such a design flow, it becomes obvious that the SOC test integrator needs support, *i.e.* tools, for the evaluation of different design decisions. For instance, if two different processor cores can solve the same problem, but the two processors have different test characteristics, which one should the SOC test integrator select?

A motivation for considering the test cost as early as in the design phase is that the test cost is becoming a bottleneck. In Figure 3, the expected cost-per-transistor and the expected test-cost-per-transistor are plotted [106]. The number of transistor per chip increases, but interestingly, the cost-per-transistor is

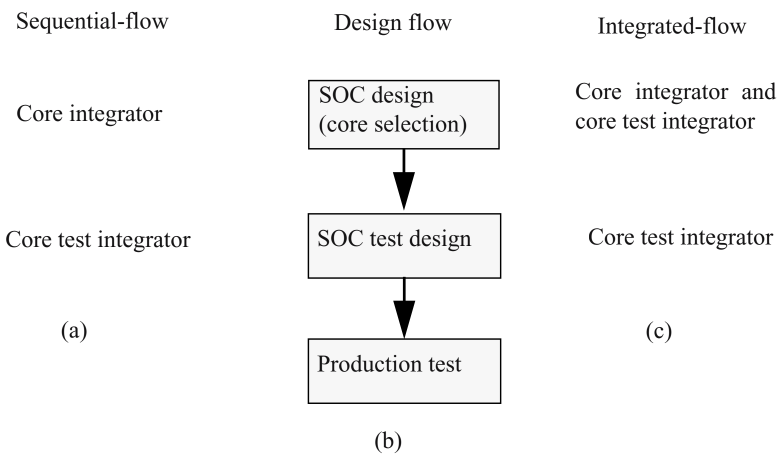


Figure 2. The system-on-chip design flow, the involved persons in (a) a sequential design flow and (b) in an integrated design flow.

decreasing. Interesting is also that the test-cost-per-transistor keeps constant, however, the trend is clear from Figure 3; the relative test cost per designed transistor is increasing.

Among expected challenges are that device size miniaturization and increasing performance lead to new fault types that require new fault models. New models will further increase the amount of test data for an SOC design, and hence the testing times are becoming longer, making test scheduling even more important. Moore [198] pointed out that the transistor count will increase drastically, but also that the power consumption-per-transistor will decrease. It is likely that there will be a time when the transistors can be designed, but the power to make them work will enforce limitations. For designers who want to implement an increasing functionality in the system this might be frustrating. However, the test design community can take advantage of this fact. It means that the cost of additional DFT logic is reduced. Logic can be introduced as logic only used for test purpose. And, logic redundancy can also be used, for instance, to improve yield. The power dissipation during testing is another important problem. In order to detect as many faults as possible per test vector, it is desirable to activate as many fault locations as possible. High activity in the system leads to higher power consumption. The system's power budget can be exceeded, the allowed power consumption for each power grid can be exceeded, and also a testable unit can during testing consume such amount of power that the system is damaged.

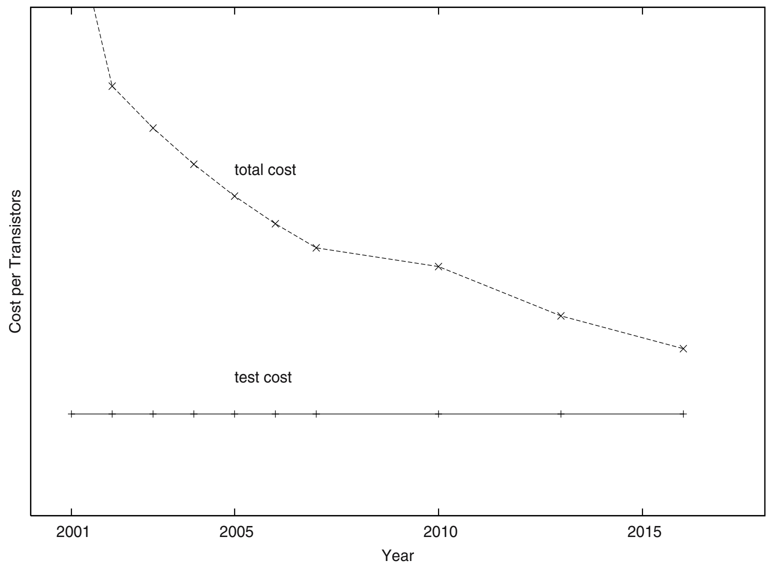


*Figure 3.*The predicted cost per transistor versus the predicted test cost per transistor [106].

This book is dived into three parts. Part one contains the following chapters, an introduction of the topic (Chapter 1), the design flow (Chapter 2) and the test problems (Chapter 3), system modeling (Chapter 5), test conflicts (Chapter 6), and test power consumption (Chapter 7). Part two contains the two chapters; Chapter 8 were TAM design approaches are described and Chapter 9 where test scheduling is discussed. The third part discusses applications in detail. The part is a compilation of a set of papers covering test scheduling using reconfigurable core wrappers, test scheduling and TAM design, core selection integrated in the test solution design flow, and defect-oriented test scheduling.

Chapter 2

Design Flow

1 INTRODUCTION

The aim with this chapter is to give a brief introduction to design flow in order to motivate the discussion on modeling granularity in the following chapters.

The pace in the technology development leads to an extreme increase of the number of transistor in a system (See *Introduction* on page 1). However, there is a gap between what is possible to design according to what the technology development allows to be designed and what is possible to design in terms of cost (time, money, manpower). The so called productivity gap, which is increaseing, is the difference between what is possible to design using available technologies and what is reasonable to accomplish in reasonable design time. The productivity is defined as [120]:

$$P = \frac{\text{total gate count}}{\text{development time} \times \text{number of designers}}$$

It means that if productivity keeps stable, the design time would increase drastically (keeping the design team size constant). An alternative is to increase the size of the design teams in order to keep design time constant. Neither of the approaches are viable. Instead, the evolution in EDA (Electronic Design Automation) will provide productivity improvements. That means that better modeling and optimization techniques have to be developed.

In order for the designers to handle the complexity of the designs due to the technology development, the system can initially be modeled at a high abstraction level. Semi-automatic synthesis and optimization steps (Computer-Aided Design (CAD) tools guided by designers) on the high-level specification transforms the design from the abstract specification to the final circuit (see Figure 4). System modeling at an high abstraction level, means that less implementation specific details are included compared to a system model at lower abstraction levels. Gajski *et al.* [64] view the design flow using a Y-chart where the three direction behavior, structure, and geometry.

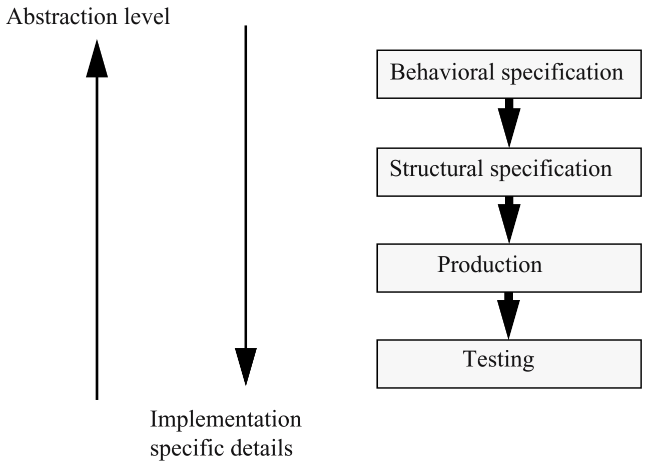


*Figure 4.*Design flow.

2 HIGH-LEVEL DESIGN

The advantage of a top-down design flow, specifying the design a high abstraction level with less implementation specific details, and through synthesis move to a model with a high degree of implementation specific details, is that design exploration, where design alternatives easily can be explored, is eased. A model at high abstraction level includes fewer details and therefore the handling of the design becomes easier, and more optimization steps can be allowed, which means a more thorough search in the design space.

In the design flow, the system is modeled. The purpose of modeling is to understand the function of the system as well as adding implementation specific details. A number of techniques have been developed. On initiative by VHSIC (Very High Speed Integrated Circuits) programme the description language for digital circuits called VHDL (VHSIC Hardware Description Language) was developed [120]. VHDL became an IEEE standard in 1987, and the latest extension took place in 1993 [268]. Graphical descriptions are useful for humans to more easily understand the design. Examples of graphical descriptions are Block Diagrams, Truth Tables, Flow Charts, State Diagrams (Mealy and Moore state machine), and State Transition Diagrams. Netlists are used to enumerate all devices and their connections. The SPICE (Simulation Program with Integrated Circuit Emphasis) is an example of a device oriented structure [265]. The EDIF (Electronic Design Interchange Format) was developed to ease transport of designs between different CAD systems. And, for capturing delay formats in the design, the SDF (Standard Delay Format) was developed [252].

3 CORE-BASED DESIGN

There are a number of disadvantages with a top-down design flow. One obvious is that at high abstraction level implementation details are less visible. It makes discussion on certain detailed design aspects impossible at high abstraction level. The problem with high abstraction levels where less information is captured has become a problem in sub-micron technology where technology development has lead to high device size miniaturization that requires knowledge of transistor-level details. Another fundamental problem is that in a top-down design flow, there is a basic assumption that the system is designed for the first time. In reality, very few systems are designed from scratch. In most cases, there exists a previous system that is to be updated. It means that small parts or even large part of the previous system can be reused for the new system. Also, other previously designed blocks that have been used in a completely different system can be included in the new system. It can also be so that blocks of logic, cores, are bought from other companies. For instance, CPU cores can be bought from different companies and used in the design.

In the core-based design environment, blocks of logic, so called cores, are integrated to a system [84]. The cores may origin from a previous design developed at the company, it may be bought from another company, or it can be a newly designed core. The *core integrator* is the person that selects which cores to use in a design, and the *core test integrator* is the person that makes a design testable - designs the *test solution*. The core integrator selects core from different *core providers*. A core provider can be a company, a designer involved in a previous design, or a designer developing a new core for the system.

A core-based design flow is typically a sequence that starts with core selection, followed by test solution design, and after production, the system is tested (Figure 5(a)). In the core selection stage, the core integrator selects appropriate cores to implement the intended functionality of the system. For each function there are a number of possible cores that can be selected, where each candidate core has its specification on, for instance, performance, power consumption, area, and test characteristics. The core integrator explores the design space (search and combines cores) in order to optimize the SOC. Once the system is fixed (cores are selected) the core test integrator designs the TAM and schedules the tests based on the test specification for each core. In such a design flow (illustrated in Figure 5(a)), the test solution design is a consecutive step to core selection. In such a flow, even if each core's test solution is highly optimized, when integrated as a system, the system's global test solution is most likely not highly optimized.

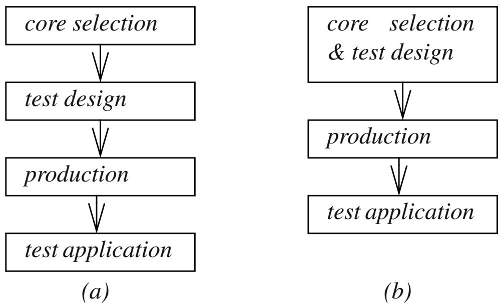


*Figure 5.*Design flow in a core-based design environment (a) traditional and (b) proposed.

The design flow in Figure 5(b), on the other hand, integrates the core selection step with the test solution design step, making it possible to consider the impact of core selection when designing the test solution. In such a design flow (Figure 1(b)), the global system impact on core selection is considered, and the advantage is that it is possible to develop a more optimized test solution.

The design flow in Figure 1(b) can be viewed as in Figure 6, where the core type is floor-planned in the system but there is not yet a design decision on which core to select. For each position, several cores are possible. For instance, for the cpu_x core there are in Figure 6 three alternative processor cores (cpu1, cpu2 and cpu3).

In general, the cores can be classified as:

- soft cores,
- firm cores, or
- hard cores.

A soft core is given as a specification that has to be synthesized, while a hard core is already a fixed netlist. A firm core is somewhere between a soft core and a hard core. A soft core is therefore the most flexible type of core;

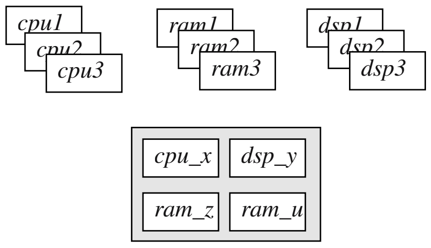


*Figure 6.*System design.

however, it requires the tools, effort and time involved in the work of synthesizing it to a gate-level netlist. A hard core, on the other hand, is already fixed and ready to be used in the design. Hence, less effort and time are required for the optimization of the core. The difference between a soft core, firm core, and a hard core is also visible when it comes to testing. For soft core there is a higher degree of freedom when determine the test method compared to hard cores where the test method and test vectors are more fixed.

3.1 Network-on-Chip

A problem when the SOC designs grow in complexity is that the on-chip interconnections, such as buses and switches, can not be used anymore due to their limited scalability. The wires are simply becoming too long. An alternative is to use networks on chip (NOC) [14, 121].

3.2 Platform-Based Design

The design flow must not only take the development of the hardware into account but also the development of software. Systems of today include an increasing number of programmable devices. The software (operating system and application programs as well as compilers) must be developed. In order to fix the architecture, the concept of platform-based design has been developed. Sangiovanni-Vincentelli [237] compare it with the PC (personal computer):

- The X86 instruction set makes it possible to re-use operating systems and software applications,
- Busses (ISA, USB, PCI) are specified,
- ISA interrupt controller handles basic interaction between software and hardware,
- I/O devices, such as keyboard, mouse, audio and video, have a clear specification.

The PC platform has eased the development for computers. However, most computers are not PCs but embedded systems in mobile phones, cars, etc [266]. Hence, a single platform-based architecture for all computer systems is not likely to be found.

4 CLOCKING

Clocking the system is becoming a challenge. Higher clock frequencies are making timing issues more important. The storage elements (flip-flops forming registers) in a system have to be controlled. For instance, Figure 7(a) shows a finite state machine (FSM), and a pipelined system is in Figure 7(b); the storage devices are controlled by the clock. Most systems are a combination of FSM and pipelined systems.

The input of a register (flip-flop) is commonly named *D*, and the output named *Q*. Figure 8 illustrates the following:

- the *setup time* (T_s) - the time required before the edge until the value is stable at the *D* input,
- the *hold time* (T_h) - the time required after the edge in order store the value in the register,
- the *clock-to-Q delay* (T_q) - the time required after the edge in order to produce the value on Q,
- the *cycle time* (T_c) - the time between two positive edges (from 0 to 1).

The registers can be design for instance as a *level-sensitive latch*, *edge-trigged register*, *RS latch*, *T register*, and *JK register* [277].

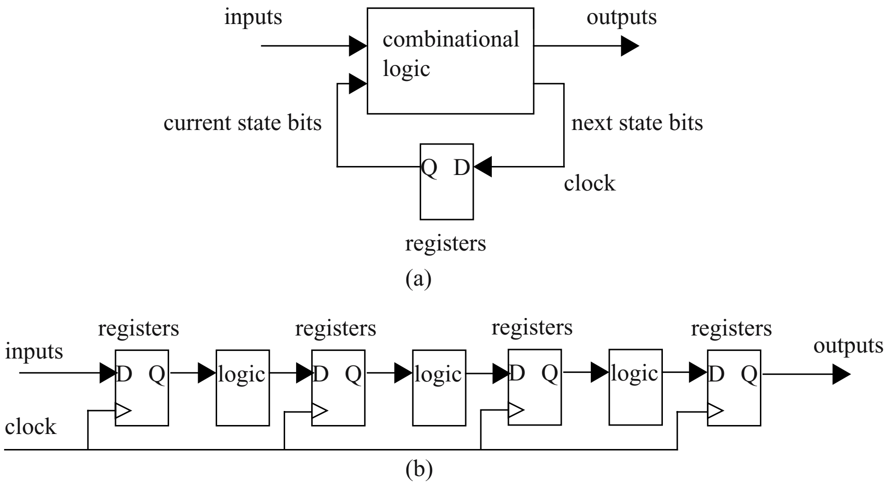


*Figure 7.*Clocked systems. (a) a finite state machine (FSM); (b) a pipelined system.

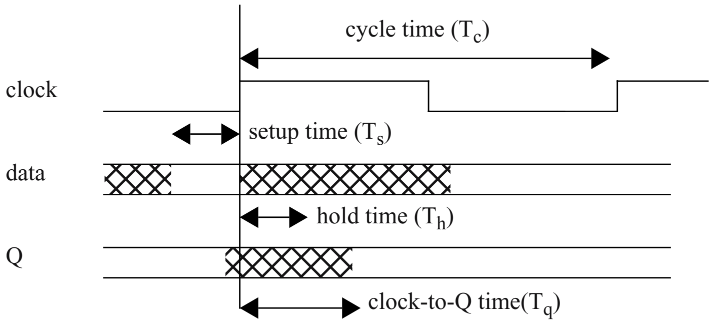


*Figure 8.*Single phase clocking and its parameters.

4.1 System Timing

Latches and registers can be used in different ways to implement a clocked system [277]. It is important that the clock signal is delivered to the registers correctly. Long wires may introduce delays in the clock distribution leading to clock skew - the clock signal is not delivered at the same time to all clocked elements.

The cycle time (T_c) is given by:

$$T_c = T_q + T_d + T_s$$

where T_q is the clock-to-Q delay, T_s is the setup time, and T_d is the worst-case delay through the combinational logic blocks.

4.2 Clock Distribution

Clock distribution is a problem. For instance, the total capacitance in the system that has to be driven can be above 1000 pF [277]. Driving such capacitance at high repetition rate (clock speed) creates high current (see example in Figure 9).

V_{DD} = 5V
C_{reg} = 2000 pF (20K register bits @ .1pF
T_{clk} = 10 ns
$T_{rise/fall}$ = 1 ns

I_{peak} = C× ×10^{-12}×5)/(1.0×10^{-9}) = 10A
Pd = C×V_{dd}^2×f = 2000×10^{-12}×25×100×10^{-6}

*Figure 9.*Example illustrating high current and power [277].

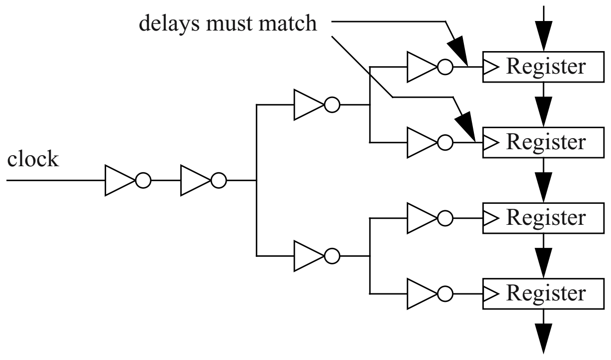


*Figure 10.*Distributed clock-tree where the logic is omitted.

The clock can be distributed using [277]:

- a single large buffer (*i.e.* cascaded inverters) that is used to drive all modules, or
- a distributed-clock-tree approach (Figure 10).

4.3 Multiple Clock Domains

In a core-based design, a set of cores are integrated to a system. Each core might require its dedicated clock speed. A design with multiple clock domains can present challenges. A typical problem is at the clock-domain boarders where one clock frequency is meeting another clock domain. Problems that can appear are, for instance, data loss and metastability. Data loss can appear when data generated by clock1 is not captured by clock2 until it has already been changed. Such problems appear if the destination clock is running at a higher speed. Metastability is due to that there is no timing relation between source and destination clock domain, and it can happen that data and clock signals reach the flip-flops at the same time. Metastability can cause reliability problems.

4.3.1 Phase Locked Loop (PLL)

A phase locked loop (PLL) is used to generate internal clocks in the system for two main reasons [277]:

- Synchronize the internal clock with an external clock.
- The internal clock should operate at a higher frequency than the external clock.

The increasing on-chip clock speed has lead to clock skew problems. A PLL allows an internal clock to be generated that is in phase with an external clock.

4.3.2 Globally-Asynchronous Locally-Synchronous

The advantage of making a system synchronous is that it is:

- a proven technique, and
- wide-spread EDA tools are available.

However, it is important to note that

- clock speed limited by slowest operation,
- clock distribution a problem,
- hard to design systems with multiple clock domains, and
- energy is consumed for idle operation.

An alternative is to make the system asynchronous. Instead of synchronize based on a clock, the system is controlled through communication protocol (often hand-shaking). The advantages with such a system are that is:

- ”average-case” performance,
- no clock signal to distribute, and
- no energy for idle operations.

The disadvantages with the asynchronous design approach are that:

- limited EDA supports, and
- extremely difficult to test.

Chapiro [36] proposed a scheme named Globally-asynchronous locally-synchronous (GALS) where there is an asynchronous wrapper around each core/block and where each core itself is synchronous. FIFO queues are inserted at the boundaries between the asynchronous and the synchronous parts.

The advantages with GALS are:

- easy to design systems with multiple clock domains,
- no global clock to distribute,

- power efficient - system only operates when data is available,
- asynchronous circuits (and related problems) limited to the wrapper, and
- main functional blocks designed using conventional EDA tools.

Among the disadvantages is the area overhead for self-timed wrapper. Implementations has shown about 20% power saving at 10% area increase.

5 OPTIMIZATION

Optimization problems can be mapped to well-known problem. The advantage by doing that is that if it is known that the well-known problem is NP-complete, the mapped problem is also NP-complete. The *Knap-sack problem* and the *fractional Knap-sack problem* are interesting for scheduling problems. The Knap-sack problem where given items of different values and volumes, the problem is to find the most valuable set of items that fit in a knapsack of fixed volume. The decision problem, if an item should be included in the knap-sack or not, is NP-hard [42]. However, the *fractional Knap-sack problem* where given materials of different values per unit volume and maximum amounts, find the most valuable mix of materials which fit in a knapsack of fixed volume. Since we may take pieces (fractions) of materials, a greedy algorithm finds the optimum. Take as much as possible of the material that is most valuable per unit volume. If there is still room, take as much as possible of the next most valuable material. Continue until the knapsack is full and the result is an optimal solution [42].

The test scheduling problem for systems where all tests are given and all are assigned a fixed testing time and the objective is to minimize the test application time is in the case when sequential testing is assumed trivial. The optimal test application time $\tau_{application}$ is for a system with N tests each with a test time τ_i (i={1..N} given by:

$$\tau_{\text{application}} = \sum_{i=1}^{N} \tau_i \quad (2.1)$$

The assumptions that only one test at a time can be active at any moment, and that all tests have to be applied, means that any order of the tests is optimal. An algorithm iterating in a loop over all tests is required and at each iteration one test is selected and given a start time. The computational complexity of such an algorithm depends linearly on the number of tests, $O(n)$ - n is the number of tests, hence the algorithm is polynomial (P). Most problems cannot be solved, that is an optimal solution is found, in polynomial time. The

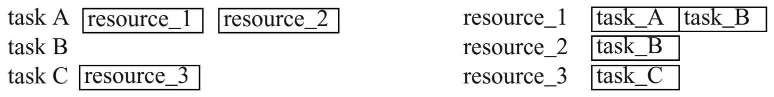


*Figure 11.*Gantt chart.

problems are then said to be *NP-complete* (non-deterministic polynomial time). Until today, there is no algorithm that in polynomial time guarantees to find the optimal solution for NP-complete problems. At present, all known algorithms for NP-complete problems require time which is exponential in the problem size [42]. Therefore, in order to solve an NP-complete problem for any non-trivial problem size, a heuristic can be used.

Optimization is the search for a solution that minimizes a given cost function where the ultimate goal is to find the *optimal solution* (no better solution can be found). An algorithm that guarantees to find the global optimum is an exact algorithm. In many cases, the problems are NP hard and a heuristic algorithm finds a solution that locally, not globally, optimizes the cost function. A heuristic *can* find the optimal solution, however, there is *no guarantee* that the optimal solution is found. A heuristic is often a trade-off between computational cost (CPU time) and computational quality (how far is the solution from the optimal solution).

In practice, the obvious problem with a heuristic is to evaluate the quality, that is how close is the solution produced from the heuristic compared to the optimal solution (the optimal solution is only guaranteed with exhaustive search, which is not possible due to time constraint). A lower bound can help a designer for instance in giving a feeling on how close to a lower bound a certain solution is. Often a resource limits a solution, and knowledge about the resources can guide the evaluation of the quality of a solution. A way to model resource utilization is to use a Gantt chart [23]. A Gantt chart can either be job (task)-oriented or resource oriented (figure 11). In figure 11(b) task_A and task_B both use resource _1 and it is obvious that resource_1 is used most, and hence, will most likely be the reasons for a bottleneck. A Gantt-chart can be used to define a lower bound on the test application time of a system, for instance. A lower bound is the lowest possible cost of a solution. Note that defining a way to compute a lower bound does not mean it is a way to find a solution corresponding to that the lower bound is found.

Above it was shown that it is trivial to develop an optimal test schedule in respect to test time for a sequential architecture when all tests are given a fixed test time and the objective is to minimize the test application time. In sequential testing, only one test at a time can be active, and that means that no constraint can limit the solution. In concurrent test scheduling, where more than one test can be applied at a time, conflicts often limits the solution. The problem to minimize the test application time using concurrent test scheduling for a system where all tests are given a fixed testing time *under no constraint* is trivial. All tests are simply started at time point zero. The optimal test application time $\tau_{application}$ is for a system with N tests each with a test time τ_i (i={1..N} given by:

$$\tau_{\text{application}} = \max\{\tau_i\} \qquad (2.2)$$

The concurrent test scheduling problem, more than one test can be executed at the same time, is in general not *NP*-complete. However, the concurrent test scheduling problem *under constraint* is *NP*-complete. We will discuss this in more detail below.

5.1 Optimization Techniques

The search for a solution can be done using exhaustive search or using some heuristics. The advantage of exhaustive methods is that an optimal solution is found. The problem is that most problems are complicated (NP-hard), which means that the computational effort (CPU time) is unacceptable for larger instances of the problem.

The most straight forward optimization approach is exhaustive search. It means that every single solution in the search space is evaluated and the best solution is reported. Below are a selection of such approaches, namely: *backtracking*, *branch-and-bound*, *integer-linear programming*. Iterative algorithms that iteratively search for an optimized solution but cannot guarantee that optimal solution is found are heuristics such as for instance: *local search*, *simulated annealing*, *genetic algorithms* and *tabu search*. These heuristics are iterative, that is from an initial solution an optimized solution is created through iterative modifications (transformations). A heuristic can be constructive - the solution is constructed from scratch. A constructive heuristic is used in an iterative transformation-based heuristic. The computational cost of an constructive algorithm is often in the range of N^x where x=2, 3, 4.

5.1.1 Backtracking and Branch-and-bound

The idea when using backtracking for an exhaustive search is to start with an initial solution where as many variables as possible are left unspecified [67]. The back-tracking process assigns values to the unspecified values in a systematic way. As soon as all variables are assigned to valid values, the cost of the feasible solution is computed. Then, the algorithm back-track to a partial solution assigning next value to the unspecified variables.

Often it is not needed to visit all solutions that are produced in the back-tracking process. Therefore the solution space is in back-tracking evaluated in a search tree. The idea in branch-and-bound is to not traverse sub-trees where the cost is higher than the best cost so far. If the cost of the solution of a partial specified solution is worse than the best solution, the sub-tree can be killed or pruned, which means that the sub-tree is not traversed. The search for a solution can be done in a depth-first or breath-first manner.

5.1.2 Integer Linear Programming

Integer Linear Programming (ILP) is a way to transform combinatorial optimization problems into a mathematical format. A high number of problems can relatively easily be reduced to ILP [67]. However, from a computational point of view, this does not help since ILP is NP-complete itself. Nevertheless, there exists a number of general ILP-solvers, which means that if the optimization problem can be modeled as an ILP instance, the software (program) solves the problem. The main advantage of using an ILP-solver is that an exact solution is found. The draw-back is, obviously, the computational cost for larger problems.

ILP is a variant of Linear programming (LP). LP problems can be solved the polynomial-time *ellipsoid algorithm*, however, it is in practice outperformed by the *simplex algorithm* which has a worst-case time complexity [67]. ILP is a variant of LP where variables can only be integers [67].

5.1.3 Local Search

Local search is an iterative method that search a *local neighborhood* instead of searching the whole search space when creating the new solution [67]. The idea is to search the neighborhood and by using modifications called move or local transform the current solution is improved. The best solution, found using exhaustive search in the local neighborhood (*steepest decent)*, is used as the next solution. In practice the search in the local neighborhood can be terminated at *first improvement*. The first improvement makes use of less computational cost compared to the steepest decent. However, in steepest decent each iteration creates a better solution, which can lead to less required iterations, hence less computational cost, compared to using first improvement. In order to avoid getting stuck at local optimum, a larger neighborhood can be used. However, if a larger neighborhood is to be considered, higher computational cost is required to search the neighborhood. Local search does not allow moves out of local optimum, so called uphill moves, as allowed in Simulated Annealing and Tabu Search.

5.1.4 Simulated Annealing

Simulated Annealing (or statistical cooling) proposed by Kirkpatrick *et al.* [131] is an optimization technique that is analogous to a physical process [67]. Initially, the material is heated up so that the molecules can move freely (the material is liquid). The temperature is slowly decreased to cool down the material. The freedom of molecules to move is decreased with temperature. At the end, the energy of the material is minimal provided that the cooling speed was very slow.

The pseudo-code for Simulated Annealing is in Figure 12. An initial solution is created and in each iteration a random modification of the solution is allowed. Initially, at high temperatures, major modifications are allowed while as the temperature decreases smaller and smaller modifications are allowed. If a better solution than the previous is created, it is kept. And at a certain probability, in order to avoid local optimum, worse solutions are accepted.

```
Construct initial solution, $x^{now}$;
Initial Temperature: T=TI;
while stop criteria not met do begin
  for i = 1 to TL do begin
    Generate randomly a neighboring solution $x' \in N(x^{now})$;
    Compute change of cost function $\Delta C=C(x')-C(x^{now})$;
    if $\Delta C \le 0$ then $x^{now}=x'$
      else begin
        Generate q = random(0, 1);
          if $q<e^{-\Delta C/T}$ then $x^{now}=x'$
      end;
  end;
   Set new temperature $T=\alpha \times T$;
end;
Return solution corresponding to the minimum cost function;
```

Figure 12. Simulated Annealing algorithm.

5.1.5 Genetic Algorithms

Genetic algorithm (pseudo-code in Figure 13) is inspired by nature and the theory of evolution [197]. An initial set of valid solutions called the population is created and in an iterative process, new solutions are created. The current population is replaced by a new population. In order to create a new solution, called a child, two solutions are selected, called parent1 and parent2. The selected parents create in a crossover process a child, which will belong to the new generation. The idea is that by combining features from the best solutions (selected parents) in current population, new better solutions (children) can be created. Each solution is characterized by its chromosome and the chromosome from two parent solutions creates a child solution. In order to avoid local optimum, mutation is allowed in the crossover process. A mutation is, as in nature, the insertion of a small modification in the creation of the child solution.

```
Create an initial population pop
Do begin
  newpop=empty
  For i=1 to size_of_population Begin
    parent1=select(pop)
    parent2=select(pop)
    child=crossover(parent1, parent2)
    newpop=newpop+child
  End
  pop=newpop;
End;
Return solution corresponding to the minimum cost function;
```

*Figure 13.*Genetic algorithm.

5.1.6 Tabu Search

Tabu search is an optimization technique where the solution is improved through search in the neighborhood. An initial solution is created and a set of modifications are defined. The initial solution is through the defined modifications improved in *n* iterations. In order to avoid having a modification followed by a re-modification, a tabu list is used. The tabu list of length *k* keep track on the *k* last moved and prohibit cycles of length <*k*.

Chapter 3

Design for Test

1 INTRODUCTION

Design for testability is basically how to make each unit in the system testable. The intention with this chapter is to give an introduction to techniques for design for test. For further studies there are a number of books on test such as Abramovici *et al*. [2], Bushnell and Agrawall [24], Lala [151], Mourad and Zorian [199], Rajsuman [225] and Tsui [264].

Testing is used to check if a design is correct according to its specification. *Failures* can appear at any time during the life of a system, and a failure has occurred if the system's behavior is different from its specification. A *fault* is a physical defect that may or may not cause a failure, and a fault can be described by its *nature*, *value*, *extent*, and *duration* [151]. The nature of a fault can be logical or nonlogical. A logical fault makes the logic at a certain point in time different from its specification, while a nonlogical fault is due to the rest of the faults, such as clock signal distribution, power failure, etc. [151]. The value of a fault is simply how the fault creates the signal. The fault value can be fixed or it may vary over time. The extent of a fault is how the fault effect is distributed. A fault may affect only a part of the system, locally, or it can distribute over a larger part in the system. For instance, a logical fault may be local while a fault on the clock distribution is likely to be distributed through the system. The duration of the fault is if the fault is permanent or temporary. Sometimes a permanent fault is referred to as a hard fault and a temporary fault is a soft fault. A temporary fault or soft fault are often more complicated to replicate (make the fault appear again). These types of faults are due to not stable power supply (transient) or degradation (intermittent) [199].

Faults may appear in the system at any time. However, a majority of the faults, if any, are introduced during the manufacturing process. Hence, focus on production testing and on faults related to the manufacturing is of high importance.

1.1 Fault models

The faults in a design can be due to defective parts, breaks in signal lines, lines unintentionally connected to ground or to the power supply, short-circuit between signal lines, and so on. Poor design may introduce faults such as hazards and races. A fault model is used to model the behavior of a fault. The most common fault models are the models for stuck-at faults (Section 1.1.1), bridging faults (Section 1.1.2), stuck-open fault (Section 1.1.3), and delay fault (Section 1.1.4).

1.1.1 Stuck-at Fault Model

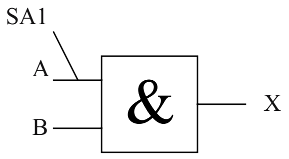
The most commonly used fault model is the single stuck-at (SSA) model. It assumes that a fault location can either be stuck-at 1 or stuck-at 0. If a fault at a wire is stuck-at 0, the logic value at the wire is always 0. In order to detect such a fault, the test stimulus should produce logic 1 at the wire. A stuck-at 0 fault at a wire is to be seen as if the wire is connected to ground. For a stuck-at 1, the wire acts as if it was connected to V_{dd} (power supply). For the generation of test vectors for SSA, it is assumed that only one fault appear in the system. Otherwise, if more than one fault is present at a time, the multiple stuck-at (MSA) fault model is used.

Assume a SA1 fault on input A at the AND-gate in Figure 14. The fault makes the truth table different compared to the correct (fault-free truth table). Regardless of what value input A is set to (0 or 1), the input is 1 (stuck-at 1). The number of faults is $2*k$ under the SSA assumption in a circuit with k lines. For instance, there are 3 lines (fault locations), namely A, B, and X, at the AND-gate in Figure 14, and each location can be either SA0 or SA1, *i.e.* 6 faults.

The stuck-at fault model is widely used. Among the advantages are the simplicity of the model and that it has been shown that it is effective to detect several types of faults. Among the disadvantages are that it is less effective to model faults in modern sub-micron CMOS systems [60, 151].

1.1.2 Bridging Fault Model

It is actually more likely that two signal lines are unintentionally connected compared to that a single line is stuck-at 0 or 1 (connected to ground or V_{dd}). A short-circuit between two signal lines is known as bridge fault. The



ABX
0 0 0
0 1 0
1 0 0
1 1 1

Fault-free (correct) behavior

ABX
1 0 0
1 1 1
1 0 0
1 1 1

Faulty behavior

*Figure 14.*An AND gate with a SA1 at input A.

bridging faults can be grouped into input bridging, feedback bridging and non feedback bridging [151].

The number of faults depends on the number of signal lines that are tested to bridge. In order to reduce the amount of bridging faults knowledge about the wire routing is needed, since signal lines routed at far distance from each other cannot bridge while signal lines routed next to each other are likely to bridge.

1.1.3 Stuck-Open Fault Model

Breaks and transistor stuck-on defects in CMOS designs may be undetected under the stuck-at fault model [151]. Breaks may occur on lines between gates, signal line breaks, or on internal lines in a gate, intragate breaks. To model these faults, transistor level fault models are used because it is required to know the structure of the circuit. It means that tests have to be done for shorts with stuck-on transistor and opens with stuck-open transistor [151]. A stuck-on transistor fault makes short-circuit between the source and the drain.

1.1.4 Delay Fault Model

Timing faults are detected with the delay fault model. Minor timing problems in the system, for instance if a signal is slightly delayed, may not affect the systems at all. However, longer delays through gates can make the system fail to meet its timing specification. Therefore, delay testing is required. Furthermore, in high performance systems running at very high clock frequencies testing for timing faults is of high importance.

Two types of delay faults have been proposed: *gate delay faults* and *path delay faults* [151]. Gate delay faults consider the delay within a certain gate. For instance, if the delay through a gate is longer than a certain value, the gate is considered faulty. The main draw-back with the gate delay model is that it only considers faults locally at each gate. The path delay model, on the other hand, considers the delay impact on a given path. It means that each gate in a path can meet its timing specification, but the propagation delay through a path (a given sequence of gates) might exceed a specified value. The disadvantage of the path delay model is the high number of possible paths in a circuit. A way to reduce the number of paths is to only consider, so called, critical paths. Critical paths are paths that have a direct impact on the delay. However, in high performance systems, the number of critical paths increases and basically every path in the system is a critical path.

1.2 Test Generation for Combinational Circuits

Fault detection is the process to determine if a system contains a fault, and fault location is the process of discovering the location of a fault. Fault detection is performed by applying a sequence of test inputs and observing the outputs which are compared to expected (fault free) outputs. Testing can be performed without knowledge about the structure, only by the function, of the circuit. For instance, to test the function completely of a 3-input adder (Table 1), all possible (2^3=8) input vectors are needed. The number of required test inputs grows exponentially (2^n - n number of input pins). And therefore alternatives where a lower number of test inputs (test stimulus) is needed have been developed.

Table 1. A 3-input adder.

a b c	*carry*	*sum*
0 0 0	0	0
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	1	0
1 1 1	1	1

The faults that can be detected with a test is determined with a fault simulator, which is basically a logic simulator where the faults are introduced usually one at a time and the test stimulus is applied and the output response is compared to the expected response (fault-free response).

The quality of a test is determined by the fault coverage (fc) given as [151]:

$$f_c = \frac{f}{x} \tag{3.1}$$

where f is the number of detected faults and x is the number of faults in the circuit. Some faults are not possible to detect, and *fault collapsing* where *equivalent faults* are ignored can be used to reduce the number of faults. For instance, an x-input logic gate can have $2*x+2$ possible faults but some faults cannot be distinguished from each other.

Test generation is the process of creating test stimulus and test response for the circuit or for parts of it (logic blocks or cores). Circuits can be divided into combinational and sequential. A combinational circuit or part of the circuit does not have any clocked elements such as flip-flops, while a sequential design or a sequential partition of the design has clock elements. Test generation for combinational circuits is compared to test generation for sequential circuits simpler because there are no clocked elements. Test generation for sequential circuits means that in order to set a certain value in the circuit, several time frames might be required, which makes the search space larger and hence the problem more complicated.

1.2.1 Path Sensitization

In path sensitization a path is selected from a failure to the outputs. In order to create test stimulus for a SA1 present at line *a* in figure 15, line *a* should be set to 0 and line *b* should be 1, since there is a stuck-at-1 fault at line a, the output value on gate G_1 (line *e*) will be 1. The value on line *e* should be propagated to line *g* and to do so line *f* should be 1. The value 1 is set on line *f* if at least one of line *c* and line *d* are 1. The input word 0111 (abcd) is the test stimulus for a stuck-at-1 on line 1. In the case of a fault-free (good circuit) the output will be 0 while in the case of a faulty circuit the output will be 1. Note that, it was enough setting line *c* or line *d* to 1. And therefore, a test stimulus is not unique for a given fault, several alternative stimuli exist.

The forward trace is the process of setting values from the fault site to the circuit's outputs, and the backward trace is the process to set necessary gate conditions in order to propagate the fault along the path. In the example (Figure 15) assigning value to line *b*, *e*, *f*, and *g* are included in the forward trace, while setting values on line *c* and *d* are referred to as the backward trace.

1.2.2 Boolean Difference

In Boolean difference two Boolean expressions, one for the fault-free circuit and one for a faulty circuit, are ORed. Assume $F(X) = F(x_1, \ldots, x_n)$ as a

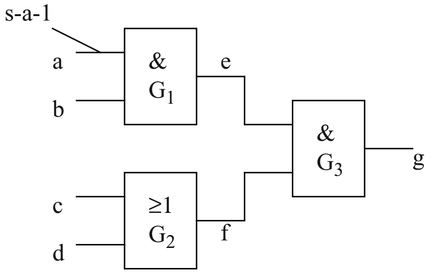
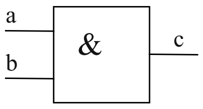


*Figure 15.*An example circuit.



a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

(a)

a	b	c
0	x	0
x	0	0
1	1	1

(b)

Figure 16.(a) Truth table; (b) singular cover.

logic function of *n* variables. The Boolean difference of *F*(*X*) with respect to a fault x_i is defined as:

$$\frac{dF(x_1\ldots, x_i, \ldots x_n)}{dx_i} = \frac{dF(X)}{dx_i} = F(x_1\ldots, x_i, \ldots x_n) \oplus F(x_1\ldots, \bar{x}_i, \ldots x_n)$$

Assume a fault at line a in Figure 15 and that line *a* is input x_1, line *b* is input x_2, line *c* is x_3, and finally line *d* is x_4, we then have [246]:

$$\frac{dF(X)}{dx_3} = \frac{dF(x_1 \cdot x_2 + x_3 \cdot x_4)}{dx_3} = \{\text{Boolean rules}\} = x_2 \cdot \overline{x_3 \cdot x_4}$$

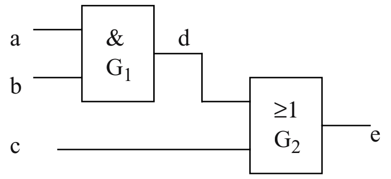
The advantage of Boolean difference approach is that it generates tests for every fault in a circuit. The algorithm is complete and does not require trial and error [151]. The disadvantage of the method is that it requires high computational cost and high memory requirements.

1.2.3 D-algorithm

The first algorithmic test generation method is the D-algorithm, and if a test exists for a fault, the algorithm finds it [233]. The algorithm makes use of *singular cover*, *D-cube propagation*, *D-cube fault representation*, and *D-intersection*. The singular cover is a compact version of the truth table. An example for an AND-gate is in Figure 16.

The singular cover for a circuit is illustrated in Figure 17.

The input-output behavior of a good and faulty gate is represented by a D-cube. The symbol D can be either 0 or 1 and $\overline{\text{D}}$ is the opposite of D. It means



	a	b	c	d	e
G_1	0	x		0	
	x	0		0	
	1	1		1	
G_2			0	0	0
			x	1	1
			1	x	1

*Figure 17.*The singular covers for a circuit.

a	b	c
0	x	0
x	0	0
1	1	1

(a)

a	b	c
0	D	$\overline{D}$
D	0	$\overline{D}$
D	D	D

(b)

Figure 18.(a) Singular cover and (b) D-cubes for an AND-gate.

that if D is 1, $\overline{D}$=0 and vice versa. The propagation of D-cubes of a gate are those that makes the output depend on the specified inputs. The propagation D-cubes for a two-input AND gate are in Figure 18. In this case, D is understood as being 1 if the circuit is fault-free and 0 if the given fault is present.

The existence of a given fault is specified using the D-cube. For instance, if there is a stuck-at 0 on the output of the AND-gate in Figure 16 the primitive D-cube is:

a	b	c
D	D	D

The D-intersection is used to build the sensitized paths. The D-cube for a test for a stuck-at 0 fault at line *b* in Figure 17 is:

a	b	d
1	1	D

In order to propagate the $\overline{D}$ on line *d* through G_2, the propagation cube of G_2 must match. The following values propagate:

c	d	e
0	$\overline{D}$	$\overline{D}$

The D-algorithm selects primitive D-cubes for the given fault. All possible paths are then sensitized from the fault location to a primary output (D-drive). The final step is a consistency operation.

An advantage of the D-algorithm is that it “proves” that it can identify redundant faults if such exists [151].

1.2.4 PODEM (Path-Oriented Decision-Making)

The PODEM algorithm enumerates all input patterns for a given fault. The process continues until a test pattern is found or the search space is exhausted. If no pattern is found, the fault is considered untestable [72].

The advantage of the PODEM approach is that it compared to the D-algorithm is more efficient in terms of required computer time.

1.2.5 FAN (Fanout-Oriented Test Generation)

The FAN algorithm is similar to the PODEM, but unlike PODEM where backtracking is performed along a single path, FAN uses multiple paths [151]. A technique called multiple backtrace is used to reduce the number of backtracks [63].

1.2.6 Subscripted D-Algorithm

The D-algorithm generates test vectors for one fault at a time [39]. It means that the path sensitization must be repeated from the output of a gate to a primary output for every fault associated with a particular gate. The sensitization must take place even if the paths are similar. The basic idea with the Subscripted D-Algorithm [195] is to remove some of the repeated work.

1.2.7 CONcurrent Test Generation

The CONcurrent Test Generation (CONT) Algorithm [259] tries to concurrently generate tests for a set of faults [39]. The algorithm makes use of PODEM but in addition it keeps track on active faults in the forward implication process. The advantage is that once a vector is found for a fault, all faults in the fault list associated with primary outputs are detected by the vectors. In the presence of conflicts, a strategy of fault switching is used. The strategy of changing target faults reduces the number of needed backtracks.

1.2.8 Fault Simulation

Fault simulation determines all faults that are detected with a given test input. The D-algorithm, on the other hand, creates a test for a given fault. In fault simulation, a list of faults is kept and as soon as a fault is detected, it is marked as detected. An example circuit with 5 lines (a to e) and hence 10 faults (line a stuck-at 0 (a_0) and stuck-at 1 (a_1) and so forth) is in Figure 19. Input pattern 000 detects three out of ten faults, c_1, d_1, and e_1, making the fault coverage 30% (Table 2). Input pattern 001 detects two faults, and in total after the two input vectors (000 and 001) five out of ten faults are detected, making the fault coverage 50%. Some input patterns, such as 011, do not contribute to increase the fault coverage; hence such input patterns can be removed. A sequence of {000, 001, 010, 100, 110} detects all ten faults.

1.2.9 Optimization

Fault simulation and test generation can be co-optimized in order to minimize the effort spent on test pattern creation. In practice, it is common to first perform fault simulation with arbitrary inputs to detect so called easy to detect

Table 2. Fault simulation results on example circuit in Figure 19.

input *a b c*	*output* *e*	a_0	b_0	c_0	d_0	e_0	a_1	b_1	c_1	d_1	e_1	*detected faults*	*fault coverage*
0 0 0	0	0	0	0	0	0	0	0	1’	1’	1’	c_1, d_1, e_1	30%
0 0 1	1	1	1	0’	1	0’	1	1				c_0, e_0	50%
0 1 0	0	0	0		0		1’	0				a_1	60%
0 1 1	1	1	1		1			1					
1 0 0	0	0	0		0			1’				b_1	70%
1 0 1	1	1	1		1								
1 1 0	1	0’	0’		0’							a_0, b_0, d_0	100%
1 1 1	1												

faults for a specified computational time. Afterwards, a test generation algorithm replaces the random test generation for the hard to detect faults.

The cost of applying a test set can be reduced be efficiently select test patterns. Based on equivalent faults - faults that are detected by exactly the same test patterns - the number of patterns can be reduced [120]. And also by knowing which fault each test pattern detects, the minimal set of patterns can be selected.

The selection of test patterns can also be used to reduce the test application time. The fault table for the example circuit (Figure 19) is in Table 3 where an x marks that a test pattern detects a fault. For example, pattern 000 detects e_1 (line e stuck-at-1). The stuck-at 1 fault e_1 can be detected by one of 000, 010, and 100. The objective is to select a minimum of patterns in such a way that every fault is covered once. However, due to new fault types, an effective way of detecting faults is to detect every fault multiple times [182].

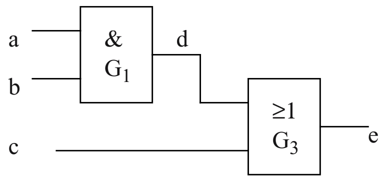


Figure 19. An example circuit.

Table 3. Fault table for the example circuit in Figure 19.

input	*ouput*	*fault locations*									
a b c	*e*	a_0	b_0	c_0	d_0	e_0	a_1	b_1	c_1	d_1	e_1
0 0 0	0								x	x	x
0 0 1	1			x		x					
0 1 0	0						x		x		x
0 1 1	1			x		x					
1 0 0	0							x	x		x
1 0 1	1			x		x					
1 1 0	1	x	x		x	x					
1 1 1	1					x					

1.2.10 Delay Fault Detection

A delay fault is detected by applying two vectors, one initialization vector to set the fault location, and in the consecutive clock cycle one transition vector or propagation vector is applied to activate the fault at a primary output. The initialization vector sets the condition in order to test a slow-to-rise or a slow-to-fall signal and the propagation vector propagates the fault effect to an output. The delay faults can be classified into *robust* and *nonrobust* [151].

1.3 Testing Sequential Circuits

A circuit usually contains both combinational blocks of logic and sequential elements such as registers formed by a set of flip-flops (see Figure 20). In order to set a fault location, sensitize, not only must the combinational logic must be considered, but also the number of clock cycles required propagating values from the inputs of the circuit to the fault location (and from the fault location to the primary outputs). For instance, assume testing for a fault in logic block B in Figure 20. The test stimulus must from the primary inputs be clocked through the sequential elements to logic block B, and the test response must also be clocked through sequential elements until primary outputs are reached.

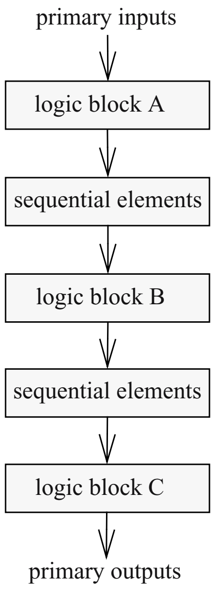


Figure 20. A circuit with both blocks of combinational logic and sequential elements.

Chen and Agrawal [39] group algorithms for sequential test generation as follows:

- *Iterative Array Approaches* - combinational test generation algorithms are used as the basis in sequential test generation. The outputs are fed to the inputs as a previous-time copy. Examples of such algorithms are Extended D-Algorithms [150,221], Nine-Value Algorithm [203], SOFTG [249], Backtrace Algorithms [37, 183, 193, 194],
- *Verification-Based Approaches* - tries to determine if the circuit under test operates as its state table. Example of such a technique is the SCIRTSS (Sequential CIRcuit Test Search System) [95].
- *Functional and Expert System Approaches*. Examples of functional approaches are proposed by Sridhar and Hayes [251], and Brahme and Abraham [21], and expert system approaches have been proposed by Bending [13] and Singh [248].

2 DESIGN-FOR-TEST METHODS

The design under test can be modified in order to make it easier to test, that is improve the design's testability. In this section we discuss several techniques to improve the testability such as test point insertion and the scan technique.

2.1 Test Point Insertion

The obvious problem when generating tests is to control and observe values at certain locations within the circuit. A way to ease the test generation process is to use test point insertion to improve controllability and/or observability at lines in the design. Test point insertion is illustrated in Figure 21. The original circuit with no test points is in Figure 21(a). Assume that it is difficult to observe the value on the line between logic block A and logic block B. The insertion of a test point to observe the value on the line is shown in Figure 21(b). If the line between logic block A and logic block B is difficult to set to 1, a test point as in Figure 21(c) can be used while if the line is difficult to set to 0, a test point as in Figure 21(d) can be used. If the value at a line is both difficult to observe and control combinations of test points can be used. The disadvantage of test point insertion is that the cost of additional test points is high. For instance, direct access is required, and it usually means that direct access must be introduced from existing or additional input/output pins of the system to the actual test point. The cost is related to additional wiring and, if needed, additional pins.

2.2 The Scan Technique

The scan technique, initially proposed by Kobayashi *et al.* [134], is a straight forward technique that is easily automized, so called plug'n play. An advantage of the technique is that all flip-flops are connected into shift regis-

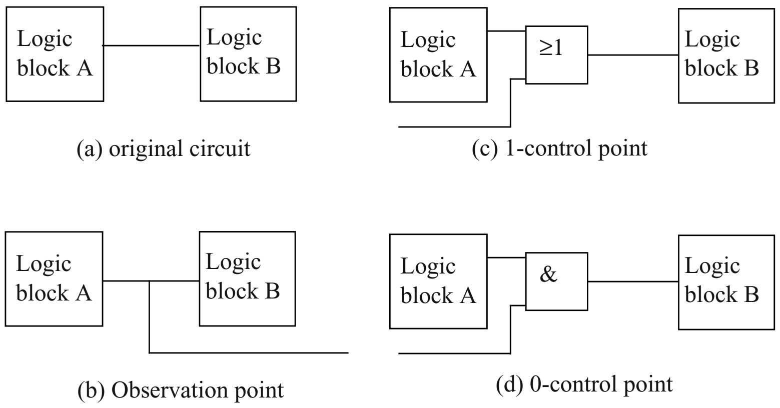


*Figure 21.*Test point insertion, (a) original circuit, (b) observation point, (c) 1-control point, and (d) 0-control point.

ters making the circuit behave as if it was a combinational design. A combinational test generation tool can be used, eliminating the problems due to sequential blocks.

The basic idea in the scan technique is to connect the sequential elements (registers) into a shift register. In test mode, test stimulus can be shifted in through the scan path. A normal (system clock) is then applied to execute the test stimulus and the test response is then captured in the scan path (the shift register). Finally, the test response can be shifted out and analyzed. An example of a circuit with a scan-chain is in Figure 22. To test logic block B, after the shift in of test stimulus into register 1, a capture cycle is applied and the test response is loaded register 2 and it can be shifted out for analysis. In order to reduce the shift time, new test stimulus can be shifted in at the same time as the test response from the previous test stimulus is shifted out.

A scan path is shown in Figure 23. The flip-flops are connected into a scan path, and the additional multiplexors are used to select between testing mode and normal operation. For scan testing, a minimum of three additional pins are required. Two inputs, one for multiplexor selection between testing mode and normal mode and one for serially inserting test stimulus in the scan path. The added output is to observe test response. It is possible to reduce the added number of pins by multiplexing with existing pins.

A single scan chain results in the lowest need of additional pins. However, a single scan chain leads to long shift in/shift out times, and the consequence

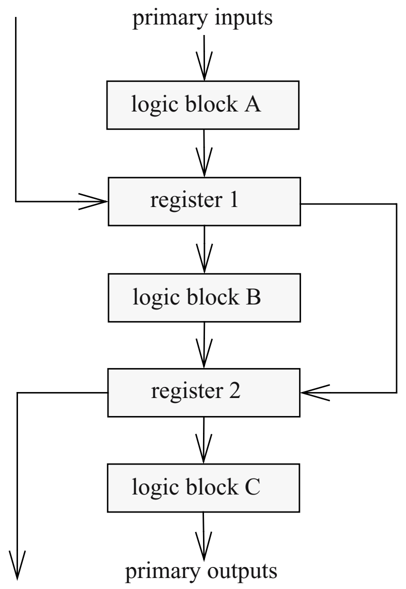


Figure 22. A circuit with both blocks of combinational logic and sequential elements.

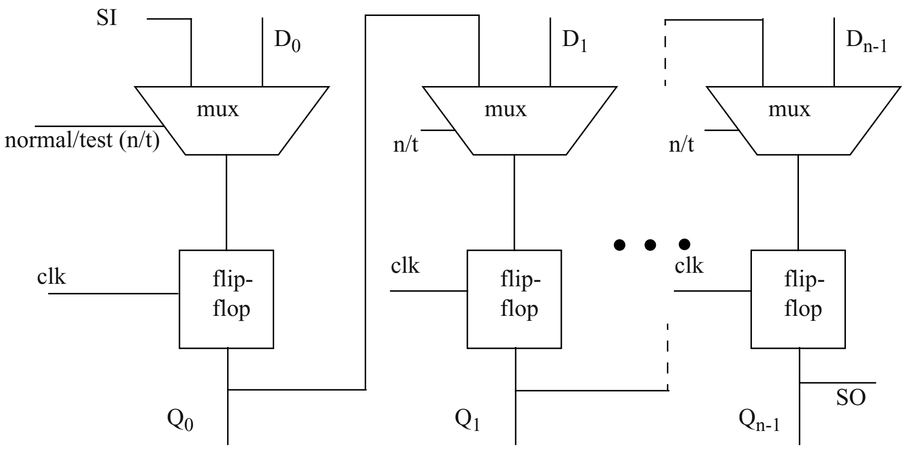


*Figure 23.*A scan path.

is that the test application time becomes high. A way to reduce the test application time is to partition the scanned elements into a higher number of scan chains, which makes each scan-chain shorter and hence the testing time is reduced.

The number of clock cycles to shift in a test vector is given by the length of the scan chain. The test time for a single test vector is given as the summation of the shift in time, the capture cycle, and the shift out time. For a set of test vectors, the time is multiplied by the number of test vectors (Figure 24(a)). Hence, the test time for a single scan chain of length *sc* tested with *tv* test vectors without using overlapping is given by:

$$\tau = (sc \times 2 + 1) \times tv \tag{3.2}$$

The testing time for a single scan chain of length *sc* tested with *tv* test vectors where overlapping is used is illustrated in Figure 24(b). Overlapping means that while the test response of the current test vector is shifted out, the following test vector is shifted in. Overlapping can be seen as a way of pipelining the test vectors. The test time when using overlapping is given by:

$$\tau = (sc + 1) \times tv + sc \tag{3.3}$$

Note, that the term last term *(+sc)* is added for unloading the test response from the final test pattern.

When discussing when scan chains are partitioned into more than a single scan chain, it is important to distinguish between soft cores and hard cores. For soft cores the number of scanned elements (scan flip-flops) and the num-

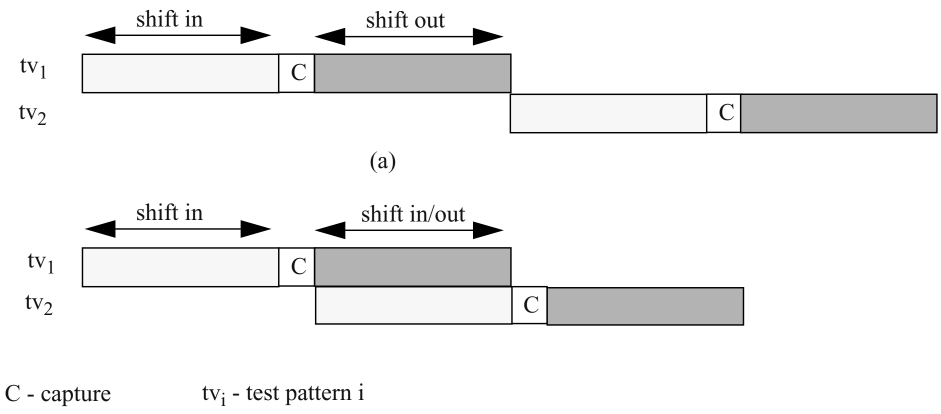


Figure 24. Scan chain testing (a) without test vector overlapping and (b) with test vector overlapping.

ber of test patterns are given. If the *ff* scan elements are partitioned into *n* scan chains the test time assuming no over lapping is given by:

$$\tau = \left(\left\lceil \frac{ff}{n} \right\rceil \times 2 + 1\right) \times tv \tag{3.4}$$

and the testing time for a soft core when overlapping is used is given by:

$$\tau = \left(\left\lceil \frac{ff}{n} \right\rceil + 1\right) \times tv + \left\lceil \frac{ff}{n} \right\rceil \tag{3.5}$$

Note that the upwards rounding is used due to that the length of scan chains must be integer. For instance, if a core has 3 scanned elements that are partitioned into 2 scan chains the shift-in/shift-out time is 2.

For hard cores the number of scan-chains is fixed. Assume that a core is tested with a set of test vectors *tv* and that the scan chains are partitioned into a *i* scan chains each of length sc_i. The testing time for a hard core when test vector overlapping is not used is given by:

$$\tau = (\max(sc_i) \times 2 + 1) \times tv \tag{3.6}$$

and the testing time for a hard core when test vector overlapping is used is given by:

$$\tau = (\max(sc_i) + 1) \times tv + \min(sc_i) \tag{3.7}$$

In the case of hard cores, we overlooked the problem of grouping scan-chains into the set of scan-chains. For instance, if 5 scan chains of different length are given and they are to be connected into 2 chains, there is a need to group the chains. However, this problem will be further discussed (see *Test Scheduling* on page 115).

In the Level-Sensitive Scan Design (LSSD) approach three clocks are used, one is for the normal operation and one clock for each of the master and slave flip-flop.

2.2.1 Scan Testing for Delay Faults

It is becoming important to not only test for stuck-at faults, but also for delay faults and timing related (Delay Fault Model, see page 23). In order to detect a delay fault, two test vectors are used. The first vector initializes the design and the second vector, applied in the consecutive clock cycle, captures the fault. Furthermore, it is important that the testing is performed at system speed, otherwise the timing fault is most likely not present.

A disadvantage of the scan technique is that it is difficult to *apply the two vectors in consecutive clock cycle*. The scan technique requires test vector shift-in, capture, and then shift-out.

2.2.1.1 Enhanced Scan

The enhanced scan flip-flop was proposed by Dervisoglu and Strong [52]. The enhancement compared to a standard scan flip-flop is an additional latch and the advantage of the additional latch is that it can hold a value while the scan flip-flop is loaded/unloaded. For delay testing, in means that for a scan path with enhanced scan flip-flops, first a vector V1 can be shifted in, and second a vector V2 can be shifted in. The first vector V1 is kept in the added latch. Figure 25 shows an illustrative design with enhanced scan flip-flops and the timing diagram for one test (shift in of vector V1 and vector V2). The disadvantage of enhanced scan is the additional silicon area that is required.

2.2.1.2 Skewed-load

Savir and Patil [235, 236, 238] propose a skew-load technique which has the advantage that it can be used in a standard scan architecture. The test is designed so that the second vector (V2) is created from the first vector (V1) through a one-bit shift. The first vector (V1) is shifted in. The second vector (V2) is then created by an additional shift and the response is captured. See the illustration in Figure 26(a).

The advantage of the technique is that it does not require additional scan area as in the case with enhanced scan. The disadvantage of the technique is that the two vectors (V1 and V2) for a test cannot be created purely independent, hence, it can be difficult to achieve a high fault coverage.

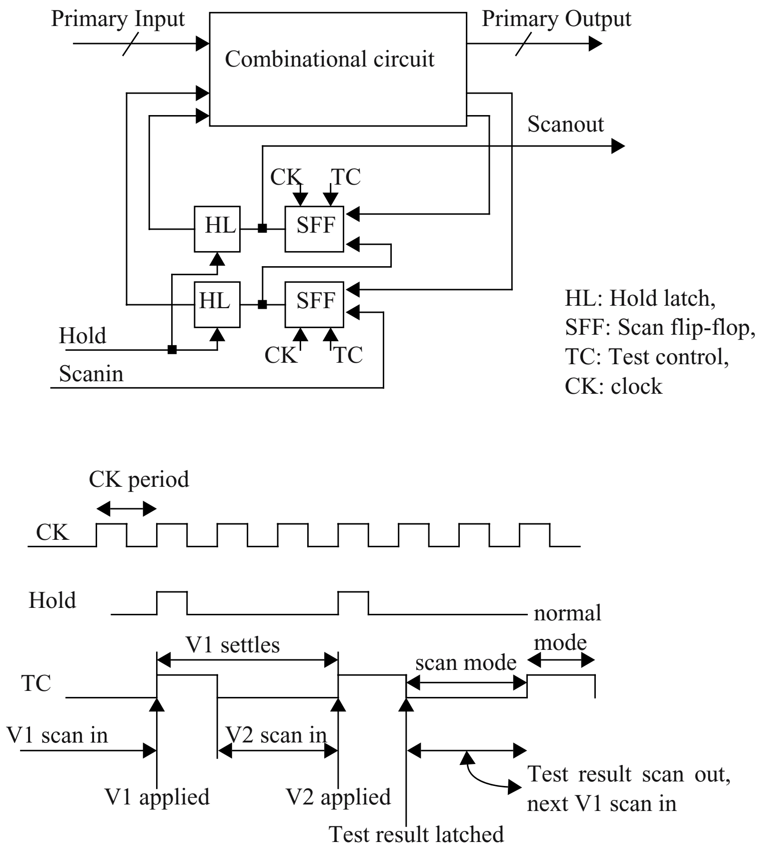


*Figure 25.*Enhanced-scan.

2.2.1.3 Broadside Test

In Broadside Test, Savir and Patil [239, 240] propose a technique where standard scan architecture is used, and the additional vector is created through an analysis of the combinational logic. Figure 27 shows an example where logic block B is to be tested. The first vector V1 is shifted in through the scan-path. The second vector V2' is also shifted in; however, for V2', the analysis of the design and the scan-path results in what V2' should be shifted in. Furthermore, the analysis of the circuit, in this case logic block A, results in V2' where the vector V2 is the output of logic block A if V2' is the input. The scheme allows the two vectors (V1 and V2) to be applied in consecutive clock cycles, when vector V1 has been applied, vector V2 has been produced by V2' and logic block A, and V2 is present in the scan register. See the illustration in Figure 26(b).

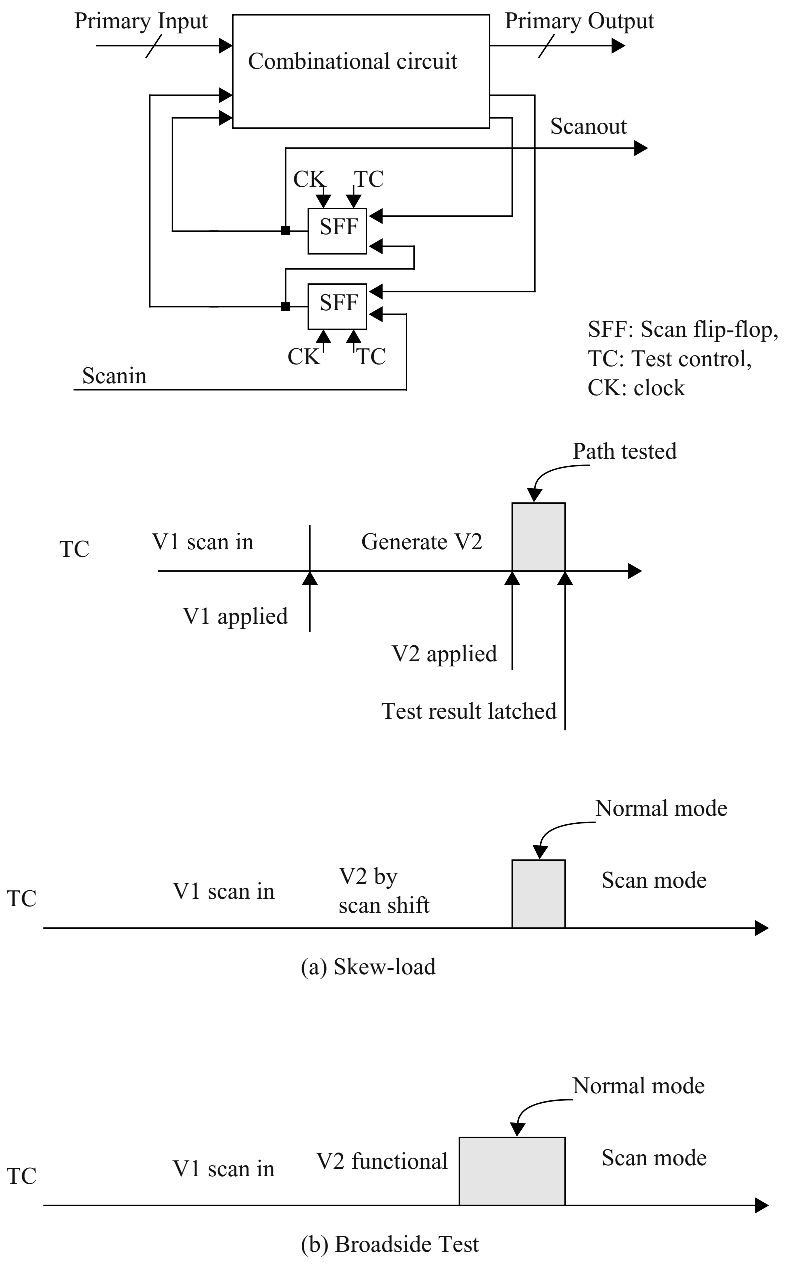


Figure 26. Skew-load and Broadside Test.

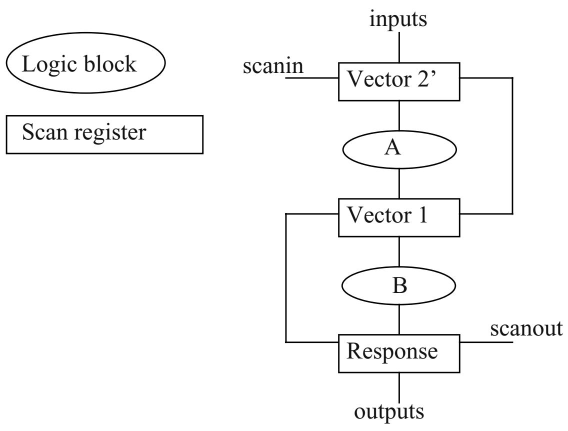


*Figure 27.*Broadside Test.

The advantage of Broadside Test is, similar to Skewed-load, that it is used using standard scan architecture, hence, no additional silicon area is needed. The disadvantage is, also similar to Skewed-load, that the fault coverage depends highly on the combintational logic.

2.3 Test Pattern Generation for BIST

A deterministic test vector set is created using an *automatic test pattern generator* (ATPG) tool, where the structure of the circuit under test is analysed and based on the analysis, test vectors are created. The size of the test vector set is relatively small compared to other techniques, which reduces test application time. If the generated test vector set is to the circuit using an external tester the following limitations are important to note [92]:

- scan usually operates at a maximum frequency of 50 MHz,
- tester memory is usually very limited, and
- it can support a maximum of 8 scan chains, resulting in long test application time for large designs.

BIST test pattern generation can be divided into:

- exhaustive,
- random/pseudo-random, and
- deterministic.

2.3.1 Exhaustive Pattern Generation

In an *exhaustive test generation approach*, all possible test patterns are applied to the circuit under test. For an n-input combinational circuit, all 2^n possible inputs are applied. Such an approach detects all possible stuck-at faults. A counter can be used to implement an exhaustive pattern generator. The area-overhead and design complexity is low and it is feasible to place such a generator on-chip. However, the approach is often not feasible since the number of possible patterns is too high: for a n-bit input design 2^n patterns are generated which results in extremely long test application time for larger designs.

A way to make the approach applicable to larger designs, is to partition the circuit into several smaller designs, where the size of each partition is defined in such a way that it is possible from a computational complexity point of view to apply all possible test patterns.

2.3.2 Pseudo-Random Pattern Generation

Another approach is to use random-based techniques. The draw-back with randomly generated test patterns is that some patterns are hard to create. For instance, generating a test pattern that creates a one on the output of an AND gate is only achieved when all inputs are one; the probability is $1/2^n$. For a 4-bit AND-gate the probability is only 0.0625 ($1/2^4$), Figure 28. This means that a large set of test vectors has to be generated in order to achieve high fault coverage, which leads to long test application time.

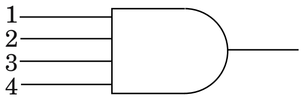
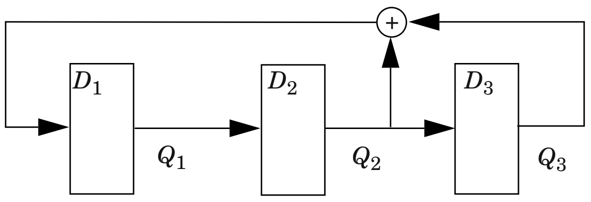


Figure 28. A 4-input AND-gate.

2.3.3 Pseudo-random-based test generation

A pseudo-random test pattern set can be achieved using a *linear feedback shift register* (LFSR). An advantage is the reasonable design complexity and the low area overhead, which allow on-chip implementation. An example of an LFSR is shown in Figure 29 where one module-2 adder and three flip-flops are used. The sequence can be tuned by defining the feedback function to suit the block under test.



S_0	0	1	1
S_1	0	0	1
S_2	1	0	0
S_3	0	1	0
S_4	1	0	1
S_5	1	1	0
S_6	1	1	1
S_7	0	1	1

*Figure 29.*Example of 3-stage linear feedback shift register based on x^3+x+1 and generated sequence where S_0 is the initial state.

2.3.4 Deterministic Testing

The deterministic test stimulus can be used in a BIST environment. An ATE can be used to apply the test stimulus and the test response of the circuit is compared to the correct test response stored in a ROM memory. It is also possible to store the test stimulus in a ROM memory and using the ATE to compare the produced test response with the expected response.

2.4 Test Response Analysis for BIST

The test response data must be compressed in order to handle it within the system. Several compression approaches have been proposed, including [151]

- transition count,
- syndrome checking and
- signature analysis.

The *transition count* technique counts and summarizes for the test response all transitions from zero to one (0->1) and one to zero (1->0) [89]. The advantage is that only the number of transitions are stored, and compared with the correct and expected transition count. In Figure 30 the transition count is shown for some test stimulus. In the presence of a fault in the circuit, the transition count will be different from expected. The advantage of transition count is that only the number of transitions are stored, which reduces the storage requirement for the test response. A disadvantage is that there is a

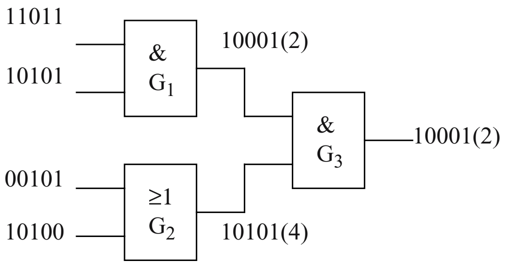


*Figure 30.*An example circuit.

possibility of fault masking error. Several different sequences results in the same transition count and therefore it is possible that the transition count of a faulty circuit can be the same as that of a fault-free one. However, as the sequences increases, this problem is reduced [151].

In *syndrome checking*, a function defined as $S=K/2^n$, where K in the number of realized min-terms by the function and n is the number of inputs [234]. The syndrome of a three-input AND gate is 1/8 - the function is true in one out of eight cases. The syndrome for a two-input OR-gate is 3/4 - there are three “true” cases out of the four possible outputs. A set of syndrome relations exists for each gate type. For an AND-gate the syndrome relation is S_1S_2 - the single min-term. And for an OR-gate the syndrome relation is $S_1+S_2-S_1S_2$. For the example circuit in Figure 31, the syndrome $S_3=S_1S_2$ (the syndrome of an AND-gate), and the syndromes of $S_1=1/4$ and $S_2=3/4$ (given by $S=K/2^n$), making $S_3=1/4*3/4=3/16$.

Signature analysis is a technique to compress the test responses into a single signature [94]. Figure 32 shows the architecture for a signature analyzer. The test response is XOR-ed with selected bits in the n-bit shift register to form the current content of the n-bit shift register. At the end of the testing, the shift register contain the signature, which is compared to the expected signature.

An n-bit generator can create 2^n signatures; however, input sequences may map to the same signature. In general, if the length of the input is m and the

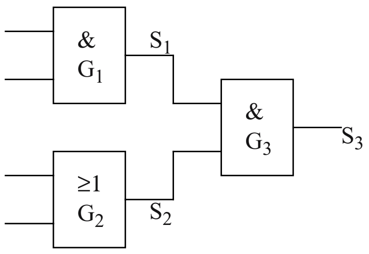


*Figure 31.*An example circuit.

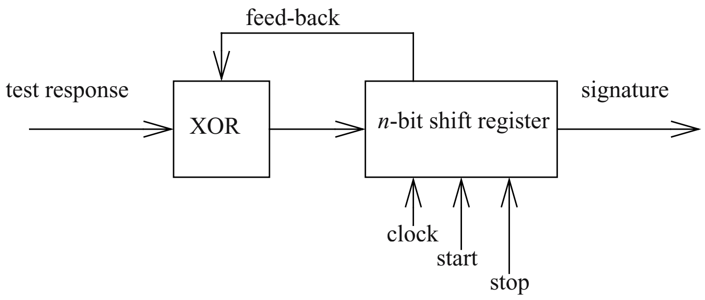


Figure 32. Signature analyzer architecture.

signature register has n stages, 2^m input sequences map into 2^n signatures. It means that 2^{m-n} input sequences map into each correct signature. Only one signature out of 2^m possible sequences is the correct signature. However, it can be so that some of the signatures map to the correct signature and, hence, even if the signature is correct, the circuit is faulty. This problem is called *aliasing* and the probability of aliasing is:

$$P = \frac{2^{m-n} - 1}{2^m - 1} \tag{3.8}$$

If only on input sequence may be good or faulty, the expression 3.1 can be reduced to:

$$P = \frac{1}{2^n} \qquad \text{for } m \gg n \tag{3.9}$$

The probability of aliasing is low if the signature generator has many stages and can generate a high number of signatures.

2.5 Circular-BIST

The circular BIST technique makes use of the existing flip-flops in the circuit to implement the test pattern generator and the signature generator [144]. The advantage of the technique is that the additional logic is reduced; however, it might be complicated to create certain test patterns for some faults in the design when the actual design itself is used as test pattern generator.

2.6 BIST-Architectures

2.6.1 BILBO (Built-In Logic Block Observer),

A BILBO can operate as both a test pattern generator and a signature analyzer [136]. Figure 33 shows a design with two logic blocks, block A and block B. Testing Block A requires that BILBO 1 operates as test pattern generator while BILBO 2 serves as the signature analyzer. Clearly, the architecture in Figure 33 prevents concurrent testing of block A and block B, since BILBO 2 serves as a signature generator when testing block A and can not concurrently produce test patterns required for the testing of Block B. BILBO 2 is a conflicting resource in this example.

Since there is no loading/unloading as in the case when using scan chains, a test can be applied in each clock cycle, and such a scheme is called *test-per-clock*.

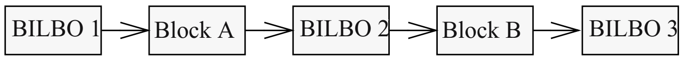


*Figure 33.*A BILBO architecture.

2.6.2 STUMPS Architecture

In the STUMPS (Self-Testing Using an MISR and Parallel Shift Register Sequence Generator) architecture, the pseudo-random pattern generator feed test stimulus to the scan chains and after a capture cycle the signature generator receives the test responses. The pseudo-random generator loads test stimulus until all scan-chains are filled and after the application of a system clock cycle, a capture cycle, the test response captured in the scan-chains is shifted out into the signature generator (Figure 34)[8]. The pseudo-random pattern generator can be implemented as an LFSR and a MISR can be used as the signature generator. A test scheme where new test stimulus is shifted into the scan chains, and after a capture cycle, the test response is shifted out while the following test stimulus is shifted in, is called t*est-per-scan*, *i.e.* one test is performed when all scan-chains are loaded/unloaded. Long scan chain lead to long testing times. In order to reduce the test application time, a higher number of scan chains can be used, which reduces the loading/unloading time since it is performed concurrent over all scan chains. In a *test-per-clock* scheme there is one test per clock cycle.

2.6.3 LOCST (LSSD On-Chip Self-Test)

In the LOCST (LSSD On-Chip Self-Test) architecture the test data is transported to the circuit under test using a boundary scan chain [179]. Figure 35 shows an example where the test stimulus is transported from the pseudo-random pattern generator via the Boundary chain to the circuit under test. The test stimulus is loaded into the scan path (the scan-chains in the system connected into a single chain). After a capture cycle, the test response is shifted out to the signature generator using the Boundary scan chain.

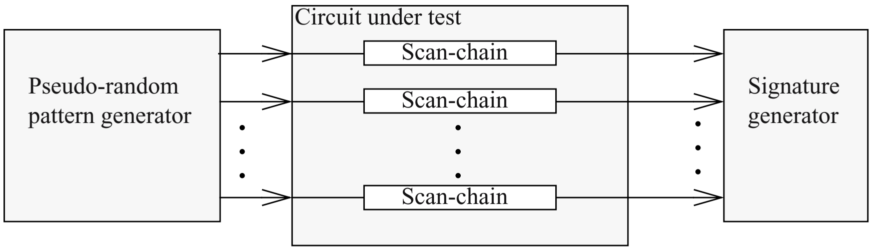


*Figure 34.*The STUMPS (Self-Testing Using an MISR and Parallel Shift Register Sequence Generator) architecture.

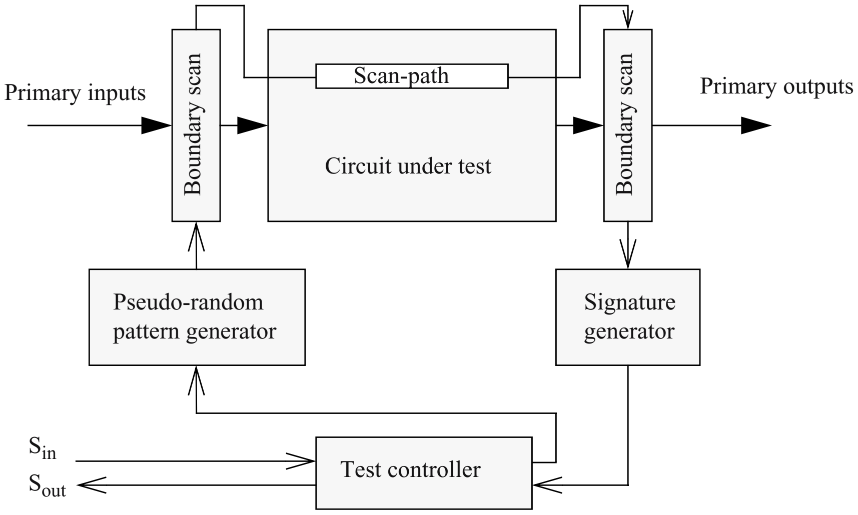


*Figure 35.*The LSSD On-Chip Self-Test (LOCST).

2.7 Memory Testing

The number of bits/chip quadruples approximately every 3.1 (π) year [24]. Figure 36 shows the number of bits (memory density) over time. A major part in terms on silicon area is usually devoted to memories; hence, it is important to discuss problems and techniques related to the testing of memories. However, the increasing memory density enforces limitations on test algorithms. Table 4 shows the test time in terms of memory size (n is the number of memory bits). As a result, the test generation techniques cannot of order $O(n^2)$ or higher [82].

The fault models in memory testing are stuck-at-fault, transition fault, coupling fault (inversion coupling fault, idempotent coupling fault, dynamic

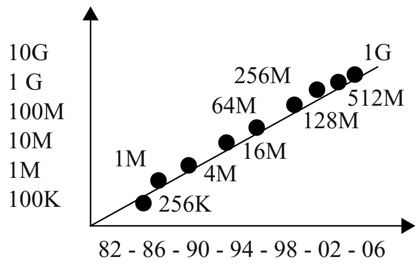


*Figure 36.*Number of bits per DRAM chip in volume production [24].

Table 4. Test time in terms of memory size *n* [24].

	Number of test algorithm operations			
n	*n*	$n \times log_2 n$	$n^{3/2}$	n^2
1 Mb	0.063 s	1.26 s	64.5 s	18.33 hr.
4 MB	0.252	5.54 s	515.4 s	293.2 hr.
16 MB	1.01	24.16 s	1.15 hr.	4691.3 hr.
64 MB	4.03	104.7 s	9.17 hr.	75060 hr.
256 MB	16.11	451.0 s	73.30 hr.	1200959.9 hr.
1 GB	64.43	1932.8 s	586.41 hr.	19215358.4 hr.
2 GB	128.9	3994.4 s	1658.61 s	76861433.7 hr.

coupling fault, bridging fault, state coupling fault, neighborhood pattern sensitive coupling fault (active, passive, and static)), address decoder fault, read/ write logic fault.

The stuck-at-fault (SAF) is when the logic of a cell or line is always 0 (SA0) or always 1 (SA1) and cannot be changed. The transition fault is when a cell fails to make a transition from 0 to 1 or from 1 to 0. The difference between a stuck-at-fault and a transition fault is illustrated in Figure 37.

A coupling fault means that a transition in memory bit *j* results in an unwanted change in memory bit *i*. An address decoder fault is a fault where the decoder logic does not become sequential.

There are a number of test algorithms for memory testing (RAM). The complexity of the algorithms vary between $O(n)$ to $O(n^3)$ where *n* is the number of bits in the RAM. In the MSCAN algorithm, a sequence or pattern consisting of all-0's (all-1's) is written and then read from each location (Figure 38). The complexity of the algorithm is O(4×*n*) where *n* is the size of the memory.

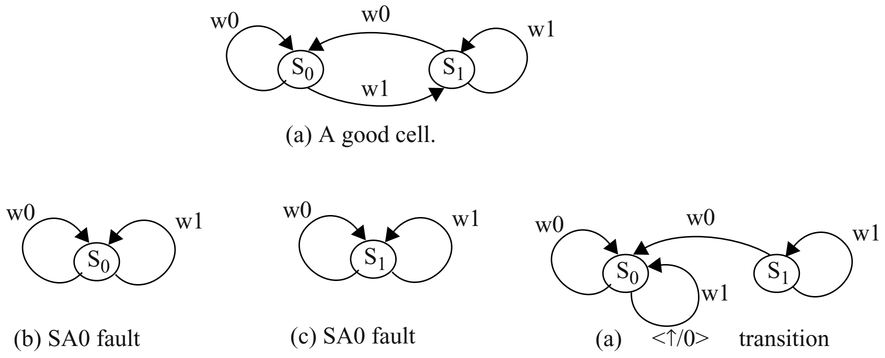


*Figure 37.*State transition diagram model for stuck-at and transition faults.

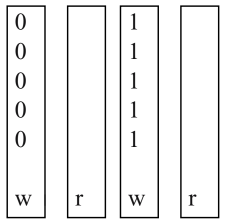


*Figure 38.*MSCAN.

The GALPAT (GALloping PATtern - Galloping 0s and 1s), or walking 1(0) or Ping-Pong test [22]. GALDIA, with complexity $O(2\times n^2)$, same as GALPAT, avoids that an entire row, column, or diagonal can hold 1s. A similar approach as GALPAT and GALDIA is “walking 0s and 1s”, $O(2\times n^2)$, where only a single memory location is assigned to 1 and all locations are checked that they are 0s [22]. In Algorithmic Test Sequence, complexity $O(4\times n)$ [133], patterns are applied to three partitions of the memory (each partition corresponds to the address modulo 3). The algorithm has been improved to the MATS [206] and the MATS+ [1] algorithms. Marching Pattern Sequences, March test, is a version of the walking 1/0 [205]. Several versions of March technique exists at different complexities $O(k\times n)$ where k is a constant. In Checkboard test, complexity $O(n)$, partition the memory into two parts and sets cells into 1 and all neighbors to 0, and all neighbors to a 0 is set to 1. All cells in partition 1 are set to 1 and all cells in partition 0 are set to 0. All cells are read, and then all cells in partition 1 are set to 0 and all cells in partition 0 are set to 1. And finally, all cells are read. More algorithms on memory testing can be found in [247, 82].

The regular structure of memories makes them easily testable and memories are often tested with BIST. However, the high density (memory capacity on a given silicon area) makes memories sensitive to faults.

2.7.1 Algorithmic Test Sequence (ATS)

In the Algorithmic Test Sequence algorithm the test stimulus is applied to three partitions of the memory array where each partition corresponds to addresses mod 3 [133]. It means that the cells of a partition does not occupy adjacent locations, the three partitions are Π_1, Π_2, and Π_3. The algorithm is illustrated in Figure 39 and its complexity is of $O(4 \times n)$. The algorithm detects stuck-at faults on the memory cells, decoders, memory data registers, and memory address registers. The algorithms has been improved to the MATS [206] and the MATS+ [1] algorithms.

2.7.2 Marching Pattern Sequences (MARCH)

The GALPAT (GALloping PATtern - Galloping 0s and 1s), or walking 1(0) or Ping-Pong test [22] is a special version of March test [82]. In March testing, after initialization, each cell is read and then complemented (written with the reverse value as it should contain). The process is repeated in reverse order when the memory array has been traversed. Figure 40 shows a March algorithm that requires 11 read/write operations, leading to a testing time of $O(11 \times n)$.

```
1. write 0 in all cells of Π_1 and Π_2.
2. write 1 in all cells of Π_0.
3. read 1 from Π_1            if 0, then no fault, else faulty
4. write 1 in all cells of Π_1
5. read 1 from Π_2            if 1, then no fault, else faulty
6. read 1 from Π_0 and Π_1    if 1, then no fault, else faulty
7. write 0 in all cells of Π_0
8. read 0 from Π_0            if 0, then no fault, else faulty
9. write 1 in all cells of Π_2
10. read 1 from Π_2           if 1, then no fault, else faulty
```

Figure 39. The algorithmic test sequence [133].

```
For all addresses j=1 to N
  write 0 in all c_j
  read c_j
  write 1 in c_j
  read c_j
  write 0
For all addresses j=N to 1
  read c_j
  write 1 in c_j
  read c_j
  write 0 in c_j
  read c_j
```

Figure 40. MARCH test algorithm.

2.7.3 Checkboard Test

The memory cells are set to 0 and 1 so that each cell is different from its neighbor (see Figure 41(a)). The bits (memory cells) are partitioned into partition Π_0 - all bits set to 0, and Π_1- all bits set to 1. The algorithms reads and writes one partition, and then the other (see Figure 41(b)). The algorithm requires 4×n read/write and its complexity is $O(n)$. The test detects all stuck-at-faults, data retention faults, and 50% of the transitions faults.

2.7.4 Memory BIST

Embedded memories are suitable for BIST testing due to the regular structure, the low sequential depth, and accessible I/Os. Furthermore, Memory BIST would be an advantage since memory testing requires a high amount of test data (test stimulus and test response). And on-chip test generation using Memory BIST would also lead to shorter testing times since the testing can be performed at a higher clock frequency. And finally, with Memory BIST per memory block means that the blocks can be tested concurrently; hence, further reduction of testing time.

Figure 42 shows a memory cell. It has an address register to address a memory cell and an input data register for writing in the addressed memory cell. There is also an output data register where the output (the contents of an addressed memory cell) from a read operation is placed.

Figure 43 shows a memory cell tested with the scan technique. The registers (address, input data, and output data) are included in the scan path. Small memories can be tested using the scan technique. However, as the size of the memories increases, the test data increases, leading to ATE limitations and long testing times (high test cost).

Figure 44 shows an example of a Memory BIST tested embedded RAM. The address register and the input data register from Figure 42 have been modified. The address register or stepper can be an LFSR or a binary counter. The output data register is modified to include a compactor and a comparator. The expected test response is compared with the actual produced test response

```
1 0 1 0
0 1 0 1
1 0 1 0
0 1 0 1
```

(b)

1. write 0 in cells of Π_0
2. write 1 in cells of Π_1
3. read all cells
1. write 1in cells of Π_0
2. write 0in cells of Π_1
3. read all cells

(b)

*Figure 41.*Checkerboard test.

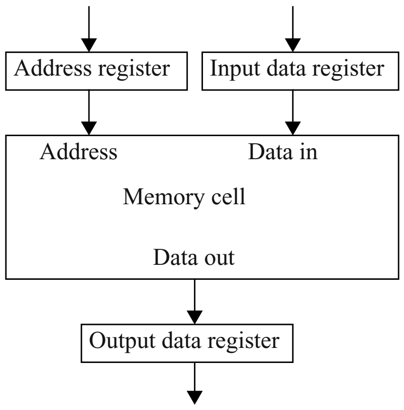


Figure 42. Embedded RAM.

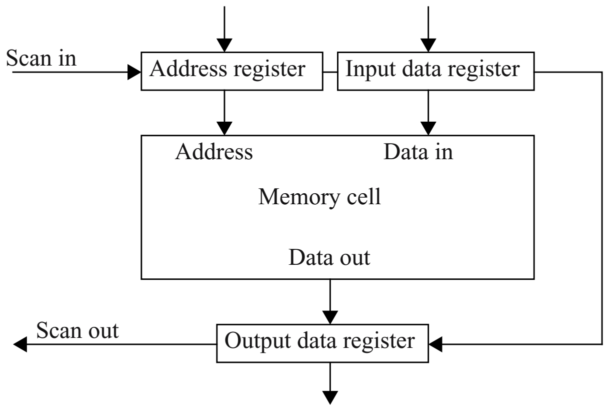


Figure 43. Scan tested embedded RAM.

from the memory cell. At the end of the testing, a go/no-go (good/no-good) value is produced.

The BIST circuits can be shared among multiple RAM blocks in order to save silicon area. The organization of the testing can be either daisy chaining or test multiplexing. However, sharing enforces constraints on the test scheduling. It is therefore important to consider the gain in saving silicon area versus the flexibility of parallel testing (reduced testing time).

2.7.5 Memory Diagnosis and Repair

The regular structure of memory arrays has made it common practice to include spare rows and columns to replace possible faulty ones. It means that testing should not only detect if a memory is faulty, it should also detect where the fault appear (diagnosis) in order to repair it. In order to reduce the

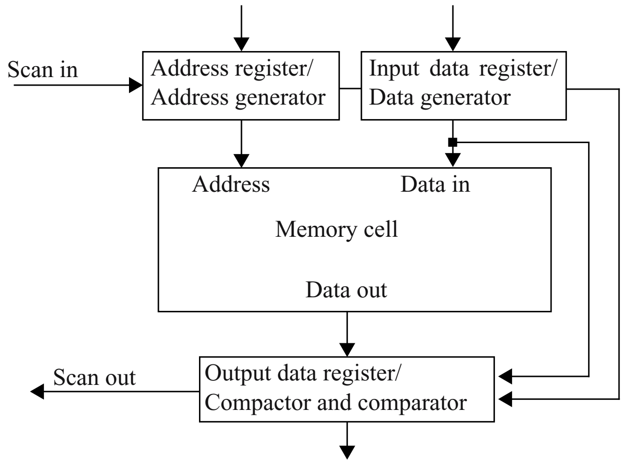


*Figure 44.*Memory BIST tested embedded RAM.

amount of test data, this (diagnosis and repair) should be embedded within the system.

3 MIXED-SIGNAL TESTING

Although only a minor part, in terms of silicon area, of the SOC is occupied by analog blocks such as analog switches, analog-to-digital (A/D) converters, comparators, digital-to-analog (D/A) converters, operational amplifiers, precision voltage references, phase-locked loops (PLL), resistors, capacitors, inductors, and transformers, these (analog blocks) must be treated with special care. Analog blocks are in general more difficult to test than digital blocks mainly due to the lack of well-accepted fault models, and the continuos range of analog circuit parameters [24]. The cost of testing analog blocks is about 30% of the total test cost [24]. Analog testing is different from digital testing. In digital parts of the system only two values are possible (0 or 1). In analog parts, continuos signals are allowed. Analog circuits are tested for their specification where the specification is expressed in terms of functional parameters, voltage, frequency, etc.

Vinnakota [269] compare analog testing with digital testing and points out (in short):

- size - analog blocks are relatively few compared to digital blocks.
- modeling - is more complicated for analog parts than for digital parts.
    - there is no widely accepted analog fault model (as stuck-at).
    - a continuos signal is to be handled compared to a discrete (0 and 1).
    - tolerance, variation, and measurement are more complicated.
    - noise has to be modeled.
- partitioning - a digital design which is partitioned into sub-components (cores) can usually be tested in parallel, not one core at a time.
- test access - the problem of sending test stimulus from primary inputs to an embedded analog block, and receiving the test response from an embedded analog block at primary outputs.
- test methods - the lack of an analog fault model makes test generation problematic.

Chapter 4

Boundary Scan

1 INTRODUCTION

The Boundary-scan test (IEEE 1149.1) standard initially developed for Printed Circuit Board (PCB) designs to ease access of the components. The common practice through the early 1980s was to use a *bed-of-nails* to access the unit under test where the nails are the probes, the connection to control and observe. For modern PCB the actual size has decreases, making it difficult for physically access. Also, the use of multiple layers of a board makes test data access complicated.

2 THE BOUNDARY-SCAN STANDARDS (IEEE 1149.1)

The main objective in PCB testing is to ensure a proper mounting of components and correct interconnections between components. One way to achieve this objective is to add shift registers next to each input/output (I/O) pin of the component to ease test access. The IEEE 1149.1 standard for the Standard Test Access Port and Boundary-scan Architecture deals primarily with the use of an on-board test bus and the protocol associated with it. It includes elements for controlling the bus, I/O ports for connecting the chip with the bus and some on-chip control logic to interface the test bus with the DFT hardware of the chip [2]. In addition, the IEEE 1149.1 standard requires Boundary-scan registers on the chip.

A general form of a chip with support for 1149.1 is shown in Figure 45 with the basic hardware elements: *test access port (TAP)*, *TAP controller*, *instruction register (IR)*, and a group of *test data registers (TDRs)* [19]. And a system with Boundary Scan is in Figure 46.

The TAP provides access to many of the test support functions built into a component. The TAP consists of four inputs of which one is optional and a single output: the *test clock input* (TCK) which allows the Boundary-scan part

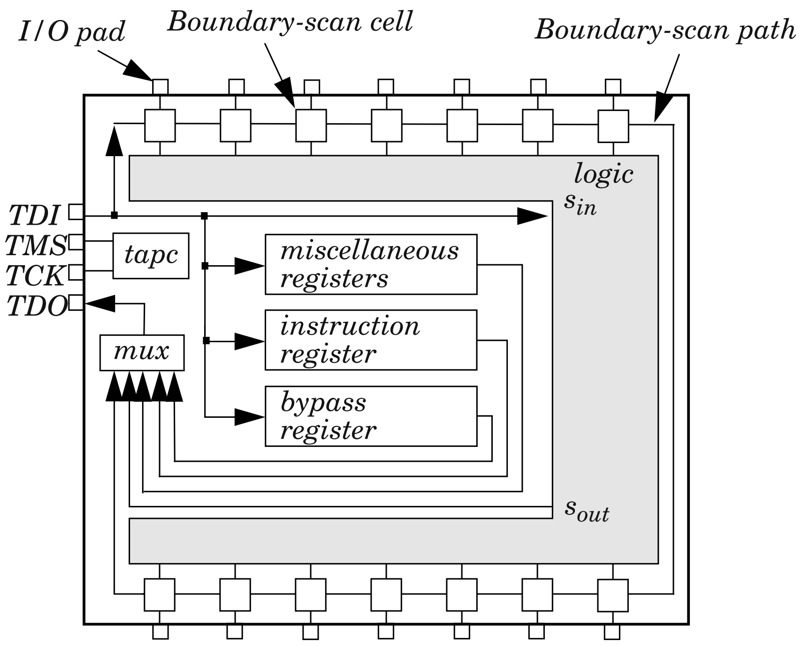


*Figure 45.*An example of chip architecture for IEEE 1149.1.

of the component to operate synchronously and independently of the built-in system clock; the *test mode select input* (TMS) is interpreted by the TAP Controller to control the test operations; the *test data input* (TDI) feeds the instruction register or the test data registers serially with data depending on the TAP controller; the *test reset input* (TRST) is an optional input which is used to force the controller logic to the reset state independently of TCK and TMS signals; and the *test data output* (TDO). Depending on the state of the TAP controller, the contents of either the instruction register or a data register is serially shifted out on TDO.

The TAP controller, named *tapc* in Figure 45, is a synchronous finite-state machine which generates clock and control signals for the instruction register and the test data registers. The test instructions are shifted into the instruction register. A set of mandatory and optional instructions are defined by the IEEE 1149.1 standard in order to define what operation to be performed. It is possible to add design-specific instructions when the component is designed.

The Boundary-scan Architecture contains at a minimum two test data registers: the Bypass Register and the Boundary-scan Register. The advantage of the mandatory bypass register, implemented as a single stage shift-register, is to shorten the serial path for shifting test data from the component's TDI to its TDO [19]. The Boundary-scan register of a component consists of series of

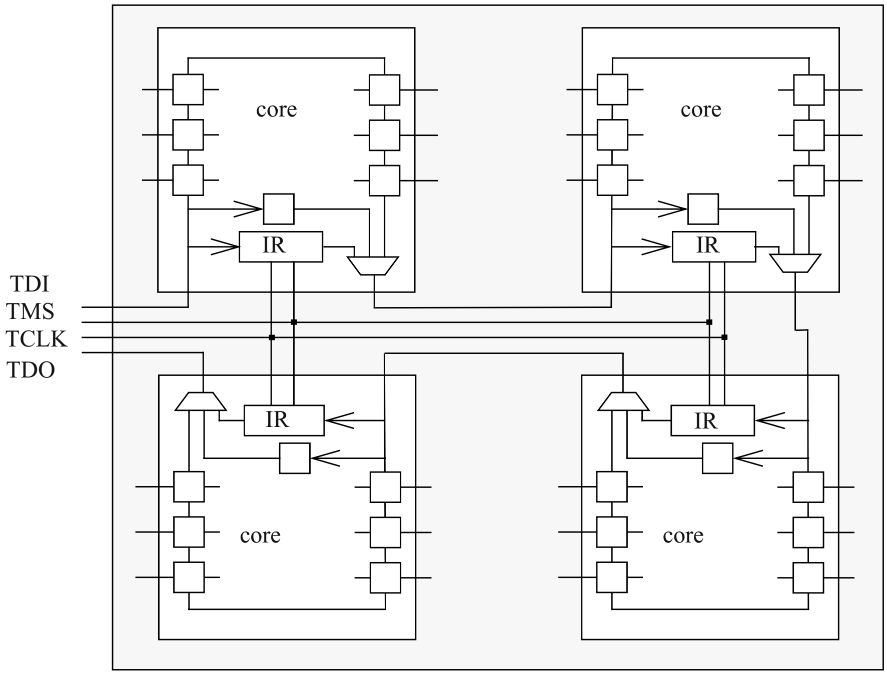


*Figure 46.*Boundary scan system view.

Boundary-scan cells arranged to form a scan path around the core, see Figure 45 [19].

A Boundary Scan cell can through the control signals (shift/load and test/normal) be in different modes. A Boundary scan cell as in Figure 47 can be seen as in Figure 48(a). Its basic modes are illustrated in Figure 48(b) - shift mode, Figure 48(c) - extest mode, and Figure 48(d) - intest mode.

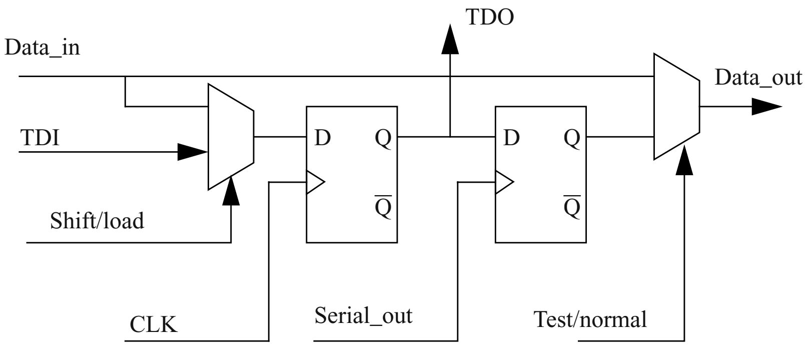


*Figure 47.*Basic Boundary scan cell.

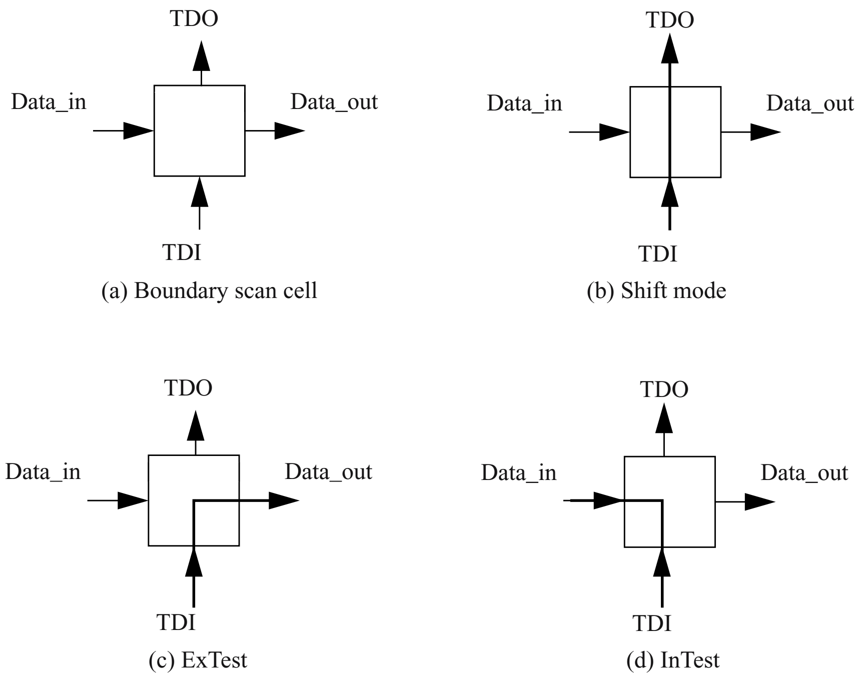


*Figure 48.*Basic Boundary scan cell.

The Bypass register (Figure 45) is a single-stage register used to bypass data directly from the TDI to the TDO of a component (module, core). The advantage is that instead of having to shift (clock) data through all the Boundary Cells in the Boundary Scan ring at a component, a single clock cycle is needed (Figure 45).

2.1 Registers

A Boundary Scan register at a component is all the Boundary Scan Cells (Figure 47) connected in a single register (similar as a scan chain) (Figure 45). The Boundary Scan Cells in the Boundary Scan register are controlled through the Instruction Register. Instructions are serially shifted into the Instruction Register. An output latch holds current instruction until a new is shifted in and an Update-IR is made (Up-date IR state in Figure 49).

The Device Identification Register is an optional 32-bit long register used to contain an identification number defined by the manufacturer. The idea is that the Device Identification Register is used to verify that correct component of the correct version is mounted at a correct position in the system.

2.2 TAP Controller

The TAP controller, trigged by the TMS signal and the positive edge of CLK (the test clock), is responsible of:

- loading instructions to IR,
- producing control signals to load and shift test data into TDI and out of TDO, and
- performing test actions such as capture, shift, and update test data.

The TAP controller is a 16-state finite state machine (Figure 49). The Test-Logic-Reset is the reset state. The controller remains in Test-Logic_Reset while TMS=1 and the system is in normal mode. The Test-Logic-Reset state is reached in five clock cycles from any state in the controller.

The testing starts from state Run-Test/Idle by loading an instruction into the instruction register (IR). It is achieved by keeping TMS=1 for two cycles to reach the state Select-IR-Scan. TDI and TDO are connected to IR. After TMS=0 the instruction is captured (state Capture-IR), the required number of shifts are applied (state Shift-IR).

2.3 Instructions

The mandatory instructions are Bypass, Extest, and Sample/Preload. Instructions as Idcode, Intest, and Runbist are the most frequently used optional instructions. The Bypass instruction is used to bypass data between TDI and TDO at a component. The Extest instruction is used to test interconnection and logic between components equipped with Boundary Scan. And the Sample/Preload instruction is used to scan the Boundary Scan Register without interrupting the normal operation. The instruction is useful for debugging purposes. The Idcode, Intest, Runbist, Clamp, Highhz instructions are all optional.

2.3.1 Example

An example to illustrate testing is in Figure 50. First, Figure 50(a), test data is shifted in, and when completed, the test data is applied to the logic under test (Figure 50(b)). Finally, the captured test response is shifted out Figure 50(c). A more detailed view on a Boundry Scan cell of the example is in Figure 51. In Figure 51(a) the test data is shifted in, and in Figure 51(b) the test data is produced at the outputs. In Figure 51(c) the test response is captured, and finally it is shifted out.

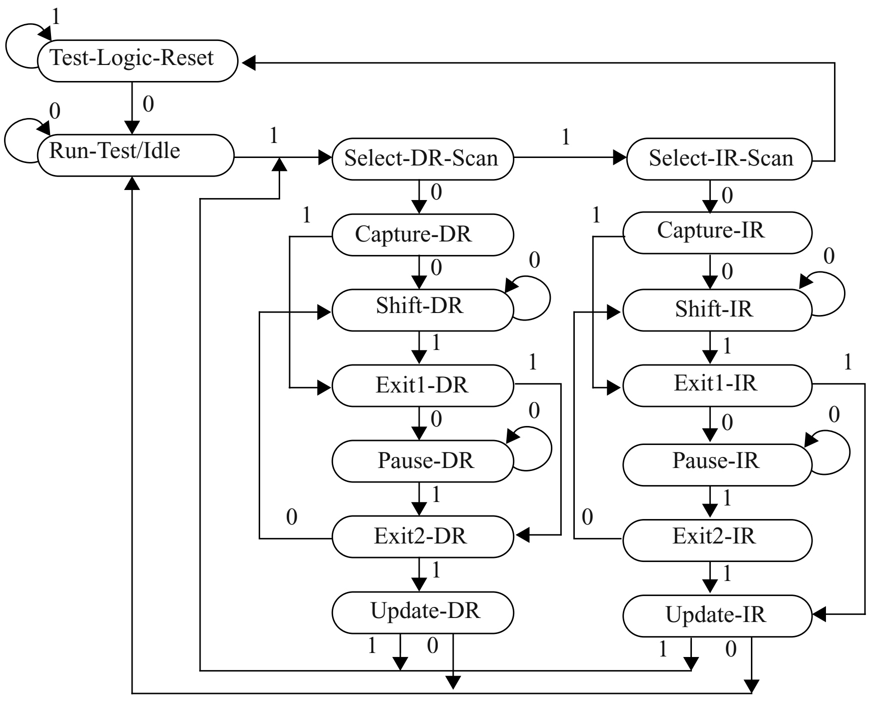


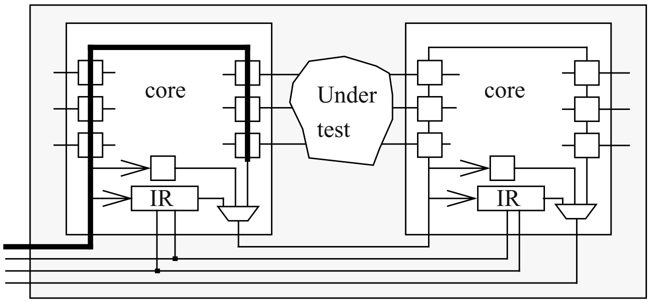
*Figure 49.*The TAP (test access port) controller.

2.3.2 Boundary-Scan Languages

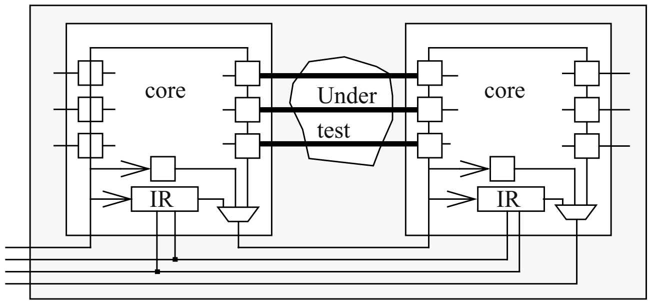
The Boundary-Scan Description Language (BSDL) is a subset of VHDL developed to connect the system (pins, clocks etc) with the test data (test stimuli and test responses) [214, 215].

2.3.3 Cost of Boundary Scan

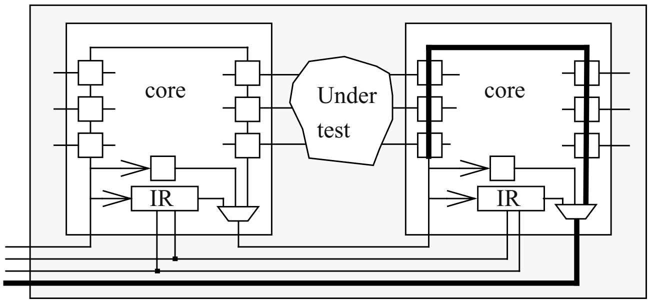
The advantage of Boundary Scan is that access to all components (modules/cores) in the system (PCB/SOC) is provided. And, adding Boundary Scan, as with the scan technique, is fairly much plug'n play, which is a major advantage when used in a design flow. Boundary Scan can be used for testing purpose, as well as for debugging. The cost of additional silicon and extra pins is relative to the system cost decreasing since the silicon cost due to Boundary Scan is relatively constant while the size of the systems is increasing. Boundary Scan introduces additional delays due to multiplexors (similar cost as with the scan technique). However, the cost is fixed, which makes it possible to predict it early in a design flow. The major disadvantage with



(a) Shift in stimulus - shift-DR

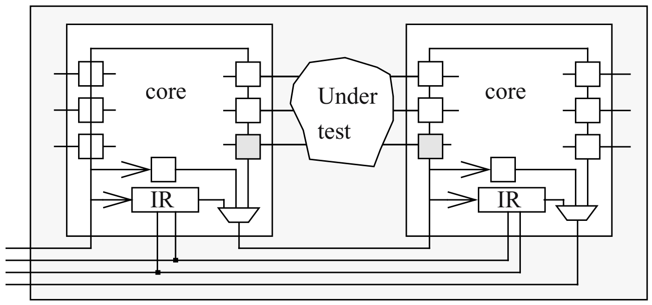


(b) Capture data - capture-DR.

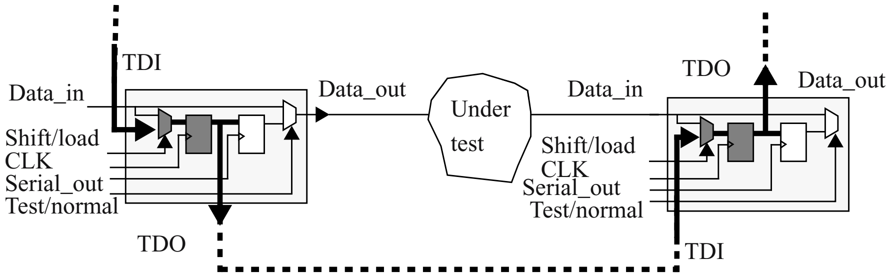


(c) Shift out response - shift-DR.

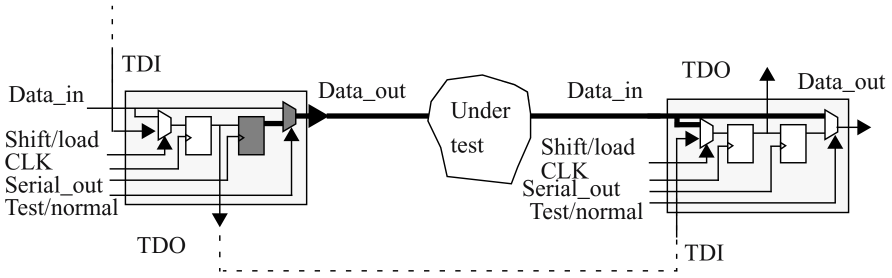
*Figure 50.*External testing of “under test” logic/interconnection.



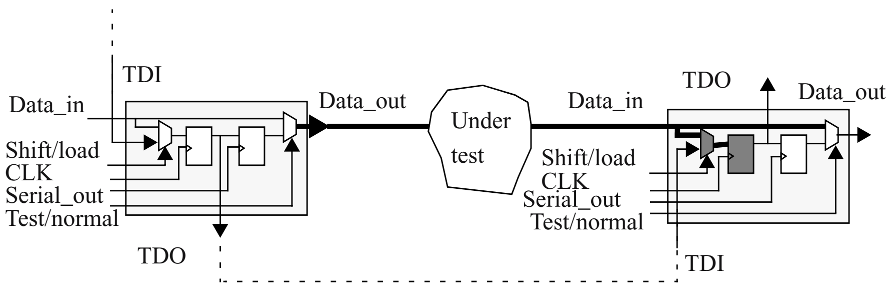
(a) System



(b) Shift in stimulus - shift-DR



(c) Capture data at the outputs - capture-DR.



(d) Capture the response data - capture-DR.

*Figure 51.*External testing of “under test” logic/interconnection.

Boundary Scan in larger SOC designs, where not only the interconnections are to be tested but also the cores (modules, components) themselves, is the long test application time due the high amount of test data that has to be transported into and out of the system using a single Boundary Scan chain.

3 ANALOG TEST BUS (IEEE 1149.4)

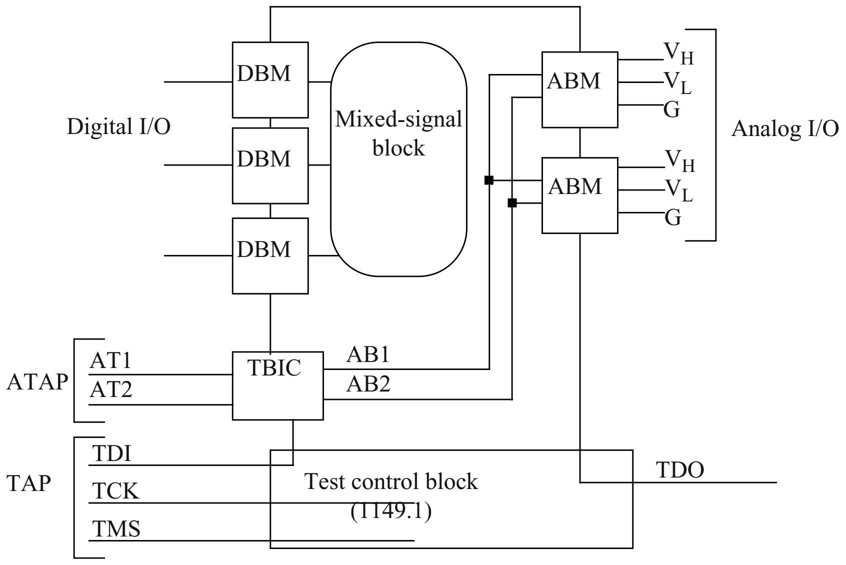
The analog test bus (ATB) (IEEE Standard 1149.4) is an extension of IEEE Standard 1149.1) intended to handle mixed-signal devices (see Figure 52) [216]. The advantage of the bus is that by using the bus, observability of analog circuits is achieved. Hence, it replaces the in-circuit tester. The bus also eliminates large analog chip area needed for extra test points. Therefore, it reduces various external impedances connected to the pins of the chip. Among the disadvantages are the measurement error (about 5%), the capacitance of all probing points, and the bus length limitations. To put it simple, the 1149.4 works as 1149.1 but instead of only being able of handling discrete values (0s and 1s), 1149.4 can handle analog continuos values.

Figure 53 shows the types of analog faults that must be tested for. The faults include opens and shorts in the interconnects (often due to solder problems). Shorts may appear between any wires; hence a short can be between two analog wires or between an analog wire and a digital wire. The aim of 1194.4 is to provide capability to find these types of faults. It means it will give test access in order to eliminate bed-of-nails testers.

The 1149.4 allow interconnection testing and the standard defines three types of interconnections:

- *simple interconnect* - wires,
- *differential interconnect* - a pair of wires transmit a signal - , and,
- *extended interconnect* - where components such a capacitors are inserted on the wires.

The aim of the analog test bus is to provide access to embedded analog block. The analog bus will not replace analog ATE; similar to that digital Boundary scan will not replace ATE. The advantage with ATB (1149.4) is that it allows testing of embedded analog blocks. The disadvantage is that the analog test bus might result inaccurate measurements. It has been shown that testing for extremely high frequencies, small amplitudes, or high precision components can be difficult.



DBM: Digital Boundary Module	AT1: Analog Test Bus 1
ABM: Analog boundary Module	AT2: Analog Test Bus 2
TAP: Test Access Port	AB1: Analog Measurement Bus 1
ATAP: Analog Test Access Port	AB2: Analog Measurement Bus 2
TBIC: Test Bus Interface Circuit	
TDI: Test Data Input	
TDO: Test Data Output	
TCK: Test Clock	
TMS: Test Mode Select	

*Figure 52.*The analog test bus architecture.

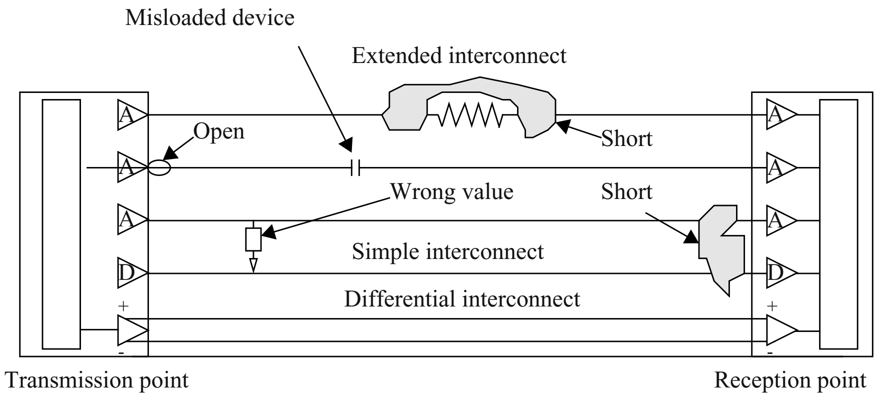


*Figure 53.*Defects in a mixed-signal circuit [216].

3.1 Analog Test Access Port (ATAP)

The 1149.4 standard is an extension of the 1149.1 standard. It means that the analog test access port (ATAP) include the four mandatory signals and one optional signal from 1149.1 (TDI, TDO, TCK, TMS, and TRST (The Boundary-Scan Standards (IEEE 1149.1) on page 53)). The 1149.4 standard adds two mandatory analog signals, AT1 and AT2. The analog test stimulus is usually sent from the ATE to AT1, while the analog test response is sent from AT2 to the ATE.

3.2 Test Bus Interface Circuit (TBIC)

The Test Bus Interface Circuit (TBIC) can [24]:

- connect or isolate the internal analog measurement buses (AB1 and AB2) to the external analog buses (AT1 and AT2),
- perform digital interconnection test on AT1 and AT2, and
- support characterization (to improve accuracy of analog measurements).

3.3 Analog Boundary Module (ABM)

The 1194.4 standard requires that every pin has a digital boundary module (DBM) (see Figure 47) or an analog boundary module (ABM). The ABM includes a one-bit digitizer that interprets the voltage on the I/O pin.

3.4 Instructions

The instructions Bypass, Preload, Idcode, and Usercode are identical in 1149.4 as in 1149.1. Other instructions, such as Extest, Clamp, Highz, clamp, Sample, are extended to handle analog signals. The Probe instruction is a mandatory instruction in 1149.4.

3.5 Chaining Example

Figure 54 shows an example where two analog blocks are chained to the analog test bus. The blocks are to be tested in a sequence, one after the other.

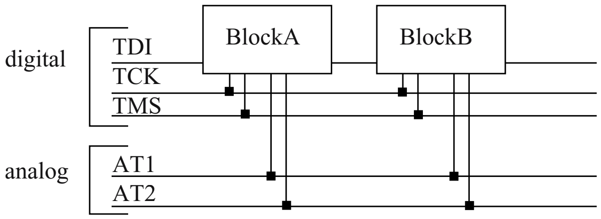


*Figure 54.*Chaining of analog blocks using 1149.4.

PART 2
SOC DESIGN FOR TESTABILITY

Chapter 5

System Modeling

1 INTRODUCTION

In this chapter we discuss modeling and concepts related to core-based system. The introduction of test methods (DFT techniques), the creation of test stimulus, and the test response analysis in Chapter *Design for Test* on *page 5* serves as the basis for this chapter. It is therefore known that a testable unit has its test stimulus and expected test response. The site where the test stimulus (test vectors, test patterns) is stored or created is called *test source* while a *test sink* is where the test response is stored or analyzed (Figure 55). Figure 55 shows a test architecture where test stimulus is stored at the test source and transported on the *test access mechanism* (TAM) to the testable unit, in this example a core. The test response is also transported on the TAM to the test sink. In order to ease test access between the core and the TAM the core is placed in a *wrapper*.

These concepts have been introduced by Zorian *et al.* [289]. We further illustrate them with the example in Figure 56. The example consists of three main blocks of logic, core A (CPU core), core B (DSP core), and core C (UDL (user-defined logic) block). A test source is where test stimulus is created or stored, and a test sink is where the test response from a test is stored or analyzed. Test resources (test source and test sink) can be placed on-chip or off-chip. In Figure 56 the ATE serves as off-chip test source and off-chip test sink, while TS1, for instance, is an on-chip test source. The TAM is the infrastructure for test stimulus transportation from a test source to the block to test and for test response transportation from a testable unit to a test sink. A wrapper is the interface between a core and the TAM. A core equipped with a wrapper is *wrapped* and a core without wrapper is *unwrapped*. If a core need test data from the TAM a wrapper is needed. The wrapper can be the cores

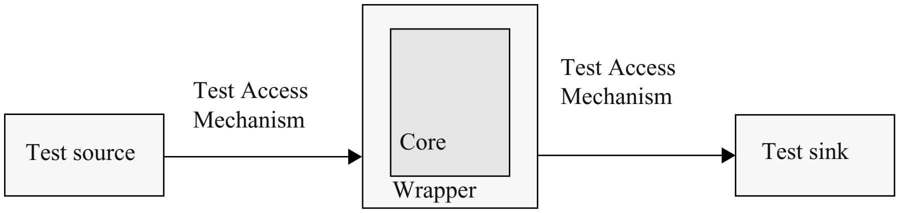


*Figure 55.*Test source, test access mechanism, wrapper, core (testable unit) and test sink.

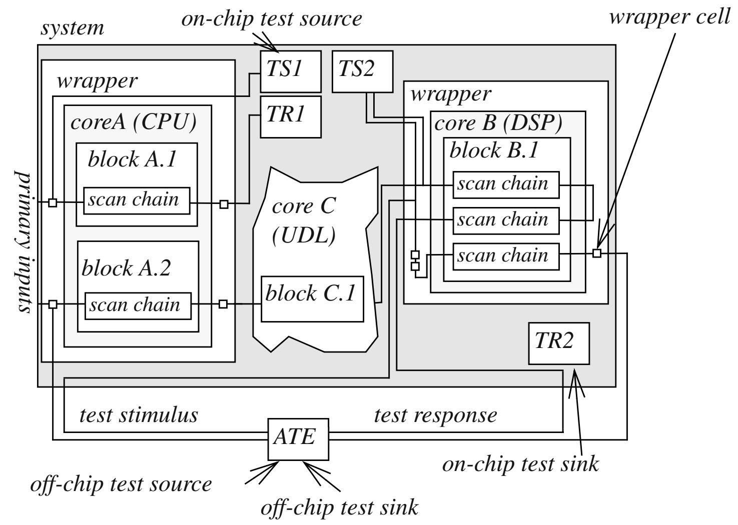


Figure 56. A system and the definition of test concepts.

dedicated wrapper or the core can make use of a wrapper at a neighboring core. For example, Core A is a wrapped core while Core C is an unwrapped core. The wrapper and the wrapper cells at the wrapper can be in one of the following modes at a time: *internal mode*, *external mode* and *normal operation mode*.

2 CORE MODELING

A core-based system is composed of a set of cores (Figure 57). The *cores* (also referred to as *modules*, *blocks*, or *macros*), may, for instance, be digital signal processing (DSP) processor, central processing unit (CPU), random access memory (RAM), field-programmable gate array (FPGA), and user-defined logic (UDL). There is no strict definition of what constitutes a core. For example, what amount of logic is required in order to call the partition of logic a core? For some parts in a core-based system, it is well-defined what constitutes a core while in other cases, such as for UDL blocks, it is less obvious. We simply say that a core is a well-partitioned piece of logic.

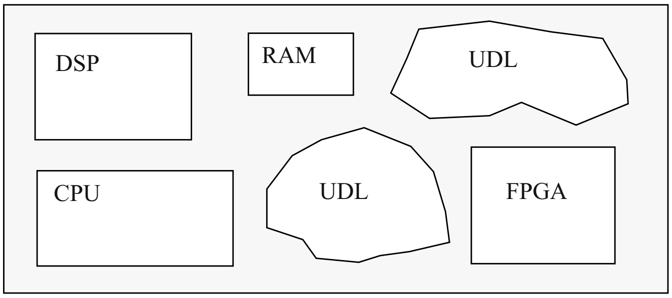


*Figure 57.*A core-based system.

The cores in a system can have different origin (core providers see *Design Flow* on page 5) and they can be classified into three categories [84, 199]:

- soft cores,
- firm cores, and
- hard cores.

Hard cores are given as layout files that cannot be modified. This type of cores are highly optimized for area and performance, and synthesized to a specific technology. And also the test sets (test stimuli and test responses) are given. Soft cores, on the other hand, are given in a high-level description (HDL) language, hence, technology independent, and the soft cores can easily be modified compared to hard cores. The soft core specification has to be synthesized and optimized, and also it is required to perform test generation. Firm cores are given as technology-dependent netlists using a library with cells that can be changed according to the core integrator's need.

Hard cores, obviously, give less flexibility to the core integrator but saves design time and effort since less synthesis and test generation are required. Soft cores give the core integrator high flexibility but they require design time for synthesis and test generation. Firm cores are somewhere between hard cores and soft cores. It means that firm cores give a little more flexibility than hard cores but not as much as soft cores, and firm cores require some more design time for optimization and test generation than hard cores but not as much as for soft cores (Figure 58).

The core-based system in Figure 57 is flat. The system consists of a set of cores where the cores are all on the same "level"; there is no hierarchy. However, a core can be embedded within a core. A parent core may have several child cores embedded inside of it. Furthermore, each child core can in turn be a parent core with embedded child cores (Figure 59). From a modeling perspective, embedding a core in core (parent-child) makes it a bit more

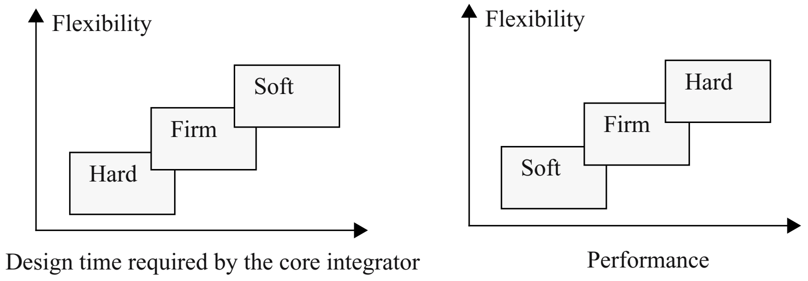


Figure 58. Trade-offs among core types.

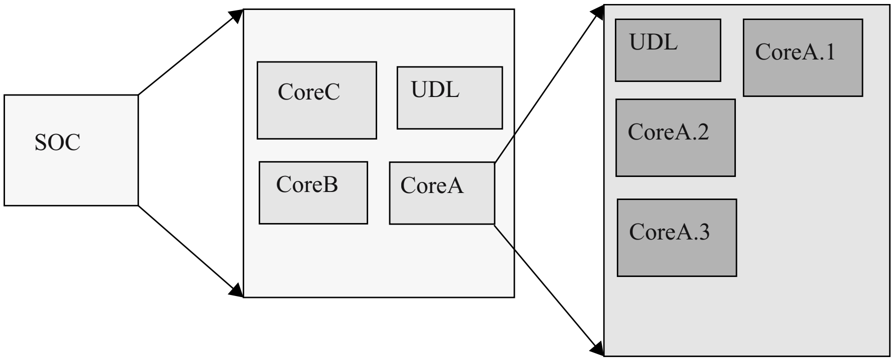


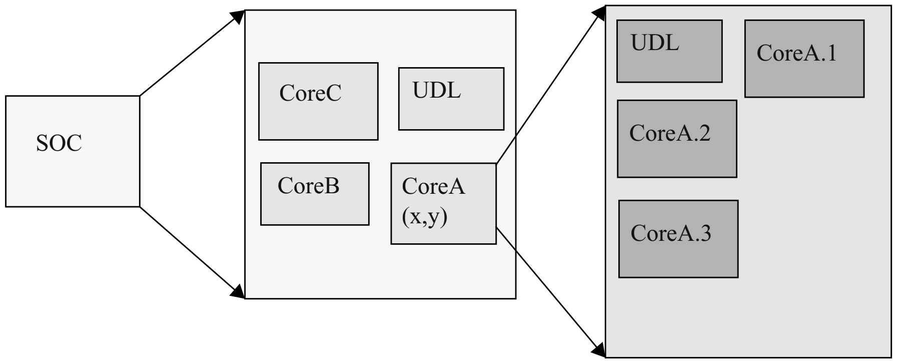
Figure 59. System-on-chip hierarchy.

complicated. How to organize the modeling? What is a testable unit? How will testing be performed when a child core is embedded in a parent core. How many testable units are there? CoreA in Figure 59 can be a single testable unit; however, coreA is a parent core with four child cores where each child core can be viewed as a testable unit.

Larsson *et al.* [158, 160, 164, 165, 166, 176] have taken the view that a core is composed of a set of blocks. The blocks are the testable units in the system and tests may, but do not have to, be defined for the blocks. Each core is given a position (x,y) in the system and the blocks are attached to a core.

Figure 60 shows an illustrative example. CoreA consists of four child cores (UDL, CoreA.1, CoreA.2, and CoreA.3). CoreA is placed at x,y-coordinate 10,10 and it consists of the four blocks (UDL, CoreA.1, CoreA.2, and CoreA.3). The tests at each block are then specified. For instance, block UDL.1 is tested with test.UDL1 and testUDL.2.

The core-block model by Larsson *et al.* [158, 160, 164, 165, 166, 176] could be more elaborate where each pin at a core would be specified with (x,y)-coordinates. It would also be possible to specify each pin at each block



(a)

```
#Syntax: name    x      y   {block1 block2 ... blockN}
       CORE.A    10    10 {UDL.1 CoreA.1 CoreA.2 CoreA.3}
       // for all cores
[Blocks]
#Syntax: name idle_power {test1 test2 ... testN}
          UDL.1  23         {test.UDL1   test.UDL2}
       //for all other blocks
```

(b)

*Figure 60.*System modeling Larsson *et al.* [176, 166, 164, 165, 160, 158].

in a core. However, such a fine-grain model would lead to additional computational cost.

The core-block model does actually also support parent-child core hierarchy. It is possible by specifying hierarchy using constraints (see *Hierarchy - Cores Embedded in Cores* on page 85).

3 TEST RESOURCE MODELING

A *test source* is where test stimulus is stored or created. It can be placed off-chip as an ATE or on-chip as an LFSR, embedded processor, memory etc. A *test sink* is where test response is stored or analyzed. It can, as a test source, be placed off-chip as an ATE or on-chip as a MISR or memory. In general, any combination is possible. For a test at a testable unit, both the test source and the test sink can be an ATE. Or, both the test source and the test sink can be placed on-chip. It is also possible to make use of an off-chip test source and an on-chip test sink, and vice versa.

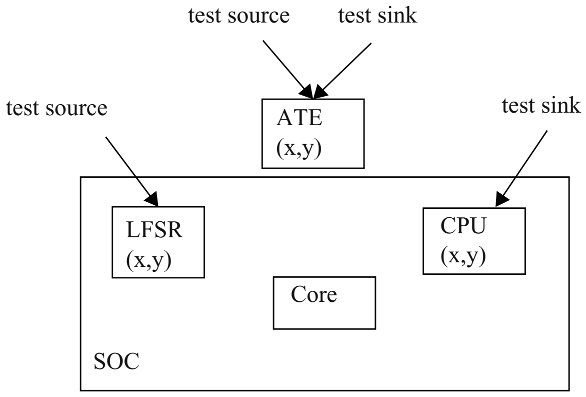


*Figure 61.*Test resources - test source and test sink.

Figure 61 shows an SOC example where an ATE is serving as an off-chip test source and an off-chip test sink, an LFSR operating as an on-chip test source, and an embedded processor (CPU) working as an on-chip test sink. Each of the tests of the core in Figure 61 has to make use of one test source and one test sink. For instance, one test can be defined so that is makes use of the ATE as test source and test sink. Another test can make use of the LFSR as test source and the CPU as test sink. It is also possible to allow a test to make use of the LFSR as the test source and the ATE as the test sink.

Larsson *et al.* [158, 160, 164, 165, 166,176] model, for instance, the test sources and the test sinks as in Figure 62. Each test source (Generator) and each test sink (Evaluator) is given a name and (x,y) co-ordinates. For instance as LFSR at (20,40). The use of test source (tpg) and test sink (tre) is specified for each test. The modeling is in more detailed discussed in Chapter Test Conflicts on page 77.

4 CORE WRAPPER

A core may be equipped with a wrapper such as P1500 [51, 104], Boundary Scan [2], TestCollar [267] and TestShell [186]. A wrapper serves as the interface between the core and the TAM. An advantage of wrapping a core is that a wrapper isolates the core during testing. A wrapper can in general be in the following modes of operation:

- normal mode,
- external test mode,
- internal core test mode

```
[Generators]
#Syntax: name    x      y
         ATE    150    50
         LFSR   20     40
[Evaluators]
#Syntax: name    x      y
         ATE    150    50
         CPU    145    50
[Tests]
#name     tpg    tre
 TestA    ATE    ATE
 TestB    LFSR   CPU
 TestC    LFSR   ATE
```

*Figure 62.*The modeling of test sources and test sinks.

For testing, the external test mode (extest) and internal core test mode (intest) are of most interest. In intest mode the core at current wrapper is tested while in extest the interconnection and logic placed between two wrappers are tested. Figure 63 shows an example with three cores, coreA, coreB, and coreC. CoreA and coreB are placed in wrapper so they have interfaces to the TAM. A core with an interface to the TAM is commonly called *wrapped*, while a core such as coreC does not have its dedicated interface to the TAM is called *unwrapped*. The testing of a wrapped core and an unwrapped core differs. Testing of a wrapped core such as coreA in Figure 63 means that the wrapper at coreA is placed in internal core test mode and test stimulus and test responses are transported via the TAM to coreA. Testing of an unwrapped core, such as coreC in Figure 63, means that the wrapper at coreA and the wrapper at coreB must be used since coreC does not have its dedicated interface to the TAM. A problem that has to be addressed when scheduling the tests is that a wrapper can only be in one mode at a time (normal, intest, or extest). In the example in Figure 63 it means that when coreA is tested its wrapper is in intest and when coreC is tested both wrappers at coreA and coreB must be in extest mode. Note also, that when coreA is tested test stimulus and test responses are transported to coreA, while when coreC is tested, the test stimulus is transported to coreA and the test responses are transported to the test sink from coreB.

In the modeling approach proposed by Larsson *et al.* [158, 160, 164, 165, 166, 176] the system in Figure 63 is modeled as shown in Figure 64. As discussed above, each core is given (x,y)-coordinates and the testable unit is a

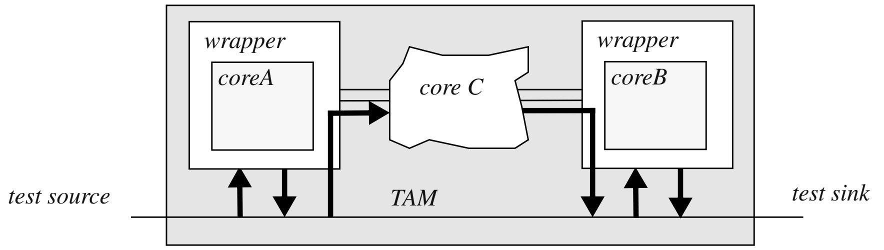


*Figure 63.*Illustration of external test mode (extest) and internal core test mode (intest).

block. The model of CoreB is a single block and a corresponding test. BlockA at CoreA is the core test in Figure 63. The interconnection test of core C is modeled as a block at CoreA and an indication at the test shows that the interconnection is to CoreB. It should be read as test stimulus should be transported from the test source to CoreA and test responses is to be transported from CoreB to the test sink.

The core wrappers are described in more detail in chapter Test Access Mechanism on page 99, and wrapper conflicts are in deeper detailed discussed in chapter Test Conflicts on page 77.

5 TEST ACCESS MECHANISM

The TAM (Test Access Mechanism) (for detailed discussion see Chapter 8 on page 99) connects test sources with testable units and testable units and test sinks. The TAM can be dedicated for testing purpose only or an existing structure can be used. The advantage of making use of an existing structure is that additional routing of wires is minimized. The advantage of a dedicated structure is the flexibility. The functional bus which is used in normal operation can during testing be used for test data transportation. The advantage of using the functional bus is that it exists in the system. There is no need for additional routing. The execution on a functional bus is sequential - one task (job) at a time and since it only allows sequential use, it can be modeled as a test conflict.

```
Syntax: name     x      y   {block1 block2 ...
blockN}
       COREA     120   50 {BlockA CoreC}
       COREB     180    50  {BlockB}
[Blocks]
#Syntax: name       {test1 test2 ... testN}
          BlockA    {TestA}
          BlockB    {TestB}
          CoreC     {TestC}
[Generators]
#Syntax: name    x      y
          ATE    100    50
[Evaluators]
#Syntax: name    x      y
          ATE    200    50
[Tests]
#name     tpg    tre        interconnection test
 TestA    ATE    ATE        no
 TestB    ATE    ATE        no
 TestC    ATE    ATE        CoreB
```

*Figure 64.*The modeling of test sources and test sinks.

Chapter 6

Test Conflicts

1 INTRODUCTION

In this chapter, we describe and discuss test conflicts that are important to consider when developing a test solution for a system. Problems related to test power consumption are discussed in *Test Power Dissipation* (page 89).

Conflicts can be classified as conflicts known prior to scheduling and conflicts that occur due to a certain test schedule, for instance, if a testable unit is to be tested with two test sets. One test set could be a deterministic set stored in the ATE while the second test set is a pseudo-random set generated with an LFSR. These two test sets can obviously not be scheduled (executed) concurrently. Only one test can be applied to a testable unit at a time. This is an example of a test conflict which is known prior to scheduling. A conflict that is not known of prior to scheduling is TAM wires sharing, for instance. If two tests are scheduled to be executed at the same time, these two tests can in general not share TAM wires, unless special care has been taken as in daisy-chaining (pipelining). A conflict such as TAM wire sharing cannot be found prior to scheduling.

We have assumed a core-based environment or a modular system view (See *System Modeling* on page 67) as in Figure 65, where the test data (test stimuli and test responses) can be stored in an ATE (test source and test sink). The test source feeds test stimuli to the system and the test sink stores the test responses.

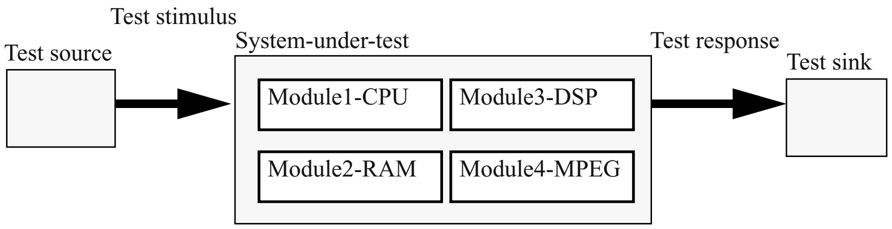


*Figure 65.*Modular view of system testing.

2 LIMITATIONS AT THE TESTER

The setup for testing a system can be as in Figure 66 where the system under test is connected to an ATE (Automatic Test Equipment). The test stimulus is transported from the channels at the ATE to the scan-chains at the system under test, and the test response captured in the scan-chains is transported back to the ATE for test evaluation. Examples of companies producing ATEs are Advantest [4], Agilent (HP) [7], Credence [48], LTX [181], and Teradyne [260].

In the example (Figure 66) the ATE serves as off-chip test source and off-chip test sink. Regardless if the test resources (test sources and test sinks) are on-chip or off-chip, there are a number of possible limitations that are to be considered.

An ATE has a number of channels (usually up to 1024) where each channel can be seen as a bit stream that is clocked out of the ATE. The ATE channels are usually grouped into ports (about 64) where a port is connected to a group of pins. Each port is controlled by a sequencer (can be seen as a controller running at some clock frequency).

Important conflicts at the ATE are:

- Bandwidth limitations (a limited number of channels are available),
- Memory limitations, test data (test stimulus and test response) has to fit the tester's memory. It is not feasible to reload the ATE.
- ATE clock-domains; the channels can be run at different clock speed.

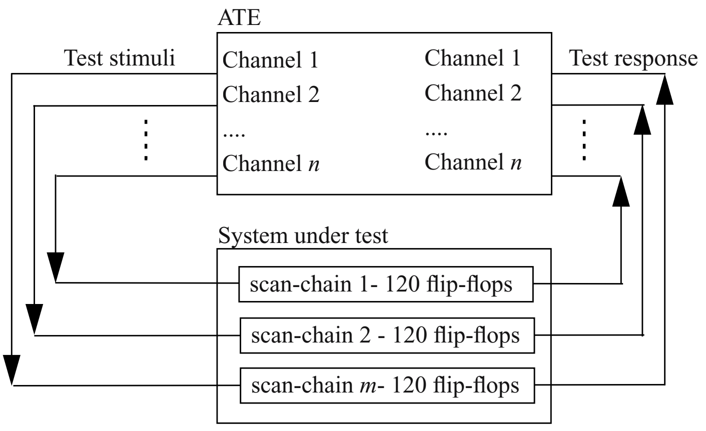


Figure 66. An Automatic Test Equipement (ATE) with *n* channels connected to the *m* scan-chains of the system under test.

2.1 Bandwidth Limitations

The number of scan-chains in a system is often higher than the number of channels at the ATE. Especially for a system designed in a modular way. In order to make the scan-chains in the system fit the number of ATE channels, the scan-chains have to be partitioned into longer chains. This problem will be in detail discussed in *Test Scheduling*, (page 115).

A way to model bandwidth limitations is illustrated for a small system in Figure 67 [166, 176]. A test source ([Generators]) r1 placed in the system at (x,y)-coordinate (10,10) has a maximal bandwidth of 1, *i.e.* a single TAM wire (scan-chain). The test sink s1 at [Evaluators] placed in the system at (x,y)-coordinate (20,10) has a maximal bandwidth of 2.

```
[Generators]
  #name   x    y    max bandwidth         memory
    r1    10   10   1                     100
[Evaluators]
  #name   x    y    max bandwidth
    s1    30   10   2
[Tests]
  #name   x    y    test sourcetest sink memory
  testA   20   10   r1           s1       10
```

*Figure 67.*Illustration of limitations at tester bandwidth and tester memory [166, 176].

2.2 Tester Memory Limitations

A test source, such as an ATE, has a fixed memory for storing test data. For an ATE, it is of most importance that the test data fits the ATE since an ATE refill is only an option in theory, in practice a memory refill is too time consuming. If test data is stored in an on-chip test source, which could be a memory, there is also a storage limitation.

The modeling of memory limitations is illustrated for a small system in Figure 67 [176, 166]. A test source r1, [Generators], has a storage capacity of 100. A test such as testA [Tests] requires 10 units out of the 100.

A way to reduce the tester memory limitation is to compress the test data. It is known that only a few bits (care bits) are specified in a test set (1 to 5%) for large cores [272]. The idea is to store compressed data in the ATE, send it to the SOC where added decompression logic uncompress' the test stimuli. For example, Koeneman [135] uses static LFSR reseeding. Several techniques have been proposed, for instance, Hellebrand *et al.* [90] proposes a approach using multiple-polynomial LFSR, Rajski *et al.* [222] suggest a technique using LFSR with variable-length seeds, and Jas *et al.* [123] describe a a technique called multiple LFSRs in virtual scan chains. The techniques using dynamic LFSR reseeding by Koenemann *et al.* [137], Krishna *et al.* [146], Krishna and Touba [147], Rajski *et al.* [223] are achieving best results.

An alternative way is to compress test data is to make use of a hardware structure such as XOR-networks proposed by Bayraktarolgu and Orailoglu [10], and folding counters proposed by Liang *et al.* [180]. The scan-chains can also be modified as in the approach proposed Hamazaoglu and Patel [87] where Illinois Scan is used, and in the RESPIN approach proposed by Dorsch and Wunderlich [54].

It is also possible to encode the test stimuli. Huffman codes and statistical codes have been used by Ichihara *et al.* [103], Iyengar *et al.* [109], and Jas *et al.* [122]. Iyengar *et al.* [109] explored the usage of run-length codes, while Chandra and Chakrabarty [35], and Hellebrand and Wurtenberg [91] investigated the use of alternating run-length codes. Golomb codes for test data compression was explored by Chandra and Chakrabarty [34], frequency-directed run-length (FDR) codes, also by Chandra and Chakrabarty [33], and packet-based codes by Koche *et al.* [129] as well as by Volkernik *et al.* [270]. Vranken *et al.* [272] propose a technique where the features of the ATE are explored instead of adding additional decompression logic to the system. The idea is that the sequencer (controller of each port) is programmed to control the flow of test data from the ATE. If a channel is to deliver x 1, the sequencer is programmed to repeat 1 x times. By doing this, the amount of test data can be reduced. Instead of storing x 1's, a single 1 is stored and a control scheme. A draw-back with the technique is that it is most efficient for testers with a high number of ports, hence expensive testers.

The techniques above are useful for compression of the test data. Especially, the test stimulus since it can be compressed off-line (before being stored in the tester). Test response has to be compressed on-chip and at run time. A disadvantage of the approaches is that they require additional silicon area. However, that can often be justified. But, the techniques also impose a more complicated design flow and tools [272].

2.3 Test Channel Clocking

The channels at the tester can all be driven by a single clock or by multiple clocks. The minimal multiple clock case is where two clocks are used and the channels are divided into two clock domains. The most general case is when there is one clock domain per channel. The clock domains can be fixed or non-fixed (flexible). If the clock in a clock domain is fixed, its frequency is fixed through out the testing while in the case of flexible clock speed the clock frequency can be changed during the application of the tests.

A test scheduling where the ATE contains two fixed clock domains is proposed by Sehgal and Chakrabarty [243]. The problem that occurs when having several clock domains is to assign modules to tester channels in such a way that the testing time is minimized and not violating test power consumption at a module. If a module is tested at a high clock frequency its power consumption increases as the switching activity increases and if the power consumption is above the limit of the module the module can be damaged. In the case with a tester where each channel has its dedicated and controllable clock, this problem can be avoided (on module-level) since each channel is clocked at a feasible frequency at any time.

3 TEST CONFLICTS

3.1 General Test Conflicts

In a system there can be general conflicts preventing tests to be executed concurrently. It can be conflicts such that a part of the system has to be used while another part is tested. In order to test part A, part B cannot be tested simultaneously. This type of conflicts can be modeled using a *resource graph* as used by Garg *et al.* (Figure 68) [65]. In a resource graph, see Figure 68, the tests in the system are on the top level and the resources are on the bottom level. An edge between nodes at different levels indicates that a test t_i tests a resource r_j or a resource r_j is needed to perform test t_i. This means that the resource graph captures information on resource conflicts. For instance, in Figure 68 both test t_1 and test t_3 make use of resource r_1, which means that test t_1 and test t_3 cannot be scheduled simultaneously.

Given a resource graph, a *test compatibility graph* (*TCG*) (Figure 69) can be obtained, where the nodes define the different test plans and the edges specify that two tests are compatible. From the test compatibility graph in Figure 69 it can be determined that test t_1 and t_2, for example, can be executed concurrently as they are connected with an edge.

The resource conflict in the example in Figure 68 can be modeled as in Figure 70 with the approach proposed by Larsson *et al.* [166, 176]. The basic idea is to list all resources (called blocks) for each test. The blocks are the

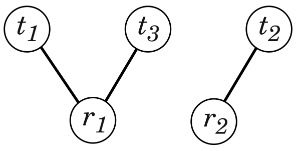


Figure 68. A resource graph.

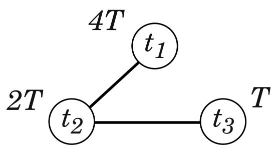
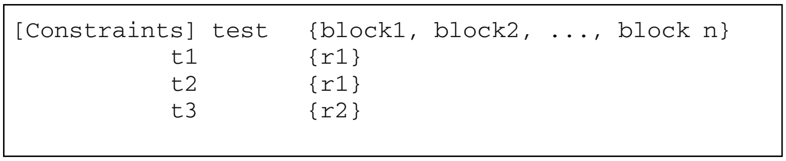


*Figure 69.*A test compatibility graph.



```
[Constraints] test   {block1, block2, ..., block n}
            t1       {r1}
            t2       {r1}
            t3       {r2}
```

*Figure 70.*Test conflict modeling where testA requires blockA and blockB to be avialable during its test execution.

testable units; however, it is not required to give a test for all blocks. It means that so called dummy blocks intended to specify a general conflict such as bus sharing.

3.2 Multiple Test Set

It can be efficient to test a testable unit with several test sets. For instance, if two sets are used; one stored off-chip in the ATE while the other is generated on-chip by an LFSR. A deterministically generated test set is often of higher quality than the quality of an LFSR-generated set. However, as discussed above, an ATE test set requires memory storage and it is becoming a limiting factor due to the high amount of test data that has to be stored in the limited memory of the ATE. If several test sets are used for a testable unit, test conflicts are unavoidable. Larsson *et al.* [166, 176] propose a way to model multiple test sets (Figure 71). A testable unit, blockA, is tested by a set of tests (testA1, testA2, and testA3) and each of the tests have their own specification on power consumption, test time, test source (TPG), test sink (TRE). The bandwidth (min_bw and max_bw) will be discussed later, as well as interconnection testing (ict).

3.3 Multiple Sets of Test Sets

A testable unit in a complex system may be tested by several test sets. It can be done in order to achieve a sufficient fault coverage. The selection of the test sets for the cores in the system affects the total test application time and the ATE usage. For instance, assume a system consisting of 4 cores as in Figure 72 and where each core is tested by a BIST and an external tester. Further, assume that the external tester can only test one core at a time. For each

```
[Tests]
 #name  pwr  time TPG  TRE   min_bw  max_bw        ict
 testA1 60   60   r1   s1    1       1          no
 testA2 1         r2   s2    4       5          no
 testA3.60   72   r1   s2    1       1          no
[Blocks]
 #name  idle_pwrpwr_grid test_sets {}
 blockA  0    p_grd1 { testA testA2 testA3}
```

*Figure 71.*Multiple test sets.

core it is possible to determine several test sets with sufficient fault coverage where the test sets differ in test time ratio between BIST test time and external test time. In Figure 72 two solutions for testing the cores are shown where in Figure 72(a) the total test time is higher than the test time in Figure 72(b) due to the use of different of test sets.

Sugihara *et al.* propose a technique for test set selection for each individual core, where each core is tested by a test set consisting of two parts, one based on BIST and the other based on external testing [256]. For each core *i* a set of test sets is defined, $v_i \in V_i$. Each test set v_i consists of a BIST part and a part using an external tester. $BC(v_i)$ is the number of BIST clock cycles for

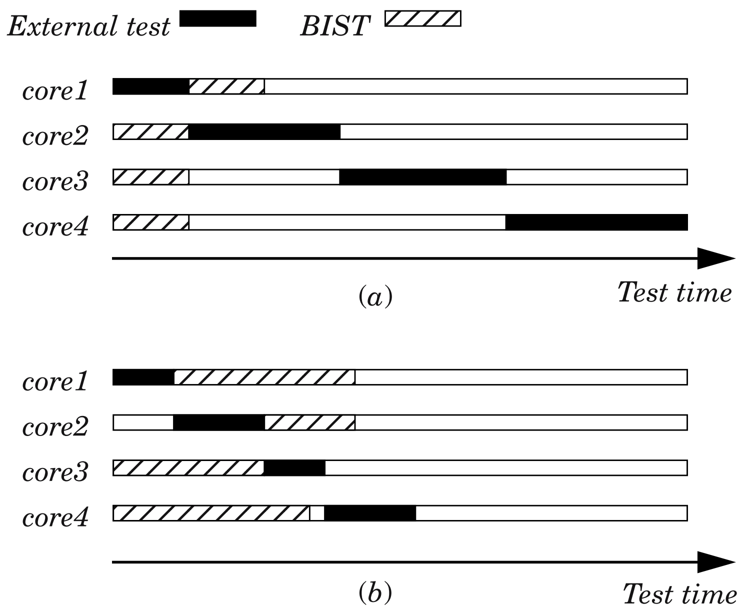


*Figure 72.*Example of test schedules.

test set v_i, and $ETC(v_i)$ is the number of clock cycles using the external tester. The total time used for external testing T_{ET} is given by:

$$T_{ET} = \sum_{i=0}^{n-1} \frac{ETC(v_i)}{FT} \tag{6.1}$$

where *FT* is the frequency of the external tester.

The total time used for external testing T_{ET} is given by:

$$T_{v_i} = \frac{BC(v_i)}{F} + \frac{ETC(v_i)}{FT} \tag{6.2}$$

where *F* is the system frequency used at the BIST. The total test application time, *T*, for the system is given by:

$$T = \max\left\{T_{ET}, \max_{i=0}^{n-1}\{T_{v_i}\}\right\} \tag{6.3}$$

The main problem is to determine the test set v_i for each core *i*, and the objective is to minimize the test application time.

For the system modeling of multiple test sets, Larsson [175, 177] proposed a scheme illustrated in Figure 73. For each testable unit (block) it is possible to specify sets of test sets where each such set is sufficient for the testing of the testable unit. For instance, Block1 can be tested either by the set {testA.1, testA.2, testA.2}, {testA1, testB} or {testC}. Each test in each set has its specification of test power test time, and test resources (test source and test sink).

3.4 Interconnection Testing - Cross-Core Testing

The interconnections between wrapped cores must also be tested. The problem with interconnection testing is that the interconnections and logic between wrapped cores do not have their own dedicated wrapper and hence no direct connection to the TAM. As wrappers are the required interface to the TAM, test stimuli and test responses for interconnection tests have to be

```
[Tests]
 #name   pwr   time  TPG  TRE  min_bw max_bw  ict
 testA.1 60    60    r1   s1   1      1       no
 testA.2 100   30    r2   s2   2      2       no
 testA.3 160   72    r3   s3   1      8       no
 testB   50    100   r4   r4   1      4       no
 testC   200   20    r3   s1   1      3       no
[Blocks]
 #name  idle_pwrpwr_grid test_sets {}, {}, ...,{}
 Block1  0    p_grd1 {testA.1 testA.2}{testA.1 testB} {testC}
```

*Figure 73.*Multiple test sets for a testable unit (block).

transported to and from the TAM via wrappers at wrapped cores. Figure 74 shows a part of a system where the UDL block is under test. The UDL block is unwrapped, and hence, it has no direct interface to the TAM. In order to access the TAM, the wrapper at CoreA is used for receiving test stimulus and the wrapper at CoreB is used to send the produced test response from the UDL to the test sink. The main problem is that wrapper cells can only be in one mode at a time, and therefore testing of CoreA and CoreB cannot be performed concurrently with the testing of the UDL block.

The testing of the wrapped core A is performed by placing the wrapper in *internal test mode* and test stimulus is transported from the required test source to the core using a set of TAM wires and the test response is transported from the core to the test sink using a set of TAM wires. In the case of an unwrapped testable unit such as the UDL block (Figure 74), the wrappers at core A and B are placed in *external test mode*. The test stimulus is transported from the required test source on the TAM via core A to the UDL block and the test response is transported via core B to the TAM and to the test sink.

The part of the system in Figure 74 where three testable units are tested with one test each can be modeled as in Figure 75 with the scheme by Larsson [175, 177]. All testable units are making use of the same test source (r1) and the same test sink (s1). The test testUDL is attached to coreA, this is the core to which test stimulus should be transported for the testing of testUDL, and at the specification of testUDL CoreB is indicated at ict (interconnection test), which means that CoreB is where test response is sent to the test sink.

3.5 Hierarchy - Cores Embedded in Cores

In a core-based environment, cores are embedded in the system. However, a core can also be embedded in a core. One or more child cores can be embed-

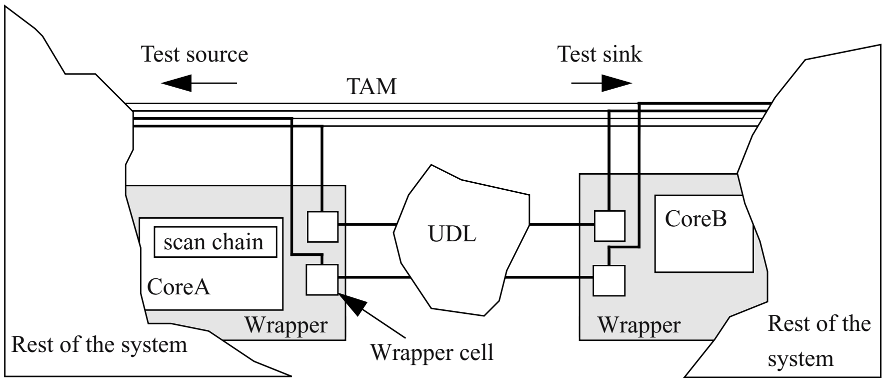


*Figure 74.*Interconnection test (cross-core testing) of the UDL block.

```
[Tests]
  #name    pwr    time   TPG  TRE  min_bw max_bw  ict
  testA    60     60     r1   s1   1      1       no
  testB    100    30     r1   s1   2      2       no
  testUDL  160    72     r1   s1   1      8       CoreB
[Blocks]
  #name  idle_pwrpwr_grid test_set
  CoreA   0    p_grd1 {testA testUDL}
  CoreB   10   p_grid2{testB}
```

*Figure 75.*Multiple test sets for a testable unit (block).

ded in a parent core (Figure 76). The embedding of cores in cores results in conflicts at testing. The testing of a parent core can usually not be performed at the same time as the testing of a child core. The required resources for the testing of the parent core are marked in bold in Figure 76(a) and the needed resources for the testing of the child core are marked in bold in Figure 76(b). From the example it is shown that the wrapper cells at the child core are the limiting resources. These wrapper cells are used both when testing the parent core and when testing the child core. For instance, the input wrapper cells (on the left side of the child core) are used to capture the test response when testing the parent core and to set up test stimulus when testing the child core.

This is a type of conflict that is known in advance and hence it can be modeled using a test resource graph as proposed by Garg *et al.* [65] and by Larsson *et al.* [176, 166] (Section 3.1). Figure 77 shows who to model the conflicts in Figure 76.

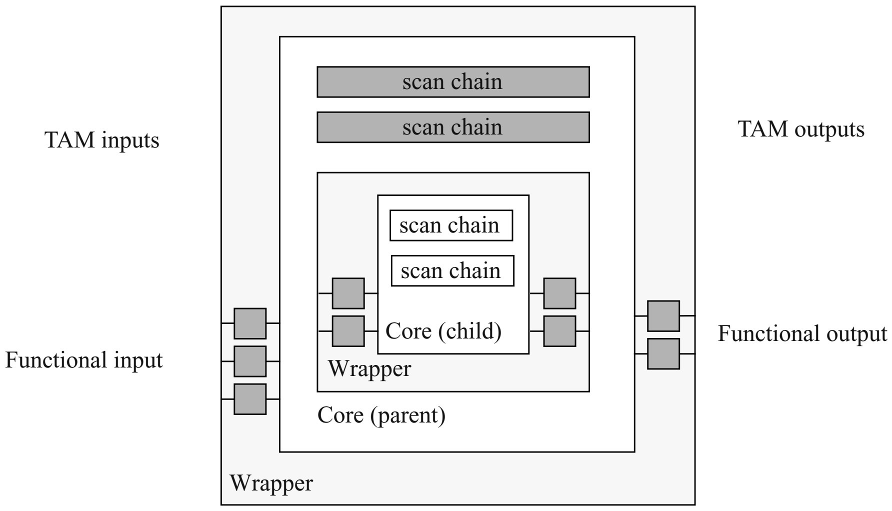
Goel [80] as well as Sehgal *et al.* [245] proposed recently wrapper extensions in order to handle hierarchical cores.

```
[Constraints] test   {block1, block2, ..., block n}
          testParent{ParentCore ChildCore}
          testChild {ParentCore ChildCore}
```

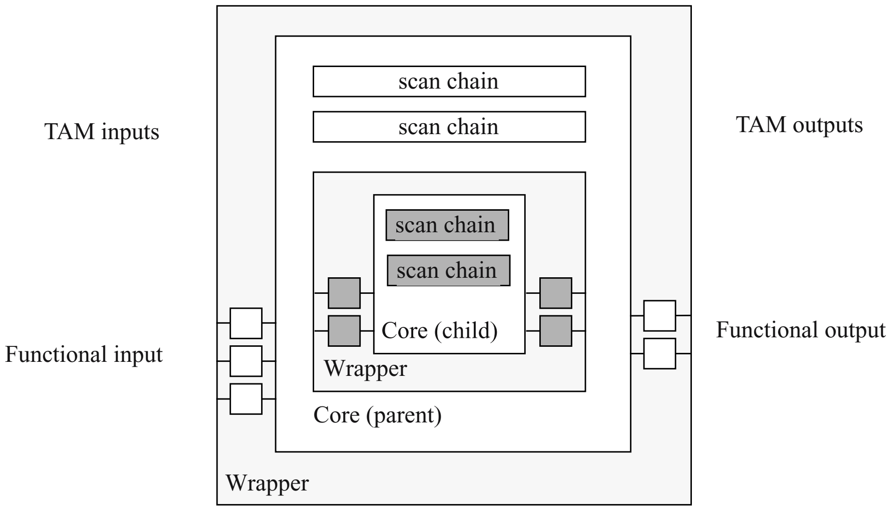
*Figure 77.*Test conflict modeling of cores embedded in cores (Figure 76).

4 DISCUSSION

We have divided the test conflicts into conflicts known prior to scheduling and conflicts not known in advance. A test conflict known prior to scheduling is, for instance if a testable unit is tested by multiple test sets, meanse that only one test can be applied to each testable unit at a time. On the other hand, a conflict that is not explicitly known prior to scheduling is TAM wire assignment, which TAM wires a particular test is assigned to use.



(a) Testing of parent core logic.



(b) Testing of child core logic.

*Figure 76.*Test conflict due to hierarchy (child core embedded in parent core).

Chapter 7

Test Power Dissipation

1 INTRODUCTION

The power consumed in a system during test mode can be higher than that consumed during normal operation. The reason is that in order to detect as many faults as possible per test stimulus (at as short time as possible) test stimulus is designed to create hyper-activity in the system. And, in order to reduce the testing time testing should be performed at the highest possible clock frequency. The test time can be reduced by activating a high number of cores concurrently, which means that during testing testable units may be activated in way that they will not be activated during normal operation. For example, a memory bank that in normal operation is activated one at a time can in testing mode be activated all at a time in order to reduce the testing time. However, systems are usually designed for normal operation, which makes it important to consider power consumption during testing otherwise the high power consumed during testing can lead to that the system is damaged.

The power dissipated during testing can be classified as:

- *system power consumption* - the testable units should not be activated in such a way that the total power budget for the system is exceeded,
- *hot spot power consumption* - the power consumed at given areas (parts of the system) should not exceed the given limit for that part,
- *unit power consumption* - the power consumed during testing can be higher than the specified limit for the testable unit.

High power consumption in a system can be tackled by:

- *design for low power* - which reduces the testing times by allowing testing at higher clock frequencies,
- *test scheduling* - where the tests are organized as concurrent as possible in order to minimize the test time while power limitations are considered, and
- a combination of design for low power and test scheduling.

In this chapter we discuss test power dissipation. In section 2 “Power consumption” the initial discussion on test power and its modeling is given and in section 3 “System-level Power modeling” power modeling at system level is discussed. In section 4 “Hot-spot modeling with Power Grids” and in section 5 “Core-level Power modeling” are discussed and the chapter is concluded with section 6 “Discussion”.

2 POWER CONSUMPTION

The power consumption during test is usually higher than during the normal operation mode of a circuit due to the increased number of switches per node which is desirable in order to detect as many faults as possible in the minimum of time [93]. However, high power consumption may damage the system because it generates extensive heat, which simply burns the system.

The power dissipation in a CMOS circuit can be divided into a static part and a dynamic part. The static power dissipation is derived from leakage current or other current drawn continuously from the power supply, while the dynamic power dissipation is due to switching transient current and charging and discharging of load capacitances [277].

The dynamic power consumption has, compared to the static part, been the dominating until the advent of sub-micron technology. The dynamic power consumption is due to loading and unloading of capacitances, and can be characterized by [277]:

$$P_{dyn} = \frac{1}{2} \times V^2 \times C \times f \times \alpha \tag{7.1}$$

where the capacitance C, the voltage V and the clock frequency f, are fixed for a given design [277]. The switch activity α, on the other hand, depends on the input to the system, which during test mode is the test stimulus. It means that the power dissipation varies depending on the test stimuli.

All parameters but the switching activity in formula (7.1) can be estimated using a design library. The switching activity depends on the input data and there are two main approaches to estimate it; based on simulation or probability. During testing the input to the design consists of the test vectors and it is possible to make use of the test vectors generated by an ATPG tool to estimate the switch activity for a circuit under test. As the order in which the vectors are applied often is not important the ordering can be used to control test power consumption. An approach where the test vectors are ordered based on Hamming distance has been proposed by Girard *et al.* [70].

The power dissipation is usually considered to originate from gates. However, power may dissipate not only from gates at blocks but also from large buses. For instance, for a wire of length 10 mm the capacitance will be about 7 pF [58]. In calculation of power consumption, the average capacitance should be used, which is close to half of the worst-case capacitance [58]. Assume a system running at 100 Mhz where the average switch activity (frequency) is 25 MHz for random input data. At 2 volts the power consumption is calculated by using formula 7.1

$$P = \frac{1}{2} \times C \times V^2 \times f \times \alpha = \frac{1}{2} \times 3.5 \times 10^{-12} \times 2^2 \times 25 \times 10^6 = 0.175\mathrm{mW}$$

In a realistic example the width of the data bus from the memory is 512 bits which results in a power dissipation of 90 mW ($512 \times 0.175 = 89.6$).

3 SYSTEM-LEVEL POWER MODELING

An example illustrating the test power dissipation variation over time τ for two test t_i and t_j is in Figure 78. Let $p_i(\tau)$ and $p_j(\tau)$ be the instantaneous power dissipation of two compatible tests t_i and t_j, respectively, and $P(t_i)$ and $P(t_j)$ be the corresponding maximal power dissipation. If $p_i(\tau) + p_j(\tau) < p_{max}$, the two tests can be scheduled at the same time. However, instantaneous power of each test vector is hard to obtain. To simplify the analysis, a fixed value $p_{test}(t_i)$ is usually assigned for all test vectors in a test t_i such that when the test is performed the power dissipation is no more then $p_{test}(t_i)$ at any moment.

The $p_{test}(t_i)$ can be assigned as the average power dissipation over all test vectors in t_i or as the maximum power dissipation over all test vectors in t_i. The former approach could be too optimistic, leading to an undesirable test schedule, which exceeds the test power budget. The latter could be too pessimistic; however, it guarantees that the power dissipation will satisfy the constraints. Usually, in a test environment the difference between the average and the maximal power dissipation for each test is often small since the objective is to maximize the circuit activity so that it can be tested in the shortest possible time [40, 41]. Therefore, the definition of power dissipation $p_{test}(t_i)$ for a test t_i is usually assigned to the maximal test power dissipation ($P(t_i)$) when test t_i alone is applied to the device. This simplification (Figure 79) was introduced by Chou *et al.* [40, 41] and has been used by Zorian [287], Muresan *et al.* [201, 202], Xia *et al.* [279], Zou *et al.* [292], Pouget *et al.* [219, 220] and Larsson *et al.* [176,169, 166].

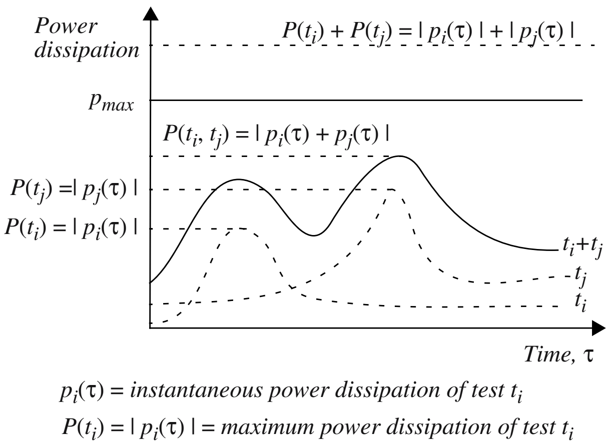


*Figure 78.*An example of the power dissipation for two tests over of time [40].

Zorian [287] and Chou *et al.* [40, 41] use an additive model for estimating the power consumption. The power dissipation for a test session s_j is defined as:

$$P(s_j) = \sum_{t_i \in s_j} P(t_i) \tag{7.2}$$

where t_i is a test scheduled in test session s_j.
The system power modeling used by Larsson *et al.* [176,169, 166] is illustrated in Figure 78. A power limit MaxPower is given as the limit that should not be exceeded at any time. For each test in the system a name, its power consumption when active, its test time, as well as test source (tpg) and test sink (tre) are specified. For instance, test tROM1 consumes 279 power units when active and it takes 102 time units to execute the test. The test source for the test is TCG and the test sink is TCE. And for each block the idle power, power consumed when the test is not active, and the tests for the block are specified. For instance, the idle power for block rom1_1 is 23 and it is tested with test tROM1.

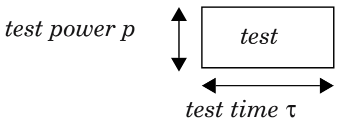


*Figure 79.*Test time and power consumption for a test.

```
[Global Options]
MaxPower = 900     # Maximal allowed simultaneous power
[Tests]
#Syntax:
# name power time tpg tre min_bandw max_bandw memory itc
tROM1   279  102  TCG TCE    1      1       4         no
tRAM4    96   23  TCG TCE    1      1       2         no
tRAM1   282   69  TCG TCE    1      1       4         no

[Blocks]
#Syntax: name idle_power {test1 test2 ... testN}
         rom1_1 23        { tROM1 }
         ram4_1 7         { tRAM4 }
         ram1_1 20        { tRAM1 }
```

*Figure 80.*System-level power modeling.

4 HOT-SPOT MODELING WITH POWER GRIDS

The system-level power model considers the global dissipated power. It means that at any time the total amount of consumed power is never above a certain limit. However, such a model does not consider where in the system the power is dissipated. A so called hot spot is a limited area where a high amount of power is consumed and it can damage the system. The power dissipated within the area does not have to exceed the system's total power budget, but can nevertheless damage the system.

Hot spots may appear during testing due to that testable units are activated in a way that they would not be activated during normal operation. Four memory blocks A, B, C, and D that are part of a system are shown in Figure 81. The memory blocks are powered by the same power grid. In normal operation the memory access to these four blocks is designed in such a way that only one block is activated at a time. The four memory blocks are designed to functionally work as a single memory. The power consumed by the memories is then not more than activating one block (P_A, P_B, P_C, P_D), *i.e* 50, which is far under the power budget of the power grid. In testing mode, all memory blocks can be activated concurrently in order to reduce the testing time. The power consumed by the four memory blocks is then $P_A + P_B + P_C + P_D = 50 + 50 + 50 + 50 = 200$, which is far above the power budget of the power grid. There is then a high risk that such concurrent testing of the memory blocks in Figure 81 will damage the system.

Larsson [177] proposed a why to model power grid. The power grid model has similarities to the approach proposed by Chou *et al*. [40] but in contrast to it the model by Larsson includes local areas (power grids). Each block (testable unit) is assigned to a power grid where the power grid has its power

Budget power grid 1=60
Normal Test
MemoryA P_A=50
Memory B P_B=50
Memory C P_C=50
Memory D P_D=50

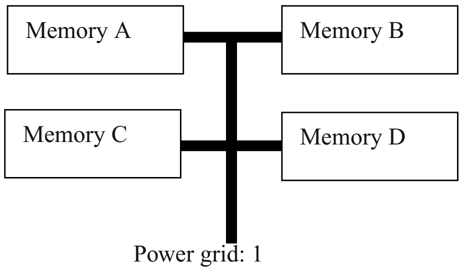


*Figure 81.*A part of a system. Four memories block powered with the same power grid.

budget and the system can contain a number of power grids. Blocks assigned to a power grid cannot be activated in such a way that the power grid budget is exceeded at any time. The example in Figure 81 is modeled as in Figure 82. Note that the modeling of test sources and test sinks are excluded. A limit is given to each power grid and the placement of the memory blocks are then given. Each memory block consists of one testable unit. For instance, MemoryA consists of the testable unit MemBlkA. Each testable unit is tested with one test. MemBlkA, for example, is tested with the test named testMemA. And each test has its specification on test time etc.

```
[PowerGrid] pwr_grid       limit
             grid1         30
[Cores]
#Syntax:
#name    x    y {block1 block2 ... blockN}
MemoryA  10   10 {MemBlkA}
MemoryB  20   10 {MemBlkB}
MemoryC  10   20 {MemBlkC}
Memoryd  20   20 {MemBlkD}
[Blocks]
#name idle_pwr pwr_grid {test1,...,test n}
MemBlkA       0    grid1    {testMemA}
MemBlkB       0    grid1    {testMemB}
MemBlkC       0    grid1    {testMemC}
MemBlkD       0    grid1    {testMemD}
[Tests]
#Syntax:
#name     power   time tpg tre min_bw max_bw mem itc
testMemA  1        255  TGA TRA      1     1       1    no
testMemA  1        255  TGB TRB      1     1       1    no
testMemA  1        255  TGC TRC      1     1       1    no
testMemA  1        255  TGD TRD      1     1       1    no
```

*Figure 82.*Power grid (hot spot) modeling.

5 CORE-LEVEL POWER MODELING

The test power dissipation at a testable unit can be above the specified limit for the unit. But it results in that when testing the unit it is damaged. This can, for instance, occur if the test clock frequency is too high (increasing the power dissipation) or due to hyper-activity. The motivation behind core-level adjustments is two-fold. First, by lowering the power consumption at a core, a higher number of cores can be activated concurrently. Second, since test power consumption often is higher than that during the normal operation, the power dissipation at a specific core can be higher that its own power budget.

The power consumed at a core can be adjusted by using clock-gating [241]. Clock-gating can reduce the power consumption so that a higher number of tests can be executed concurrently, but also, it can be use for testable units where its own power dissipation is higher than its allowed power consumption due to for instance a too high test clock frequency.

The power consumption during testing is often higher than that consumed during normal operation. A testable unit that consumes power below the power budget during normal operation can during testing consume power above the power budget. In Figure 83 each module consumes 50 units during normal operation, and during testing each module consumes 80 units, which is above the power budget for power grid 1. The power grids in the system can be over-designed in order to make sure that testable units can be tested. An alternative is to use clock-gating.

The test power consumption depends highly on the switching activity. During testing in scan-based systems, switches appear not only during the application of test vectors, at the capture cycles, but also in the shift process when a new test vector is shifted in while the test response from the previous test vector is shifted out. Actually, the shift process contributes to a major part of the test power consumption [68]. Gerstendörfer and Wunderlich [68] proposed a technique to isolate the scan flip-flops during the shift process. However, the approach may cause an effect on the critical path.

Budget power grid 1=60

Block	Normal	Test
MemoryA	P_{AN}=50	P_{AT}=80
Memory B	P_{BN}=50	P_{BT}=80
Memory C	P_{CN}=50	P_{CT}=80
Memory D	P_{DN}=50	P_{DT}=80

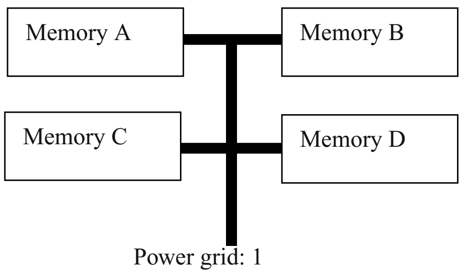


*Figure 83.*A part of a system. Four memories block powered with the same power grid.

Saxena *et al*. [241] proposed a gating scheme to reduce the test power dissipation during the shift process that does not have a direct impact on the critical path. Given a set of scan-chains as in Figure 84 where the three scan-chains are forming a single chain. During the shift process, all scan flip-flops are active and it leads to high switch activity in the system and high power consumption. However, if a gated sub-chain scheme as proposed by Saxena *et al*. is introduced (Figure 85), only one of the three chains is active at a time during the shift process while the others are switched off and as a result no switching activity is taking place in them. The test time in the examples (Figure 84 and Figure 85) are the same while the switch activity is reduced in the gated example and also the activity in the clock tree distribution is reduced [241].

The experimental results presented by Saxena *et al*. [241] indicate that the test power consumption can be reduced to a third using a gated scheme with three sub-chains as in Figure 85 compared to the original scheme in Figure 84. Larsson [177] used this and created generalized model based on the experimental results presented by Saxena *et al*. [241] assuming a linear dependency between the test time and the test power consumption. If x scan chains exist at a core, it is possible to form x number of wrapper chains. In such a case, the test time will be reduced since the shift process is minimized; however, the test power consumption will be maximized since all of the x scan chains are active at the same time. On the other hand, if a single wrapper chain is assumed where all scan chains are connected into a single wrapper chain, the test time will increase but the test power consumption can be reduced by gating the x scan chains.

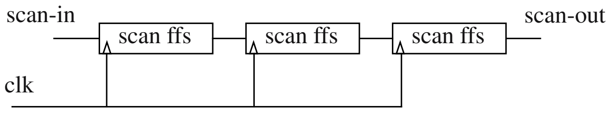


*Figure 84.*Original scan chain [241].

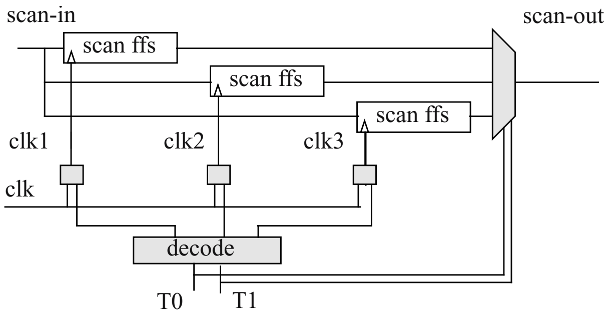


*Figure 85.*Scan chain with gated sub-chains[241].

An illustrative example is in Figure 86. In Figure 86(a) the scan chains at the core are all connected into a single wrapper chain, which is connected to the TAM. In such a scheme, the testing time becomes long due to long shift in time and shift out time. Also, the test power consumption is high since all scanned elements are active at the same time. Figure 86(b) shows how the test time can be reduced. A high number of TAM wires are connected to the scan chains. There is one wire per scan chain. The test time is reduced since all scan chains can be loaded concurrently. However, the test power consumption remains as in Figure 86(a) since all scanned elements are activated concurrently. In order to reduce the test power, a scheme as Figure 86(c) can be used (clock-gating). In this scheme, as few scan chains as possible are loaded concurrently. If one TAM wire is assigned to the core, one scan chain is loaded at a time. The testing time is the same as compared to the scheme in Figure 86(a). However, since less scanned elements are active at a time, the test power consumption is lower.

Larsson [177] defined a model as illustrated in Figure 87. There are two tests, testA and testB. TestA does not allow clock-gating (indicated by flex_pwr no). It can be a test at a testable unit where the core provider fixed the test power. On the other hand, testB, allows clock gating (indicated by flex_pwr yes). The power consumption for a test is given as a single value and within the specified bandwidth range, the power can change. Note that we include the possibility to specify if clock-gating can be used by setting *flexible_pwr* to yes or no. If power can be modified, we assume a linear dependency:

$$p_i = p_1 \times \mathrm{tam} \tag{7.3}$$

where p_1 is the power at a single tam wire, and *tam* is the number of TAM wires, which has to be in the specified range [*minbw*:*maxbw*].

The advantage of the model is that it is possible to model systems tested with a combination of tests with fixed testing time and fixed test power, flexible testing time and fixed power consumption, and flexible testing time and flexible power consumption.

```
[Tests]
#Syntax:
#name     power   time tpg tre min_bw max_bw mem flex_pwr
testA     15       255  TGA TRA      1     3       1    no
testB     10       155  TGA TRA      1     3       1    yes
```

*Figure 87.*Core-level power modeling.

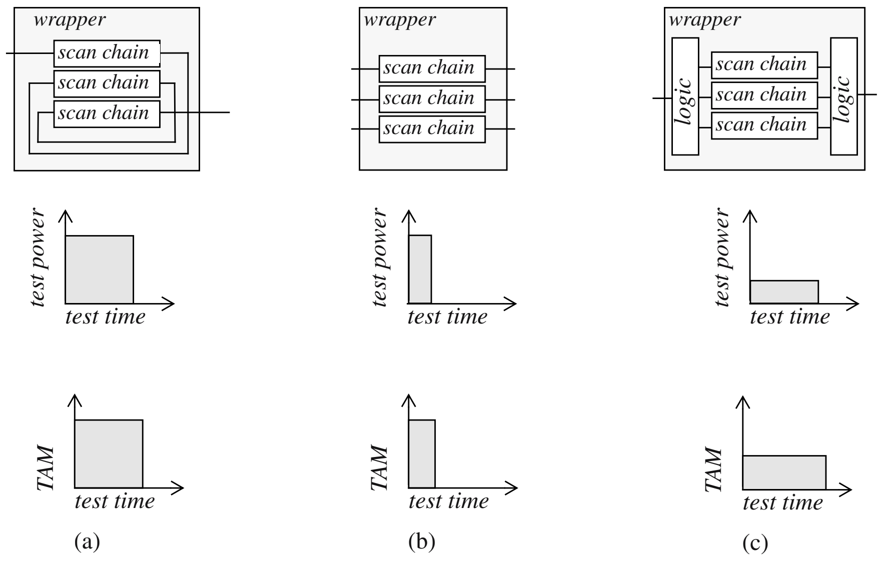


*Figure 86.*Core design alternatives.

6 DISCUSSION

Power consumption is becoming a problem in testing. Several test scheduling techniques have been proposed that take power consumption into consideration. However, the used power model has been a single fixed value, *i.e.* a not very precise measure.

This chapter has given an introduction to power consumption and its modeling. System level (global) power modeling as well as power grid (hot spot) modeling, and core-level adjustments such as clock gating have been discussed. Questions that remains to be answered is if it make sense to consider power consumption with only a single power value when test time is counted at clock cycle granularity? At what granularity can power consumption be modeled? How does it behave? Is peak power consumption worse than high average power consumption? Is static power really negligible in comparison with dynamic power consumption for CMOS designs?

Chapter 8

Test Access Mechanism

1 INTRODUCTION

The TAM (test access mechanism), the topic of this chapter, is responsible for the transportation of test data (test stimulus and test response). The TAM can be dedicated for test purposes only, or it can be an existing structure, such as the functional bus, used not only in testing mode but also during normal operation. The TAM connects, for each testable unit, its test sources with the testable unit and the testable unit with its test sinks (Figure 88). To ease the connection between a testable unit and the TAM, an interface called a core test wrapper is used.

The test infrastructure, the TAM, consists of two parts; one part for the actual test data transportation and one part that control the test data transportation. The cost of the infrastructure depends on its length and width; that is the length and the number of TAM wires. In the case of reusing the existing functional structure, the additional TAM cost is negligible. In a fully BISTed system where each testable unit in the system has its own dedicated test source and its dedicated test sink there is a minimal TAM requirement assuming that the test source and the test sink for each testable unit are placed at the location of the testable unit. Only an infrastructure controlling the start and end of the tests is required.

We will discuss Boundary-scan (IEEE 1149.1)[19], TestShell [184], TestCollar [267] and P1500 [51, 104] with the emphasis of SOC testing and their use in different TAM architectures.

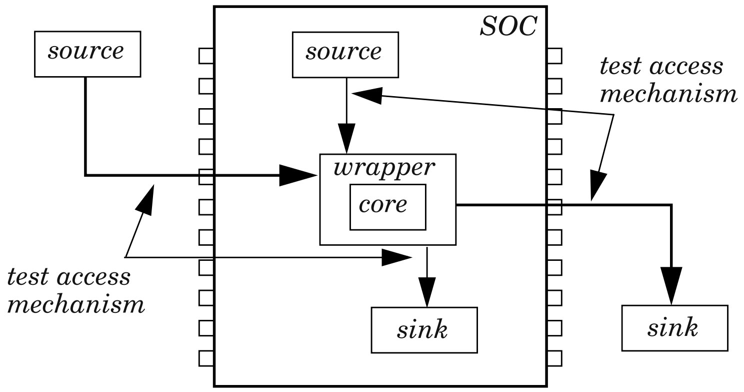


Figure 88. Test sources and sinks.

1.1 System-on-Chip Test Data Transportation

The Boundary Scan technique (see page 53) is mainly developed for PCB systems. The main objective is to test interconnections and logic placed between components. The infrastructure for test data and test control is minimal as it is shared. The advantage of that is that minimal additional silicon is required. The disadvantage of Boundary Scan is the long testing times due to serial access. For SOC designs where modules (cores, components) are not tested prior to mounting, testing includes external testing, as in PCB, and also internal testing of the cores. Also, the increasing complexity of SOCs leads to an increasing amount of test data volume to be transported in a system. For that reason, core wrappers such as TestShell, TestCollar, and P1500 have be developed.

1.1.1 The TestShell and P1500 Approach

The TestShell is proposed to reduce the test access and test isolation problems for SOC designs proposed by Marinissen *et al.* [186]. The TestShell consists of three layers of hierarchy, see Figure 89, namely:

- the *core* or the *IP module*,
- the *TestShell*, and
- the *host*.

The *core* or the *IP module* is the object to be tested and it is designed to include some DFT mechanism. No particular DFT technique is assumed by the TestShell. The *host* is the environment where the core is embedded. It can be a complete IC, or a design module which will be an IP module itself. Finally, the *TestShell* is the interface between the core and the host and it contains three types of input/output terminals, see Figure 90:

- *function* input/output corresponds one-to-one to the normal inputs and outputs of the core.
- *TestRail* input/outputs are the test access mechanism for the TestShell with variable width and an optional bypass.
- *direct* test input/outputs are used for signals which can not be provided through the TestRail due to their non-synchronous or non-digital nature.

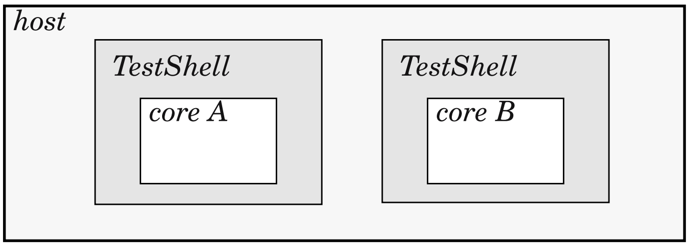


*Figure 89.*Three hierachy layers: core, TestShell and host.

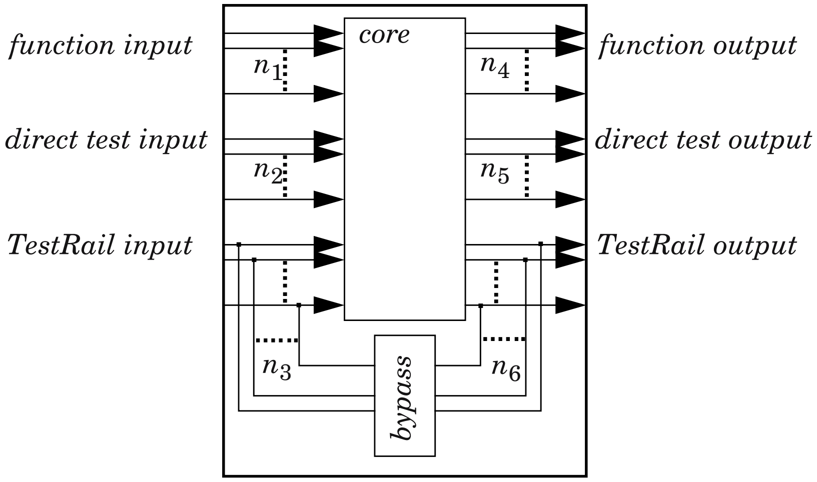


*Figure 90.*Host-TestShell interface.

The conceptual view of a TestCell is illustrated in Figure 91 and it has four mandatory modes:

- *Function* mode, where the TestShell is transparent and the core is in normal mode, *i.e.* not tested. It is achieved by setting the multiplexers m_1=0 and m_2=0.
- *IP Test* mode, where the core within a TestShell is tested. In this case the multiplexers should be set as: m_1=1 and m_2=0 where test stimulus comes from s_1 and test response is captured in r_1.
- *Interconnect Test* mode, where the interconnections between cores are tested. The multiplexers are set to m_1=0 and m_2=1 where r_2 captures the response from a function input and s_2 holds the test stimulus for a function output.
- *Bypass* mode; test data is transported through the core regardless if the core has transparent modes. It may be used when several cores are connected serially into one TestRail to shorten an access path to the

core-under-test, see Bypass using Boundary-scan in see page 53 (Bypass is not shown in Figure 91). The bypass is implemented as a clocked register.

Figure 92 illustrates the TestShell approach where a Test Cell is attached to each functional core terminal (primary input and primary output). Every TestShell has a TestRail which is the test data transport mechanism used to transport test patterns and responses for synchronous digital tests. The width n ($n \geq 0$) of the TestRail is a trade-off between the following parameters:

- *Host pins* available for test form an important limiting factor with respect to the maximal TestRail width.
- *Test time* is dependent on the test data bandwidth.
- *Silicon area* required for wiring the TestRail increases with the width of the TestRail.

The TestRail is designed to allow flexibility; see Figure 93, where an example is shown to illustrate some possible connections. Within the TestShell the connections may vary. The three basic forms of connection are as follows, see Figure 94:

- *Parallel* connection means that the TestRail is a one-to-one connected to the terminal of the core.
- *Serial* connection means that a single TestRail wire is connected to multiple IP terminals forming a shift register, similar to Boundary-scan (see page 53).
- *Compressed* connection refers to decompression hardware at core inputs or compression hardware at core outputs.

It is also possible to use a combination of the above types of connections. The type of connection selected for a particular core depends mainly on the width of the available TestRail.

A standardized Test Control Mechanism to control the operation of the TestShell means that instructions are loaded in the Test Control Block, see Figure 92.

A similar approach to the TestShell is the P1500 proposal (see Figure 95) [104]. The P1500 consists of a Core Test Wrapper and a Core Test Language. The wrapper uses *Wrapper Boundary Cells* with functionality similar to the Test Cell in TestShell and the Boundary-scan Cell in the Boundary-scan approach. Instructions are loaded to the *Wrapper Instruction Register* (WIR)

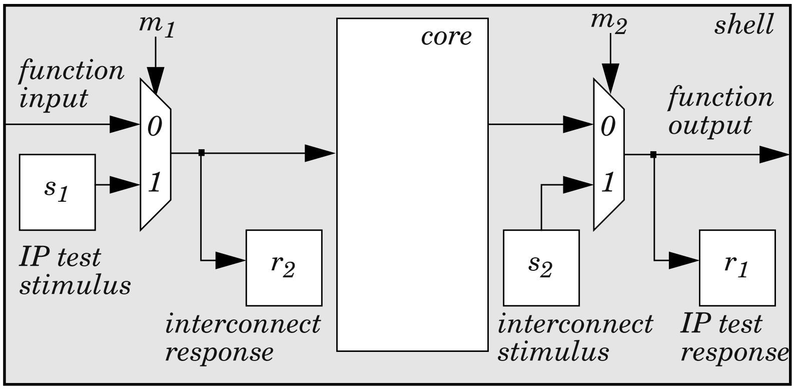


*Figure 91.*Conceptual view of a Test Cell.

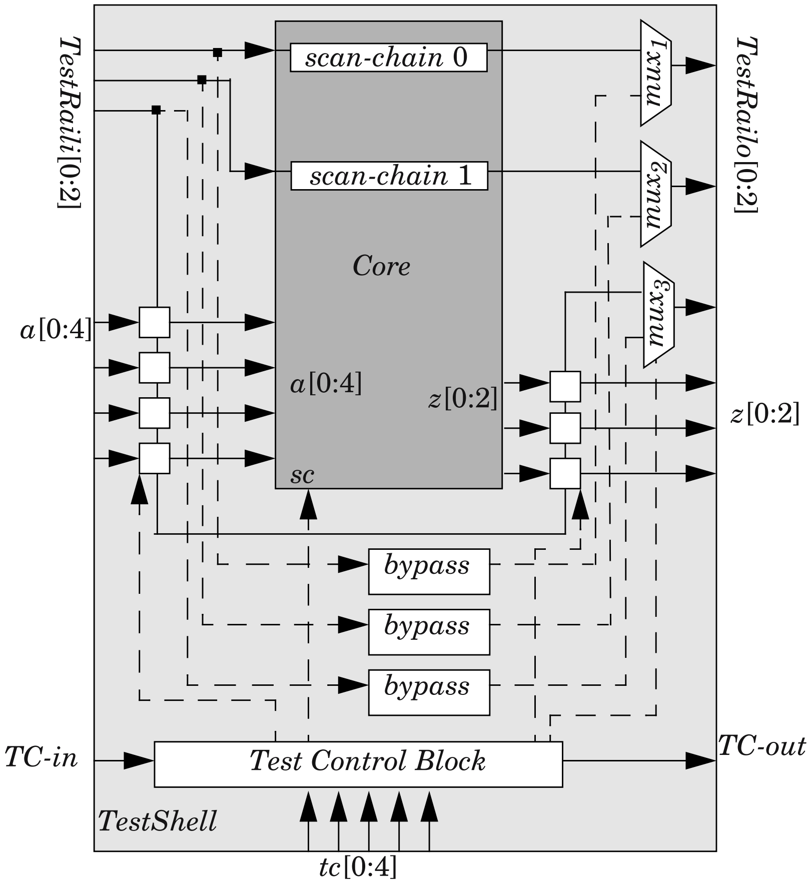


*Figure 92.*The TestShell approach.

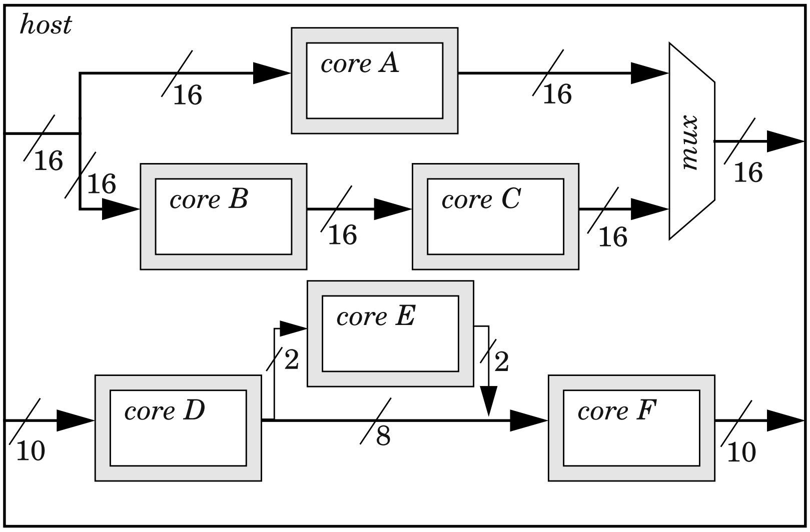


*Figure 93.*Example of possible host-level TestRail connections.

which is similar to the Test Control Mechanism in TestShell and the instruction register in Boundary-scan.

The differences between the TestShell wrapper and the P1500 approach is that the former allow a bypass of the test access mechanism (TAM) width while the P1500 only has a single-bit bypass, the *single-bit TAM plug* (STP). The P1500 wrapper connects to one mandatory one-bit wide TAM and zero or more scalable-width TAM, *multi-bit TAM plug* (MTP). The P1500 allows different widths of the *multi-bit TAM plugs* (MTP) input and output.

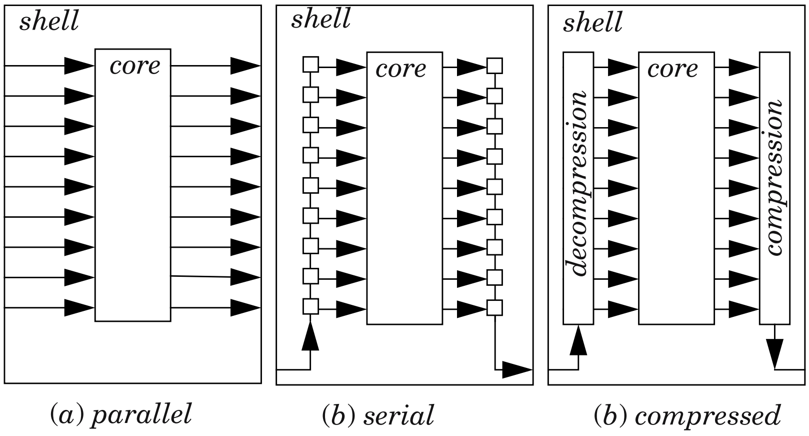


*Figure 94.*The core-level TestRail connections.

Another wrapper approach, called TestCollar, is proposed by Varma and Bhatia [267]. The approach is similar to the TestShell; however, the approach does not have the bypass feature which reduces the flexibility by allowing only one core to be served at a time.

An approach combining P1500 and the TestShell approach has been proposed by Marinissen *et al.* [187]. A major advantage is the flexible bypass introduced, see Figure 96. Two types of bypass styles are defined; wrapper bypass and scan-chain bypass. The wrapper bypass is the same as used in the TestShell while the scan-chain bypass is a flexible structure which can be inserted at any place between a terminal input and a terminal output. The advantage is that it allows a non-clocked bypass structure which can be used to bypass the complete core.

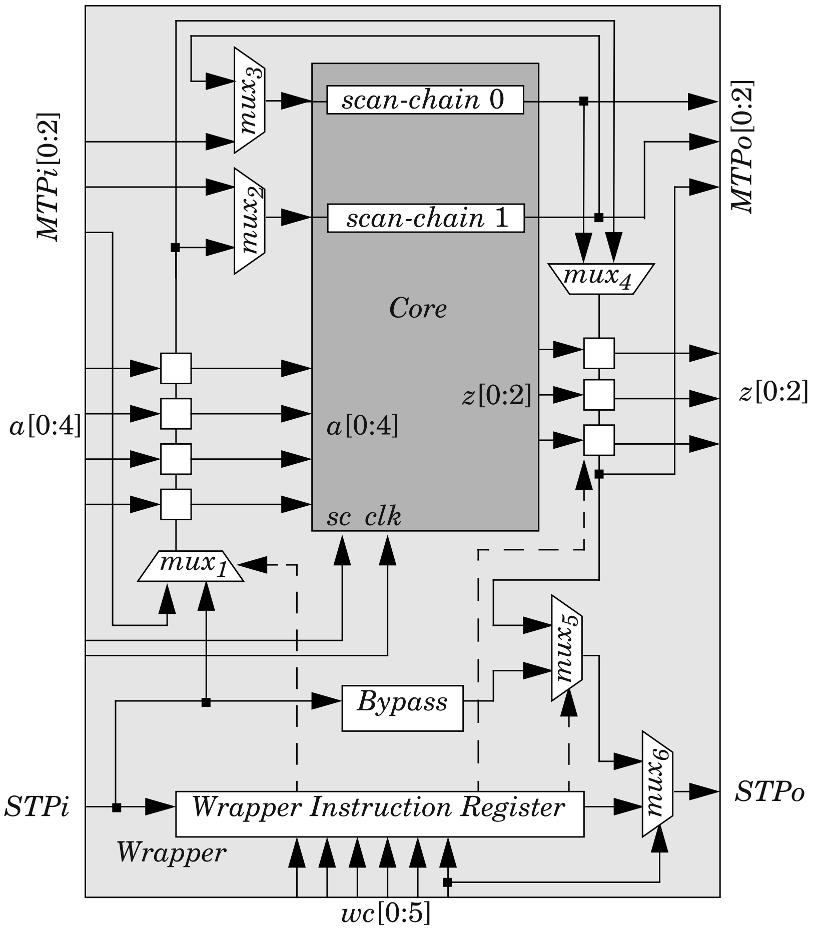


*Figure 95.*The P1500 approach.

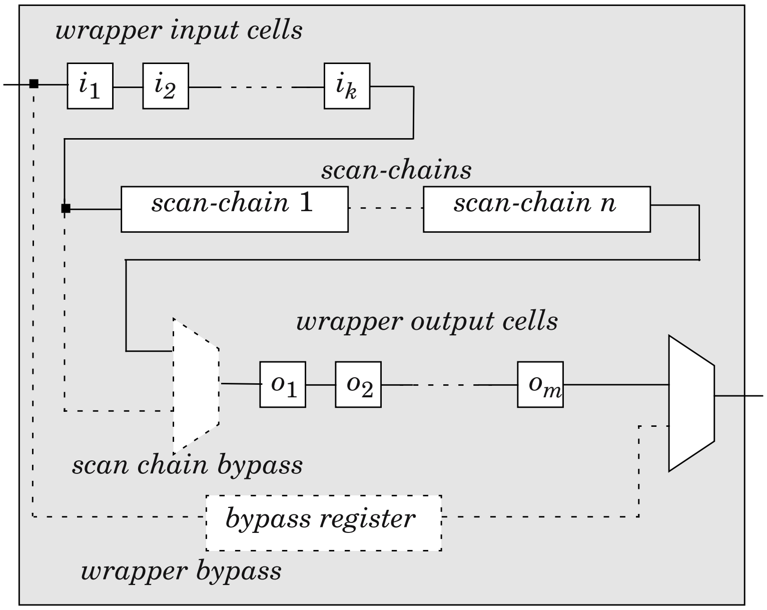


*Figure 96.*Proposed bypass structures where optional items are dashed [187].

1.2 Reconfigurable Core Wrappers

The core wrapper proposed by Koranne allows, in contrast to approaches such as Boundary Scan, TestShell and P1500, several wrapper chain configurations [139, 140]. The main advantage is the increased flexibility in the scheduling process. We use a core with 3 scan chains of length {10, 5, 4} to illustrate the approach. The scan-chains and their partitioning into wrapper chains are specified in Table 5.

TAM width	*Wrapper chain partitions*	*Max length*
1	[(10,5,4)]	19
2	[(10),(5,4)]	10
3	[(10),(5),(4)]	10

Table 5. Scan chain partitions.

For each TAM widths (1, 2, and 3) a di-graph (directed graph) is generated where a node denotes a scan-chain and the input TAM, node I (Figure 97). An arc is added between two nodes (scan-chains) to indicate that the two scan-chains are connected, and the shaded nodes are to be connected to the output TAM. A combined di-graph is generated as the union of the di-graphs.

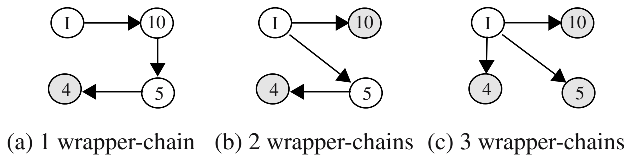


*Figure 97.*Di-graph representations.

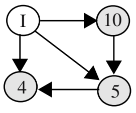


*Figure 98.*The union of di-graphs in Figure 97.

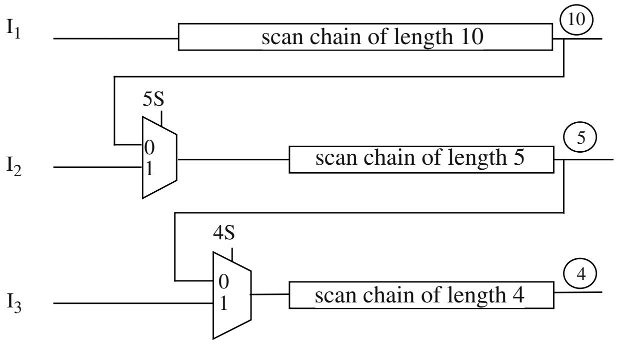


*Figure 99.*Multiplexing strategy [139, 140].

Figure 98 shows the result of the generated combined di-graph from the three di-graphs in Figure 97 The indegree at each node (scan-chain) in the combined di-graph gives the number of signals to multiplex. For instance, the scan chain of length 5 has two input arcs, which in this example means that a multiplexer selecting between an input signal and the output of the scan chain of length 10 is needed. The multiplexing for the example is outlined in Figure 99.

2 TEST ACCESS MECHANISM DESIGN

The Test Access Mechanism (TAM) is responsible for transporting test data in the system. It connects test sources with cores and the cores with the test sinks. The TAM can be dedicated for test purposes only, or it can be an existing structure. It is also, for a system, possible to combine the two.

2.1 Multiplexing Architecture

Aerts and Marinissen propose the *Multiplexing architecture* [5], see Figure 100, where each of the cores are assigned to all TAM wires and the cores are tested one at a time. The TAM is routed to and from all cores in the system. The outputs from all cores are multiplexed, which means that only one core at the time can use the outputs. The result is that the cores have to be tested in sequence [5].

Assume that the following is given:

f_i: the number of scannable flip-flops,

p_i: the number of test patterns, and

N: the scan bandwidth for the system, maximal number of scan-chains.

In scan-based systems it is common to use a pipelined approach where the test response from one pattern is scanned out, the next pattern is scanned in simultaneously. The test application time t_i for a core i is given by [5]:

$$t_i = \left\lceil \frac{f_i}{n_i} \right\rceil \cdot (p_i + 1) + p_i \qquad (8.1)$$

In the muliplexed architecture $n_i = N$. The term $+p_i$ in Equation (8.1) is added due to the fact that the pipelining can not be used for scanning out the last pattern. The pipelining approach can be used when several cores are tested in a sequence. While the first pattern is scanned in for a core, the test response from previous pattern can be scanned out for the previous core under test. The test application time using the multiplexed architecture is given by:

$$T = \sum_{i \in C} \left(p_i \cdot \left\lceil \frac{f_i}{N} \right\rceil + p_i \right) + \max_{i \in C} \left\lceil \frac{f_i}{N} \right\rceil \qquad (8.2)$$

where the maximum results from filling the largest core.

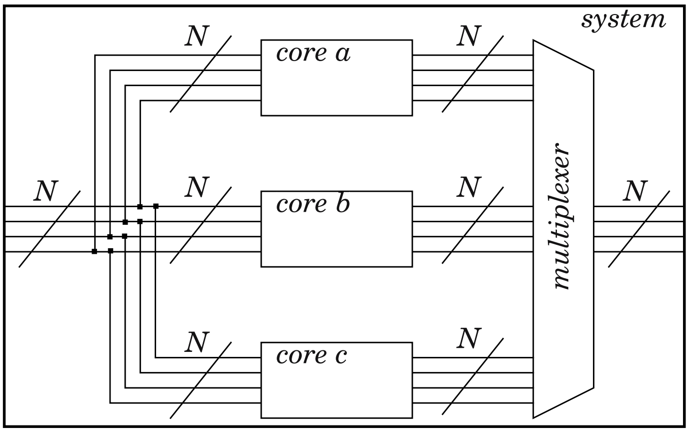


*Figure 100.*Example of the Multiplexing architecture [5].

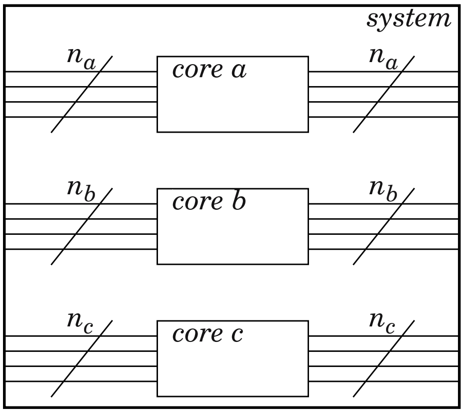


*Figure 101.*Example of the Distribution architecture [5].

2.2 Distribution Architecture

Aerts and Marinissen proposed also the *Distributed architecture* where each core is given a private dedicated number of TAM wires [5], Figure 101. The problem is to assign TAM wires and connect the flip-flops into that number of scan chains at each core i in order to minimize the test time, *i.e.* assign values to n_i where $0<n_i\leq N$. The test application time for a core i in the distribution architect is given by Equation (8.1) and the total test time for the system is given by:

$$T = \max_{i \in C}(t_i) \tag{8.3}$$

An algorithm is proposed to assign the bandwidth n_i for each core i, Figure 102, where the goal is to find a distribution of scan chains such that the test time of the system is minimized while all cores are accessed, expressed as:

$$\min_{\bar{n} \in N^{|C|}}(\max_{i \in C}(t_i)),\ \sum_{i \in C} n_i \leq N \wedge \forall i \in C\{n_i > 0\} \tag{8.4}$$

The algorithm outlined in Figure 102 works as follows. Each core is assigned to one TAM wires and the scanned elements at each core are forming a single scan-chain at the core which is required to test the system. In each iteration of the loop, the core with the highest test time is selected and another TAM wire is assigned. The scanned elements at that core are formed into a higher number of scan-chains; hence the testing time is reduced. The iterations are terminated when no more TAM wires can be distributed.

```
forall $i \in C$ begin
  $n_i=1$
  $t_i=\lceil f_i/n_i \rceil \cdot (p_i+1)+p_i$
sort elements of C according to test time
$L=N-|C|$
end
while $L \neq 0$ begin
  determine $i^*$ for which $t_{i^*}=\max_{i \in C}(t_i)$
  let $n_{i^*}=n_{i^*}+1$ and update $t_{i^*}$ accordingly
  let L=L-1;
end
$n_i$ gives the number of scan chains for core i
$\max_{i \in C}(t_i)$ gives the test time
```

*Figure 102.*Algorithm for scan chain distribution.

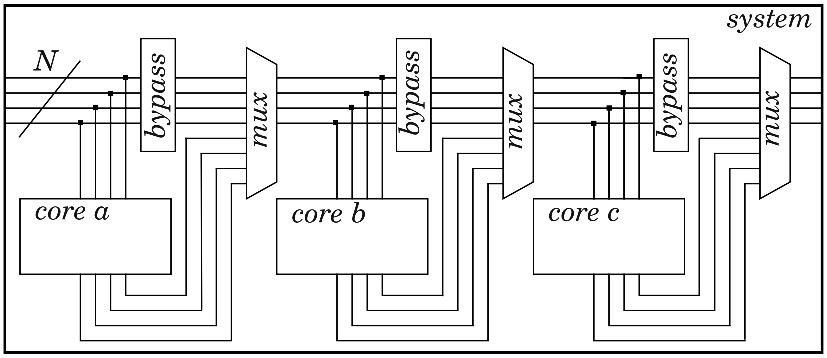


*Figure 103.*Example of the Daisychain architecture [5].

2.3 Daisychain Architecture

In the *Daisychain architecture* proposed by Aerts and Marinissen [5], Figure 103, a bypass structure is added to shorten the access path for individual cores. The bypass register and 2-to-1 multiplexer allow flexible access to individual cores, which can be accessed using the internal scan chain of the cores and/or by using the bypass structure.

The bypass offers an optional way to access cores and a bypass selection strategy is proposed by Aerts and Marinissen [5], where all cores are tested simultaneously by rearranging the test vectors. The approach starts by not using any of the bypass structures and all cores are tested simultaneously. At the time when the test of one core is completed, its bypass is used for the rest of the tests. Due to the delay of the bypass registers, the Daisy-chain approach is more efficient compared to testing all cores in sequence.

Assume the system in Figure 103 where p_a=10, p_b=20, p_c=30 (number of test patterns (vectors)) and f_a=f_b=f_c=10 (number of flip-flops). When the cores are tested in a sequence the test time of the system is 720 (10×(10+1+1)+(20×(10+1+1)+ (30×(10+1+1)). Note that the terms +1+1 are due to the bypass registers. However, using the approach proposed by Aerts and Marinissen, the test time for the system is reduced to 630 (10×30+10×(20+1)+10×(10+1+1)). The test application using this scheme is given by:

$$T = \sum_{i=1}^{|C|}\left((p_i - p_{i-1}) \cdot \left(i - 1 + \sum_{j=i}^{|C|}\left\lceil \frac{f_j}{N} \right\rceil\right)\right) + p_{|C|} \qquad (8.5)$$

where p_0=-1. Note that the indices in Equation (8.5) are rearranged into a non-decreasing number of patterns.

2.4 Test Bus Architecture

The Test Bus architecture proposed by Varma and Bahtia [267] is a combination of the Multiplexing Architecture and the Distribution Architecture. If a single Test Bus is assumed, all tests are tested in a sequence where the full TAM width is given to each core at a time; as in Multiplexing Architecture (Section 2.1). However, if several Test Buses are assumed, only one test can be active at a time on each Test Bus since the testing on a Test Bus is scheduled sequentially. But, concurrent testing is achieved since multiple Test Buses can be active at a time. An example of the Test Bus architecture where the *N* TAM wires are partitioned into three Test Buses each of width w_1, w_2, and w_3, respectively, in such a way that w_1+w_2+w_3=N is in Figure 104.

2.5 TestRail Architecture

Marinissen *et al.* [184] proposed the TestRail architecture, which basically is a combination of Daisychain and Distribution architecture. The advantage of Daisychain is that it allows concurrent as well as sequential testing. The

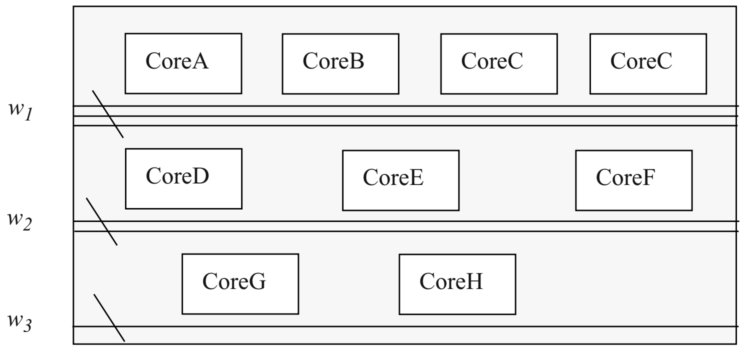


*Figure 104.*Test Bus architecture.

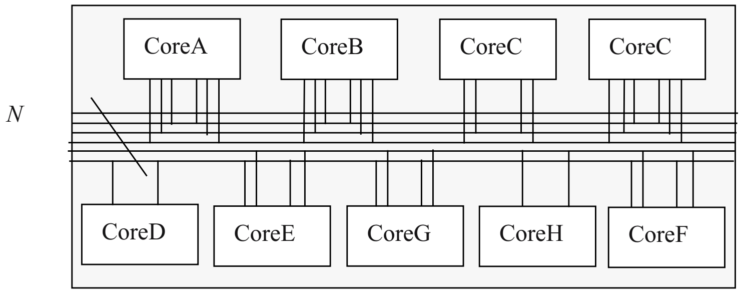


Figure 105. Test Bus architecture.

concurrent execution means that more than one wrapper can be active at a time, which makes it possible to perform external testing (interconnection testing or cross-core testing). The N TAM wires are partitioned into n TestRails each of width w_i in such a way that $w_1+w_2+...+w_n=N$.

2.6 Flexible-Width Architecture

It is also possible to view the TAM wires as a set of single wires that are assigned to cores in a flexible way; Flexible-Width Architecture proposed by Iyengar *et al.* [117]. An example of the Flexible-Width Architecture is in Figure 105.

Wang *et al.* [276] proposed a Test Access Control System (TACS) suitable for flexible-width TAM design. The architecture can be used for one-level or multi-level test hierarchy. It means it is possible to design an architecture for systems where cores are embedded within cores (Section 3.5, Hierarchy - Cores Embedded in Cores, page 85). Figure 106 shows an example of TACS. In Figure 106(a) the cores are divided into two sessions (session 1 with coreA, coreB, and coreC, and session 2 with core D and core E). The two sessions (session 1 and session 2) must be scheduled in a sequence while cores in the same session are executed concurrently (coreD and coreE can be tested at the same time). Figure 106(b) shows an example of the multi-tier architecture which allows a flexible width schedule.

2.7 Core Transparancy

Macro Test [11] and Socet [69] make use of the core-internal functional paths as TAMs. Transparent paths are created using neighboring cores. Yoneda and Fujiwara [283, 284, 285, 286] have also proposed techniques making use of transparent paths. The advantage of such approaches is that the need for additional TAMs is reduced. The disadvantage is that the testing of a core depends on the availability of paths at its neighboring cores.

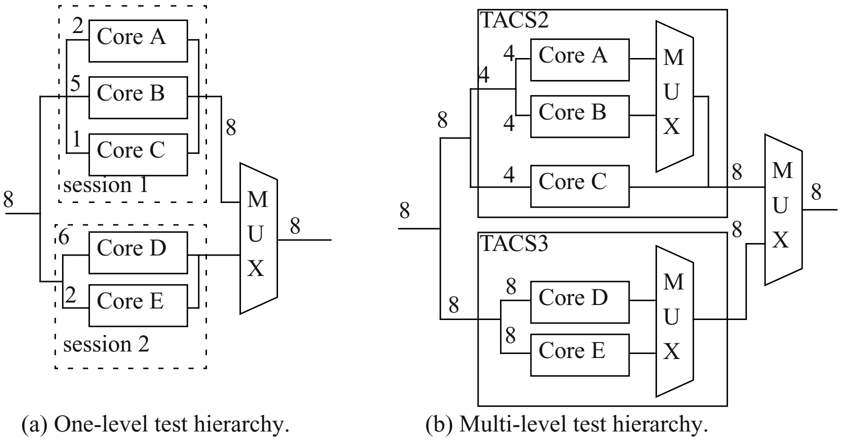


*Figure 106.*Examples of TACS [276].

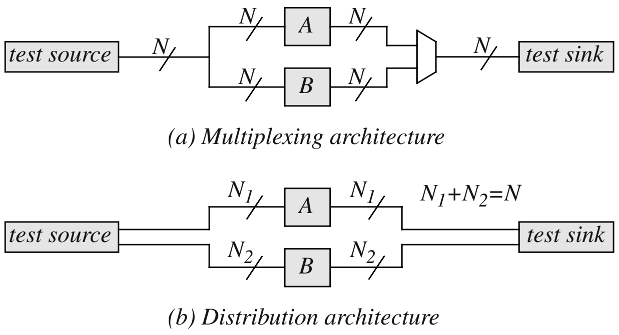


*Figure 107.*Multiplexing architecture and distribution architecture [5].

3 TEST TIME ANALYSIS

Larsson and Fujiwara [162, 163, 173] analyzed the MA (Multiplexing architecture) [5] and the DA (Distribution architecture) [5] (Figure 107) from a test time perspective. In MA each core is given all TAM bandwidth when it is to be tested, which means the tests are scheduled in a sequence (see Section 2.1). For cores where the number of scan-chains is smaller than the TAM bandwidth, the TAM is not fully utilized. Furthermore, since the test time is minimized at each core, the test power is maximized, which could damage the core.

In DA, each core is given its dedicated part of the TAM, which means that initially all cores occupy a part of the TAM (see Section 2.2). The DA approach assumes that the bandwidth of the TAM is at least as large as the number of cores, ($N_{tam} \geq |C|$).

In the analysis of the test time the IC benchmark (data as in Table 6) is used. Both for the MA and the DA each scan chains must include at least 20 flip flops and the size of the TAM is in the range $|C| \le N_{tam} \le 96$, Figure 108. The lower bound of the test time, excluding the capture cycles and the shift out of the last response, is given by [5]:

$$\sum_{i=1}^{|C|} \frac{ff_i \times tv_i}{N_{tam}} \qquad (8.6)$$

Core c_i	Flip-flops ff_i	Test vectors tv_i
1	6000	1100
2	3000	900
3	2600	1100
4	1500	1000
5	1500	800
6	800	1000
7	800	400
8	600	500
9	300	300
10	150	400
11	120	150

Table 6. Design data for benchmark IC [5].

The results in Figure 108 indicate that the DA is not efficient for low TAM size while MA is less efficient as the TAM size increases. It is important to note that the reason why the MA performs worse as the TAM with increases is the limitation that no scan-chain can be of shorter length than 20 flip-flops.

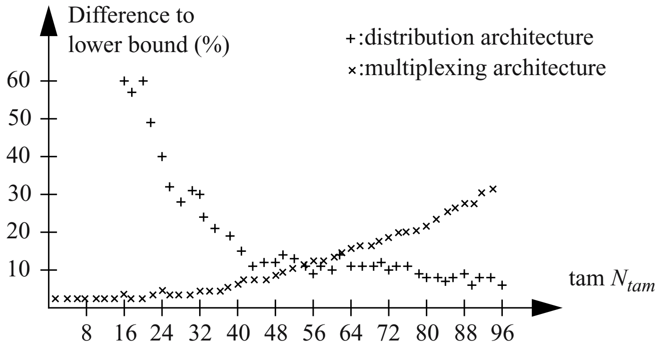


Figure 108. Difference to lower bound for the Multiplexing and the Distribution Architecture at various TAM widths.

Chapter 9

Test Scheduling

1 INTRODUCTION

In this chapter we will discuss test scheduling. The fundamental test scheduling problem is to determine the order in which the tests (the sets of test vectors) are to be applied; that is to give a start time to all tests, where the objective is to minimize a cost function while ensuring that no constraints are violated. The cost function is often related to the test application time, which either is to be minimized or given as a constraint, and/or to the need of additional DFT logic and wires, such as routing of a dedicated TAM. The constraints that must be considered are, for instance, test conflicts, resource sharing (see Chapter 6, Test Conflicts, page 77) and power constraints (see Chapter 7, Test Power Dissipation, page 89). It is important when discussing the test scheduling problem to clearly describe which assumptions, such as TAM architecture, and constraints, such as power consumption, that are taken into account.

System testing can be viewed as black-box testing (no knowledge is known of the system) as in Figure 109 where the test response is the produced outputs for the given input test stimulus. If the test stimulus for the system is applied and the produced test response is equal to the expected test response, the system is fault free in respect to the applied test set. The test patterns (test stimulus and test response) can be stored in an ATE, for instance. A finer grain view of system testing is in Figure 110 where a core-based system is shown.

As an example to illustrate the importance of test scheduling, assume a design with two modules each with one scan-chain. The length of the scan-chains is 100 flip-flop respectively 200 flip-flops, where the former is tested with 50 test vectors and the latter with 10 test vectors. A straight forward approach to organize the testing is to connect the two scan-chains into a single

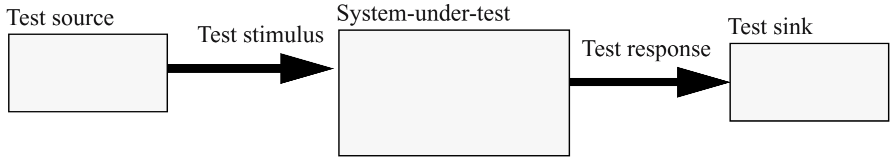


Figure 109. System testing.

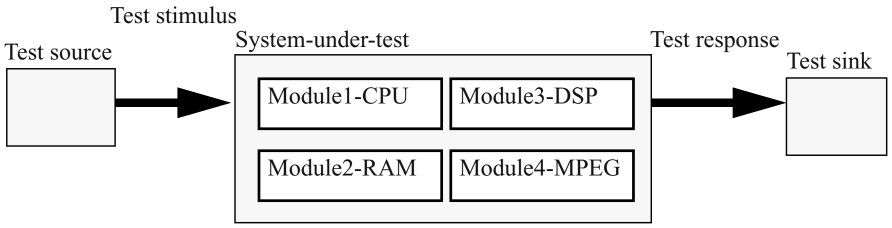


*Figure 110.*Modular view of system testing.

longer scan-chain of 300 flip-flops where 50 test vectors are required (the 10 test vectors for the scan-chain of length 200 are loaded simultaneously with 50 test vectors for the scan-chain of length 100). The testing time for such an approach can be estimated to 300×50=15000 cycles (capture cycles and shift-out of the last test response are neglected). However, if the design consists of two separate modules, the testing can be performed as a sequence where the modules are tested one after the other. The testing time is then approximately 100×50+200×10=7000. The example demonstrates that rather trivial considerations during the organization of the tests can lead to substantially lower testing time, and further as the testing times often are related to the ATE memory size, an effective organization of the tests means that an ATE with a smaller memory can be used.

In general, the tests in a system can be organized sequentially (Figure 111(a)) or concurrently (Figure 111(b)). In sequential test scheduling, all tests are ordered in a sequence where only one test is applied at a time, while in concurrent test scheduling more than a single test can but does not have to be active at a time. The test application time can be minimized by ordering tests in an efficient manner. Four basic scheduling strategies can be distinguished, namely:

- *Nonpartitioned testing*,
- *Partitioned testing with run to completion,*
- *Partitioned testing* or *preemptive testing*, and
- *Pipelined testing* or *daisychained testing*.

The four scheduling strategies are illustrated in Figure 112. In *nonpartitioned testing* no new tests are allowed to start until all tests in a session are completed. As examples of techniques using this scheduling scheme are the ones proposed by Zorian [287], Chou *et al.*[41], and Larsson and Peng [158]. In *partitioned testing with run to completion* a test may be scheduled to start as soon as possible. For instance, $test_3$ is started before $test_1$ is completed. It

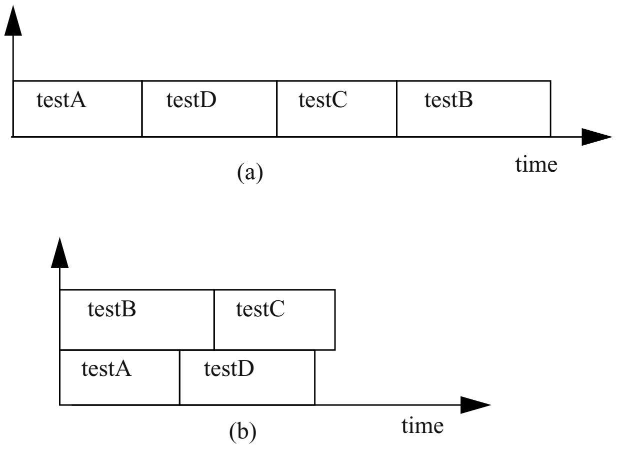


*Figure 111.*Sequential test scheduling (a) versus concurrent test scheduling (b).

means that the idea of test sessions is neglected. Chakrabarty [25], Muresan *et al.* [202] and Larsson and Peng [158] have proposed techniques for this type of test scheduling. Finally, in *partitioned testing* or *preemptive testing* the tests may be interrupted at any time. The requirement is that all tests must be completed by the end of testing. In Figure 112(c) $test_1$ is interrupted and is scheduled to be executed as two segments with indexes *a* and *b*. Iyengar and Chakrabarty [110], Larsson and Fujiwara [162], and Larsson and Peng [171] have proposed preemptive test scheduling techniques. *Pipelined testing (daisy-chain testing)* - the tests are pipelined through the testable units. An example of pipelining is scan testing where new test stimulus is shifted in at the same time as the test response from previous test stimulus is shifted out. Aerts and Marinissen have investigated this type of test scheduling [5].

The above scheduling strategies do not separate between the *transportation* and the *application* of test data. The transportation of test data can make use of a strategy while the actual application can make use of another strategy. For instance, Larsson *et al.* investigated the use of a test bus where the test data transportation is sequential and buffers are introduced at each core making it possible to apply more than one test at a time [152, 153]. The classification of testing in Figure 112 apply both to sequential scheduling and concurrent scheduling, however, partitioned testing with run to completion will in sequential scheduling be the same as nonpartitioned testing.

The test scheduling problem for systems where all tests are given and all are assigned a fixed testing time and the objective is to minimize the test application time is in the case when sequential testing is assumed trivial. The

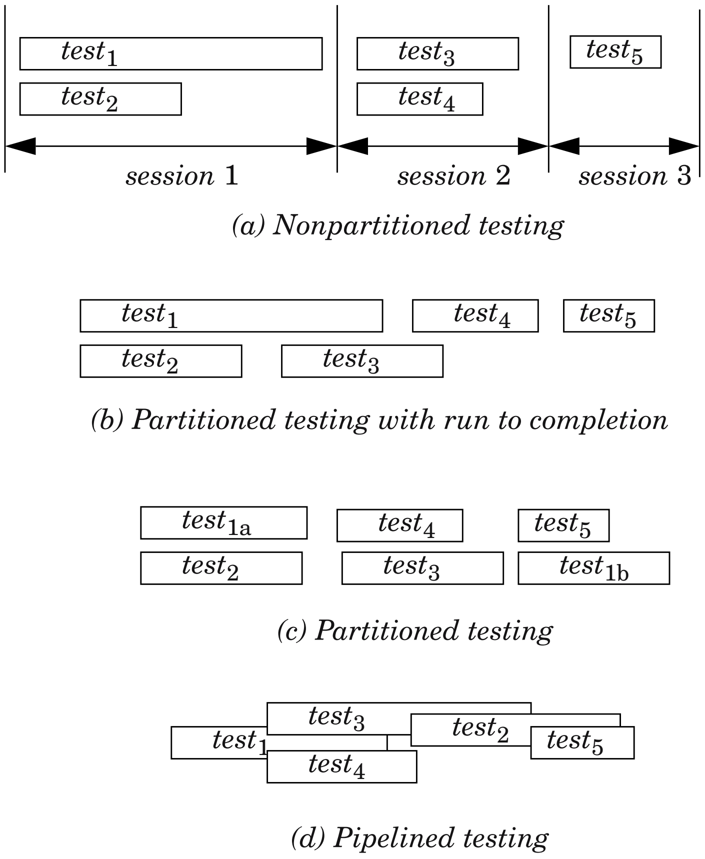


Figure 112. Scheduling approaches.

optimal test application time $\tau_{application}$ is for a system with N tests each with a test time τ_i (i={1..N} given by:

$$\tau_{\text{application}} = \sum_{i=1}^{N} \tau_i \qquad (9.1)$$

The assumptions that only one test at a time can be active at any moment, and that all tests have to be applied, means that any order of the tests is optimal. An algorithm iterating in a loop over all tests is required and at each instance of the iteration one test is selected and given a start time. The computational complexity of such an algorithm depends linearly on the number of tests, $O(n)$ - n is the number of tests and hence the algorithm is polynomial (P).

Above it was shown that it is trivial to develop an optimal test schedule in respect to test time for a sequential architecture when all tests are given a fixed test time and the objective is to minimize the test application time. In sequential testing, only one test at a time can be active, and that means that no constraint can limit the solution. In concurrent test scheduling, where more than one test can be applied at a time, a conflict often limits the solution. The problem to minimize the test application time using concurrent test scheduling for a system where all tests are given a fixed testing time *under no constraint* is trivial. All tests are simply started at time point zero. The optimal test application time $\tau_{application}$ is for a system with N tests each with a test time τ_i ($i=\{1..N\}$ given by:

$$\tau_{\text{application}} = \max\{\tau_i\} \tag{9.2}$$

The concurrent test scheduling problem, more than one test can be executed at the same time, is in general not *NP*-complete. However, the concurrent test scheduling problem *under constraint* is *NP*-complete. We will discuss this in more detail below.

2 SCHEDULING OF TESTS WITH FIXED TEST TIME UNDER TEST CONFLICTS

A test scheduling approach is proposed by Garg *et al.* for systems where the test time for each test is fixed in advance and the objective is to minimize the test application time while the constraints among the tests are considered [65]. Test conflicts among the tests make it impossible to execute all test concurrently. A system and its tests can be modeled using a *resource graph*, see Figure 113, where the tests in the system are on the top level and the resources are on the bottom level. An edge between nodes at different levels indicates that a test t_i tests a resource r_j or a resource r_j is needed to perform test t_i. This means that the resource graph captures information on resource conflicts. For instance, in Figure 113 both test t_1 and test t_3 use resource r_1 which means that test t_1 and test t_3 can not be scheduled simultaneously.

Given a resource graph, a *test compatibility graph* (*TCG*) (Figure 114) can be obtained where the nodes define the different test plans and the edges specify that two tests are compatible. From the test compatibility graph in Figure 114 it can be determined that test t_1 and t_2, for example, can be executed concurrently.

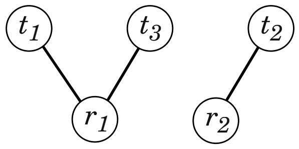


Figure 113. A resource graph.

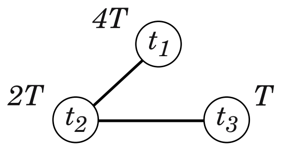


Figure 114. A test compatibility graph.

The problem of finding the minimal number of test groups such that tests within a group can be executed concurrently can be formulated as a clique partitioning problem [65]. Finding the minimal clique cover on a TCG is a *non-deterministic polynomial* (*NP)* complete problem, which justifies the use of heuristics [65].

Given a TCG, Garg *et al.* construct a binary tree called *time zone tree* (*TZT*) [65]. Each node in the TZT represents a time zone and its constraints, *i.e.* tests associated with the zone. An illustrative example of the approach proposed by Garg *et al.* is presented in Figure 115. The example is based on the test compatibility graph shown in Figure 114 which is obtained from the resource graph illustrated in Figure 113.

Initially the root $R = <\varnothing, \Sigma\, l(t_i)>$ is unconstrained ($\varnothing$) and of length $7T$ ($\Sigma\, l(t_i)=4T+2T+T$). When a test t_k is assigned to R, two branches are created with two nodes, the first with the constraint t_k and length $l(t_k)$, and the second with no constraint ($\varnothing$) and length $\Sigma\, l(t_i) - l(t_k)$.

For the first test, the test with the maximum length is selected. If several such tests exist, favour is given to the test with the highest compatibility. For all other tests, the selection is based on the cost function $CF(t_i)$, where the selected test t_i has the least value according to the cost function:

$$CF(t_i) = \sum_{j=1}^{|T|} (l(t_j) - Opp(t_j/t_i)) \tag{9.3}$$

where:

$$Opp(t_j/t_i) = \begin{cases} l(Z_k), \text{ if } t_j \text{ is compatible with } t_i \\ l(Z_k), \text{ if } t_j \text{ is not compatible with } t_i \text{ and } l(Z_k) > l(Z_k) \\ 0, \text{ otherwise.} \end{cases}$$

In the example, given in Figure 115, t_1 is selected first and when appended to the tree, two branches (or zones) Z_1 and Z_2 are created, see Figure 115(a), (b). Next when t_2 is assigned to zone Z_1, node 3 is appended to the tree with constraints and length as shown in Figure 115 (c). Node 4 is also created at this time, denoting that Z_4 is of length $2T$ and constrained by t_1 only. Finally, test t_3 is assigned, resulting in the TZT shown in Figure 115(e) and the corresponding diagram is in Figure 115(f). The scheduling diagram is directly

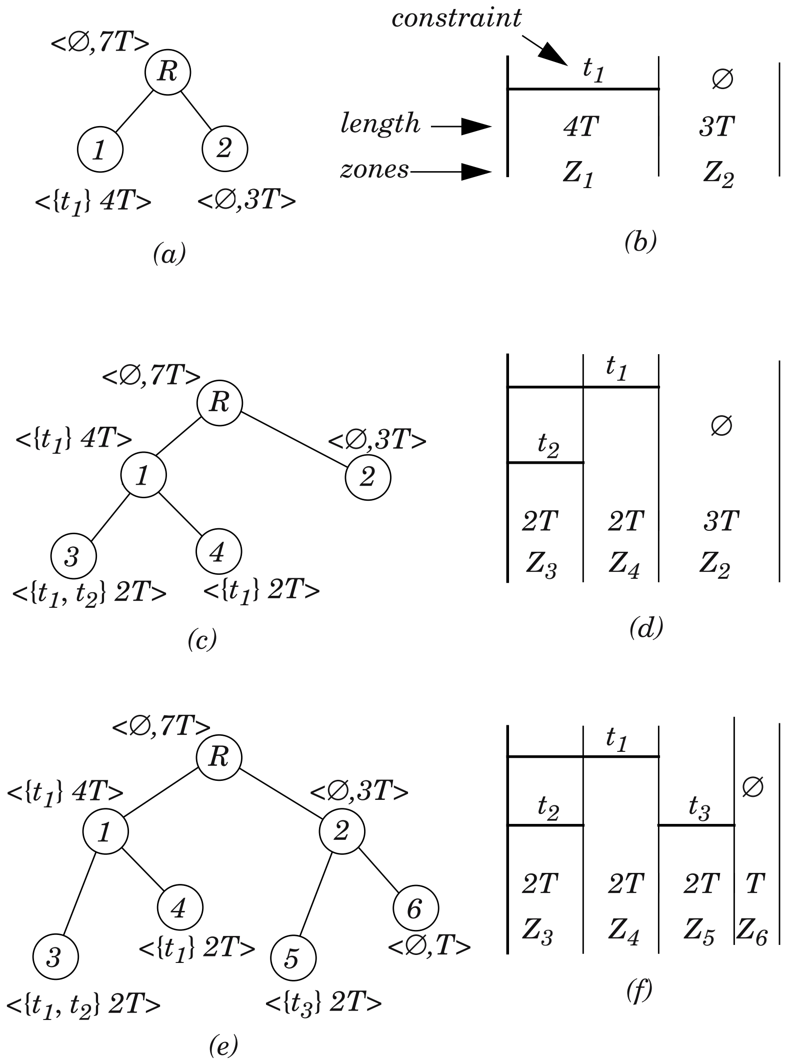


Figure 115. The test scheduling approach proposed by Garg *et al.* [65].

derived by inspection of the leaves of the *TZT* from left to right. And the worst case computational cost of the approach is of the order $O(n^3)$ [65].

Chakrabarty proposes a test scheduling algorithm where test time is minimized while test constraints are considered. Chakrabarty shows that the test scheduling problem is equivalent to open-shop scheduling [81]. And then a test scheduling algorithm is proposed, see Figure 116 [27]. In the approach tests are scheduled as soon as possible. If a conflict among two tests occurs

```
Procedure SHORTEST_TASK_FIRST({$t_i$})
begin
for $i$:= 1 to $m$ do /* there are $m$ tasks */
  $start\_time_i$ := 0;
while $flag$ = 1 do begin
  $flag$ = 0;
  for $i$:= 1 to $m$ do
    for $j$ := $i$ + 1 to $m$ do
      if $x_{ij}$=1 then
        /* $x_{ij}$=1 if $i$ and $j$ are conflicting */
        if OVERLAP($i,j$) then begin
          if $start\_time_i+l_i>start\_time_j+l_j$ then
            $start\_time_i+l_i:=start\_time_j+l_j$
          else
            $start\_time_i+l_i:=start\_time_j+l_j$;
          $flag$ := 1;
        end;
      end;
end;
```

*Figure 116.*The shortest-task-first procedure [28].

the test with the shortest test time is scheduled first. The algorithm in Figure 116 has a worst case execution time of $O(n^3)$ for n tests.

Other test scheduling approaches where test time is minimized while considering test conflicts are proposed by Kime and Saluja [132], Craig *et al.* [47] and Jone *et al.* [128].

An approach where the test application time is minimized while constraints on power consumption are considered is proposed by Zorian [287]. The tests in the system are partitioned in such a way that tests in a partition can be executed concurrently and the power dissipation within each partition is below the maximal allowed power dissipation. The partitioning is guided by the placement of the blocks in the system. Tests at blocks which are physically close to each others are placed in the same partition. This approach to partitioning minimizes the amount of control lines added for controlling the tests of the system since the same control line is used for a complete partition.

The system ASIC Z is used to illustrate the approach by Zorian, see Figure 117, where the design is partitioned into four partitions, marked with numbers 1 to 4. Table 7 gives the design data for this example and the test schedule for ASIC Z is shown in Figure 118.

Another approach to test scheduling where test application time is minimized while constraints among tests and test power consumption are considered is proposed by Chou *et al.* [40]. The approach works on a TCG with added power constraints and test length information constructed from a resource graph (Figure 119). In order to minimize the complexity of the test controller, the tests are assigned to test sessions and no new tests are started

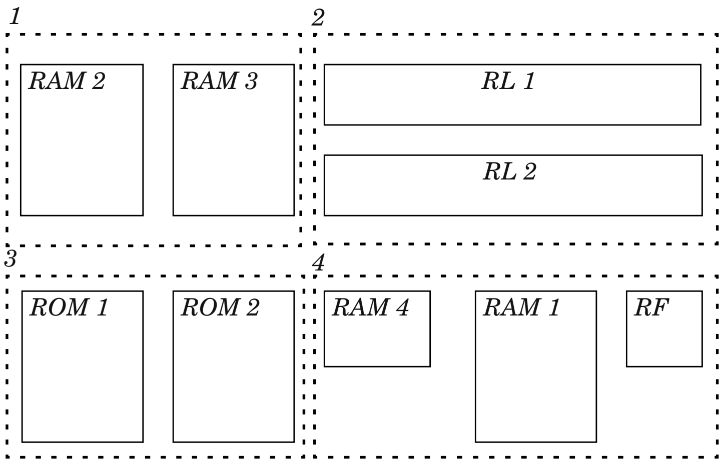


*Figure 117.*ASIC Z floor-plan and test partitioning.

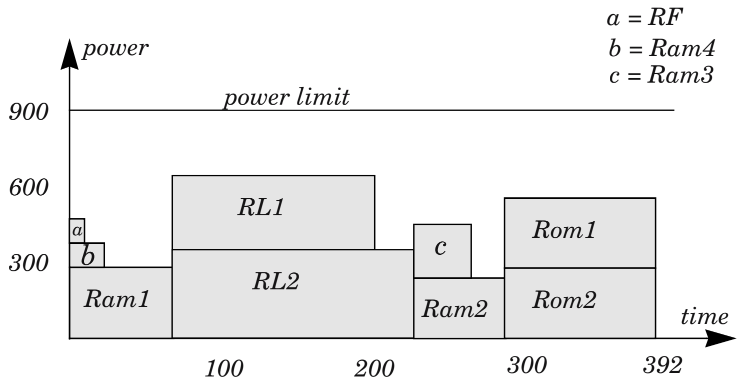


*Figure 118.*ASIC Z test schedule using the approach proposed by Zorian [287].

until all tests in a session are completed. The power dissipation for a test session s_j is given by:

$$P(s_j) = \sum_{t_i \in s_j} P(t_i) \qquad (9.4)$$

The power constraint is defined as:

$$P(s_j) \le P_{max} \qquad \forall j \qquad (9.5)$$

From the TCG a *power compatible set* (PCS) is derived where the tests in each set (clique) are time compatible with each other and do not violate the power constraints. For instance PCS={t_4,t_3,t_1} in such a set, as illustrated in Figure 119.

A *power compatible list* (PCL) *H* is a PCS such that the elements in *H* are arranged in descending order of length. For instance, the PCL for PCS={t_4, t_3,

Block	*Test Time*	*Idle Power*	*Test Power*
RL1	134	0	295
RL2	160	0	352
RF	10	19	95
RAM1	69	20	282
RAM2	61	17	241
RAM3	38	11	213
RAM4	23	7	96
ROM1	102	23	279
ROM2	102	23	279

Table 7. ASIC Z characteristics.

$t_1\}$ is $H=\{t_1, t_3, t_4\}$ since $l(t_1) \geq l(t_1) \geq l(t_1)$. A *derived* PCL (DPCL) is an ordered subset of a PCL or DPCL such that the test length of the first element is strictly less than the test length of the first element in the original PCL. For instance the DPCLs of the PCL $H=\{t_1, t_3, t_4\}$ are $H'=\{t_3, t_4\}$ and $H''=\{t_4\}$. A *reduced* DPCL (RDPCL) set is the set of all DPCLs derivable from all possible PCLs such that each DPCL appears only once. Furthermore, if DPCL $h_1=(t_1, t_2,...,t_m)$ and DPCL $h_2=(t_{i1}, t_{i2},..., t_{ik})$ such that $t_{ij} \in h_1, j=1, 2,..., k$ and $l(h_1)=l(h_2)$, then h_2 is removed from the TDPCL set.

Given a TCG, as shown in Figure 119, the steps in the approach by Chou *et al.* are as follows:

1. All possible cliques are identified: $G_1=\{t_1, t_3, t_5\}$, $G_2=\{t_1, t_3, t_4\}$, $G_3=\{t_1, t_6\}$, $G_4=\{t_2, t_5\}$, $G_5=\{t_2, t_6\}$.
2. All possible PCLs are: (t_1, t_3), (t_1, t_5), (t_3, t_5) obtained from G_1, (t_1, t_3, t_4) from G_2, (t_1, t_6) from G_3, (t_2, t_5) from G_4 and finally (t_2, t_6) from G_5.
3. The reduced DPCLs are: (t_1, t_5), (t_5), (t_3, t_5), (t_1, t_3, t_4), (t_3, t_4), (t_4), (t_1, t_6), (t_2, t_5), (t_2, t_6).
4. Using a minimum cover table, see Table 8, to find an optimum schedule over the compatible tests, the test schedule is: (t_3, t_4), (t_2, t_5), (t_1, t_6) with a total test time of 120.

The test schedule achieved on the ASIC Z system by the approach proposed by Chou *et al.* is shown in Figure 120. The total test application time is 331; the approach proposed by Zorian needs 392 time units, see Figure 118.

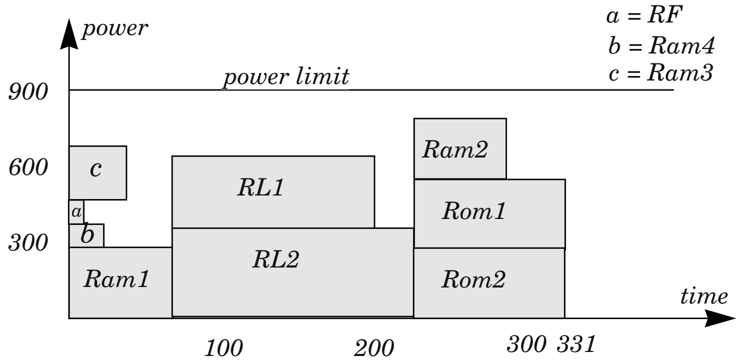


*Figure 120.*ASIC Z schedule using the approach proposed by Chou *et al.* [40].

The identification of all cliques in the TCG graph is an NP-complete problem and therefore a greedy approach such as proposed by Muresan *et al.* is justified where test time is minimized while test constraints and power consumption are considered [202].

A basic assumption in the approaches by Chou *et al.* [40] and by Zorian [287] is that no new tests are started until all tests in a test session are all completed. Due to this assumption the test controller is minimized. However, this assumption is not valid in the approach proposed by Muresan *et al.* [202].

Muresan *et al.* define an extension called the *expanded compatibility tree* (ECT) of the compatibility tree introduced by Jone *et al.,* where the number of children is generalized. For instance, assume tests t_1, t_2, t_3 and t_4, where t_2, t_3 and t_4 are compatible with t_1, see Figure 121. However, t_2, t_3 and t_4 are not compatible with each other. Assume that the test length $l(t_2)+l(t_3)<l(t_1)$ and t_4 is to be scheduled. If $l(t_4)\leq l(t_1)-(l(t_2)+l(t_3))$ then t_4 can be inserted in the ECT.

Neither the approach by Chou *et al.* [40,41] nor that by Muresan *et al.* [202] consider the routing of control lines, which was considered by Zorian [287] by partitioning the tests due to their physical placement in the system.

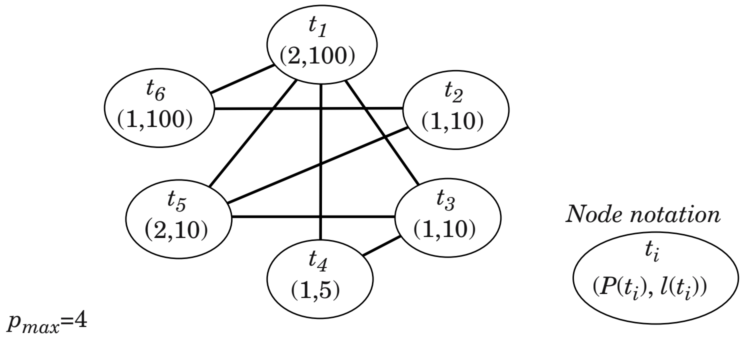


*Figure 119.*TCG with added power constraint and test length for each test.

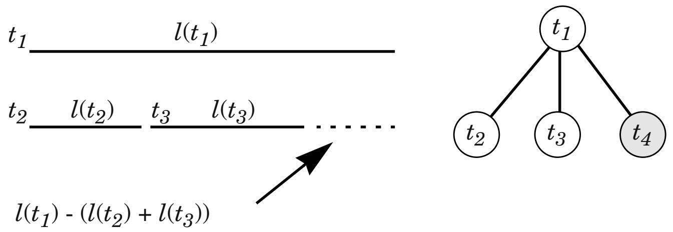


*Figure 121.*Merging example by Muresan *et al.* [202].

```
sort tests according to a key and put them in a list L.
time point $\tau$=0
while L is not empty begin
  for i = 1 to |P| begin //all tests (items) in the list
    if $item_i$ = ok to schedule then begin
      remove $item_i$ from L
      schedule $item_i$ at time $\tau$
    end //if
  end // for
  increase $\tau$ to the next timepoint when a test stops
end //while
end.
```

*Figure 122.*Test scheduling technique by Larsson and Peng.

Larsson and Peng proposed a constructive power-constrained test scheduling approach running at $O(n^2)$ - n is the number of tests [158]. The tests are initially sorted based on test length and scheduled in order. Table 9 collects the results from the approach by Zorian, Chou *et al.* and Larsson and Peng on the design ASIC Z. The results indicate that a straight-forward algorithm can produce high quality solutions.

Wang *et al.* [275] proposed a Simulated Annealing technique ([131]) to schedule memory BIST cores in such a way that the test time is minimized.

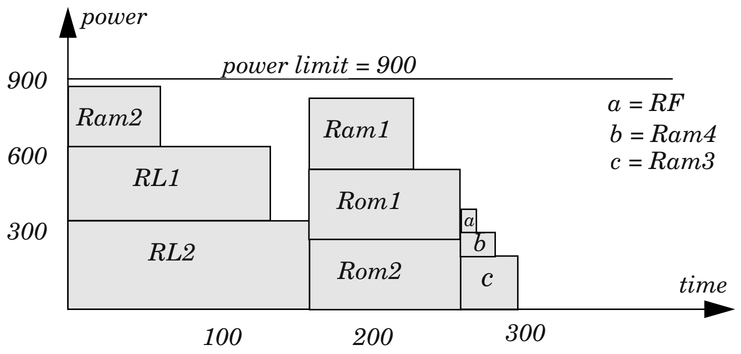


*Figure 123.*Test schedule achieved using the heuristi proposed by Larsson and Peng [161, 164, 165] on ASIC Z.

RDPCL	t_1	t_2	t_3	t_4	t_5	t_6	*Cost*
(t_1, t_3, t_4)	x		x	x			100
(t_1, t_5)	x				x		100
(t_1, t_6)	x					x	100
(t_2, t_6)		x				x	100
(t_3, t_5)			x		x		10
(t_2, t_5)		x			x		10
(t_3, t_4)			x	x			10
(t_5)					x		10
(t_4)				x			5

Table 8. Covering table.

Test session	*Zorian*		*Chou et al.*		*Larsson and Peng*	
	Time	*Blocks*	*Time*	*Blocks*	*Time*	*Blocks*
1	69	Ram1, Ram4, RF	69	Ram1,Ram3, Ram4,RF	160	RL2, RL1, Ram2
2	160	RL1, RL2	160	RL1, RL2	102	Ram1,Rom1, Rom2
3	61	Ram2, Ram3	102	Rom1, Rom2, Ram2	38	Ram3, Ram4, RF
4	102	Rom1, Rom2				
Total time:	392		331		300	

Table 9. A comparison of different test scheduling approaches on ASIC Z.

Each core is given a fixed testing time and a fixed power consumption value when active, and there are no constraint among the tests since each testable unit (memory core) has its dedicated BIST. The objective is to schedule the tests in such a way that the test application time is minimized while power constraints are met. Flottes *et al.* [61, 62] compared the control over-head for session-based (nonpartitioned testing) and session-less scheduling (partitioned testing with run to completion). The results indicate that relative to the size of the system, the type of controller has minor impact.

2.1 Preemptive test scheduling

A way to minimize the impact of test conflicts is to make use of preemptive scheduling (partitioned testing) (Figure 112). The idea is to stop a test as soon as a conflict appears, and by selecting an alternative test, the conflict is avoided. Iyengar and Chakrabarty proposed preemptive test scheduling [110].

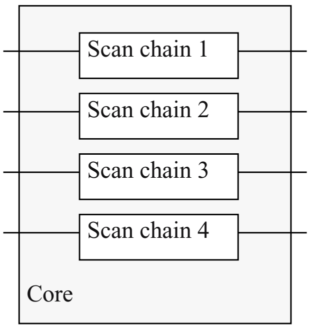
3 SCHEDULING OF TESTS WITH NON-FIXED (VARIABLE) TESTING TIMES

The test time for a testable unit (core) does not have to be fixed. For instance, the scan-chains at a scan-tested unit can often be configured into one or more chains. If a single chain is used the testing time becomes long while if the scanned elements are connected into *n* chains, it is possible to load all *n* chains concurrently, and hence reduce the test time. For instance, assume a core with four scan-chains (Figure 124). The scan-chains can be assigned to a wire each or as in Figure 124(b) and (c) where two respectively three wires are used. There is a clear trade-off between test time and the number of used wires. Few wires means that the scanned elements are configured into a few longer wrapper chains, while if a high number of wires are used, the chains become shorter. Shorter wrapper chains lead to shorter loading/unloading time and hence shorter testing times. However, as can be seen in Figure 124(b), increasing number of wires does not guarantee in lower testing times for a core since the length of the wrapper chain is actually not decreased when going from a configuration using two wires Figure 124(b) to a configuration using three wires Figure 124(c). Further, all cores in the system have to be considered.

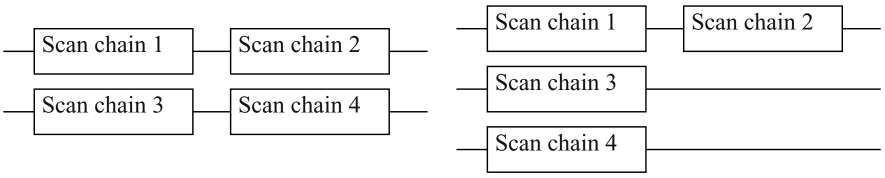
Several techniques have been proposed to solve the problem. The techniques have different assumptions and take different constraints into account. The constraints when designing test solutions have been described in Chapter 6, Test Conflicts, on page 77. Below in Section 3.1 the introduced idle bits, which should be minimized, are described. The general problem is to configure the scan-chains into wrapper-chains and assign a start time for each test in such a way that the total test time is minimized while considering test conflicts.

3.1 Idle Types

Test scheduling having the objective to minimize the test application time means that the idle times (idle bits) in a test schedule should be minimized. The idle bits are useless data that have to be stored in the ATE. The idle bits



(a) A core with four scan chains.



(b) (c)

Figure 124.(a) A core with four scan chains. (b) The four scan chians configured into two wrapper chains making use of two wires. (c) The four scan chains configured into three wrapper chains making use of three wires.

can be found at different places in a test schedule. Therefore, Goel and Marinissen [78] defined the following three *idle types* for a test schedule:

- Type 1 Idle Bits: Imbalanced TAM Test Completion Times,
- Type 2 Idle Bits: Module Assigned to TAM of Non Pareto Optimal Width, and
- Type 3 Idle Bits: Imbalanced Scan Chains in Module.

The idle bit types are described below. And for flexible-width TAM, an additional idle type is defined.

3.1.1 Imbalanced TAM Test Completion Times

The *Type 1 idle bits* are by Goel and Marinissen [78] defined as the idle time that is:

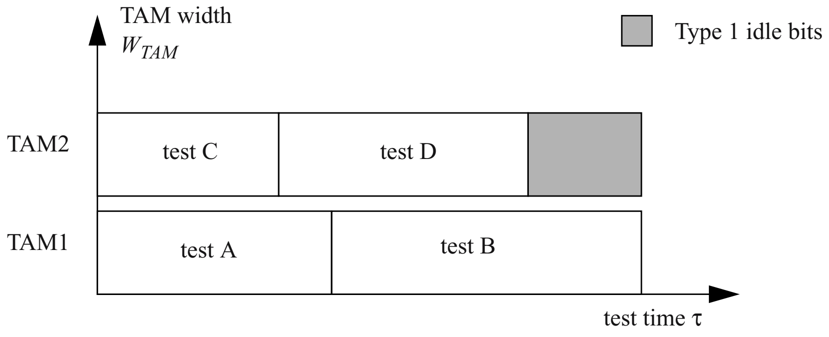


*Figure 125.*Illustration of Type 1 Idle Bits.

"between the completion time of an individual TAM and the overall completion time, the TAM in question is not utilized".

For an example system with four tests, A, B, C, and D, scheduled on two TAMs, TAM1 and TAM2, as in Figure 125, the Type 1 idle bits are at the end of TAMs that are not limiting the solution.

3.1.2 Module Assigned to TAM of Non Pareto Optimal Width

For scan-tested cores the scanned elements (scan-chains and wrapper cells) are to be configured into at least one wrapper chain. A single wrapper-chain leads to long testing times, and therefore, increasing the number of wires assigned to a core, will reduce the test time. However, increasing the number of wrapper-chains will not always lead to lower testing times. Assume that the four scan-chains in Figure 124 each include 100 flip-flops. The shift-in/shift-out time if two wrapper-chains are used is 200 cycles (Figure 126(a)). However, if the number of wrapper-chains is increased to three (Figure 126(b)) the shift-in/shift-out time will not decrease.

The test time versus the number of wrapper chains (TAM wires) are in Figure 127 plotted for Core 11 the ITC'02 [189] design P93791. The test time when all scanned elements (scan chains and wrapper cells) are connected into a single wrapper chain is the longest, and the test time decreases as the num-

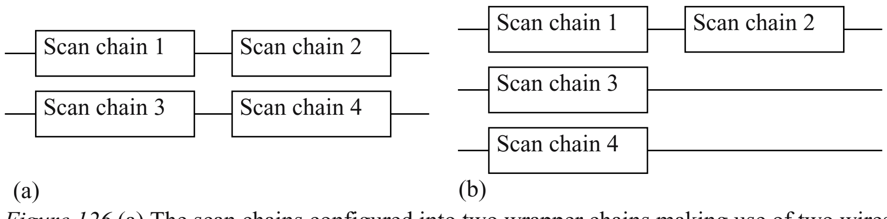


Figure 126.(a) The scan chains configured into two wrapper chains making use of two wires. (b) The scan chains configured into three wrapper chains making use of three wires.

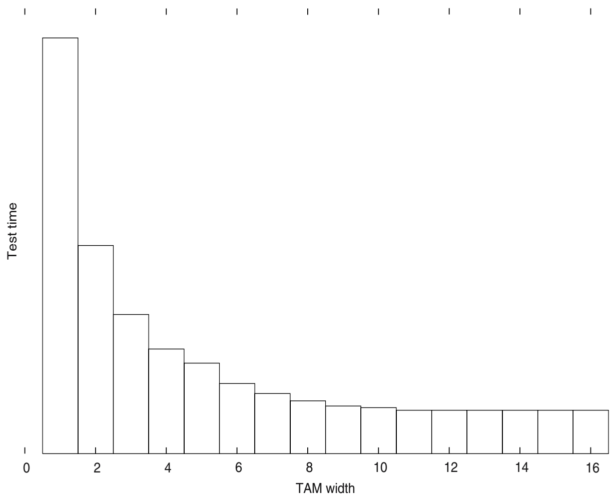


*Figure 127.*Test time versus number of wrapper chains (TAM wires) for Core 11 in ITC’02 benchmark P93791.

ber of wrapper chains increases. However, at some points, the test time does not decrease even if the number of wrapper chains increases. It means that for several TAM widths the test time is constant. For all points with the same test time, a *Pareto-optimal point* is the point where the least number of TAM wires is used [113]. Type 2 Idle Bits occur when a module is assigned to a *n* number of wrapper-chains where *n* is not a Pareto-optimal point.

3.1.3 Imbalanced Scan Chains in Module

The type 2 idle bits occur when assigning the scanned elements at a core into *n* wrapper-chains where the configuration at *n* is not a Pareto-optimal point. However, even if assigning the TAM width to Pareto-optimal points the cost $\tau \times w$ is not constant. For instance, Edbom and Larsson [57] plotted for TAM widths up to 64 TAM wires the Wrapper Design Cost (WDC), defined as $w_n \times \tau_n - \tau_1$, for core 1 in the ITC’02 design P93791 [189, 190] (Figure 128). Figure 128 shows that the cost test time×TAM is not constant, not even for the Pareto-optimal points. Type 3 idle bits come from the imbalance at scan-chains.

3.1.4 Other Types of Idle Bits

Idle bits may also occur due to test conflicts. In the flexible-width architecture idle bits may occur as indicated in Figure 129.

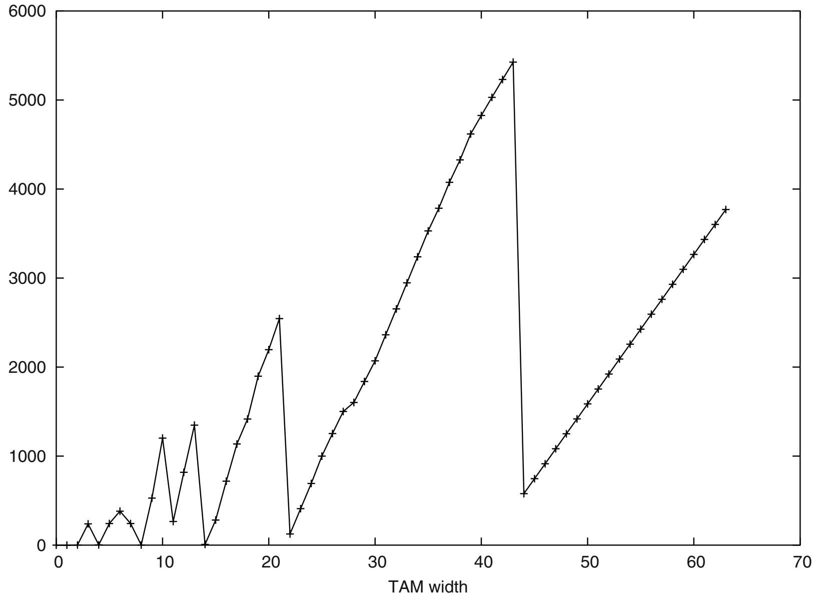


Figure 128. The plot of the wrapper design quality (WDC) - $\tau \times w - \tau_1 \times w_1$, test time ($\tau$) times the number of wrapper chains (w) for Core 1 in P93791.

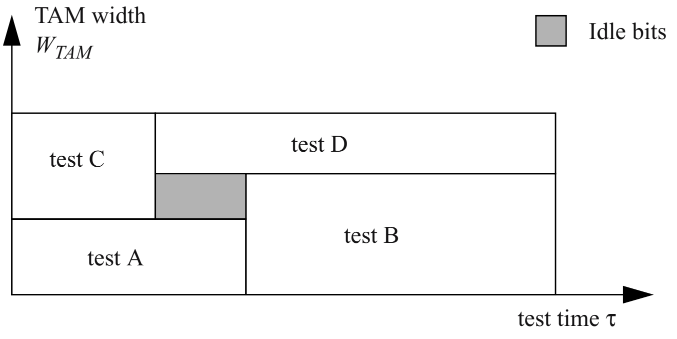


Figure 129. Illustration of idle bits.

3.2 SOC Test Scheduling with Fixed-Width TAM

Given for the problem is a core-based architecture with n cores $i=\{1..n\}$ and for each core i the following is given:

$sc_{ij}=\{sc_{i1}, sc_{i2},..., sc_{im}\}$ - the length of the scanned elements at core i are given where m is the number of scanned elements,

wi_i - the number of input wrapper cells,

wo_i - the number of output wrapper cells,
wb_i - the number of bidirectional wrapper cells,
tv_i - the number of test vectors,

For the system, a maximal TAM bandwidth W_{tam} which can be partitioned onto k number of TAMs, w_1, w_2, ..., w_k, in such a way that:

$$W_{tam} = \sum_{i=1}^{k} w_i \tag{9.6}$$

Problem 1: Find a test schedule that minimizes the test application time for a SOC system where each core is wrapped and scan-tested by one ATE test per testable unit for a fixed width TAM architecture.

Iyengar *et al.* [111, 113] formulated and proposed an algorithm to chain the scanned elements (wrapper inputs, wrapper outputs, wrapper bidirectionals, and scan-chains) for a core into *w* number of wrapper-chains and modeled the test scheduling problem in such a way that an ILP solver could be used. Due to the fact that for larger designs, the ILP becomes to costly, Iyengar *et al.* [112] proposed a heuristic for the test scheduling.

Goel and Marinissen [74, 75, 78] proposed the TR-Architect algorithm, which is composed of four main steps; *CreateStartSolution*, *Optimize-BottumUp*, *Optimize-TopDown*, and *CoreReshuffle*. The initial solution (*CreateStartSolution)* is created by assigning a single TAM wire to each core (similar to Distribution Architecture, page 109)). A limitation of Distribution Architect is that it is not applicable if the number of TAM wires are less then the number of cores (W_{TAM}<*i*). In the case when W_{TAM}<*i* the largest cores in respect to test time are each assigned to a dedicated TAM wire while test with shorter test times are share wires (the tests will be executed in a sequence on the wires).

In *Optimize-BottomUp* TAMs with shortest test time are merged with another TAM so that TAM wires are freed up. The free wires are then distributed in order to minimize the test time. In the *Optimize-TopDown* the focus is on TAMs that limits the solutions, the bottleneck. The idea is to merge TAM that limits the solution with another TAM. If TAM wires can be freed, they are distributed to reduce the testing time. In the *CoreReshuffle* step, cores at a TAM that is a bottleneck are moved to other TAMs in an attempt to reduce the total test time.

Note that pipelining (daisy-chaining) (see page 110) is used on each TAM in TR-Architect. An advantage of TR-Architect is the fact the width of each TAM is nor fixed; the algorithm defines the number of TAMs as well as their width.

Ebadi and Ivanov [56] proposed a technique where a Genetic algorithm [197] (Genetic Algorithms at page 18) is used for the optimization in assigning TAM wires to the cores.

3.3 SOC Test Scheduling with Flexible-Width TAM

The problem, similar to problem 1 (see page 133), however, the difference is in the TAM architecture where an architecture such as the one proposed by Wang *et al.* [276] is used. In this problem, instead of partitioning the TAM wires into a set of TAMs, there is a flexibility to basically use a TAM wire as soon as it is free.

Problem 2: The scheduling problem for systems with core tests with variable testing times, and one ATE test per testable unit for a flexible-width TAM architecture.

Huang *et al.* [97, 99] mapped the problem to the two-dimensional bin-packing problem and made use of a Best Fit Decrease algorithm for the optimization. Koranne proposed a technique based on the shortest job first algorithm [138]. Koranne and Iyengar [141] proposed a k-tuple technique based on sequence-pair proposed by Murata *et al.* [200]. The sequence-pair technique was developed for placement of VLSI components where the problem is to pack the components on a limited silicon area. A sequence pair for n modules is a pair of sequences of the n module names. Figure 130 show an example of packing and its sequential-pair representation (*abc, bac*), and it should be read as {*a* should be placed to the left of *c*, *b* should be placed to the left of *c*, *b* should be placed below *a*}.

Zou *et al.* [292] formulate the problem as a two-dimensional bin packing problem, similar to Huang *et al.* [97, 99], and is as Koranne and Iyengar [141] making use of sequence-pair [200]. For the optimization, Zou *et al.* [292] make use of Simulated Annealing (see Section 5.1.4). Xia *et al.* [279] are also making use of the sequence-pair technique [200] and a Genetic Algorithm for the test application time optimization.

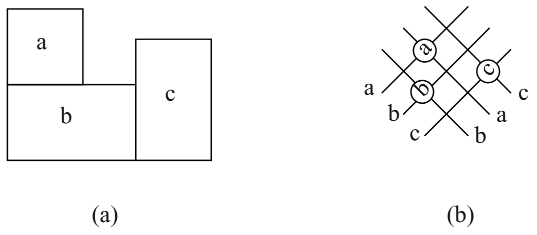


Figure 130. A packing example and its sequence-pair representation (*abc, bac*) [200].

Koranne proposed on-the fly reconfigurable wrappers [139, 140, 143] (see Reconfigurable Core Wrappers at page 106) and corresponding scheduling techniques. Larsson and Fujiwara [167, 172] make use of reconfigurable wrappers as proposed by Koranne [139, 140, 143], but in contrast to previous work, Larsson and Fujiwara [167, 172] modeled the problem as the independent job scheduling on identical machines, and demonstrated that a preemptive algorithm can produce optimal solution in linear time [23]. The test scheduling problem of core tests is equal to the independent job scheduling on identical machines [167, 172] since each test t_i at a core c_i, (i=1, 2, ..., n) with testing time τ_i is independent on all other core tests, and each TAM wire w_j (j=1, 2, ..., N_{tam}) is an independent machine used to transport test data [172, 171]. The LB (lower bound) of the test time for a given TAM width N_{tam} can be computed by [23]:

$$\mathrm{LB} = \max\left\{\max(\tau_i),\ \sum_{i=1}^{n} \tau_i / N_{tam}\right\} \tag{9.7}$$

The problem of independent job scheduling on identical machines can be solved in linear time ($O(n)$ for n tests) by using preemption [23]: assign tests to the TAM wires successively, assign the tests in any order and preempt tests into two parts whenever the LB is reached. Assign the second part of the preempted test on the next TAM wire starting from time point zero.

An example (Figure 131) illustrates the approach where the five cores and their test times are given. The LB is computed to 7 (Equation (9.7)) and due to that τ_i≤LB for all tests; the two parts of any preempted test will not overlap.

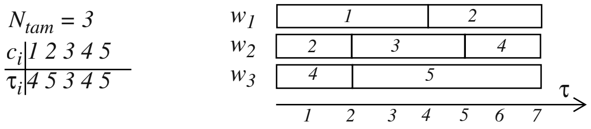


*Figure 131.*Optimal TAM assignment and preemptive scheduling.

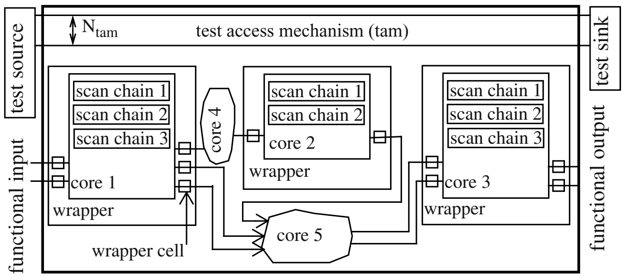


*Figure 132.*An example system with three wrapped cores (1,2,3) and two unwrapped cores (4,5).

The scheduling proceeds as follows: The tests are considered one by one, for instance, starting with a test at c_1 scheduled at time point 0 on wire w_1. At time point 4, when the test at c_1 is finished, the test at c_2 is scheduled to start. At time point 7 when LB is reached, the test at c_2 is preempted and the rest of the test is scheduled to start at time 0 on wire w_2. Therefore the test at c_2 is partitioned into two parts. In execution of the test at c_2, the test starts at wire w_2 at time point zero. At time point 2, the test is preempted and resumed at time point 4. The test ends at time point 7. At the preemption of a test, another wire is assigned to the core and a multiplexer is added for wire selection. For the test of c_2, a multiplexer is added to select between w_1 and w_2.

In general preemptive scheduling, extra time is introduced at each preemption point due to the need to set up a job and also to save its state. In this case, the machines are the wires and no extra time is needed to set up and save the state. Also, in testing no other tasks are performed at the cores but testing, *i.e.* the core's state can be left as it is until the testing continues. The advantage is that the state of the core is already set and testing of it can start at once.

Assume that a core has a wrapper-chain of length l (l cycles are needed to perform a shift-in of a new vector and a shift-out of the previous test response). If the test is preempted when x% of the l cycles are shifted in it means that when the test restarts x% of the new test vector is already loaded and x% less cycles are needed in the first shift process, *i.e.* there is no time overhead due to setting up and saving the state of a core; all tests can be stopped at LB.

Finally, in some cases, such as for some types of memories such as DRAMs, the testing cannot be preempted. For instance, assume that test t_2 cannot be preempted as in Figure 131. In such a case, when LB is met, the scheduling algorithm restarts at LB (and not at time 0) and moves towards zero. The resulting schedule is in Figure 133. Note that, test t_2 now makes use of one wire during time point 4 to 5 and two wires during time 5 to 7, which is possible using the reconfigurable wrapper (more details can be found in Chapter 10, A Reconfigurable Power-Conscious Core Wrapper and its Application to System-on-Chip Test Scheduling, page 163).

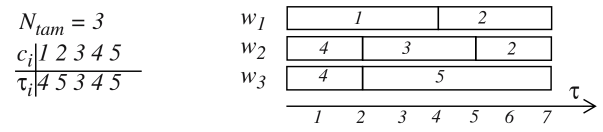


*Figure 133.*Optimal TAM assignment and preemptive scheduling assuming that test t_2 cannot be interrupted (preempted).

3.3.1 Test Power

It is becoming important to consider test power consumption during testing (see Chapter 7, Test Power Dissipation, page 89). Huang *et al.* [100] included test power by extending the two-dimensional model (TAM width versus test time) into a three-dimensional problem. The power model is based on the model proposed by Chou *et al.* [40, 41] which means that each test is assigned a fixed power value. Pouget *et al.* [218, 219, 220] have also proposed heuristics for the wrapper design problem under power constraint. Pouget *et al.* tries a pseudo-exhaustive approach, and the experiments shows that it can only be used at limited sized designs.

Hsu *et al.* [96] proposed scheduling technique for single-tier and multi-tier TACO architectures. For single-tier, three techniques are proposed; based on limited-polymorphic, similarity, and grouping. In order to minimize the test time further the multi-tier architecture can be used. However, Hsu *et al.* [96] reports that the multi-tier architecture only results in a slight test time reduction at the expense of a 583% higher hardware over-head compared to single-tier architecture. It should be noted that the comparison is between the additional hardware and not of the relative additional hardware. Flottes *et al.* [61, 62] demonstrated that the additional hardware overhead in relative terms is modest.

Zhao and Upadhyaya [291] proposes a technique where test conflicts and test power consumption are taken care of. The modeling of test conflicts and test power consumption is similar to the idea by Chou *et al.* [40, 41]. The difference is that in the approach by Zhao and Upadhyaya [291] the test times are not fixed for each testable unit and tests are not scheduled according to the partitioned testing scheme. Not fixed test times per testable unit means that special care has to be taken when analyzing the conflict graph since the testing times can be modified.

An approach that also tackles the test scheduling problem under power constraints and test conflicts is proposed by Su and Wu [258]. Similar to the approach by Zhao and Upadhyaya [291], Su and Wu [258] have to take special care when scheduling tests since the testing times for each test can be modified. For the optimization, Su and Wu [258] make use of a Tabu-search technique (see Section 5.1.6). Cota *et al.* [44, 45, 46] have proposed scheduling techniques for network-on-chip designs.

Xia *et al.* [279] make use of an Evolutionary Algorithm (EA) to solve the test scheduling problem. The power model is assumed to be non-fixed per test. A non-fixed power model is a result of clock-gating (Saxena *et al.* [241]) and has been used by for instance Larsson and Peng [158, 161].

Larsson and Peng [171] extended the work by Larsson and Fujiwara [167, 172] to also include test power consumption and clock-gating as proposed by

Saxena *et al*. [241]. The approaches by Larsson and Fujiwara [167, 172] and Larsson and Peng [171] are in more detail described in Chapter 10, A Reconfigurable Power-Conscious Core Wrapper and its Application to System-on-Chip Test Scheduling, page 163.

3.3.2 Multiple Test Sets

Huang *et al.* have proposed an extension allowing multiple tests per testable unit [101]. Iyengar *et al.* [118] proposed techniques to handle hierarchy (cores embedded in cores). Larsson and Fujiwara [167, 172] showed that conflicts between core tests and interconnection tests can be solved in an optimal way.

3.4 Other Test Scheduling Techniques

There are a number of additional problems that must be taken into account when designing the test solution.

3.4.1 Problem: Control lines and Layout.

The cost of adding additional pins to the SOC or pins to a component used in a PCB is high. It is therefore important to not only discuss the pins used for test data transportation but also the pins required for the control of the test data transportation. In the work by Aerts and Marinissen [5] the control pins required for scan-chain control was taken into account. Goel and Marinissen [77] as well as Wayers [273, 274] have proposed techniques where the additional control lines are taken into account. However, the cost of additional input and outputs at a core embedded in a SOC is by far lower than the cost of additional pins.

Goel and Marinissen [76] proposed a scheduling scheme were the floorplan of the system is taken into account when designing the test architecture. Larsson and Peng [171] showed that their approach minimizes the additional TAM routing.

3.4.2 Problem: Power Modeling

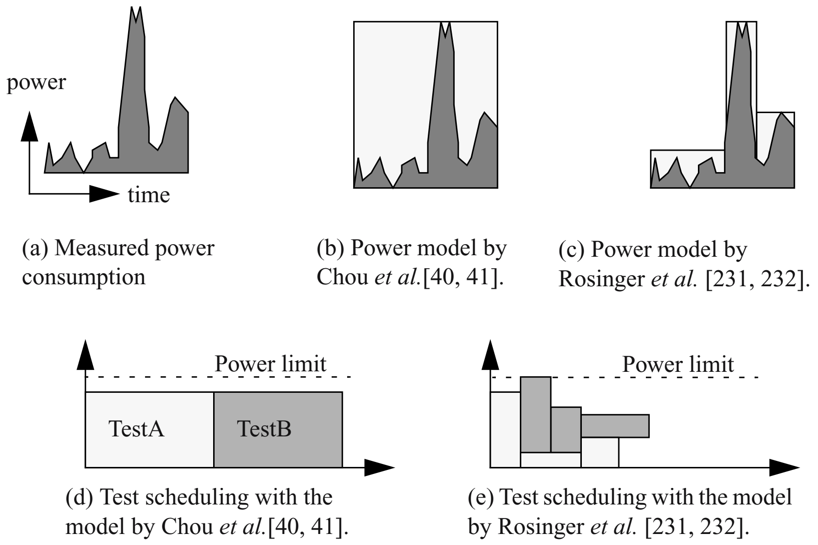
The power consumed during testing can be high due to the fact that it is desirable to activate as many fault locations as possible in order to reduce the testing times (see Chapter 7, Test Power Dissipation, page 89). Rosinger *et al.* [231, 232] propose a scheme where better power profiles for each test are used. The idea is that if a better model, not only a fixed power value per test, it is possible to schedule the tests in a way that minimizes the test time. An example is in Figure 134. The measured power over time for a test is in Figure 134(a) and the power model proposed by Chou *et al.*[40, 41] is in

Figure 134(b) where one value is given to each test. The model proposed by Rosinger *et al.* [231, 232] allows more the one value per test. The advantage is that a more accurate model of the real power consumption is given. The advantage during test scheduling is illustrated in Figure 134(d) and Figure 134(e). One can see that by having multiple power values per test (several boxes) it is possible to get a lower testing time for the system.

The model proposed by Chou *et al.*[40, 41] (one fixed test power value per test) is supported by the assumption that there is only a small difference between the peak test power and the average test power. If the difference is small between peak power and average power, the impact of the more accurate model as proposed by Rosinger *et al.* [231, 232] becomes less significant. It should also be noted that it is complicated to measure test power. The power depends highly on the activity in the rest of the system. A test that consumes a certain power when activated alone can use a different power when activated concurrent with other tests. One could also put a question mark on: why one measures the number of clock cycles at an extreme granularity while test power is given as a single value which is not very precise.

3.4.3 Problem: Fixed Test Resources

Multiple clock domains are common in systems. Therefore Xu and Nicoloci [282] propose a core wrapper that can handle related problems. Sehgal and Chakrabarty [243] has proposed optimization techniques where the ATE can deliver test data a dual speed; the ATE channels are partitioned into



(a) Measured power consumption

(b) Power model by Chou *et al.*[40, 41].

(c) Power model by Rosinger *et al.* [231, 232].

(d) Test scheduling with the model by Chou *et al.*[40, 41].

(e) Test scheduling with the model by Rosinger *et al.* [231, 232].

*Figure 134.*Power profiling and its impact on test scheduling.

two partitions each with its clock speed. To minimize the test time, as many cores as possible should be assigned to the high-speed channels, however, test power consumption puts a limitation.

3.4.4 Problem: Multiple Clock Domains

Multiple clock domains are common in SOC designs. It is common that cores operate at different clock frequencies, and furthermore, cores may operate internally at multiple frequencies. For stuck-at testing, lock-up latches could be inserted at the boundary between clock domains however, for at speed testing, clock skew during at-speed capture might occur and corrupt the test response [282]. Xu and Nicolici [282] propose a technique (wrapper and algorithm) for at-speed testing using a low speed ATE. It means that the system is running at a higher frequency compared to the ATE. Xu and Nicolici [282] propose a core wrapper architecture where the low speed ATE is synchronized with the higher clock frequency of the SOC. In order to apply at-speed testing, the last launch and the capture must be performed at-speed [92].

3.4.5 Problem: Delay Fault Testing

Testing for timing faults is becoming important due to deep submicron process variations. Xu and Nicolici [280] proposed a technique for broadside delay fault testing (see Broadside Test, see page 37) for SOCs with P1500 wrappers. The objective is to detect delay faults, which are becoming increasingly important to detect due to higher clock frequencies. The main problem in delay fault testing is that two test patterns must be applied in consecutive clock cycles.

Chakrabarty *et al.* [29] proposed a DFT technique where multiplexers are inserted in order to bypass a core. In such a way, two patterns can be applied in consecutive clock cycles. Yoneda and Fujiwara [284, 285, 286] proposed a technique where instead of adding multiplexers, test vectors are transported through the cores (instead of around the cors as in the approach by Chakrabarty *et al.* [29].

3.4.6 Defect-Oriented Scheduling

The aim of testing is to ensure that the circuit under test is fault-free in respect to the applied test set. That means to ensure that no mistakes have been inserted during production. If a fault exists in the circuit, the system is faulty and should not be sent for use. It means that as soon as a fault appears in the system, the testing can be aborted. If one fault appears, the system is faulty. In high-volume production (a high number of chips are tested), it is desirable to schedule the tests in such a way that the cores with a high proba-

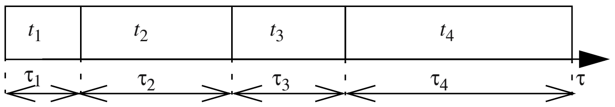


Figure 135. Sequential schedule of the example (Table 10).

bility to fail, are scheduled as early as possible. In such a way, the expected test time can be minimized. Larsson *et al.* [168, 169, 174] have proposed formulas to compute the expected test time for a system where each core is given a probability of having a defect.

For illustration of the computation of the expected test time, we use an example with four tests (Table 10). The tests are scheduled in a sequence as in Figure 135. For test t_1, the expected test time is given by the test time τ_1 and the probability of success p_1, $\tau_1 \times p_1 = 2 \times 0.7 = 1.4$. Note if there is only one test in the system, our above formula will give the expected test time to be 2 since we assume that every test set has to be fully executed before we can determine if the test is a successful test or not.

Core i	Test t_i	Test time τ_i	Probability to pass, p_i	Cost ($\tau_i \times p_i$)
1	t_1	2	0.7	1.4
2	t_2	4	0.6	2.4
3	t_3	3	0.9	2.7
4	t_4	6	0.8	4.8

Table 10. Example data.

The expected test time for the completion of the complete test schedule in Figure 135 is:

```
τ1×(1-p1) +                    // test t1 fails
(τ1+τ4)×p1×(1-p4) +            // test t1 passes, test t4 fails
(τ1+τ4+τ3)×p1×p4×(1-p3) +      // test t1, t4 pass, test t3 fails
(τ1+τ4+τ3+τ2)×p1×p4×p3×(1-p2) + //test t1, t4, t3 pass, test t2 
fails
(τ1+τ4+τ3+τ2)×p1×p4×p3×p2 =    // all tests run until completion,
                               // i.e. correct system.
2×(1-0.7) +
(2+6)×0.7×(1-0.6) +
(2+6+3)×0.7×0.6×(1-0.9) +
(2+6+3+2)×0.7×0.6×0.9×(1-0.8) +
(2+6+3+2)×0.7×0.6×0.9×0.8 = 8.2
```

As a comparison, for the worst case schedule where the test with highest passing probability is scheduled first, the order will be t_3, t_4, t_2, t_1, and the expected test time is 12.1. In the case of executing all tests until completion, the total test time does not depend on the order, and is $\tau_1+\tau_2+\tau_3+\tau_4=15$ (more details can be found in Chapter 14, Defect-Aware Test Scheduling, page 277).

An alternative approach to defect-oriented scheduling is taken by Edbom and Larsson [57] where a time constraint (given by the ATE memory depth) is given and the aim is to select test vectors and schedule them in such a way that the quality of the testing is maximized. The assumption in the approach by Edbom and Larsson is that the amount of test data increases and in cases where all needed test vectors cannot be applied, it is important to select the test vectors that contribute the most in increasing the systems quality. Each core is as in the approach by Larsson *et al.* [168, 169, 174] attached with a defect probability. Different from Larsson *et al.* [168, 169, 174], Edbom and Larsson [57] assume that the first test vectors detects a higher number of faults compared to test vectors applied later. Edbom and Larsson [57] approximate the fault detection to an exponential curve (Figure 136).

For each core i the CTQ_i (core test quality) given as:

$$\mathrm{CTQ_i} = \mathrm{dp_i} \times \mathrm{fc_i}(\mathrm{stv_i}) \tag{9.8}$$

and for the system with n cores, the STQ (system test quality) metric given as:

$$\mathrm{STQ} = \sum_{i=1}^{n} \mathrm{CTQ_i} / \sum_{i=1}^{n} \mathrm{dp_i} \tag{9.9}$$

The CTQ_i value depends on the defect probability (dp_i) and increases according to the exponential function fc_i with the number of applied test vectors (stv_i).

To illustrate the approach, the d695 (a design from the ITC’02 benchmark set [189, 190]) is used, see Figure 137. The time constraint is set to 5% of the total time if all test vectors would have been applied. In Figure 137 (a) only test scheduling is used, defect probability, fault coverage, and test set selec-

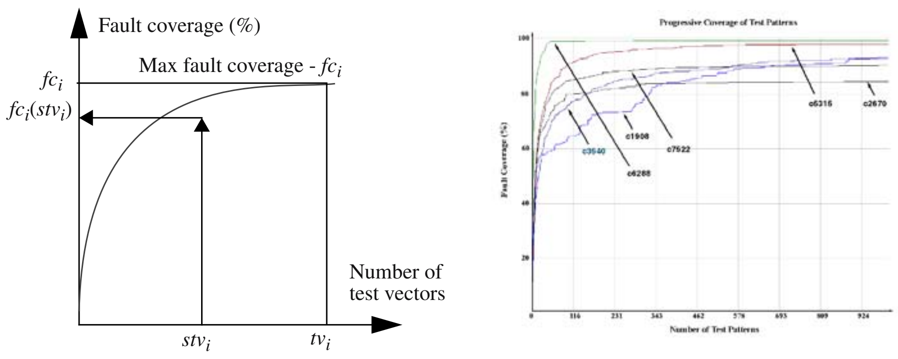


Figure 136. (a) Exponential estimation of the fault coverage and (b) the actual fault coverage for some ISCAS designs [57]

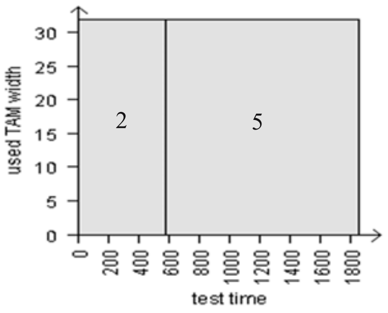
tion are not used. It means that core 2 and core 5 are tested. All vectors from core 2 are applied and 20% of the vectors from core 5 are applied. At the point when 20% vectors are applied from core 5 the time constraint limits further testing; hence the testing is terminated. The STQ value for such an approach on this design is 0.0322. In Figure 137 (b) test scheduling and defect probability are considered but not fault coverage and test set selection. Only one core is selected to be tested (core 7) and 54% of the vectors are applied. The STQ value in this approach is improved to 0.167. Figure 137 (c) where test scheduling, defect probability and fault coverage are considered, the STQ is improved to 0.203. If test set selection (selection of number of test vectors per core), defect probability, and fault coverage all are taken into account, the STQ value increases to 0.440. If a higher number of TAMs are allowed (Figure 137(e), (f)), the STQ value can be even increased.

The STQ value versus the test time is plotted in Figure 138. As the test application time increase (the time constraint becomes closer to the test application time when all test vectors are applied) the STQ value increases. Obviously, the STQ value is highest when all test vectors are applied. However, it is interesting to note that the same STQ value can be reached at a much lower test time when using test vector selection and test scheduling when considering defect probability and fault coverage compared to when it is not used. For instance, the STQ value at 20% of the test time for a technique where defect probability, fault coverage and test vector selection are taken into account is as high as the STQ value for a technique at 50% of the testing time when only considering defect probability and fault coverage. It means that a significant amount of test time can be gained by an effective selection of test vectors. The approach by Edbom and Larsson is in more detail described in Chapter 15, An Integrated Technique for Test Vector Selection and Test Scheduling under ATE Memory Depth Constraint, page 291.

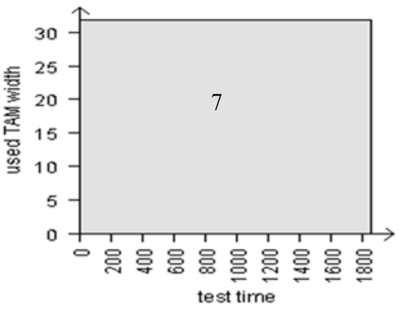
4 OPTIMAL TEST TIME?

The main objective with test scheduling is to organize the test in such a way that the test application time is minimized. The optimal goal is to find a test schedule with the lowest possible test application time; the optimal test time. In this section discuss possibilities and limitations for achieving optimal test application time.

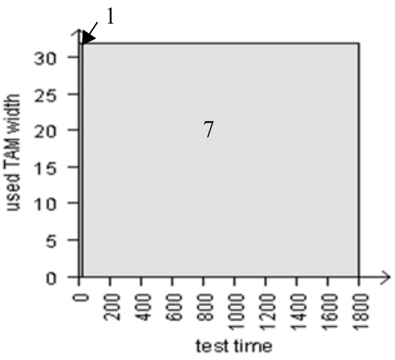
In the case when each test is given a fixed test time and a fixed resource usage, and the scheduling (packing) problem is to minimize the test time while not violating any given constraint the problem is NP-complete (dis-



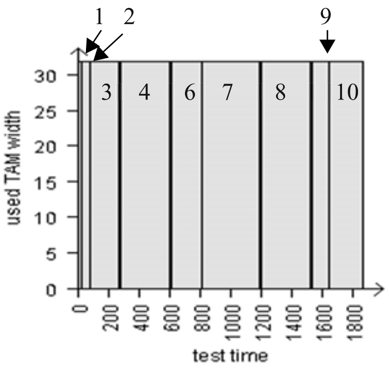
(a) Not considering test vector selection, defect probability, and fault coverage.



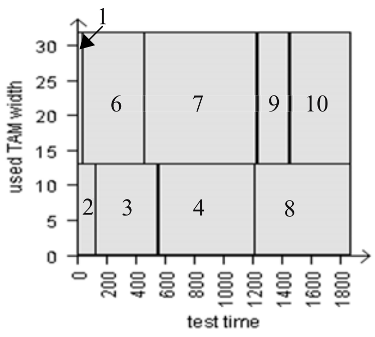
(b) Not considering test vector selection and fault coverage, but defect probability.



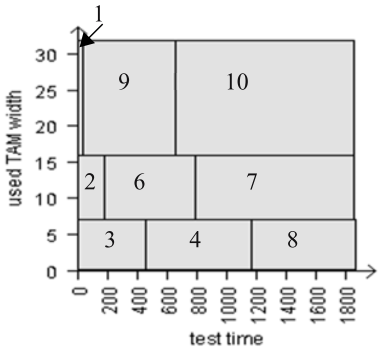
(c) Not considering test vector selection but defect probability and fault coverage.



(d) Considering test vector selection, defect probability and fault coverage on single TAM.



(e) Considering test vector selection, defect probability and fault coverage on two TAMs.



(e) Considering test vector selection, defect probability and fault coverage on three TAMs.

*Figure 137.*Illustration of the approach by Edbom and Larsson [57].

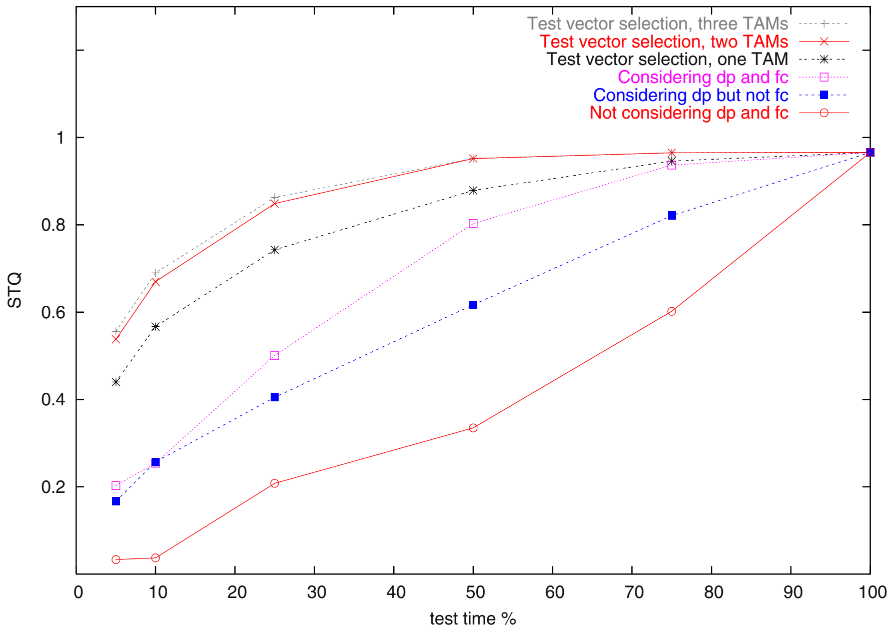


*Figure 138.*The STQ (system test quality) versus the test time [57].

cussed in Section 2, page 119). Note, that sequential scheduling is not NP-complete and concurrent test scheduling is NP-complete only when a resource constraint is to be met (see page 118). Figure 139 shows an example where a test named testA has its specified fixed test time and its fixed resource usage (the fixed resource usage can for instance be test power or TAM wire usage). Figure 139 also shows a schedule where the tests are assigned a start time.

Optimal test schedule can be defined as a test schedule having minimal idle bits (see Section 3.1 for discussion on idle bits). However, a test schedule where idle bits exist can be optimal in test time, *i.e.* it is the best possible test schedule from test time perspective. Figure 140 shows an example with two

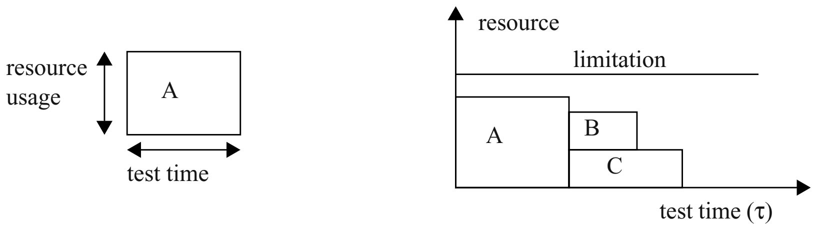


*Figure 139.*Test scheduling with tests having fixed test time and fixed resource usage.

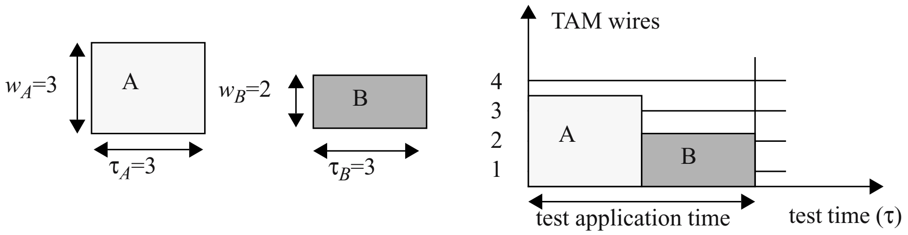


*Figure 140.*Test scheduling with tests having fixed test time and fixed resource usage.

tests (testA and testB). Each of the test has a fixed test time and a fixed TAM wire usage. The two tests are scheduled on 4 TAM wires (Figure 140). The test schedule results in a test application time of 6 time units (τ_A+τ_B=3+3=6). The test schedule in Figure 140 is optimal even if it contain idle bits. The solution is optimal from test application time perspective but obviously not from a TAM wire perspective (three TAM wires would result in the same solution).

A way to compute the lower bound, assuming no idle bits at all, would be:

$$\tau_{opt} = \frac{\sum_{\forall i} \tau_i \times r_i}{R_{max}} \tag{9.10}$$

For the example in Figure 140, $\tau_{opt}=(\tau_A \times w_A + \tau_B \times w_B)/N_{tam}=(3\times3+3\times2)/4=3.75$. Larsson and Peng [159, 161] defined, similar to Equation (9.10), to compute the lower bound when the resource is test power:

$$\tau_{opt} = \frac{\sum_{\forall i} \tau_i \times p_i}{P_{max}} \tag{9.11}$$

The Equations (9.10) and (9.11) gives a feeling for the lower bound. In order to achieve a solution near to the lower bound, the testing times must, if possible, be modified.

4.1 Soft Cores - No Fixed Scan-chains

For a system with soft cores where for each core *i* a number of scanned elements (flip-flops) ff_i, a number of test vectors tv_i, and the width of the TAM N_{tam} are given. The problem is to configure the flip-flops at each core to scan-chains, and determine the organization of the testing in such a way that the test time is minimized. We assume that Multiplexing Architecture (Chapter 8, Section 2.1, page 108) is used, which means that the cores are tested one at a time and the full TAM width is given to each core when tested.

The test time τ_i for a scan test at core c_i where the scanned elements (flip-flops) are not fixed into scan-chains is given by Aerts and Marinissen [5]:

$$\tau_i = (tv_i + 1) \times \left\lceil \frac{ff_i}{n_i} \right\rceil + tv_i \qquad (9.12)$$

at a core with ff_i scanned flip-flops partitioned into n_i scan chains and tv_i test vectors. Assuming Multiplexing Architecture means that n_i is equal to N_{tam}(number of TAM wires).

Figure 141 (a) shows an example with a core having 16 scanned flip-flops that are partitioned into 3 scan-chains (Figure 141 (b)). The shift-in time is then 6 (=$\lceil 16/3 \rceil$) cycles. At the same time as the test response from the previous test vector is shifted out, a new test vector is shifted in. Such pipelining can be applied for all test vectors but the last test response Figure 141 (c).

For each test vector, idle bits are introduced due to the imbalanced scan-chains ($\lceil ff_i/N_{tam} \rceil$) (see Section 3.1). The worst case is when there is N_{tam}-1 idle bits per test vector for each core. It means that the number of idle bits at a core i tested with tv_i test vectors is:

$$tv_i \times (N_{tam} - 1) \qquad (9.13)$$

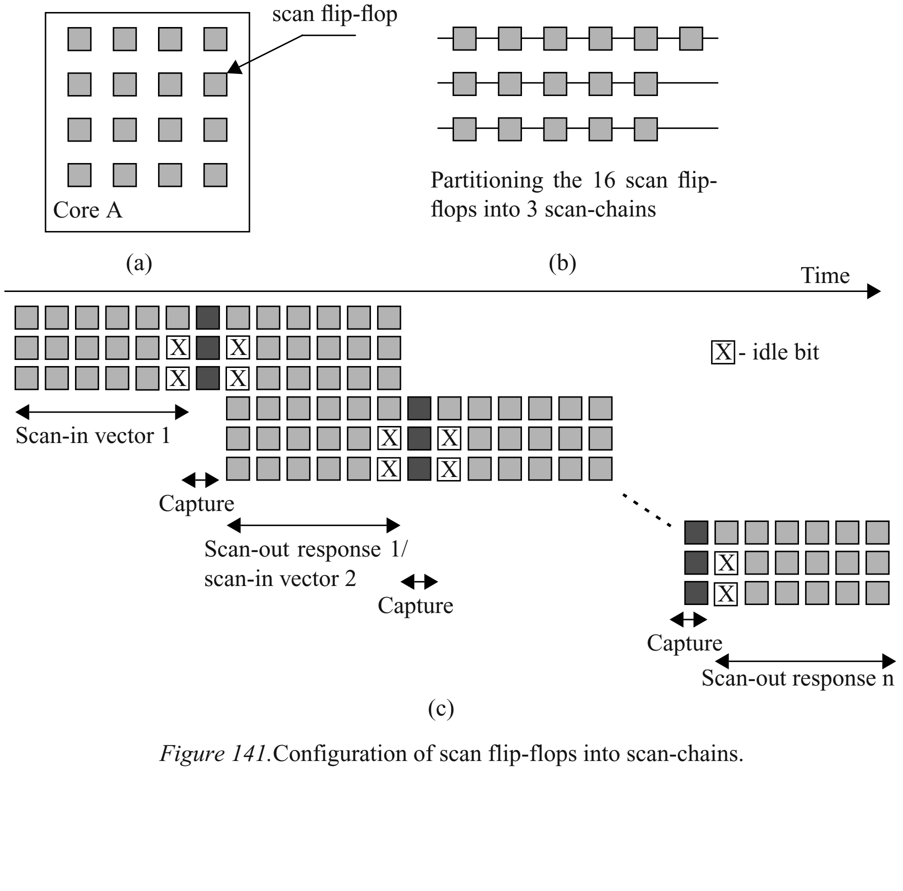


*Figure 141.*Configuration of scan flip-flops into scan-chains.

For a system with |C| cores, the number of idle bits is then given by:

$$\sum_{i=1}^{|C|} tv_i \times (N_{tam} - 1) \qquad (9.14)$$

Equation (9.14) can if N_{tam}-1 almost equal to N_{tam} be approximated to:

$$\sum_{i=1}^{|C|} tv_i \times (N_{tam} - 1) \approx \sum_{i=1}^{|C|} tv_i \times N_{tam} \qquad (9.15)$$

In order to minimize the idle bits, Equation (9.14) actually suggests that the TAM width should be minimized.

It is of interest to explore the number of idle bits in relation to the number of *non* idle bits. Taking Equation (9.12) and by assuming that the last test vector at each core has a minor impact, the following simplification $((tv_i+1)\times ff_i/N_{tam} >> tv_i)$ can be made for each core:

$$(tv_i + 1) \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil + tv_i \approx (tv_i + 1) \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil \qquad (9.16)$$

Assuming that the number of test vectors is high, Equation (9.16) can by assuming that tv_i+1=tv_i be further simplified to:

$$(tv_i + 1) \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil + tv_i \approx (tv_i + 1) \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil \approx tv_i \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil \qquad (9.17)$$

If the number of flip-flops is high compared to the TAM width, Equation (9.17) can be approximated to:

$$(tv_i + 1) \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil + tv_i \approx (tv_i + 1) \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil \approx tv_i \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil \approx tv_i \times \frac{ff_i}{N_{tam}} \qquad (9.18)$$

If the simplifications above are acceptable, Equation (9.18) demonstrates that there is a linear dependency between the test time and the number of TAM wires; hence the scheduling problem assuming soft cores is trivial. Assign the full TAM bandwidth to each core in a sequence (Multiplexing Architecture).

If the approximations leading to Equation (9.18) are not acceptable, starting with Equation (9.12),

$$\tau_i = (tv_i + 1) \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil + tv_i \qquad (9.19)$$

the difference (Δ) between considering idle bits and not considering idle bits is given by:

$$\Delta_i = \left((tv_i + 1) \times \left\lceil \frac{ff_i}{N_{tam}} \right\rceil + tv_i\right) - \left((tv_i + 1) \times \frac{ff_i}{N_{tam}} + tv_i\right) \qquad (9.20)$$

Equation (9.20) can be reduced to:

$$\Delta_i = \left\lceil \frac{ff_i}{N_{tam}} \right\rceil - \frac{ff_i}{N_{tam}} \qquad (9.21)$$

Equation Δ_i is obviously minimal as ff_i can be evenly divided by N_{tam}. Δ_i is minimized when the number of scan flip-flops is much larger than the TAM width, $ff_i >> N_{tam}$.

In the study made by Larsson and Fujiwara [162, 163, 173] it is assumed that tests can be partitioned into smaller pieces. It means that a set of TV test vectors for the test of a core can be partitioned into $\{tv_1, tv_2, ..., tv_m\}$ where $TV = \Sigma\, tv_i$; all test vectors are applied. Larsson and Fujiwara make use of preemption and reconfigurable wrappers to allow the most suitable wrapper configuration at a core. Figure 142 shows an example of a produced test schedule where core 3 is tested in two sessions. Note, that the TAM width at core 3 is different in the two sessions.

For the experiments, Larsson and Fujiwara make use of the ITC'02 benchmarks [189, 190] where it is assumed that all cores are soft cores. As a reference Larsson and Fujiwara make use of the lower bound of the test application time, excluding the capture cycles and the shift out of the last response, defined by Aerts and Marinissen [5] as:

$$\sum_{i=1}^{|C|} \frac{ff_i \times tv_i}{N_{tam}} \qquad (9.22)$$

Larsson and Fujiwara compare the three techniques; Multiplexing Architecture [5], Distribution Architecture [5], and preemptive scheduling. The result from the experiments on design p93791 is in Figure 143. The reason why Distribution Architect does not perform well at low TAM widths is that the design contains some small cores with low testing time. It means that there are cores assigned to dedicated TAM wires but these cores only make use of the wires for a short period of time. The reason why Multiplexing Architecture does not perform well at high TAM width is that Aerts and Mari-

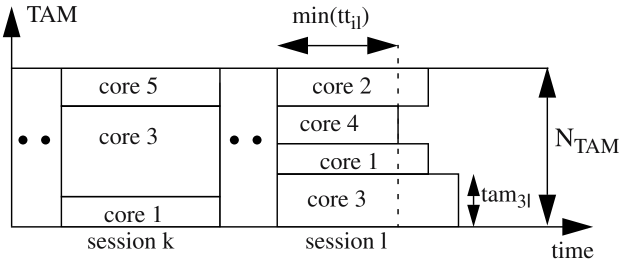


Figure 142. Session length based on preemption.

nissen [5] impose the constraint the a scan-chain must at least contain 20 flip-flops.

The results for the designs ic, p22810, p34392, and p93791 is in Figure 144 where for the ic design each scan-chain must contain at least 20 flip-flops, p22810 at least 5 flip-flops, p34392 at least 60, and p93791 at least 30. Figure 144 shows that the preemption-based technique proposed by Larsson and Fujiwara [162, 163, 173] produces results close to the lower bound. It also supports the statement in Equation (9.21) that the number of idle bits are reduced when $N_{tam} << ff_i$.

4.2 Hard Cores - Fixed Number of Scan-chains

Hard cores are cores where the number and the length of the scan-chains are fixed prior to the design of the test solution for the system. It means that a fixed number of scan-chains are given. The test time versus the number of TAM wires (wrapper chains) at a core is illustrated in Figure 127. From Figure 127 it is clear that at some TAM widths, the testing time is actually not decreased compared to a lower TAM width.

Larsson [177] made an analysis of the test time, the TAM width and the number of scan-chains at core 11 in the ITC'02 benchmark p93791

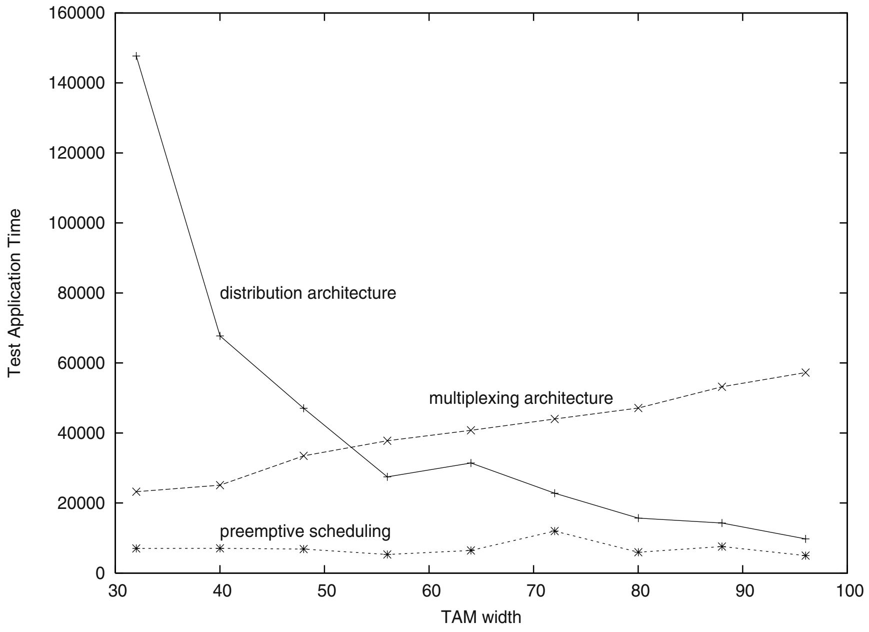


*Figure 143.*Comparing the test application time for multiplexing architecture, distribution architecture and preemptive scheduling.

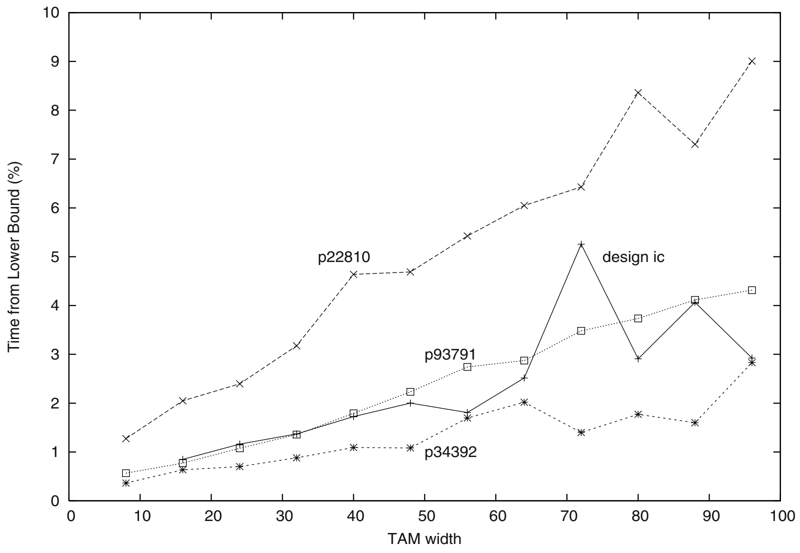


*Figure 144.*Test time analysis by Larsson and Fujiwara [162, 163, 173].

(Figure 145). Figure 145 plots the test time ($\tau(w)$ at w TAM wires) times the number of TAM wires (w), *i.e* ($\tau(w) \times w$). For Core 11 - *original* - the original design as specified in the p93791 design. However, the flip-flops are for design Core 11 - *balanced* - partitioned into 11 balanced scan-chains. The design Core X - *balanced* - means that the scanned elements (flip-flops) are partitioned into X balanced chains (X=11, 22, 44, and 88). As the number of scan-chains increases, the more linear the term $\tau(w) \times w$ becomes. It means that by carefully designing the scan-chains at a module, a near linear behavior is achieved. And hence, the scheduling problem becomes trivial (if no other constraints but test time and TAM wires are considered).

5 INTEGRATED TEST SCHEDULING AND TAM DESIGN

The possibility to acheive a low test application time for a system depends highly on the test schedule and the TAM design. Narrow TAMs reduces the routing cost at the expense of the testing time. Larsson *et al.* [166, 176] proposed therefore an integrated technique where the test schedule and the TAM are designed. In the approach several constraints are taken into account.

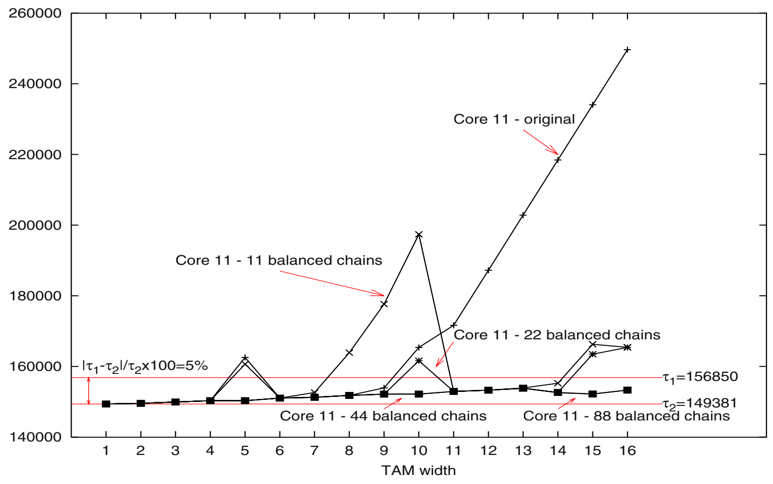


Figure 145. The test time versus the number of TAM wires for different scan design versions of Core 11 in P93791 made by Larsson [177].

5.1 Test Time and Test Power Consumption

An important constraint to consider is test power. The test power problem and its modeling have been discussed in Chapter 7, “Test Power Dissipation”, on page 89. The power consumption can be assumed to be fixed for each testable unit or it can be assumed to be non-fixed. A non-fixed test power model based on the assumption by Saxena *et al.* [241] has been used by Larsson *et al.* [158, 159, 161, 164, 166, 176] (Eq. 7.1, page 90).

5.2 Bandwidth Assignment

Test parallelization allows a flexible bandwidth assignment for each test depending on the bandwidth limitations at the block under test and the bandwidth limitations at the test resources. The test time (see Section 5.1) for a test t_{ijk} at block b_{ij} at core c_i is given by:

$$\tau'_{ijk} = \left\lceil \frac{\tau_{ijk}}{bw_{ij}} \right\rceil \quad (9.23)$$

and the test power (see Section 5.1):

$$p'_{ijk} = p_{ijk} \times bw_{ij} \quad (9.24)$$

where bw_{ij} is the bandwidth at block b_{ij} at core c_i [161].

Combining the TAM cost and the test time (Equation 9.23), the following is given for each block b_{ij} and its tests t_{ijk}:

$$cost(b_{ij}) = \sum_{\forall k} l_l \times bw_{ij} \times \beta + \tau_{ijk} / (bw_{ij}) \times \alpha \qquad (9.25)$$

where: $l_l = [source(t_{ijk}) \rightarrow c_i \rightarrow sink(t_{ijk})]$ and k is the index of all tests at the block. To find the minimum cost of Equation 9.25, the derivative in respect to bw gives the bandwidth bw_{ij} at a block b_{ij}:

$$\sqrt{\frac{\alpha}{\beta} \times \left(\sum_{\forall k} \tau_{ijk}\right) / \left(\sum l_l\right)} \qquad (9.26)$$

Naturally, when selecting bw_{ij}, the bandwidth limitations at each block are considered.

5.3 Test Scheduling

The test scheduling algorithm is outlined in Figure 146. First, the bandwidth is determined for all blocks (Section 5.2) and the tests are sorted based on a key (time, power or time×power). The outmost loop terminates when all tests are scheduled. In the inner loop, the first test is picked and after calling *create_tamplan* (Section 5.4), the required number of TAM wires are selected or designed for the test based on the cost function. If the TAM factor is important, a test can be *delayed* in order to use an existing TAM. This is determined by the cost function. If all constraints are fulfilled, the test is scheduled and the TAM assignment is performed using the technique in section 5.4. Finally, all TAMs are optimized according to the technique discussed in section 5.5.

```
for all blocks bandwidth = bandwidth(block)
sort the tests descending based on time, power
or time×power
τ=0
until all tests are scheduled begin
    until a test is scheduled begin
        tamplan = create_tamplan(τ,test) // see
Figure 147 //
        τ' = τ+delay(tamplan)
        if all constraints are fulfilled then
            schedule(τ')
            execute(tam plan) // see Figure
148 //
            remove test from list
        end if
    end until
    τ=first time the next test can be sched-
uled
end until
order (tam) // see Figure 149 //
```

*Figure 146.*Test scheduling algorithm.

5.4 TAM Planning

In the TAM planning phase, our algorithm:

- creates the TAMs,
- determines the bandwidth of each TAM,
- assigns tests to the TAMs, and
- determines the start time and the end time for each test.

The difference compared to the published approaches [164, 165] is that in the planning phase the existence of the TAMs is determine but not their routing.

For a selected test, the cost function is used to evaluate all options (*create_tamplan*(τ', *test*)) (Figure 147). The time (τ') when the test can be scheduled to start and its TAM is determined using the cost function and if all constraints are fulfilled, the TAM floor plan is determined (*execute* (*tam-plan*)) (Figure 148). To compute the cost of extending a TAM wire with a node, the length of the required additional wires is computed. Since the order of the cores on a TAM is not yet decided, an estimation technique for the wire length is needed. For most TAMs, the largest wiring contribution comes from connecting the nodes with the largest distance from each other. The rest of the nodes can be added on the TAM at a limited additional cost (extra routing). However, for TAMs with a high number of nodes, the number of nodes becomes important.

The estimation of the wire length considers both of these cases. We assume that the nodes (test sources, test sinks and the wrapped cores) in the system are evenly distributed over the area, *i.e.* A = width×height = $(N_x \times \Delta) \times (N_y \times \Delta) = N_x \times N_y \times \Delta^2$, where N_x and N_y are the number of cores on the x and y axis, respectively. Therefore Δ, the average distance between two nodes, is computed as:

$$\Delta = \sqrt{A/(N_x \times N_y)} \tag{9.27}$$

The estimated length, el_i, of a wire, w_i, with k nodes is:

$$el_i = \max_{1 \le j \le k}\{l(n_{source} \rightarrow n_j \rightarrow n_{sink}), \Delta \times (k+1)\} \tag{9.28}$$

It means that the maximum between the length of the longest created wire and the sum of the average distances for all needed arcs (wire parts) is computed. For example, let $n_{furthest}$ be the node creating the longest wire, and n_{new}

```
for all tams connecting the test source and test sink used
by the test, select the one with lowest total cost
  tam cost=0;
  demanded bandwidth=bandwidth(test)
  if bandwidth(test)>max bandwidth selected tam then
    demanded bandwidth=max bandwidth(tam)
    tam cost=tam cost+cost for increasing bandwith of tam;
  end if
  time=first free time(demanded bandwidth)
  sort tams ascending according to extension (τ, test)
  while more demanded bandwidth
    tam=next tam wire in this tam;
    tam cost=tam cost+cost(bus, demanded bandwidth)
    update demanded bandwidth accordingly;
  end while
  total cost=costfunction(tam cost, time, test);
```

Figure 147. TAM estimation, *i.e. create_tamplan(τ, test)*.

```
demanded bandwidth = bandwidth(test)
if bandwidth(test)>max bandwidth selected virtual tam then
  add a new tam with the exceeding bandwidth
  decrease demanded bandwidth accordingly
end if
time=first time the demanded bandwidth is free sufficient
long
sort tams in the tam ascending on extension (test)
while more demanded bandwidth
  tam=next tam in this tam;
  use the tam by adding node(test) to it, and mark it busy
  update demanded bandwidth accordingly;
end while
```

Figure 148. TAM modifications based on *create_tamplan* (Figure 147), i.e. *execute (tamplan)*.

the node to be added, the estimated wiring length after inserting n_{new} is given by (Eq. 9.27):

$$el'_i = \max\begin{Bmatrix} \min\begin{Bmatrix} l(n_{source} \rightarrow n_{new} \rightarrow n_{furthest} \rightarrow n_{sink}) \\ l(n_{source} \rightarrow n_{furthest} \rightarrow n_{new} \rightarrow n_{sink}) \end{Bmatrix} \\ \Delta \times (k+2) \end{Bmatrix} \tag{9.29}$$

For a TAM, the extension is given as the summation of all extensions of the wires included in the TAM that are needed in order to achieve the required bandwidth. The TAM selection for a test t_{ijk} is based on the TAM with the lowest cost according to:

$$(el'_l - el_l) \times \quad + \text{delay}(tam_l, t_{ijk}) \times \alpha. \tag{9.30}$$

Using this cost function, a trade-off between adding a new TAM and delaying a test on an existing TAM is given. For a newly created TAM, the delay for a test is 0 (since no other test is scheduled on the TAM and the test can start at time 0): $newcost(t_{ijk}) = l(source(t_j) \rightarrow c_i \rightarrow sink(t_j)) \times \beta$.

5.5 TAM Optimization

Above the TAMs for the system was created, each test was assigned to a TAM, the bandwidth of the TAMs was determined, and every test was given a start time and an end time in such a way that no conflicts and no limitations were violated. In this section, we discuss the routing of the TAMs, *order*(*tam*) in Figure 146. The approach is based on a simplification of an algorithm presented by Caseau and Laburthe [38]. The notation TG→[A,D]→SA was above used to indicate that core A and D were assigned to the same TAM, however, the order of [A,D] was not determined (Equation 9.25), which is the objective in this section. The following is used:

$$n_{source} \rightarrow n_1 \rightarrow n_2 \ldots \rightarrow n_n \rightarrow n_{sink} \qquad (9.31)$$

to denote that a TAM from n_{source} (the test source) to n_{sink} (the test sink) connects the cores in the order n_{source}, n_1, n_2,..., n_n, n_{sink}.

The TAM routing algorithm is outlined in Figure 149. The algorithm is applied for each of the TAMs and initially in each case the nodes (test sources, wrapped cores, and test sinks) of a TAM are sorted in descending order according to:

$$dist(n_{source}, n_i) + dist(n_i, n_{sink}) \qquad (9.32)$$

where the function *dist* gives the distance between two cores, or between a test source and a core, or between a core and a test sink, *i.e*:

$$dist(n_i, n_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (9.33)$$

First the test source and the test sink are connected (Figure 149). In the loop over the list of nodes to be connected, each node is removed and added to the final list in such a way that the wiring distance is minimized according to:

$$\min\{dist(n_i, n_{new}) + dist(n_{new}, n_{i+1}) - dist(n_i, n_{i+1})\} \qquad (9.34)$$

where $1 \leq i < n$ (all nodes on the TAM).

```
add test source and test sink to a final list
sort all cores descending according to Eq. 9.34
while cores left in the list
  remove first node from list and insert in the final list
  insert direct after the position where Eq. 9.32 is fulfilled
end while
```

Figure 149. Routing optimization of all TAMs.

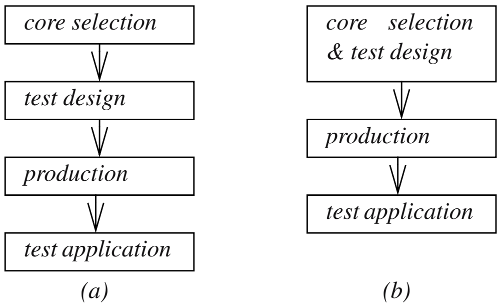


*Figure 150.*Design flow in a core-based design environment (a) traditional and (b) proposed.

Further details on the above approach is to be found in “An Integrated Framework for the Design and Optimization of SOC Test Solutions” (page 187) and “Efficient Test Solutions for Core-based Designs” (page 215).

6 INTEGRATING CORE SELECTION IN THE TEST DESIGN FLOW

A core-based design flow is typically a sequence that starts with core selection, followed by test solution design, and after production, the system is tested (Figure 150(a)). In the core selection stage, the core integrator selects appropriate cores to implement the intended functionality of the system. For each function there are a number of possible cores that can be selected, where each candidate core has its specification on, for instance, performance, power consumption, area, and test characteristics. The core integrator explores the design space (search and combines cores) in order to optimize the SOC. Once the system is fixed (cores are selected) the core test integrator designs the TAM and schedules the tests based on the test specification for each core. In such a design flow (illustrated in Figure 150(a)), the test solution design is a consecutive step to core selection. And, even if each core’s specification is highly optimized, when integrated as a system, the system’s global test solution is most likely not highly optimized.

The design flow in Figure 150(b), on the other hand, integrates the core selection step with the test solution design step, making it possible to consider the impact of core selection when designing the test solution. In such a design flow (Figure 1(b)), the global system impact on core selection is considered, and the advantage is that it is possible to develop a more optimized test solution.

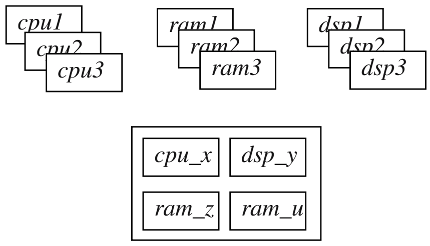


*Figure 151.*System design

The design flow in Figure 150(b) can be viewed as in Figure 151, where the core type is floor-planned in the system but there is not yet a design decision on which core to select. For each position, several cores are possible. For instance, for the cpu_x core there are in Figure 151 three alternative processor cores (cpu1, cpu2 and cpu3).

The modeling technique proposed by Larsson *et al.* [166, 176] allows a flexible specification of tests. Each testable unit can have multiple test sets, and each test set is modeled in a flexible way. For instance, a testable unit can be tested by three test sets. Each test set has its specification on test time, test power and test resources (test source and test sink). It is, for instance, possible to specify a test having an external test source and an internal test sink. However, each testable unit has its specified tests. Larsson [175, 177] proposes a technique where for each testable unit *a set of* test sets are specified for each testable unit. It means that for each testable unit, one such test should be selected. Figure 152 shows an example. The testable unit blockB1 can be tested by the two tests tB1.1 and tB1.2 or by tB1.3.

The approach by Larsson [175, 177] is outlined in Figure 153. The approach consists of:

- Creation of initial solution,
- Test scheduling and TAM design,
- Identification of limiting resources,
- Design modifications.

```
[Blocks]
  #name      idle_pwr   pwr_grid     test_sets {} {} ...{}
   blockA1 0    p_grd1 { tA1.1 }{ tA1.2, tA1.3}
   blockA2 0    p_grd1 { tA2.1 }{ tA2.2 }
   blockB1 5    p_grd1 { tB1.1 tB1.2 } { tB1.3 }
```

*Figure 152.*Extention to allow test set selection.

The initial solution is created by selecting the best test set for each testable unit by only considering the test set itself. It means that only a local view is taken.

After the creation of the initial solution, each testable unit has its tests and the tests and the TAM are to be designs. For this task the approach by Larsson *et al.* [166, 176] is used.

The bottlenecks or limiting resources are to be identified. A machine-oriented Gantt chart where the resources are the machines and the tests are the jobs can be used to show the allocation of jobs on machines [23]. A Gantt chart is in Figure 154 where for instance, test B2 needs TG: r2 and TRE: s2. An inspection of Figure 154 shows that TG:r2 and TRE:s2 are not limiting the solution. On the other hand, all resources that are used more than τ_{opt} are limiting the solution. The test source TG:r1 is the most critical one followed by the test sink TRE:s1. These two test resources are the obvious candidates for modification.

For tests that are using a limiting resource TAM size modification and test set modifications and the novel extensions are the bandwidth modifications. The search for an improved solution continues until no better solution can be found (Figure 153).

Further details on the above described approach can be found in "Core Selection in the SOC Test Design-Flow" (page 253).

```
Create initial solution
  Do {
    Create test schedule and TAM
    Find limiting resources
    Modify tests at limiting resources
} Until no improvement is found
Return best solution
```

*Figure 153.*Core selection integrated in test solution design.

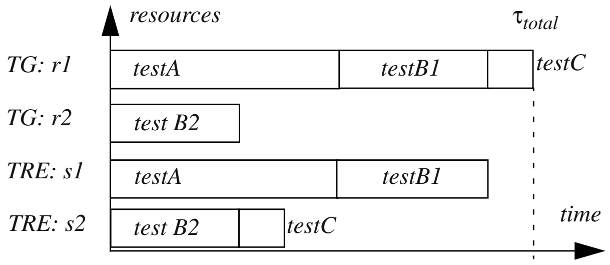


*Figure 154.*A machine-oriented Gantt chart [23].

7 FURTHER STUDIES

We have discussed test scheduling and the impact of the test architecture (TAM), test conflicts and test power consumption.

7.1 Combined Test Time and TAM Design Minimization

Below, in the chapter "An Integrated Framework for the Design and Optimization of SOC Test Solutions" (page 187) and "Efficient Test Solutions for Core-based Designs" (page 215) techniques where the cost of both test time and TAM routing cost are minimized.

7.2 Core Selection in the Test Design Flow

Larsson proposes in "Core Selection in the SOC Test Design-Flow" (page 253) a technique where the test sets are not yet fixed for the system. The idea is to include the test solution design already at the core integration phase. Each core has its specification on performance, power consumption, as well as its test specification. In order to design a test solution with a minimal cost (test time and TAM routing cost), the core selection, and test resource partitioning and placement should be taking into account with a system-level perspective. A system-level perspective is needed since only considering a solution at core-level will not show the impact for the whole system.

7.3 Defect-Oriented Test Scheduling

In a large volume production a high number of systems are to be tested. Some are faulty will some a non-faulty (correct) to be shipped. In the case when a system is faulty the testing is terminated not at the end when all tests are applied but as soon as a fault appears - abort-on-fail. In abort-on-fail the cores with a high probability to have a fault should be scheduled prior to cores with a lower defect probability. The idea is to minimize the expected test time.

In the chapter "Defect-Aware Test Scheduling (page 277) formulas to compute the expected test time for different test architectures are presented, as well as experiments to illustrate the importance of considering the defect probability per testable unit (core). In "An Integrated Technique for Test Vector Selection and Test Scheduling under ATE Memory Depth Constraint" (page 277) a technique is proposed to select test sets for a system to meet the time constraint while maximizing the test quality.

PART 3
SOC TEST APPLICATIONS

Part 3 is a collection of the following longer papers:

- The chapter **A Reconfigurable Power-Conscious Core Wrapper and its Application to System-on-Chip Test Scheduling** is based on the papers presented at International Test Conference (ITC'03) [171], Asian Test Symposium (ATS'03) [172], and Workshop on RTL and High-Level Testing WRTLT'02 [167].
- The chapter **An Integrated Framework for the Design and Optimization of SOC Test Solutions** is based on the papers presented at Design Automation and Test in Europe (DATE), 2001 [158], International Conference on Computer-Aided Design (ICCAD), 2001 [160], and Journal on Electronic Testing: Theory and Applicationd (JETTA) 2002 [164, 165].
- The chapter **Efficient Test Solutions for Core-Based Designs** is based on the following papers Asian Test Symposium (ATS), 2002 [166], and Transactions on Coputer-Aided Design for Integrated Circuits, 2004 [176].
- The chapter **Integrating Core Selection in the System-on-Chip Test Solution Design-Flow** is based on the papers presented at International Test Resource Workshop (TRP), 2004, [175], and International Test Conference (ITC), 2004 [177].
- The chapter **Defect-Aware Test Scheduling** is based on the papers presented at International Test Synthesis Workshop (ITSW'03) [168], 2003, Design and Diagnostics of Electronic Circuits & Systems (DDECS'03), 2003 [169], and VLSI Test Symposium (VTS'04), 2004 [174].
- The chapter **An Integrated Technique for Test Vector Selection and Test Scheduling under Test Time Constraint** is based on the work submitted to Asian Test Symposium (ATS), 2004, [57].

Chapter 10

A Reconfigurable Power-Conscious Core Wrapper and its Application to System-on-Chip Test Scheduling[1]

1 INTRODUCTION[2]

Test power consumption and test time minimization (test scheduling) are becoming major challenges when developing test solutions for core-based designs. Test power consumption is usually high during testing since it is desirable to activate as many fault locations as possible in a minimum of time, and if the tests are scheduled concurrently in order to reduce the testing times the test power increases. In this chapter, we demonstrate that the scheduling of tests on the test access mechanism (TAM) is equivalent to independent job scheduling on identical machines, and we make use of an existing preemptive scheduling algorithm and reconfigurable core wrappers to produce optimal solution in linear time. We propose a novel reconfigurable power-conscious core test wrapper and discusses its application to optimal power-constrained SOC (system-on-chip) test scheduling. The advantage of the proposed wrapper is that at each core it allows (1) a variable number of wrapper-chains, and (2) a possibility to select the appropriate test power consumption for each core. Our scheduling technique produces optimal solutions in respect to test time, and selects wrapper configurations in a systematic way that implicitly minimizes the TAM routing and the wrapper logic. We have implemented the technique and the experimental results show the efficiency of our approach. The results from our technique are less than 3% from the theoretical computed lower bound.

Long testing times are required due to the increasing amount of test data needed to test complex SOC (System-on-Chip). The testing times can be reduced by concurrent execution of tests, however, constraints and limitations must be considered. Concurrent testing leads to higher activity in the system

[1]. The chapter is based on the papers presented at International Test Conference (ITC'03) [171], Asian Test Symposium (ATS'03) [172], and Workshop on RTL and High-Level Testing WRTLT'02 [167].

[2]. The research is partially supported by the Swedish National Program STRINGENT.

and hence higher power consumption. The high test power consumption and the test time minimization problem can be tackled by:

- *design for low power testing* where the system is designed to minimize test power consumption, which allows consequently testing at a higher clock frequency [68, 207, 230, 241], and
- *test scheduling* where the tests are organized in such a way that the test time is minimized while considering test power limitations and test conflicts [5, 41, 75, 74, 73, 100, 115, 113, 112, 114, 140, 141, 164].

A core-based design is composed of pre-defined cores, UDL (user-defined logic) blocks and interconnections. From a testing perspective, all testable units, that is core logic, UDL blocks and interconnections, are defined as cores. The TAM (Test Access Mechanism), the set of wires connecting the ATE (Automatic Test Equipment) and the cores under test, is responsible for the test data transportation. A wrapper is the interface between the TAM and a core and it can be in one of the following modes at a time: *normal operation*, *internal test* or *external test*. A core can be either wrapped or unwrapped, which means that there are have two types of tests in a system: *wrapped core test* at *wrapped* cores and *cross-core test* (interconnection test) at *unwrapped* cores [191].

In this chapter, we propose a reconfigurable power-conscious (RPC) core test wrapper. The advantage is that it can be used to regulate the test power at core-level. We describe also the application of the RPC core wrapper in optimal test scheduling. The main contributions of the chapter are that we:

1. show that the problem of scheduling *wrapped core tests* on the TAM is equivalent to independent job scheduling on identical machines,
2. make use of a preemptive approach to produce a solution with optimal test time in linear time [23] using reconfigurable wrappers [140],
3. extend the scheduling algorithm to handle wrapped core tests and cross-core tests while preserving the production of an optimal solution in linear time.
4. develop RPC core test wrapper which combines the gated sub-chain scheme presented by Saxena *et al.* [241] and the reconfigurable core test wrapper introduced by Koranne [140],

5. integrate the RPC wrapper in the preemptive test scheduling approach, and
6. formulate a power condition that, if satisfied, guarantees that our preemptive test scheduling scheme produces optimal test application time for the system.

The main advantages with our approach are that:

1. optimal test time is achieved in linear time,
2. the TAM routing is minimized by assigning a minimum of TAM wires per core,
3. the reconfigurable wrappers are selected and inserted in a systematic manner to minimize the number of configurations, which minimizes the added logic,
4. it is possible to regulate the power consumption at each individual core, which allows the test clock speed to increase,
5. it is possible to regulate the test power consumption at system level, which should be kept within a given value in order to reduce the risk of over-heating which might damage the system under test,
6. it is possible select the best TAM size, and
7. minimization of the TAM routing and the overhead when using the RPC wrapper is performed.

The rest of the chapter is organized as follows. Related work is reviewed in Section 2 and our reconfigurable power-conscious test wrapper is introduced in Section 3. In Section 4 we present our optimal preemptive test scheduling technique, based on a known preemptive scheduling algorithm [23]. In the experiments we have made a comparison with previous approaches and we illustrate the advantages with our wrapper and its use in the proposed scheduling approach in Section 5. The chapter is concluded with conclusions in Section 6.

2 BACKGROUND AND RELATED WORK

Figure 155 shows an example of a core-based system consisting of five cores, a TAM and an ATE (serving as test source and test sink). The test source, where the test stimulus are stored, and the test sink, where the test responses are saved, are connected to the N_{tam} wire wide TAM. The test source feed test stimulus to the cores via the TAM and the produced test responses from the cores are also transported via the TAM to the test sink. The cores are equipped with a test method; core 1 is, for instance, scan tested. A wrapper is the interface between a core and the TAM. Its main task is to connect the scan chains and the wrapper cells at a core into a set of wrapper chains, which are connected to the TAM. Cores with a dedicated wrapper, such as core 1, 2, and 3, are wrapped while cores without a dedicated wrapper, such core 4 and 5, are unwrapped. A core test is a test at a wrapped core, and a cross-core test is a test at an unwrapped core. The execution of a core test and a cross-core test therefore differs from each other. A core test is performed by placing its wrapper cells in internal test mode, and the test stimulus are transported direct from the test source via the TAM to the core. The test responses are transported from the core via the TAM to the test sink. The testing of an unwrapped core (such as core 4 in Figure 155), cross-core test, requires that the wrapper at core 1 and the wrapper at core 2 are both placed in external test mode. Test stimulus are then transported from the test source on the TAM via the wrapper cells at core 1 to core 4. The test responses are transported from core 4 via the wrapper cells at core 2 and the TAM to the test sink.

In the particular example given in Figure 155, the testing of core 1 and core 4 cannot be performed concurrently due to that in both cases the wrapper at core 1 is needed, and a wrapper can only be in one mode at a time. In general, test conflicts such as this one must be considered in the scheduling process. There are mainly two types of test conflicts: (1) TAM wire conflicts and (2) core wrapper conflicts, respectively.

Several test scheduling techniques have been proposed [5, 41, 75, 74, 73, 100, 115, 113, 112, 114, 140, 141, 167, 164]. Chou *et al.* proposed a technique for general systems where each test has a fixed test time and a fixed power consumption value [41]. The objective is to organize the tests in such a way that the total test application time is minimized while test conflicts and test power limitation are taken in to account [41]. The testing time for a test

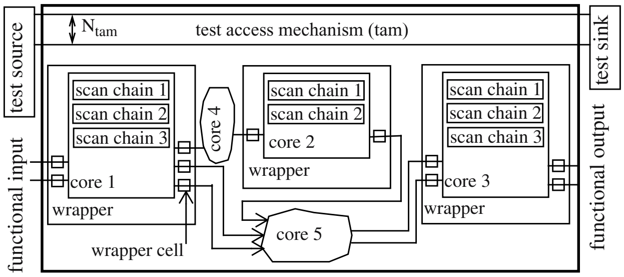


Figure 155. An example system with three wrapped cores (1,2,3) and two unwrapped cores (4,5).

can often be modified. In scan testing, for example, the test time at a core can be modified by adjusting the number of wrapper chains that the scanned elements are configured into. A low number of wrapper chains at a core reduces the number of occupied TAM wires at the expense of higher testing time. And vice versa. Several wrapper chain configuration and TAM wire assignment algorithms to minimize the test time for *core tests* at *wrapped cores* have been proposed [75, 74, 73, 100, 115, 113, 112, 114, 140, 141]. For instance, Iyengar *et al.* [113] used an ILP (Integer-Linear Programming) approach. Koranne [140] introduced a reconfigurable wrapper with the advantage of allowing N_{tam} wrapper chain configurations per wrapped core. In order to minimize the added overhead due to the reconfigurable wrapper, a limited number of cores are selected to have a reconfigurable wrapper prior to the scheduling. The fundamental reason that the SOC wrapper configuration problem is hard is that the test time is not linear with the number of wrapper-chains. Figure 156 shows the test time versus the number of wrapper chains and the test time decreases as the number of wrapper chains increases. However, it is not a strict linear function. And, it is a staircase function where certain points (number of wrapper chains) does not reduce the test time.

Iyengar *et al.* [115] and Huang *et al.* [100] proposed scheduling techniques for core tests under power constraints with fixed test power consumption for each test. Nicolai and Al-Hashimi [207] proposed a technique to minimize useless switches, and Gerstendörfer and Wunderlich [68] introduced a technique to disconnect the scan-chains from the combinational logic during the shift process, both leading to a lower power consumption during test, which reduces test time by allowing clocking at a higher frequency. For the same purpose, Saxena *et al.* [241] proposed an approach to gate sub-chains.

3 A RECONFIGURABLE POWER-CONSCIOUS CORE WRAPPER

The RPC (reconfigurable power-conscious) test wrapper we propose combines the gated sub-chain approach proposed by Saxena *et al.* [241] and the reconfigurable wrapper introduced by Koranne [140]. The basic idea in the approach proposed by Saxena *et al.* [241] is to use a gating scheme to lower the test power dissipation during the shift process. Given a set of scan-chains as in Figure 157 where the three scan-chains are connected into a single chain. During the shift process, all scan flip-flops are active, leading to high switch activity in the system and therefore high power consumption. However, if a gated sub-chain scheme is introduced (Figure 158), only one of the three

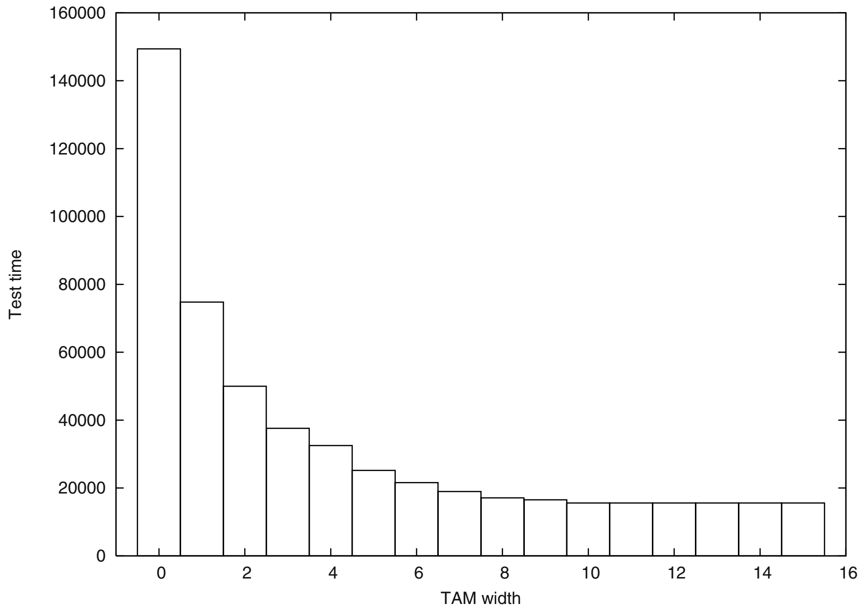


*Figure 156.*The test time versus the number of wrapper chains at core 11 in design P93791.

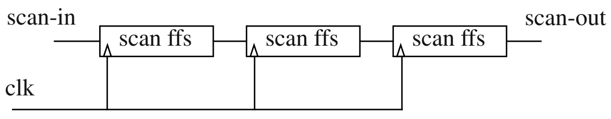


*Figure 157.*Original scan chain [241].

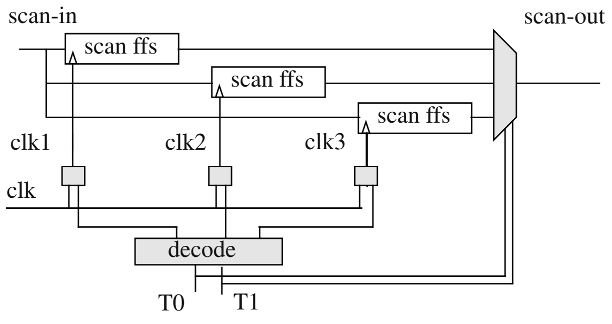


*Figure 158.*Scan chain with gated sub-chains[241].

chains is active at a time during the shift process. The advantage is that the switch activity is reduced in the scan-chains and also in the clock tree distribution while the test time remains the same in the two cases [241].

The wrapper proposed by Koranne allows, in contrast to approaches such as Boundary Scan, TestShell and P1500, several wrapper chain configurations

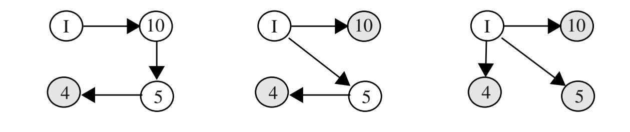
TAM width	*Wrapper chain partitions*	*Max length*
1	[10,5,4]	19
2	[(10),(5,4)]	10
3	[(10),(5),(4)]	10

Table 11. Scan chain partitions.

[140]. The main advantage is the increased flexibility in the scheduling process. We use a core with 3 scan chains of length {10, 5, 4} to illustrate the approach. The scan-chains and their partitioning into wrapper chains are specified in Table 11.

For each TAM widths (1, 2, and 3) a di-graph (directed graph) is generated where a node denotes a scan-chain and the input TAM, node I (Figure 159). An arc is added between two nodes (scan-chains) to indicate that the two scan-chains are connected, and the shaded nodes are to be connected to the output TAM. A combined di-graph is generated as the union of the di-graphs. Figure 160 shows the result of the generated combined di-graph from the three di-graphs in Figure 159. The indegree at each node (scan-chain) in the combined di-graph gives the number of signals to multiplex. For instance, the scan chain of length 5 has two input arcs, which in this example means that a multiplexer selecting between an input signal and the output of the scan chain of length 10 is needed. The multiplexing for the example is outlined in Figure 161.

Our combined approach works in two steps. First, we generate the reconfigurable wrapper using Koranne's approach. Second, we add clock gating, which means we connect the *inputs* of each scan-chain to the multiplexers, which is to be compared to connecting the *outputs* of each scan-chain as in the approach by Koranne. We illustrate our approach using the scan chains speci-



(a) 1 wrapper-chain (b) 2 wrapper-chains (c) 3 wrapper-chains

Figure 159. Di-graph representations.

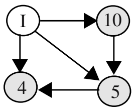


Figure 160. The union of di-graphs in Figure 159.

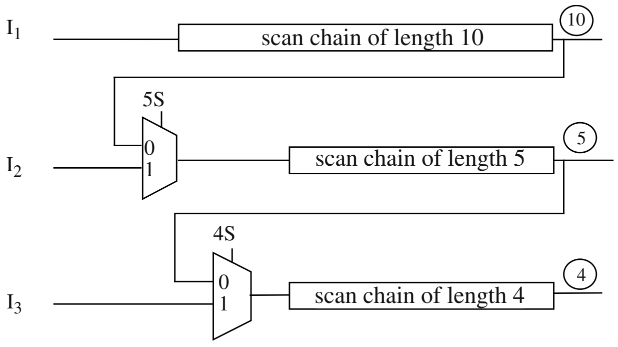


*Figure 161.*Multiplexing strategy [140].

fied in Table 11. The result is given in Figure 162, and the generated control signals are in Table 12.

The advantages with our approach are that we gain control of the test power consumption at each core, and we do not require the extra routing needed with Koranne's approach, as illustrated in Figure 163.

We could make use of the RPC wrapper at all cores, which would lead to a high flexibility since we could reconfigure the wrapper into any configuration. However, in order to minimize the overhead, we will use a systematic approach to select cores and number of configurations at each core (described below).

Wrapper chains	*T0 T1 T2*	*5S 4S*	*S1 S2*	*clk10 clk5 clk4*
3	0 0 0	1 1	0 0	1 1 1
2	0 0 1	1 x	0 0	1 0 0
	0 1 0	1 0	0 1	0 1 1
1	0 1 1	x x	0 x	1 0 0
	1 0 0	0 x	1 0	0 1 0
	1 0 1	0 0	1 1	0 0 1

Table 12. Control signals.

4 OPTIMAL TEST SCHEDULING

In this section we describe our power-constrained test scheduling technique that produces optimal test time in linear time while scheduling both core tests and cross-core tests. The approach also selects the wrapper configurations.

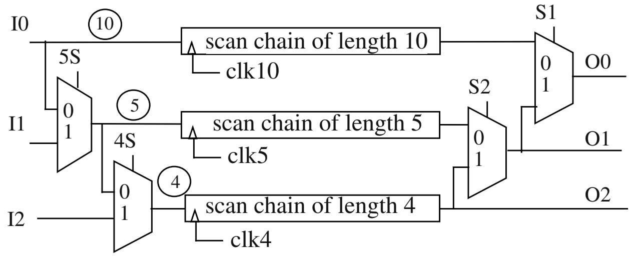
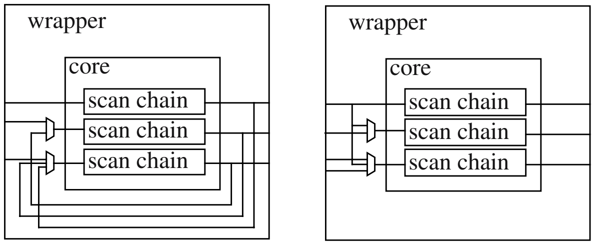


*Figure 162.*Our multiplexing and clocking strategy.



(a) Koranne's routing [140]. (b) Our approach.

*Figure 163.*Wrapper routing.

4.1 Optimal Scheduling of Core Tests

The test scheduling problem of core tests is equal to the independent job scheduling on identical machines since each test t_i at a core c_i, (i=1, 2, …, n) with testing time τ_i is independent on all other core tests, and each TAM wire w_j (j=1, 2, …, N_{tam}) is an independent machine used to transport test data [167]. The LB (lower bound) of the test time for a given TAM width N_{tam} can be computed by [23]:

$$LB = \max\left\{\max(\tau_i), \sum_{i=1}^{n} \tau_i / N_{tam}\right\} \tag{10.1}$$

The problem of independent job scheduling on identical machines can be solved in linear time ($O(n)$ for n tests) by using preemption [23]: assign tests to the TAM wires successively, assign the tests in any order and preempt tests into two parts whenever the LB is reached. Assign the second part of the preempted test on the next TAM wire starting from time point zero.

An example (Figure 164) illustrates the approach where the five cores and their test times are given. The LB is computed to 7 (Equation (10.1)) and due to that τ_i≤LB for all tests; the two parts of any preempted test will not overlap. The scheduling proceeds as follows: The tests are considered one by one, for instance, starting with a test at c_1 scheduled at time point 0 on wire w_1. At time point 4, when the test at c_1 is finished, the test at c_2 is scheduled to start.

At time point 7 when LB is reached, the test at c_2 is preempted and the rest of the test is scheduled to start at time 0 on wire w_2. Therefore the test at c_2 is partitioned into two parts. In execution of the test at c_2, the test starts at wire w_2 at time point zero. At time point 2, the test is preempted and resumed at time point 4. The test ends at time point 7. At the preemption of a test, another wire is assigned to the core and a multiplexer is added for wire selection. For the test of c_2, a multiplexer is added to select between w_1 and w_2.

In general preemptive scheduling, extra time is introduced at each preemption point due to the need to set up a job and also to save its state. In our case, the machines are the wires and no extra time is needed to set up and save the state. Also, in testing no other tasks are performed at the cores but testing, *i.e.* the core's state can be left as it is until the testing continues. The advantage is that the state of the core is already set and testing of it can start at once.

Assume that a core has a wrapper-chain of length l (l cycles are needed to perform a shift-in of a new vector and a shift-out of the previous test response). If the test is preempted when x% of the l cycles are shifted in it means that when the test restarts x% of the new test vector is already loaded and x% less cycles are needed in the first shift process, *i.e.* there is no time overhead due to setting up and saving the state of a core; all tests can be stopped at LB.

Finally, in some cases, such as for some types of memories such as DRAMs, the testing cannot be preempted. For instance, assume that test t_2 cannot be preempted as in Figure 164. In such a case, when LB is met, the scheduling algorithm restarts at LB (and not at time 0) and moves towards zero. The resulting schedule is in Figure 165. Note that, test t_2 now makes use of one wire during time point 4 to 5 and two wires during time 5 to 7, which is possible using the reconfigurable wrapper. This overlapping is further discussed below.

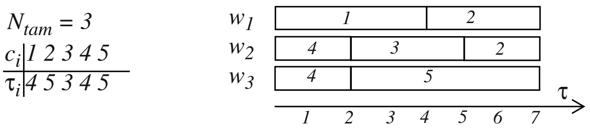


Figure 165. Optimal TAM assignment and preemptive scheduling where test t_2 cannot be interrupted.

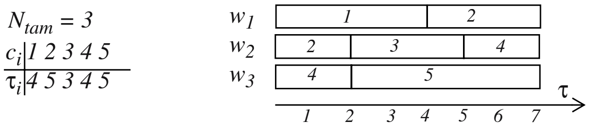


Figure 164. Optimal TAM assignment and preemptive scheduling.

4.2 Transformations for Optimal TAM Utilization

A long test time for one of the test in the system may limit the solution, *i.e.* LB is given by the test time of a test ($max(\tau_i)$ in Equation 10.1). In such a case, the test time can be reduced by assigning more TAM wires so that the length of the wrapper chains becomes shorter. Our approach is straight forward, we remove $max(\tau_i)$ from Equation 10.1:

$$LB = \max \sum_{i=1}^{n} \tau_i / N_{tam}. \tag{10.2}$$

When LB is computed, we use the scheduling approach illustrated above (Figure 164). For illustration, we use the same example but with a wider TAM (N_{tam}=7). The scheduling result is presented in Figure 166. A test may now overlap in using the wires (machines). For instance, the test at c_1 uses wire w_1 and w_2 during time period 0 to 1 and only wire w_1 during period 1 to 3. A reconfigurable wrapper is required to solve this problem.

We solve the overlapping problem in two consecutive steps: partitioning of the tests, and inserting of reconfigurable wrappers. After assigning TAM wires to all tests, we determine the partitions, which is illustrated in Figure 166. For instance, in partition 1 of the test at c_2, w_3 is used during period τ_{21} and in partition 2 of the test at c_2, w_2 and w_3 are used during period τ_{22}. From this we can determine that two wrapper chains are initially needed and then a single wrapper chain is needed. In total, two configurations are needed for core c_2. The generic partitioning of a test's usage of wires over the testing time is given in Figure 167. For each test, a start time $start_i$ and an end end_i are assigned by the algorithm, respectively. The number of partitions, which will be the number of configurations, is computed for each test by the algorithm given in Figure 168. If the test time τ_i for a test t_i is below LB, only one configuration is needed. A multiplexer might be required for wire selec-

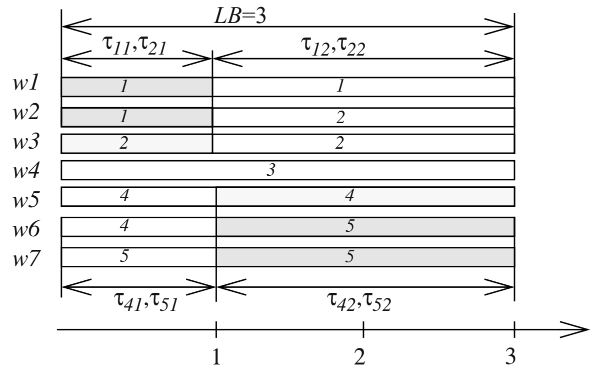


Figure 166. Partitioning of the schedule into sessions.

```
for all $t_i$ begin
  if $\tau_i \le$ LB then begin
    if $start_i \le end_i$ then begin
      no pre-emption for $t_i \rightarrow$ no wrapper logic is added.
    end else begin
      the test is split into two parts at different wires $\rightarrow$
      1 multiplexer is inserted.
    end
  end else begin
    if $start_i \le end_i$ then begin
      one configuration needed from time point 0 to $start_i$,
      one configuration needed from time point $start_i$ to $end_i$,
      one configuration needed from time point $end_i$ to LB,
    end else begin
      one configuration needed from time point 0 to $end_i$,
      one configuration needed from time point $end_i$ to start,
      one configuration needed from time point $start_i$ to LB,
    end
  end
end
```

*Figure 168.*Algorithm to determine wrapper logic.

tion if $start_i > end_i$. From the algorithm, we find that the maximal number of partitions per test is three, which means we in the worst case have to use three configurations per core. The wrapper logic is then in range |C|×3×*technology parameter* (maximum 3 configurations per core). In the approach by Koranne [140] the added logic, if a reconfigurable wrapper is added at all cores, is given by |C|×N_{tam}×*technology parameter.*

The preemption based approach, assumes that a job can be divided over machines. And that the job time is reduced linearly with the number of machines. In the case of testing, it means that the testing times reduces in a linear way with the number of wrapper chains. However, Figure 156 showed that at some points the testing time does not decrease as the number of wrapper chains are increased. A major reason is that the scan-chains are

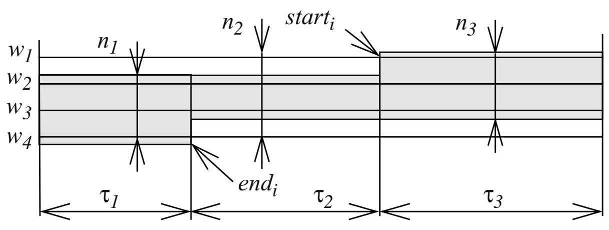


*Figure 167.*Bandwidth requirement for a general test.

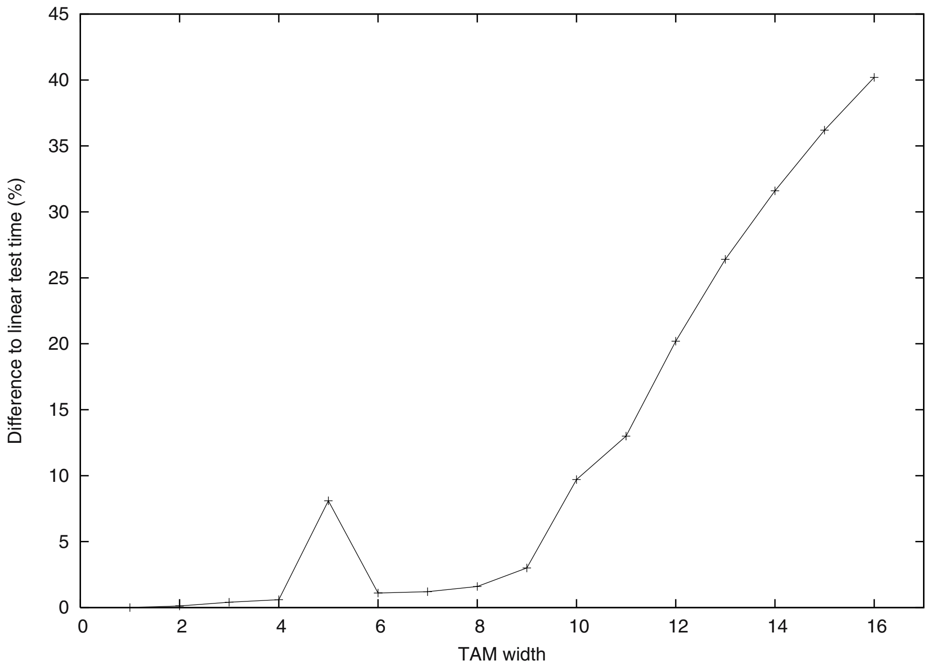


*Figure 169.*The test time over a set of TAM widths (wrapper chains) for core 11.

unbalanced. We have made further studies on the test time versus the number of wrapper chains. We took core 11 from design P93791 and plotted the difference to linear test time (τ_x-τ_1/x at x tam wires where τ_1 is the test time at a single TAM wire) (Figure 169). The difference to the assumption that the testy time is linear increases as the number of wrapper chains increases. However, note that the difference is extremly low when the number of wrapper chains are few. It means that an efficient test scheduling technique should make use of as few wrapper chains as possible.

4.3 Cross-Core Test Scheduling

There are no wrapper conflicts among *core tests* since each core has its dedicated interface to the TAM. For *cross-core tests,* on the other hand, there is not a dedicated interface to the TAM and wrapper conflicts must therefore be taken into account.

Test conflicts can be modeled using a resource graph [41]. The system in Figure 155 is modeled as a resource graph as in Figure 170, where the nodes represents the tests and the resources. A resource may consist of cores and wrapper cells, which are explicitly captured as shown in Figure 170. An edge between a test and a resource (core or a wrapper cell) indicates that the test requires the resource during testing. The test conflicts are due to that the wrapper cells can only be in one mode at a time: In *core testing* the wrapper cells are in internal test mode while in *cross-core test* they are in external test mode. In Figure 170, we denote arcs from core tests with (*i*) - *internal mode* and arcs from cross-core tests with (*e*) - *external mode*. Breaking the graph into two resource graphs, one for core tests and one for cross-core tests makes it possible to schedule all core tests with the algorithm in section 4.1.

For a cross-core test, test vectors are transported from the TAM to a wrapped core placed in external test mode, which is feeding the test vectors to the unwrapped core (the target for the cross-core test). The test responses are captured at a wrapped core also in external test mode and then the test responses are fed to the TAM. A cross-core test involves therefore three cores. In Figure 155 a cross-core test at c_4 is performed by setting the wrappers at c_1 and c_2 in external test mode, and then test vectors are transported to c_4 through the wrapper at c_1 and the test response from c_4 is transported to the TAM using the wrapper at c_2. This demonstrates a cross-core test with a *one-to-one* mapping where the wrapper cells at the functional outputs at c_1 are connected via c_4 to the functional input wrapper cells at c_2.

Several other mapping combinations are possible for the wrapper input and wrapper output cells, including *one-to-many*, *many-to-one* and *many-to-many*. These mappings cover all combinations and we assume that each functional input and output can be in only one such mapping and in only one test set. In Figure 155, for instance, a functional output wrapper cell at c_1 cannot be in one test set with an input wrapper cell at c_5 and in another test set with an input cell at c_3. However, a wrapper cell at c_1 can be in the same test set as a wrapper cell at c_3 and at c_5. In some cases, the functional inputs and outputs at a wrapped core may be connected to different cores. Figure 155 shows such an example where the outputs at c_1 are partitioned into two sets, one set used by c_2 and c_4 and another set used by c_3 and c_5. However, these partitions operates independently when the wrapper is in external test mode. Therefore there is no conflict.

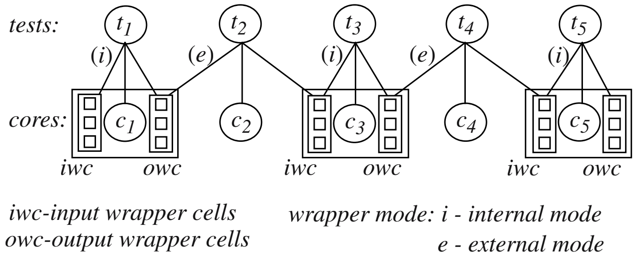


Figure 170. A resource graph of the system in Figure 155

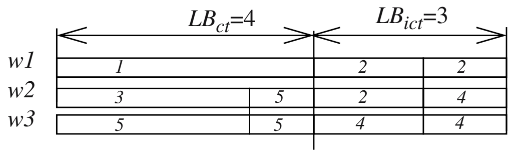


*Figure 171.*Partitioning of the schedule in Figure 166.

We have above shown that partitioning the tests into two distinct test phases, *core tests* and *cross-core tests,* eliminates the wrapper conflicts between the set of core tests and the set of cross-core tests. We make use of this property and divide the test scheduling into two separate parts, core testing followed by cross-core testing. The partition of the tests means we divide the tests into a core test part given by LB_{ct} and a cross-core test part given by LB_{ict}. To illustrate, we take the example in Figure 164 assuming that the test at c_2 and c_4 are cross-core tests, which means that executing the test at c_2 entails concurrent testing at c_1 and at c_3, and testing t_4 entails concurrent testing at c_3 and at c_5. The core tests are performed at core c_1, c_3 and c_5 and the cross-core tests are at core c_2 and c_4. The lower bound for the core tests: LB_{ct}=(4+3+5)/3=4 and the lower bound for the cross-core tests: LB_{ict}=(5+4)/3=3, *i.e.* $LB=LB_{ct}+LB_{ict}$. One test schedule is presented in Figure 171.

The test scheduling algorithm consists of four steps:

1. Compute LB_{ct} (lower bound) for the core tests,
2. Schedule all core tests,
3. Compute LB_{ict} for the cross-core tests, and
4. Schedule all cross-core tests.

The algorithm starts by selecting a core and assigning it to wire zero at time zero. If the core's test time is higher than LB, a new wire is used. The test time is reduced until it reaches zero and each time LB is reached, a new wire is added to the test. The start time and the end time of the tests are used when creating the partitions. We observe that the LB defines the test application time and also that all TAM wires are fully utilized, all tests ends at the same time (Figure 166). It means that partitioning the tests into two partitions (core tests and cross-core tests) will still produce an optimal solution.

4.4 Optimal Power-Constrained Scheduling

Chou *et al.* introduced a power model where each test is denoted with a fixed power consumption value [41]. Recently a more elaborate model was presented by Rosinger *et al.* [231]. The power consumption depends on the switching activity in the circuit, and by reducing the switching activity, the

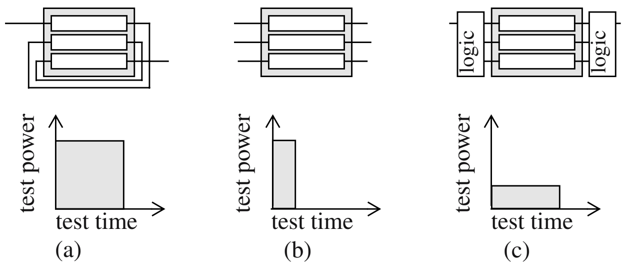


*Figure 172.*Core design alternatives.

power consumption is reduced. Saxena *et al.* [241] showed by experiments that sub-gating a single scan-chain into three sub-chains reduces the test power to approximately a third. Rosinger *et al.* proposed a technique to reduce both shift and capture power consumption. The experimental results indicate, in most cases, that the power consumption is lower than the intuitive approximation of dividing the consumption at a single chain with the number of partitions [230]. In this chapter, we use a power model based on the results by Saxena *et al*, which means the power depends on the number and the length of the wrapper chain partitions. However, a more elaborate power model can easily be adopted in our approach.

We use an example to illustrate the test power modeling at scan-chain level (Figure 172). In Figure 172 (a) a single *wire* is assigned to the core where the three *scan chains* form a single *wrapper chain*. The result is that the wire usage is minimized but both the test time and the test power are relatively high. In Figure 172 (b) three TAM wires (one per wrapper chain) are used resulting in a lower test time while the test power consumption remain the same as in Figure 172(a). However, in Figure 172(c), our approach, the same test time is achieved as in Figure 172(a) but at a lower test power consumption. The reduction in test power in this example is due to that each scan-chain is loaded in a sequence, and not more than one scan-chain is activated at a time.

Our test scheduling technique [167] minimizes the number of TAM wires at each core by assigning as few wires as possible to each core. It means that each wrapper chain includes a high number of scanned elements. This is an advantage since it maximizes the possibility to gate scan-chains at each wrapper chain and hence control the test power consumption at the cores.

We assume that the test power at a core is evenly distributed over the scanned elements. The algorithm to compute the power limit (P_{limit}) for a system is in Figure 173. At step 2, the LB is computed, and at step 3, the maximal number of required TAM wires are computed. At step 4, the amount of test power consumed by each scan chain, and wrapper cell is computed. At steps 5 and 6, the N_{tam} values with highest test power are summarized which is the P_{limit}. If P_{limit} is below P_{max} ($P_{limit} \le P_{max}$), optimal test time can be achieved.

We have now a relationship between the TAM bandwidth and the test power. The advantage is that we can use it when we determine the TAM bandwidth; N_{tam} can be increased as long as $P_{limit} \le P_{max}$. It is also possible to increase the frequency of the test clock in order to minimize the test time as long as $P_{limit} \le P_{max}$.

4.5 Minimization of TAM Wiring

The test scheduling approach above minimizes the number of TAM wires assigned to each core. The advantage is that, even if the floor-plan is unknown, the TAM routing cost is minimized since as few TAM wires as possible are assigned to each core. If the floor-plan is known, we can further minimize the TAM routing since the scheduling approach does not require any particular sorting of the tests. We take the system in Figure 164 with N_{tam}=7 resulting in a test schedule as in Figure 166 where the cores are sorted (and numbered) clock-wise as in Figure 174. The advantage is that neighbouring cores share TAM wires. For instance core 2, which makes use of TAM wire w_2 as soon as core 1 finish its use of w_2. Cores placed far away from each other are not sharing TAM wires, such as core 5 and core 3.

```
1. Given: a system with $i$ cores, where each core $i$ con-
   sists of $ff_i$ scanned flip-flops and wrapper cells parti-
   tioned into $j$ partitions each of length $sc_{ij}$ (including
   wrapper cells). The test time $\tau_i$ is computed as if all $ff_i$
   elements are connected to a single wrapper chain. The
   test power when all $ff_i$ elements are active is given by
   $p_i$. $N_{tam}$ is the TAM bandwidth.
2. Compute $LB$ (lower bound) (algorithm in Section 4.3)
3. For each core $i$ compute $w_i$ as the maximal number of TAM
   wires that can be assigned to it assuming preemptive
   scheduling:
        $w_i = \left\lceil \frac{\tau_i}{LB} \right\rceil$
4. For each scan-chain partition $s_{ij}$ compute its power:
        $p_{ij} = \frac{p_i}{ff_i} \times sc_{ij}$
5. For all cores: select the $w_i$ scan elements with highest
   power value and sort them descending in a list $L$.
6. For all scan elements in L select the $N_{tam}$ first and the
   $P_{limit}$ is equal to the summation of the $N_{tam}$ values.
```

Figure 173. Algorithm to compute the power limit.

5 EXPERIMENTAL RESULTS

We have above shown that the test scheduling problem can be solved in linear time by using our proposed RPC wrapper. For illustration, we have made experiments using the P93791 design, one of the largest ITC'02 benchmarks [189]. We have made a test time comparison between our approach and techniques proposed by Goel and Marinissen [75, 74, 73], Huang *et al.* [100], Iyengar *et al.* [113, 112, 114], Koranne [140], and Koranne and Iyengar [141]. In our implementation we made use of the wrapper chain algorithm proposed by Iyengar *et al.* [113]. Similar to all previous approaches, we assume that all tests are core tests and that the designs are flat (no hierarchy). The results are collected in Table 13. In some cases the previously reported results are below the lower bound computed by Goel and Marinissen [75]. These results are excluded and placed within parentheses. In six out of the seven TAM bandwidths our approach finds the solution with the lowest test time. In the other case, our approach is the second best. The results are also illustrated as the difference to lower bound (τ-LB)/LB×100 for all approaches in Figure 175 and in more detail for the best approaches in Figure 176. It should be noted that most approaches are less than 10% from the lower bound (Figure 175) and that the best approaches are less than 6% from the lower bound. Our approach is less than 3% from lower bound at all bandwidths. It should be noted that the lower bound of the test time is computed assuming that the last stimulus from the previous core can be shifted out on wires concurrently as the first vector of the following core is loaded on the same wires [75]. We have in our approach not considered such optimization. Hence, our approach can be further optimized.

We have in Table 14 collected the overhead due to the use of our reconfigurable wrapper. The overhead is computed as follows. For cores with a single

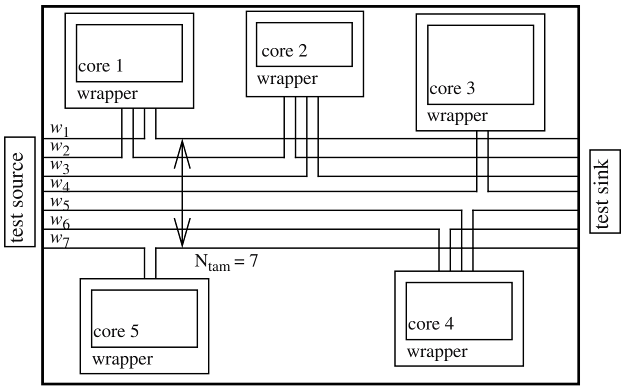


*Figure 174.*The example system assuming the five wrapped cores to be floor-planned.

Approach	*Test application time: T*						
	TAM=16	*TAM=24*	*TAM=32*	*TAM=40*	*TAM=48*	*TAM=56*	*TAM=64*
Lower bound [75]	1746657	1164442	873334	698670	582227	499053	436673
Enumerate [113]	1883150	1288380	944881	929848	835526	537891	551111
ILP [113]	1771720	1187990	887751	(698583)	599373	514688	460328
Par eval [112]	1786200	1209420	894342	741965	599373	514688	473997
GRP[114]	1932331	131084	988039	794027	669196	568436	517958
Cluster [73]	-	-	947111	816972	677707	542445	467680
TRA [74]	1809815	1212009	927734	738352	607366	529405	461715
Binpack[97]	1791860	1200157	900798	719880	607955	521168	459233
CPLEX[140]	1818466	(1164023)	919354	707812	645540	517707	453868
ECTSP[140]	1755886	(1164023)	919354	707812	585771	517707	453868
ECTSP1[140]	1807200	1228766	967274	890768	631115	562376	498763
TB-serial [75]	1791638	1185434	912233	718005	601450	528925	455738
TR-serial [75]	1853402	1240305	940745	786608	628977	530059	461128
TR-parallel [75]	1975485	1264236	962856	800513	646610	540693	477648
K-tuple [141]	2404341	1598829	1179795	1060369	717602	625506	491496
Our approach	1752336	1174252	877977	703219	592214	511925	442478

Table 13. Test time comparison on P93791.

TAM bandwidth assigned to it, only one bandwidth is required and the cost is assumed to be zero. In some cases, only a multiplexer is to be added for the selection between wires and we assumed such cost to be equal to 1. For cores with three configurations, we assumed the cost to be equal to 3. We have collected the overhead at each scanned core and for the rest of the cores. The test times for the cores without scan-chains are in general shorter and in only a few cases additional logic is required.

We have also made experiments considering test power consumption. First, we illustrate the use of the RPC wrapper at core 12 assuming a fixed test time at a single TAM wire (Table 15). The test time remains the same while the test power consumption can be adjusted depending on the number of gated wrapper-chains. The added wrapper logic depends on the number of wrapper chains which indicate how many partitions are to be gated. An advantage of our scheduling approach is that we assign as few TAM wires as possible to each core, which makes it possible to have a high ratio between the number of gated wrapper chains and the number of TAM wires at each core. In other words, we have a high possibility to regulate the test power at each core.

We have compared our approach with the multiplexing and the distribution architecture [5]. In the multiplexing approach all cores are tested in a sequence where the full bandwidth is given to each core at a time. In the distribution architecture, every core is given its dedicated TAM wires. The distribution architecture is sensitive to test power consumption since the testing of all cores are started at the same time. All results are collected in Table 16. The distribution architect is not applicable when the TAM bandwidth is below the number of cores (32 in P93791). At the 50000 power limit the distribution architecture can not be used since activating all cores exceeds the power limit. At the limit 20000, the multiplexing approach is not applicable since core 6 limits the solution with its consumption of 24674. Our approach results in the same test time, however, the wrapper logic is increased in order to gate the wrapper chains.

TAM band-width	*Test time*	*Wrapper logic at core c_i*													*Total*
		c_1	c_6	c_{11}	c_{12}	c_{13}	c_{14}	c_{17}	c_{19}	c_{20}	c_{23}	c_{27}	c_{29}	*Cores with no scan chains*	
4	6997584	0	1	0	0	0	1	0	0	1	0	0	0	0	3
8	3498611	0	3	0	1	0	1	0	1	1	0	1	0	1	9
16	1752336	1	3	0	1	1	1	1	1	3	1	1	1	1	16
24	1174252	2	3	0	3	3	3	3	1	3	3	3	1	1	29
32	877977	2	3	1	3	3	3	3	3	3	3	3	3	1+1	35
40	703219	2	3	0	3	3	3	3	3	3	3	3	3	3+1	36
48	592214	2	3	0	3	3	3	3	3	3	3	3	3	3+2	37
56	511925	2	3	0	3	3	3	3	3	3	3	3	3	3+2+3	40
64	442478	2	3	1	3	3	3	3	3	3	3	3	3	3+3+3	42

Table 14. Number of configurations per core for our approach on P93791 (Table 13).

6 CONCLUSIONS

In this chapter we have proposed a reconfigurable power-conscious core test wrapper, and described its application to preemptive test scheduling. The main advantages with the wrapper is the possibility to (1) control the test power consumption at each individual core and (2) the possibility to achieve several TAM bandwidths for a given core test. Test power control at each core is important since it allows testing at a higher clock frequency, which can

Wrapper chains	*Power consumption*	*Wrapper logic*
1	4634	0
2	2317	1
3	1545	3
4	1159	4
5	927	5
6	773	6

Table 15. Test power consumption options at core 12 (P93791) with the RPC wrapper at a fixed test time (=1813502) and fixed TAM width (=1).

be used to further decrease the test time. Test power control also allows a higher number of cores to be tested simultaneously. Variable bandwidth is important since it increases the flexibility during the scheduling process. The advantage of the proposed test scheduling scheme, besides the production of an optimal solution in respect to test time, is that it also considers cross-core testing, which are used to test unwrapped testable units such as interconnections and used-defined logic.

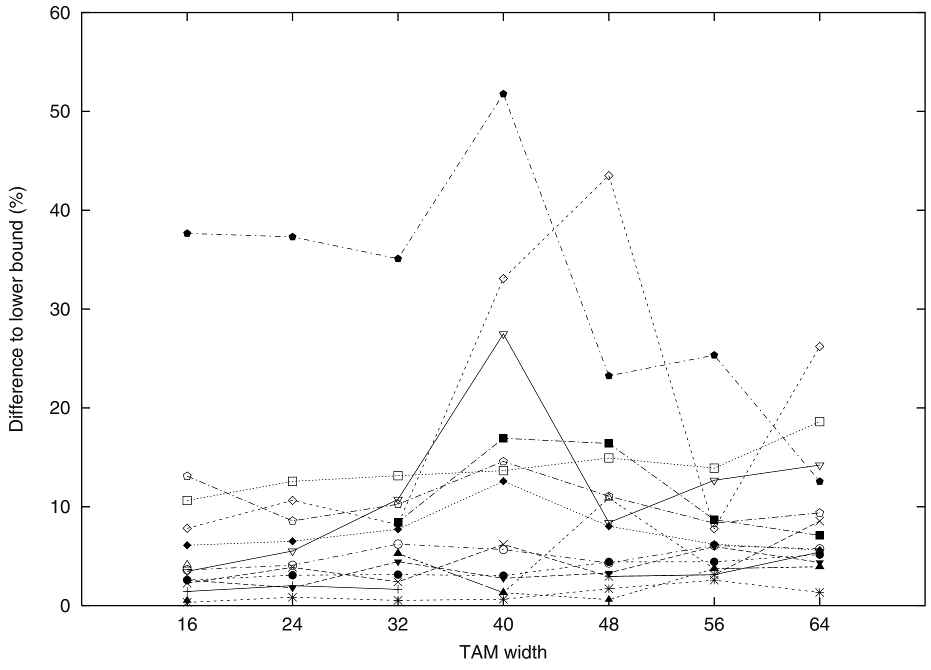


*Figure 175.*Difference to lower bound for the approaches in Table 13.

P_{max}	*TAM width*	*Multiplexing architecture*	*Distribution architecture*	*Our approach*
		Test time	*Test time*	*Test time*
100000	4	7113317	Not applicable	6997584
	8	3625510	Not applicable	3498611
	16	1862427	Not applicable	1752336
	24	1262427	Not applicable	1174252
	32	1210398	5317007	877977
	40	1119393	1813502	703219
	48	660143	1126316	592214
	56	645698	907097	511925
	64	645682	639989	442478
50000	4	7113317	Not applicable	6997584
	8	3625510	Not applicable	3498611
	16	1862427	Not applicable	1752336
	24	1262427	Not applicable	1174252
	32	1210398	Not applicable	877977
	40	1119393	Not applicable	703219
	48	660143	Not applicable	592214
	56	645698	Not applicable	511925
	64	645682	Not applicable	442478
20000	4	Not applicable	Not applicable	6997584
	8	Not applicable	Not applicable	3498611
	16	Not applicable	Not applicable	1752336
	24	Not applicable	Not applicable	1174252
	32	Not applicable	Not applicable	877977
	40	Not applicable	Not applicable	703219
	48	Not applicable	Not applicable	592214
	56	Not applicable	Not applicable	511925
	64	Not applicable	Not applicable	442478

Table 16. Test time on P93791 for the multiplexing architecture [5], the distribution architecture [5], and our approach at different power limitations.

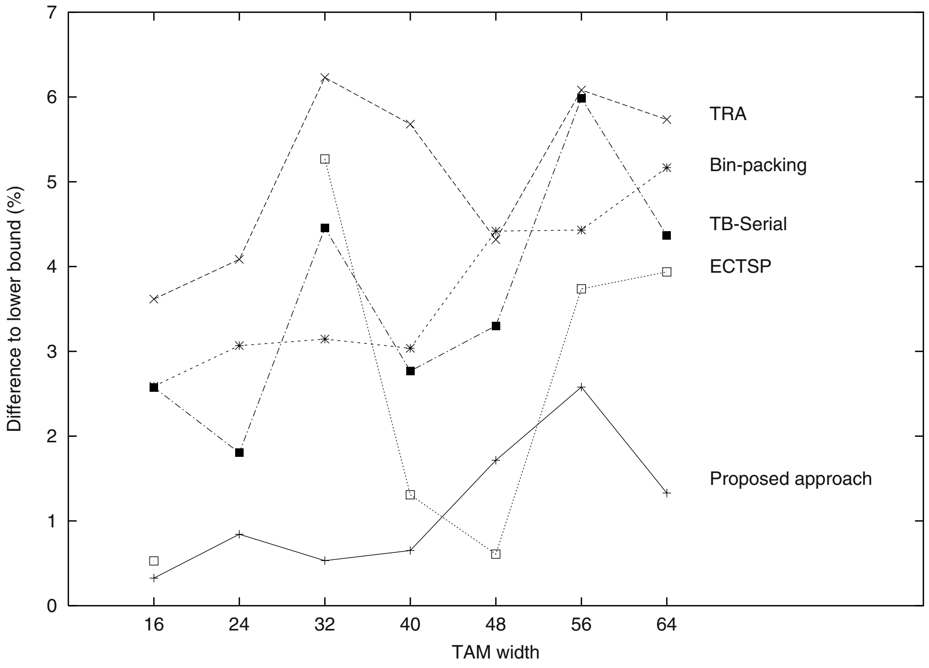


*Figure 176.*Difference to lower bound for the best approaches in Table 13.

Chapter 11

An Integrated Framework for the Design and Optimization of SOC Test Solutions[1]

1 INTRODUCTION[2]

We propose an integrated framework for the design of SOC test solutions, which includes a set of algorithms for early design space exploration as well as extensive optimization for the final solution. The framework deals with test scheduling, test access mechanism design, test sets selection, and test resource placement. Our approach minimizes the test application time and the cost of the test access mechanism while considering constraints on tests and power consumption. The main feature of our approach is that it provides an integrated design environment to treat several different tasks at the same time, which were traditionally dealt with as separate problems. We have made an implementation of the proposed heuristic used for the early design space exploration and an implementation based on Simulated Annealing for the extensive optimization. Experiments on several benchmarks and industrial designs show the usefulness and efficiency of our approach.

The testing of System-on-Chip (SOC) is a crucial and time consuming problem due to the increasing design complexity. Therefore it is important to provide the test designer with support to develop an efficient test solution.

The work-flow for a test designer developing a test solution consists typically of two consecutive parts: an early design space exploration and an extensive optimization for the final solution. During the process, conflicts and limitations must be carefully considered. For instance, tests may be in conflict with each other due to the sharing of test resources; and power consumption must be controlled, otherwise the system may be damaged during test. Furthermore, test resources such as external testers support a limited number of scan-chains and have a limited test memory, which also introduce constraints.

Research has been going on in developing techniques for test scheduling, test access mechanism (TAM) design and testability analysis. For example, a framework to support the design of test solutions for SOC with Built-In Self-Test (BIST) has been proposed by Benso *et al.* [15]. In this chapter, we com-

1. The chapter is based on the papers presented at Design Automation and Test in Europe (DATE), 2001 [158], International Conference on Computer-Aided Design (ICCAD), 2001 [160], Journal on Electronic Testing: Theory and Applicationd (JETTA) 2002 [164, 165].

2. The research is partially supported by the Swedish National Program STRINGENT.

bine and generalize several approaches in order to create an integrated framework for the development of SOC test solutions where:

- tests are scheduled to minimize the total test time,
- a minimal TAM is designed,
- test resources are floor-planned, and
- test sets for each core with test resources are selected.

The above set of tasks is performed in a single algorithm, which considers test conflicts, power limitation and test resource constraints [158,160,157]. The algorithm is suitable for early design space exploration due to its low computational complexity, which is an advantage since it will be used iteratively many times.

Chakrabarty showed that test scheduling is equal to the open-shop scheduling [28], which is known to be NP-complete and the use of heuristics are therefore justified. Several heuristics have been proposed [5,28,25,41,65,110,202,287]; however, they have been evaluated using rather small benchmarks. For such benchmarks, a technique based on Mixed-Integer Linear-Programming (MILP) can be used [28]. A disadvantage is the complexity of solving the MILP model since the size of it quickly grows with the number of tests making it infeasible for large industrial designs. We have therefore made an implementation based on Simulated Annealing for the extensive optimization of the test schedule and the TAM design for the final solution [131]. We have performed experiments on several benchmarks and on an Ericsson design consisting of 170 tests to show the efficiency and usefulness of our approach.

The rest of the chapter is organised as follows. Related work is outlined in Section 2 and the system modeling is introduced in Section 3. Factors affecting the test solution are presented in Section 4. The integrated algorithm is described in Section 5 and how we used Simulated Annealing is outlined in Section 6. The chapter is concluded with experimental results in Section 7 and conclusions are in Section 8.

2 RELATED WORK

The basic problem in test scheduling is to assign a start time for all tests. In order to minimize the test application time, tests are scheduled as concurrent as possible, however, various types of constraints must be considered.

A test to be scheduled consists of a set of test vectors produced or stored at a test source (placed on-chip or off-chip). The test response from the test is evaluated at a test sink (placed on-chip or off-chip). When applying a test, a test conflict may occur, which must be considered during the scheduling process. For instance, often a testable unit is tested by several test sets (usually an external test set and a on-chip test set are required to reach high test quality). If several tests are used for a testable unit, only one test can be applied to the testable unit at a time.

Constraints on power consumption must be considered otherwise the system can be damaged. During testing mode, the power consumption is usually higher compared to normal operation [68]. For instance, consider a memory, which often is organized in memory banks. During normal operation, only a single bank is activated. However, during testing mode, in order to test the system in the shortest possible time it is desirable to concurrently activate as many banks as possible [41].

The use of different test resources may entail constraints on test scheduling. For instance, external testers have limitations of bandwidth due to that a scan chain operates usually at a maximum frequency of 50 MHz [92]. External testers can usually only support a maximum of 8 scan chains [92], resulting in long test application time for large designs. Furthermore, an external tester's memory is limited by its size [92].

Zorian proposed a test scheduling technique for fully BISTed systems where test time is minimized while power constraint is considered [287]. The tests are scheduled in sessions where tests at cores placed physically close to each other are grouped in the same test session. In a fully BISTed system, each core has its dedicated test source and test sink; and there might not be any conflicts among tests, *i.e.* the tests can be scheduled concurrently. However, in the general case, conflicts among tests may occur. Garg *et al.* proposed a test scheduling technique where test time is minimized for systems with test conflicts [65] and for core-based systems a test scheduling technique is proposed by Chakrabarty [28, 25]. Chou *et al.* proposed an analytic test scheduling technique where test conflicts and power constraints are considered [41]. A resource graph is used to model the system where an arc between a test and a resource indicate that the resource is required for the test, Figure 177. From the resource graph, a test compatibility graph (TCG) is generated (Figure 178) where each test is a node and an arc between two nodes indicates that the tests can be scheduled concurrently. For instance t_1 and t_4 can be scheduled at the same time. Each test is attached with its test time and its power consumption and the maximal allowed power consumption is 10. The tests t_1 and t_5 are compatible, however, due to the power limit they cannot be scheduled at the same time.

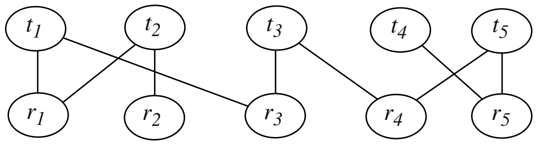


*Figure 177.*Resource graph of an example system.

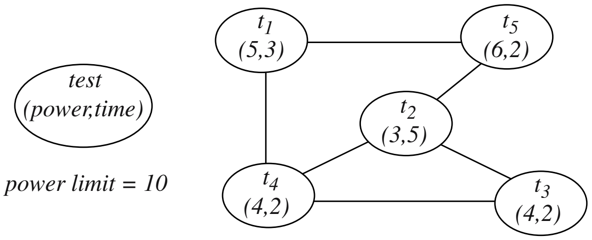


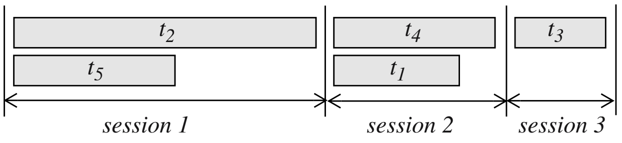
*Figure 178.*Test compatibility graph (TCG) of the example system (Figure 177).

Another test scheduling approach is proposed by Muresan *et al.* where constraints among tests and on power consumption are considered [202]. Favour is given to reduce the test time by allowing new tests to start even if all tests in a session are not completed. Iyengar and Chakrabarty proposed a pre-emptive test scheduling technique where the test for a testable unit may be interrupted and resumed later, *i.e.* the test set is partitioned into several test sets [110]. Using a scheme by Craig *et al.*, the above discussed scheduling techniques can be grouped in [47]:

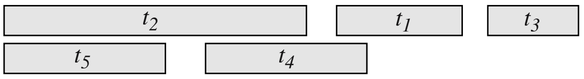
- nonpartitioned testing,
- partitioned testing with run to completion, and
- partitioned testing.

The differences among the grouping are illustrated with five tests (t_1,..., t_5) in Figure 179. The scheduling strategies proposed by Zorian [287] and Chou *et al.* [41] are nonpartitioned (Figure 179(a)), the strategy proposed by Muresan *et al.* is partitioned testing with run to completion (Figure 179(b)) and the approach proposed by Iyengar and Chakrabarty [110] is partitioned testing (Figure 179(c)).

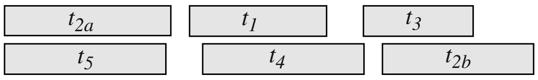
A test infrastructure is responsible for the transportation of test vectors from test sources to cores under test and test responses from cores under test to test sinks. It consists of two parts; one for the test data transportation and



(a) Nonpartitioned testing



(b) Partitioned testing with run to completion



(c) Partitioned testing

*Figure 179.*Scheduling approaches.

one for the control of the transportation. In the approach for fully BISTed systems proposed by Zorian [287], tests at cores placed physically close to each other are grouped in the same test session. The main advantage is that the same control structure can be used for all tests in the session, which minimizes the routing of control wires. In general, systems are not tested with a BIST structure only for each testable unit and therefore a TAM is required [27,30,26,212]. Chakrabarty proposed an integer linear programming (ILP) for the allocation of TAM width [27,30,26] and the effect on test time for systems using various design styles for test access with the TestShell wrapper is analysed by Aertes and Marinissen [5].

To ease test access the cores can be placed in wrappers such as Boundary scan [19], TestShell [187] or P1500 [104, 186]. The Boundary scan technique, developed for PCB designs, suffers from long testing time due to the shifting process, and it becomes even worse for SOC designs since the amount of test data to be transported increases. An alternative is to use an architectures as proposed by Aertes and Marinissen (Figure 180.(a,b,c)) where the test time only depends on the number of flip-flops within the cores and the number of test vectors, *i.e.* the transportation to and from a core is free [5]. The daisy-chained architecture uses a clocked bypass, however, Marinissen *et al.* have also proposed a wrapper design library allowing bypass structures without clock delay [187].

The above scheduling techniques all assume that all testable units have their selected test sets. Sugihara *et al.* proposed a technique for selecting test sets where each core may be tested by one test set from an external tester and one test set from a dedicated test generator for the core [256].

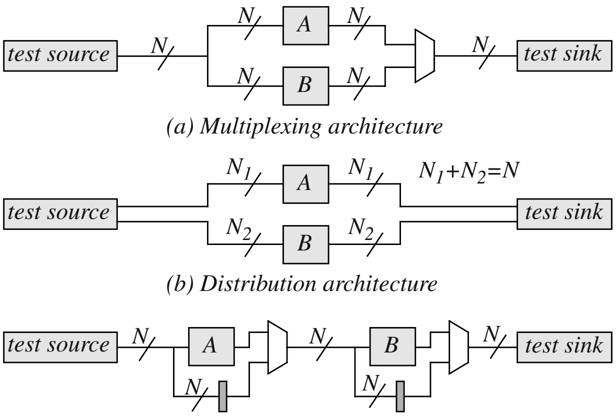

test source
N
A
B
test sink
(a) Multiplexing architecture
N_1
N_2
$N_1+N_2=N$
(b) Distribution architecture
(c) Daisy-chain architecture

*Figure 180.*Multiplexing, distribution and daisy-chain architecture [5].

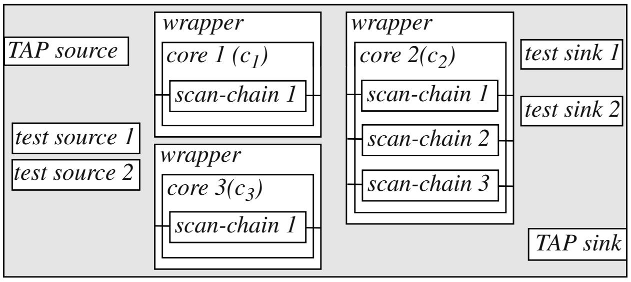


Figure 181. An example system.

3 SYSTEM MODELING

This section describes the formal notation we use to model the SOC under test. An example of a system under test is given in Figure 181 where each core is placed in a wrapper to ease test access. A DFT technique is added to each core and in this example all cores are tested using the scan technique. The test access port (*tap*) is the connection to an external tester and for instance the test source, *test source* 1, and test sink, *test sink* 1, are implemented on-chip [158,157]. Applying several sets of tests, where each test set is produced or stored at a test source and the test response is analysed at a test sink, tests the system.

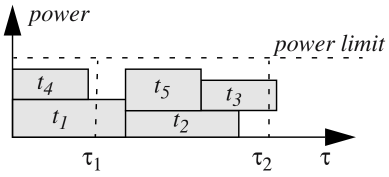


*Figure 182.*Example of test scheduling.

The system in Figure 181 can be modelled as a *design with test*, $DT = (C, R_{source}, R_{sink}, p_{max}, T, source, sink, core, constraint, mem, bw)$, where:

$C = \{c_1, c_2,..., c_n\}$ is a finite set of cores where each core $c_i \in C$ is characterized by $p_{idle}(c_i)$: idle power;

$R_{source} = \{r_1, r_2,..., r_p\}$ is a finite set of test sources;

$R_{sink} = \{r_1, r_2,..., r_q\}$ is a finite set of test sinks;

p_{max}: maximal allowed power at any time;

$T = \{t_1, t_2,..., t_o\}$ is a finite set of tests, each consisting of a set of test vectors. Several tests form a *core test* (*CT*) and each core, c_i, can be associated with several core tests, CT_{ij} (j=1,2,...,l).

Each test t_i is characterized by:

$\tau_{test}(t_i)$: test time for test t_i,

$p_{test}(t_i)$: test power for test t_i,

$bw(t_i)$: bandwidth required for t_i

$mem(t_i)$: memory required for test pattern storage.

source: $T \rightarrow R_{source}$ defines the test sources for the tests;

sink: $T \rightarrow R_{sink}$ defines the test sinks for the tests;

core: $T \rightarrow C$: the core where a test is applied;

constraint: $T \rightarrow 2^C$: the set of cores required for a test;

$mem(r_i)$: memory available at test source $r_i \in R_{source}$;

$bw(r_i)$: bandwidth at test source $r_i \in R_{source}$.

For each test, one test sink and one test source are required. In our model, it is possible for every test to combine any type of test source (on-chip or off-chip) with any type of test sink (on-chip or off-chip), which is important to allow high flexibility. To further give the designer a high flexibility to explore different combinations of tests, it is possible to define several set of tests (CT_{ij}) for each core where each such set tests the core.

Given a system as in Figure 181 where for each of the cores c_1, c_2, c_3, one CT exists. For core c_1, $CT_{11}=\{t_1, t_2\}$, for core c_2, $CT_{21}=\{t_4, t_5\}$ and for c_3, $CT_{31}=\{t_3\}$. The on-chip test source 1 is required by t_1 and t_3 and the on-chip test sink 1 is required by t_3 and t_5. The resource graph in Figure 177 and the test compatibility graph in Figure 178 model the system when r_3 (test source 1) is needed by t_1 and t_3 and r_4 (test sink 1) needed by t_3 and t_5.

4 THE SOC TEST ISSUES

In this section we discuss the different issues considered by our SOC test framework. We also demonstrate the importance of considering them in an integrated manner.

4.1 Test Scheduling

The test scheduling problem can be seen as placing all tests in a diagram as in Figure 182 while satisfying all constrains. For instance, t_1 and t_2 cannot be applied at the same time since both tests the same core, c_1 (example in Section 3).

The basic difference between our scheduling technique [158,157] and the approaches proposed by Zorian [287] and Chou *et al.* [41] is, besides that we design the TAM while scheduling the tests, that Zorian and Chou *et al.* do not allow new tests to start until all tests in a session are completed. It means that if t_3 is about to be scheduled it cannot be scheduled as in Figure 182. In the approach proposed by Muresan *et al.* [202], t_3 can be scheduled if it is completed no later than t_2.

In our approach it is optional if tests may start before all tests in a session are completed (nonpartitioned testing) or not (partitioned testing with run to completion). The advantage of using the latter is that it gives more flexibility.

Let a schedule *S* be an ordered set of tests such that:

$$\{S(t_i) < S(t_j) | \tau_{start}(t_i) \leq \tau_{start}(t_j), i \neq j, \forall t_i \in S, \forall t_j \in S\},$$

where $S(t_i)$ defines the position of test t_i in S; $\tau_{start}(t_i)$ denotes the time when test t_i is scheduled to start, and $\tau_{end}(t_i)$ its completion time: $\tau_{end}(t_i) = \tau_{start}(t_i) + \tau_{test}(t_i)$.

For each test, t_i, the start time and the wires for test data transportation have to be determined before it is inserted into the schedule, *S*.

Let the Boolean function *scheduled*(t_i, τ_1, τ_2) be true if test t_i is scheduled in such a way that the test time overlaps with the time interval [τ_1, τ_2], *i.e.*,

$$\{t_i \in S \wedge \neg(\tau_{end}(t_i) < \tau_1 \vee \tau_{start}(t_i) > \tau_2)\}.$$

An example to illustrate the function *scheduled* for a set of scheduled tests is shown in Figure 183.

The Boolean function *scheduled*(r_i, τ_1, τ_2) is true if a source r_i is used by a test t_j between τ_1 and τ_2, *i.e.*:

$$\{\exists t_j \in S | r_i = \text{source}(t_j) \wedge \text{scheduled}(t_j, \tau_1 \tau_2)\}.$$

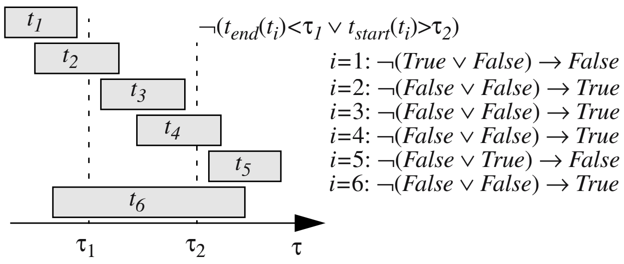

t_1
t_2
t_3
t_4
t_5
t_6
τ_1
τ_2
τ
$\neg(t_{end}(t_i)<\tau_1 \vee t_{start}(t_i)>\tau_2)$
i=1: $\neg(True \vee False) \rightarrow False$
i=2: $\neg(False \vee False) \rightarrow True$
i=3: $\neg(False \vee False) \rightarrow True$
i=4: $\neg(False \vee False) \rightarrow True$
i=5: $\neg(False \vee True) \rightarrow False$
i=6: $\neg(False \vee False) \rightarrow True$

*Figure 183.*The function *scheduled.*

A similar definition is used if a sink r_i is scheduled (used by any test) between τ_1 and τ_2. The Boolean function *scheduled*(*constraint*(t_i), τ_1, τ_2) is true if:

$$\{\exists t_j \in S \,|\, \mathrm{core}(t_j) \in \mathrm{constraint}(t_i) \wedge \mathrm{scheduled}(t_j, \tau_1, \tau_2)\}.$$

4.2 Power Consumption

Generally speaking, there are more switching activities during the testing mode of a system than when it is operated under the normal mode. The power consumption of a CMOS circuit is given by a static part and a dynamic part where the latter dominates and can be characterized by:

$$p = C \times V^2 \times f \times \alpha$$

where the capacitance C, the voltage V, and the clock frequency f are fixed for a given design [277]. The switch activity α, on the other hand, depends on the input to the system, which during testing are test vectors and therefore the power dissipation varies depending on the test vectors.

An example illustrating the test power dissipation variation over time τ for two test t_i and t_j is in Figure 184. Let $p_i(\tau)$ and $p_j(\tau)$ be the instantaneous power dissipation of two compatible tests t_i and t_j, respectively, and $P(t_i)$ and $P(t_j)$ be the corresponding maximal power dissipation.

If $p_i(\tau) + p_j(\tau) < p_{max}$, the two tests can be scheduled at the same time. However, instantaneous power of each test vector is hard to obtain. To simplify the analysis, a fixed value $p_{test}(t_i)$ is usually assigned for all test vectors in a test t_i such that when the test is performed the power dissipation is no more then $p_{test}(t_i)$ at any moment.

The $p_{test}(t_i)$ can be assigned as the average power dissipation over all test vectors in t_i or as the maximum power dissipation over all test vectors in t_i. The former approach could be too optimistic, leading to an undesirable test schedule, which exceeds the test power constraints. The latter could be too

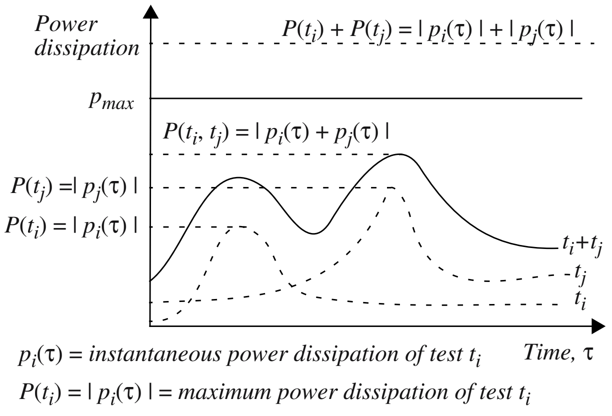


*Figure 184.*Power dissipation as a function of time [41].

pessimistic; however, it guarantees that the power dissipation will satisfy the constraints. Usually, in a test environment the difference between the average and the maximal power dissipation for each test is often small since the objective is to maximize the circuit activity so that it can be tested in the shortest possible time [41]. Therefore, the definition of power dissipation $p_{test}(t_i)$ for a test t_i is usually assigned to the maximal test power dissipation ($P(t_i)$) when test t_i alone is applied to the device. This simplification was introduced by Chou *et al.* [41] and has been used by Zorian [287] and by Muresan *et al.* [202].We will use this assumption also in our approach.

Let $p_{sch}(\tau_1, \tau_2)$ denote the peak power between τ_1 to τ_2:

$$\max\left\{ \sum_{\forall t_i \text{scheduled}(t_i,\, \tau)} p_{test}(t_i) - p_{idle}(core(t_i)) + \sum_{\forall c_i \in C} p_{idle}(c_i),\ \tau \in [\tau_1, \tau_2] \right\},$$

where *scheduled*(t_i, τ)=*scheduled*(t_i, τ, τ).

As an example, applying the function $p_{sch}(\tau_1, \tau_2)$ on the schedule for a system with 5 tests as in Figure 182, with τ_1 and τ_2 as indicated in the figure, returns $p_{test}(t_2) + p_{test}(t_5) + p_{idle}(c_3)$, the peak power consumption between τ_1 and τ_2.

In our approach, the maximal power consumption should not exceed the power constraint, p_{max}, for a schedule to be accepted. That is, $p_{sch}(0, \infty) \leq p_{max}$.

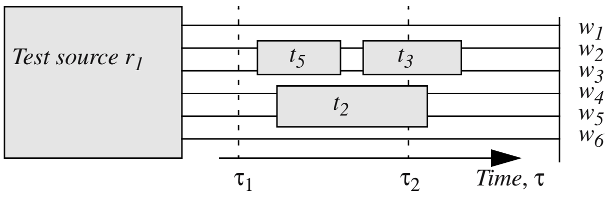


*Figure 185.*The TAM allocation.

4.3 Test Source Limitations

A test source usually has a limited bandwidth. For instance, external tester may only support a limited number of scan chains at a time [92]. There could also be a limitation in the number of available pins. For each test source, this information is given in the attribute bandwidth.

The function $bw_{alloc}(r_i, \tau_1, \tau_2)$ gives the maximal number of wires allocated between τ_1 and τ_2 for a source r_i, *i.e.*:

$$\max\left\{ \sum_{\forall t_j \text{scheduled}(t_j, \tau) \wedge r_i = \text{source}(t_j)} |t_{\text{wires}}(t_j)|, \tau \in [\tau_1, \tau_2] \right\}.$$

For instance, in Figure 185 a test source r_1 feeds test t_5 using wire w_2 and w_3. In this example, $bw(r_1)$=6 and $bw_{alloc}(r_i, \tau_1, \tau_2) = 4$ since wire w_2 and w_3 are used by t_3 and t_5 and w_5 and w_6 are used by t_2.

A test generator (test source) may use a memory for storing the test patterns. In particular, external test generators use such a memory with a limited size, which may lead to additional constraints on test scheduling [92].

The function $mem_{alloc}(r_i, \tau_1, \tau_2)$ gives the allocated memory between time τ_1 and τ_2 for a given source r_i, *i.e.*:

$$\max\left\{ \sum_{\forall t_j \text{scheduled}(t_j, \tau) \wedge r_i = \text{source}(t_j)} \text{mem}(t_j), \tau \in [\tau_1, \tau_2] \right\}.$$

4.4 Test Set Selection

A test set is attached with its test power consumption, test application time, bandwidth requirement and memory requirement (Section 3). The required test source and the test sink are also defined for each test set. In the approach proposed by Sugihara *et al.* each testable unit can be tested by two test sets using a dedicated BIST resource and an external tester [256]. In our approach, we assume that an arbitrary number of test sets can be used for each testable unit where each test set can be specified using any type of test resource. For instance, we allow the same test resource to be used by several test sets at different testable units. The major advantage is that the designer can define and explore a wider range of test sets. Due to that the test resources are defined for each test set it is possible to make a comparison of different test sets not only in terms of the number of test vectors but also in respect to test resources.

For each testable unit, the designer defines a core test set where each set is a set of test vectors. For instance, if the following $CT_{11}=\{t_1, t_2\}$, $CT_{12}=\{t_6\}$, $CT_{13}=\{t_7, t_8, t_9\}$ is given for core c_1. One of CT_{11}, CT_{12} or CT_{13} must be selected and all tests within the selected CT must be scheduled and no other tests from any other CT for core c_1 should be scheduled.

4.5 Test Access Mechanism

A test infrastructure transports and controls the flow of test data in the system under test. The Boundary scan technique can be used for these purposes, however, it suffers from long testing times and due to the amount of test data to be transported in the system, we assume an infrastructure consisting of test wires (lines). It means we add needed number of test wires from test sources to cores and from cores to test sinks. The time to transport a test vector from a test source to a core and the time to transport test response from a core to a test sink is neglected, which means that the test time is determined by the test vectors and design characteristics of each core.

When adding a TAM between a test source and a core or between a core and a test sink, and the test data has to pass another core, c_i, several routing options are possible:

1. through the core c_i using its transparent mode;
2. through an optional bypass structure of core c_i; and
3. around core c_i and not connecting it to the TAM.

The model in Figure 186(a) of the example system in Figure 181 illustrates the advantage of alternatives 1 and 2 (Figure 186 (b)) compared to alternative 3 (Figure 186 (c)) since the TAM can be reused in the former two cases. However, a delay may be introduced when the core is in transparent mode or its by-pass structure is used as in the TestShell [186]. On the other hand, Marinissen *et al.* recently proposed a library of wrapper cells allowing a flexible design where it is possible to design non-clocked bypass structures of TAM width [187]. In the following, we assume that bypass may be solved using such non-delay mechanism.

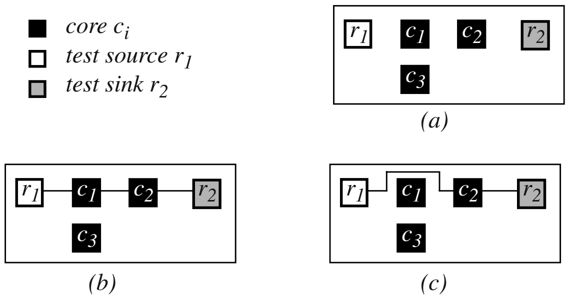


*Figure 186.*Illustrating TAM design alternatives using the example in Figure 181.

When designing the TAM, two problems must be solved:

- the design and routing of the TAMs, and
- the scheduling of tests using the TAMs.

In order to minimize the routing, few and short wires are desired. However, such approach increases the test time of the system. For instance, consider System S [25] (Table 17) where we added the floor planning (x, y co-ordinates for each core). A minimal TAM would be a single wire starting at the TAP, connecting all cores and ending at the TAP. However, such TAM

Core	*Index i*	*External test cycles, e_i*	*BIST cycles, b_i*	*Placement*	
				x	*y*
c880	1	377	4096	10	10
c2670	2	15958	64000	20	10
c7552	3	8448	64000	10	30
s953	4	28959	217140	20	30
s5378	5	60698	389214	30	30
s1196	6	778	135200	30	10

Table 17. Test data for the cores in System S.

leads to high test application time since no tests can be applied concurrently, all tests have to be applied in a sequence.

The system (for instance the example system in Figure 181 or System S) can be modelled as a directed graph, $G=(V,A)$, where V consists of the set of cores (the testable units), C, the set of test sources, R_{source}, and the set of test sinks, R_{sink}, *i.e.* $V=C \cup R_{source} \cup R_{sink}$ [158,157]. An arc $a_i \in A$ between two vertices v_i and v_j indicates a TAM (a wire) where it is possible to transport test data from v_i to v_j. Initially no TAM exists in the system, *i.e.* $A=\emptyset$. However, if the functional infrastructure may be used, it can be included in A initially. A test wire w_i is a path of edges $\{(v_0,v_1),..,(v_{n-1},v_n)\}$ where $v_0 \in R_{source}$ and $v_n \in R_{sink}$. Let Δy_{ij} be defined as $|y(v_i) - y(v_j)|$ and Δx_{ij} as $|x(v_i) - x(v_j)|$, where $x(v_i)$ and $y(v_i)$ are the *x-placement* respectively the *y-placement* for a vertex v_i and the distance between vertex v_i and v_j is:

$$\mathrm{dist}(v_i, v_j) = \sqrt{(\Delta y_{ij})^2 + (\Delta x_{ij})^2}.$$

The Boolean function *scheduled*(w_i, τ_1, τ_2) is true when a wire w_i is used between τ_1 to τ_2:

$$\{\exists t_j \in S | w_i \in \mathrm{tam}(t_j) \wedge \mathrm{scheduled}(t_j, \tau_1, \tau_2)\},$$

where *tam*(t_j) is the set of wires allocated for test t_j.

The information of the nearest core in four directions, *north, east, south* and *west*, are stored for each vertex. The function *south*(v_i) gives the closest vertex south of v:

$$\mathrm{south}(v_i) = \left\{ v_i \middle| \left(\frac{\Delta y_{ij}}{\Delta x_{ij}} > 1 \vee \frac{\Delta y_{ij}}{\Delta x_{ij}} < -1 \right), y(v_j) < y(v_i), i \neq j, \min\{\mathrm{dist}(v_i, v_j)\} \right\}.$$

The functions *north*(v_i), *east*(v_i) and *west*(v_i) are defined in similar ways. The function *insert*(v_i, v_j) inserts a directed arc from vertex v_i to vertex v_j *if and only if* the following is true:

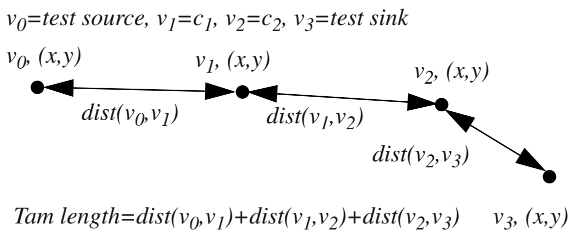
$$\{\mathrm{south}(v_i, v_j) \vee \mathrm{north}(v_i, v_j) \vee \mathrm{west}(v_i, v_j) \vee \mathrm{east}(v_i, v_j)\}$$

where *south*(v_i,v_j) is true if v_j=*south*(v_i). The function *closest*(v_i, v_j) gives a vertex, v_k, in the neighbourhood of v_i with the shortest distance to v_j. The function *add*(v_i, v_j) adds arcs from v_i to v_j in the following way: (1) find v_k=*closest*(v_i, v_j); (2) add an arc from v_i to v_k; (3) if $v_k = v_j$, terminate otherwise let v_i=v_k and go to (1).

The total length of a path is the sum of the length of all individual edges. An example to illustrate the calculation of the length of a test wire on the example system (Figure 181) defined as a path is in Figure 187 (in this path core c_3 is not included).

4.6 Test Floor-planning

In the approach proposed by Sugihara *et al.* each testable unit is tested by a test sets using a dedicated BIST resource and an external tester [256]. However, we allow the sharing of test resources and if a BIST resource is shared among several cores, the floor-planning is not trivial.



v_0=test source, $v_1=c_1$, $v_2=c_2$, v_3=test sink
v_0, (x,y)
v_1, (x,y)
v_2, (x,y)
v_3, (x,y)
dist(v_0,v_1)
dist(v_1,v_2)
dist(v_2,v_3)
Tam length=dist(v_0,v_1)+dist(v_1,v_2)+dist(v_2,v_3)

*Figure 187.*Computing the TAM length.

For instance, if two cores in the example design (Figure 181) both use a single on-chip test source and test sink, it is most feasible to place the test source at one core while the test sink is placed at the other core.

Initially, the test resources may not be placed. Our algorithm, described in the next section, determines their placement in this case.

5 THE HEURISTIC ALGORITHM

In this section the issues discussed above are combined into an algorithm. The algorithm assumes that the tests are initially sorted according to a key *k*, which characterizes *power*(*p*), *test time*(*t*) or *power*×*test time*(*p*×*t*).

Let *T* be the ordered set of the tests, which are ordered based on the key *k*. If the scheduling approach is partitioned testing with run to completion the function *nexttime*(τ_{old}) gives the next time where it is possible to schedule a test:

$$\{\tau_{end}(t_i) | \min(\tau_{end}(t_i)), \tau_{old} < \tau_{end}(t_i), \forall t_i \in S\},$$

otherwise if nonpartitioned testing is used the function *nexttime*(τ_{old}) is defined as:

$$\{\tau_{end}(t_i) | \max(\tau_{end}(t_i)), \tau_{old} < \tau_{end}(t_i), \forall t_i \in S\}.$$

The algorithm depicted in Figure 188 can basically be divided into four parts for:

- constraint checking,
- test resource placement,
- test access mechanism design and routing, and
- test scheduling.

```
Sort T according to the key $(p, t \text{ or } p \times t)$;
$S=\emptyset$; $\tau=0$;
until $\forall b_{pq} \exists CT_{pq} \forall t_s \in S$ do
  for all cur in T do
    $v_a=source(cur)$;
    $v_b=core(cur)$;
    $v_c=sink(cur)$;
    $\tau_{end}=\tau+t_{test}(cur)$;
    if all constraints are satisfied then begin
      $\neg scheduled(v_a, 0, t_{end})$ floor-plan $v_a$ at $v_b$;
      $\neg scheduled(v_c, 0, t_{end})$ floor-plan $v_c$ at $v_b$;
      for all required test resources begin
        new=length of a new wire $w_j$ connecting $v_a$, $v_b$ and $v_c$;
        u=number of wires connecting $v_a$, $v_b$ and $v_c$ not
          scheduled from $\tau$ to $\tau_{end}$;
        v=number of wires connecting $v_a$, $v_b$ and $v_c$;
        for all min(v-u, bw(cur))wires $w_j$
          extend=extend+length of an available wire($w_j$);
        if (bw(cur)>u)
          extend=extend+new$\times$(par-u);
          move=par($v_a$) $\times$ min{dist($v_a$, $v_b$),dist($v_b$, $v_c$)};
          if (move$\leq$min{extend, new $\times$ bw(cur)})
            $v_x$, $v_y$=min{dist($v_a$, $v_b$), dist($v_b$, $v_c$)}, dist($v_a$, $v_b$)>0,
                dist($v_b$,$v_c$)>0
            add par($v_a$) wires between $v_x$ and $v_y$;
            if ($v_x$=source(cur)) then floorplan $v_a$ at $v_b$;
            if ($v_y$ = sink(cur)) then floorplan $v_c$ at $v_b$;
      end
      for r = 1 to bw(cur)
        if there exists a not scheduled wire during $\tau$ to $\tau_{end}$
          connecting $v_a$, $v_b$ and $v_c$ it is selected
        else
          if (length of a new wire < length of extending a wire $w_j$)
            $w_j$=add($v_a$, $v_b$)+add($v_b$, $v_c$);
          else
            extend wire;
      schedule cur and remove cur from T;
    end;
  $\tau$ = nexttime($\tau$).
```

*Figure 188.*The system test algorithm.

A main loop is terminated when there exists a core test (CT) for each core such that all tests within the selected CT are scheduled. In each iteration at the loop over the tests in *T* a test *cur* is checked. A check is also made to determine if all constraints are fulfilled, *i.e.* it is possible to schedule test *cur* with a start at τ and an end time at $\tau_{end}=\tau+\tau_{test}$:

- $\neg\exists t_f\,(t_f\in CT_{ij}\wedge t_f\in S\wedge cur\notin CT_{ij})$ checks that another core test set for current core is not used in the schedule,
- $p_{sch}(\tau, \tau_{end})+p_{test}(cur)<p_{max}$ checks that the power constraint is not violated,
- $\neg scheduled(v_a, \tau, \tau_{end})$ checks that the test source is not scheduled during τ to τ_{end},
- $\neg scheduled(v_c, \tau, \tau_{\mathrm{end}})$ checks that the test sink is not scheduled during τ to τ_{end},
- $\neg scheduled(constraint(cur), \tau, \tau_{end})$ checks that all cores required for *cur* are available during τ to τ_{end},
- the available bandwidth at test source v_a is checked to see if: $bw(v_a) > bw(cur) + bw_{alloc}(v_a, \tau, \tau_{end})$ and
- the available memory test source v_a is checked to see if: $mem(v_a)>mem(cur)+mem_{alloc}(v_a, \tau, t_{end})$.

Then the placement of the test resources is checked. If the test resources are on-chip resources and not placed in the system, they are placed at core c_i. If they are floor-planned, a check is performed to determine if they are to be moved.

When the placement of the test resources for the selected test is determined, the test access mechanism is designed and routed. The basic question is if existing wires can be used or new wires must be added. If no routed connection is available connecting all required cores, the distance for adding a completely new connection is re-calculated due to a possible moving of test resources.

Examples of the produced results from the algorithm using System S [25] (Table 17) are the TAM design as in Figure 189 and the test schedule as in Figure 190. The TAM wires 1 to 5 in Figure 189 correspond to the TAM 1 to 5 in Figure 190. For instance, b_5 is the BIST test of core indexed 5 (s5378) and e_5 is the external test of s5378 (note that the BIST tests b_i do not require a TAM but they are placed in Figure 190 to illustrate when they are scheduled).

The computational complexity for the above algorithm, where the test access mechanism design is excluded in order to make it comparable with other approaches, comes mainly from sorting the tests and the two loops. The sorting can be performed using a sorting algorithm at $O(n\times log\ n)$. The worst

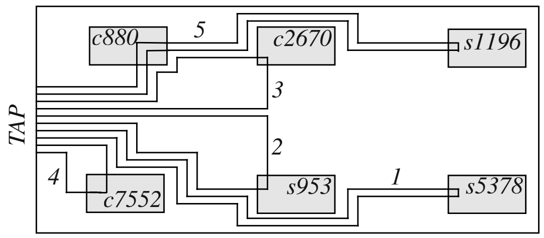


*Figure 189.*TAM design using our heuristic on System S.

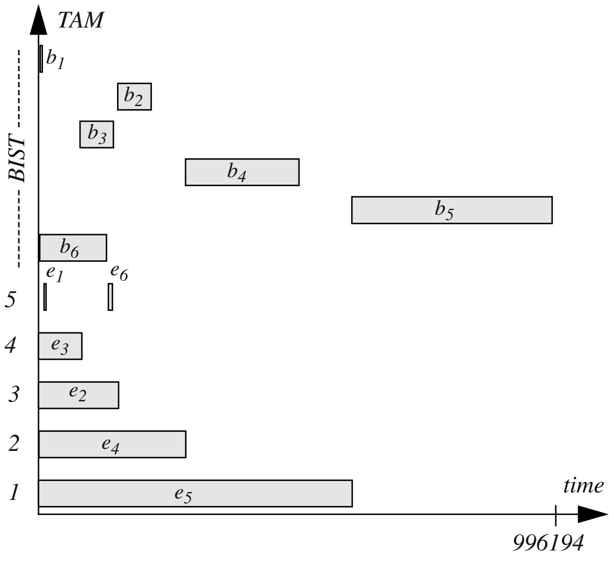


*Figure 190.*TAM schedule on System S using our heuristic.

case occurs when in each loop iteration for the loops only one test is scheduled giving a complexity:

$$\sum_{i=0}^{|T|-1} (|T| - i) = \frac{n^2}{2} + \frac{n}{2}$$

The total worst case execution time is $n \times \log n + n^2/2 + n/2$, which is of $O(n^2)$. For instance, the approach by Garg *et al.* [65] and by Chakrabarty [25] both has a worst case complexity of $O(n^3)$.

```
1:  Construct initial solution, $x^{now}$;
2:  Initial Temperature: $T=TI$;
3:  while stop criteria not met do begin
4:      for $i = 1$ to $TL$ do begin
5:          Generate randomly a neighboring solution
                $x' \in N(x^{now})$;
6:          Compute change of cost function
                $\Delta C=C(x')-C(x^{now})$;
7:          if $\Delta C \leq 0$ then $x^{now}=x'$
8:              else begin
9:                  Generate $q = random(0, 1)$;
10:                 if $q<e^{-\Delta C/T}$ then $x^{now}=x'$
11:             end;
12:     end;
13:     Set new temperature $T=\alpha \times T$;
14: end;
15: Return solution corresponding to the minimum
        cost function;
```

*Figure 191.*Simulated Annealing algorithm.

6 SIMULATED ANNEALING

In this section we outline the Simulated Annealing (SA) technique and describe its adoption to be used for scheduling and TAM design. The technique proposed by Kirkpatrick *et al.* [131] uses a hill-climbing mechanism to avoid getting stuck in at local optimum.

6.1 The Simulated Annealing Algorithm

The SA algorithm (Figure 191) starts with an initial solution and a minor modification creates a neighbouring solution. The cost of the new solution is evaluated and if it is better than the previous, the new solution is kept. A worse solution can be accepted at a certain probability, which is controlled by a parameter referred to as temperature.

The temperature is decreased during the optimization process, and the probability of accepting a worse solution decreases with the reduction of the temperature value. The optimization terminates when the temperature value is approximately zero.

6.2 Initial Solution and Parameter Selection

We use our heuristic described in Section 5 with an initial sorting of the tests based on *power* (using *time* and *power*×*time* results after optimization in the same cost) within the integrated test framework (Section 5) to create the initial solution [158,157]. An example of an initial solution produced for System S is in Figure 189 and Figure 190.

The parameters, initial temperature *TI*, the temperature length *TL* and the temperature reduction factor α ($0<\alpha<1$) are all determined based on experiments.

6.3 Neighbouring Solution in Test Scheduling

In the case when only test scheduling is considered, *i.e.* the TAM is not considered or it is fixed and seen as a resource, we create a neighbouring solution by randomly selecting a test from an existing schedule and schedule it as soon as possible but not at the same place as it was in the original schedule. For instance, creating a neighbouring solution given a test schedule as in Figure 182 with resource graph as in Figure 177 and test compatibility graph as in Figure 178, we randomly select a test, let say t_1. We try to schedule t_1 as soon as possible but not with the same starting time as it had while fulfilling all constraints. Test t_1 was scheduled to start at time 0 and no new starting point exists where constraints are fulfilled until end of t_3 where t_1 now is scheduled. In this case, the test time increases after the modification (getting out of a possible local minimum), however, only temporarily since in the next iteration a test may be scheduled at time 0 (where t_1 used to be).

6.4 Neighbouring Solution in Scheduling and TAM Design

When both the test time and the TAM design are to be minimized, a neighbouring solution is created by randomly adding or deleting a wire and then the tests are scheduled on the modified TAM.

If the random choice is to add a wire, a test is randomly selected and a wire is added from its required test source to the core where the test is applied and from the core to the test sink for the test. For instance, if e_3 in System S (Table 17) is selected, a wire is added from the TAP to core c7552 and from core c7552 to the TAP.

If the random choice is to delete a wire, a similar approach is applied. However, a check is performed to make sure that all tests can be applied.

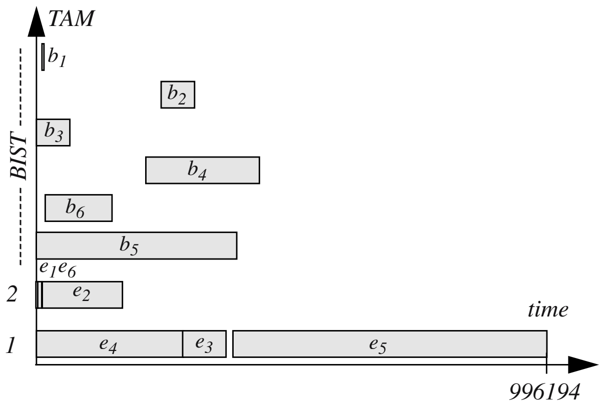


Figure 192. Test schedule on System S using SA.

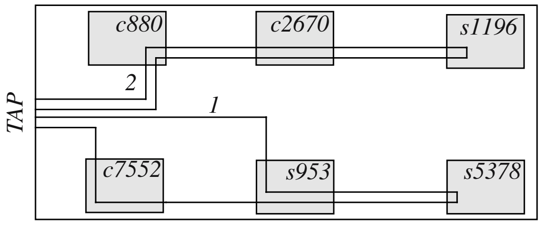


Figure 193. TAM design using SA on System S.

6.5 Cost function

The cost function of a test schedule, *S*, and the TAM, *A*, is:

$$C(S, A) = \beta_1 \times T(S) + \beta_2 \times L(A)$$

where: *T*(*S*) is the test application time for a sequence of tests, *S*, *L*(*A*) is the total length of the TAM, β_1, β_2 are two designer-specified constants used to determine the importance of the test time and the TAM.

The test application time, *T*(*S*), for a schedule, *S*, is:

$$T(S) = \{\forall t_i(\max\{t_{end}(t_i)\}), t_i \in S\}$$

and the length, *L*(*A*), of the TAM, *A,* is given by:

$$\sum_{w_j \in A} \sum_{j=0}^{|w_i|-1} \mathrm{dist}(v_j, v_{j+1}), v_j, v_{j+1} \in w_i$$

For the test schedule, S, produced by SA for System S (Figure 192) the test time, T(S) is 996194 (the end time of test e_5) and the length of the TAM (Figure 193) is 160. Comparing this to the results produced by our heuristic [158] shows that test time is the same while the TAM is reduced from 320 (Figure 189) to 160 (Figure 193).

7 EXPERIMENTAL RESULTS

7.1 Benchmarks

We have used the System S [25] (Appendix 1, page 349), ASIC Z design (Appendix 1, page 329). We have also used an extended ASIC Z design where each core is tested by two test sets (one external test and one BIST) and an interconnection test to a neighbouring core [157]; in total 27 tests.

We have used a design with 10 tests presented by Muresan *et al.* [202] and an industrial design. The largest design is the Ericsson design [160,157] consisting of 8 DSP cores plus additional logic cores and memory banks. All data for the benchmarks can be found in Appendix 1.

For the implementation, we have simplified our system model (Section 3) regarding the TAM wire design and only allowing a single wire per test. When discussing about our algorithm we use *our*1, *our*2 and *our*3, which corresponds to the initial sorting based on *test power*(*p*), *test time*(*t*) and *test power*×*test time*(*p*×*t*).

Unless stated, we use partitioned testing with run to completion, for the cost function (Section 6.5) $\beta_1=\beta_2=1$ and we have used a Sun Ultra10, 450 MHz CPU, 256 MB RAM.

7.2 Test Scheduling

We have compared our algorithm (nonpartitioned testing) with the nonpartitioned testing approaches proposed by Zorian [287] and Chou *et al*. [41]. We have used the same assumptions as Chou *et al*. and the results are in Table 18. Our approaches (our1, our2, and our3) results, in all cases, in a test schedule with three test sessions (*ts*) at a test time of 300 time units, which is 23% better than Zorian's approach and 9% better than the approach by Chou *et al.*

ts	*Zorian*		*Chou et al.*		*Our1, Our2, Our3*	
	Cores	*Time*	*Cores*	*Time*	*Cores*	*Time*
1	RAM1,RAM4, RF	69	RAM1,RAM3, RAM4, RF	69	RL2, RL1,RAM2	160
2	RL1, RL2	160	RL1, RL2	160	RAM1,ROM1, ROM2	102
3	RAM2,RAM3	61	ROM1, ROM2, RAM2	102	RAM3, RAM4, RF	38
4	ROM1, ROM2	102				
	Test time:	392		331		300

Table 18. ASIC Z test scheduling.

In System S, no power constraints are given and therefore only our2 can be used. Our approach finds using partitioned testing with run to completion after 1 second the optimal solution; see Table 11.1 (first group).

The results on the industrial design are in Table 11.1 (second group) where the industrial designer's solution is 1592 time units while our test scheduling achieve a test time of 1077 time units, the optimum, in all cases (our1, our2, and our3), which is 32.3% better than the designer's solution.

The results on design Muresan are in the third group of Table 11.1. The test time using the approach by Muresan *et al.* is 29 time units and the results using our approaches our1, our2, and our3 are 28, 28 and 26, respectively, all produced within 1 sec. Our SA (TI=400, TL=400, α=0.97) improves to 25 time units using 90 seconds.

When not considering idle power on ASIC Z, the test schedules using our1, our2, and our3 (fourth group in Table 11.1) all result in a test time of 262. The SA (TI=400, TL=400 and α=0.97) did not find a better solution.

In the experiments considering idle power (fifth group of Table 11.1), our heuristic approaches our1, our2, and our3 resulted in a solution of 300, 290 and 290, respectively, each produced within 1 second. The SA (TI=400, TL=400 and α=0.99) produced a solution of 274 requiring 223 sec., *i.e.* a cost improvement in the range of 6% to 10%.

The results on Extended ASIC Z when not considering idle power are 313 (our1), 287 (our2) and 287 (our3) (sixth group in Table 11.1), which were produced by our heuristic after 1 second. The SA optimization (TI=TL=400, α=0.97) produced a solution at a cost of 264 running for 132 seconds, *i.e.* a cost improvement in the range of 9% to 18%.

The results on the Ericsson design (seventh group of Table 11.1) are 37226, 34762, 34762 produced by our heuristic our1, our2, and our3 (within 3 sec.). The SA algorithm (TI=200, TL=200, α=0.95) produced a solution at 30899 after 3260 seconds.

7.3 Test Resource Placement

In the ASIC Z design all cores have their own dedicated BIST structure. Let us assume that all ROM cores share one BIST structure and all RAM memories share another BIST structure; the rest of the cores have their own dedicated BIST structure.

7.4 Test Access Mechanism Design

Assume for ASIC Z that all tests are scan-based (1 scan-chain per core) applied with an external tester allowing a maximum of 8 scan chains to operate concurrently. The results using our1, our2, and our3 considering idle

Design	*Approach*	*Test time*	*Diff. to SA/optimum*	*CPU*
System S [25]	Optimal solution	1152180	-	-
	Chakrabarty	1204630	4.5%	-
	Our2 (t)	1152180	0%	1 sec.
Industrial design	Optimal solution	1077	-	-
	Designer	1592	47.8%	-
	Our1 (p)	1077	0%	1 sec.
	Our2 (t)	1077	0%	1 sec.
	Our3 (p×t)	1077	0%	1 sec.
Muresan [202]	SA	25	-	90 sec.
	Muresan [202]	29	16%	-
	Our1 (p)[158]	28	12%	1 sec.
	Our2 (t)[158]	28	12%	1 sec.
	Our3 (p×t)[158]	26	4%	1 sec.
ASIC Z (1)	SA	262	-	74 sec.
	Our1 (p)	262	0%	1 sec.
	Our2 (t)	262	0%	1 sec.
	Our3 (p×t)	262	0%	1 sec.
ASIC Z (2)	SA	274	-	223 sec.
	Our1 (p)[158]	300	10%	1 sec.
	Our2 (t)[158]	290	6%	1 sec.
	Our3 (p×t)[158]	290	6%	1 sec.
Extended ASIC Z (3)	SA	264	-	132 sec.
	Our1 (p)	313	18%	1 sec.
	Our2 (t)	287	9%	1 sec.
	Our3 (p×t)	287	9%	1 sec.
Ericsson	SA	30899	-	3260 sec.
	Our1 (p)	37336	20%	3 sec.
	Our2 (t)	34762	12%	3 sec.
	Our3 (p×t)	34762	12%	3 sec.

Table 11.1. Test scheduling results.

power are collected in Table 19. The test schedule and the TAM schedule achieved with heuristic our3 are in Figure 194. The total length of the test

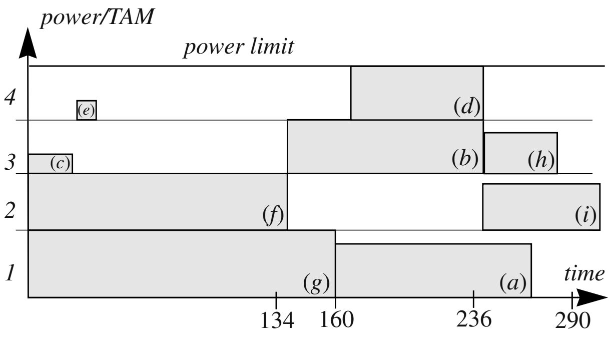


*Figure 194.*Test schedule for ASIC Z.

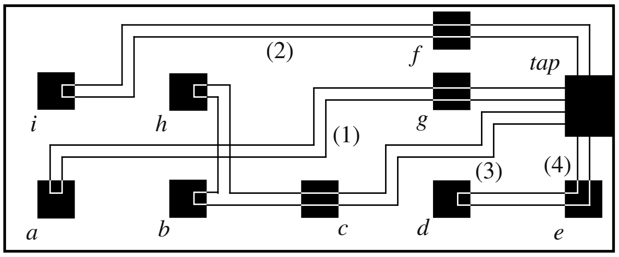


*Figure 195.*ASIC Z with test data access mechanism.

access mechanism is 360 units and it is routed as in Figure 195. All solutions were produced within 1 second.

7.5 Test Scheduling and TAM Design

We have made experiments using System S and ASIC Z to demonstrate the importance of integrating test scheduling and TAM design. The ASIC Z is fully BISTed; however, in this experiment we assume all tests are applied using an external tester able of supporting several tests concurrently. We have used two naive approaches, Naive1 and Naive2. Naive1 uses a minimal TAM design connecting all cores in a ring and Naive 2 uses an extensive TAM where each core gets its own dedicated TAM.

For the cost function, we have for ASIC Z used β_1=β_2=1 and for System S β_1=1/1000 and β_2=1 and the results are collected in Table 20. Naive 1 produces a low cost TAM design but the test time is long leading to high total cost due to that all tests have to be scheduled in a sequence. The total cost of using the approach Naive 2 is also high due to the extensive TAM design. Such extensive TAM gives more flexibility, however, other constraints in the design limits its use. Our approaches (our1, our2, and our3) produces better results indicating the importance of considering TAM design and test sched-

Approach	*Test time*	*TAM cost*
Our1 (p)	300	360
Our2 (t)	290	360
Our3 (p×t)	290	360

Table 19. Results on ASIC Z.

Approach	*System S*			*ASIC Z*		
	Time	*TAM*	*Total*	*Time*	*TAM*	*Total*
Naive 1	1541394	100	1641	699	110	809
Naive 2	996194	360	1356	300	500	800
Our 1 (p)	-	-	-	300	360	660
Our 2 (t)	996194	320	1316	290	360	650
Our 3 (p×t)	-	-	-	290	360	650
SA	996194	160	1156	334	180	514

Table 20. TAM design and test scheduling.

uling in an integrated manner. The TAM design (Figure 189) and the test schedule (Figure 190) achieved using our heuristic for System S (Table 17) have a test time of 996194 and a TAM length of 320 computed within 1 second. The SA (TI=TL=100, α=0.99) was running for 1004 seconds producing a test schedule (Figure 192) and a TAM design (Figure 193) with a test time of 996194 and a TAM length of 160, a TAM improvement of 50%. For ASIC Z, the SA (TI=TL=300, α=0.97) produced after 855 seconds a solution with a cost of 514 (334 for test time and 180 for TAM).

The results comparing our heuristic with SA are collected in Table 21 where, for instance, our heuristic our1 produced a solution for ASIC Z with a total cost of 660 (a test time of 290 and a TAM cost of 360) after 1 second. The test time results are in the range of 10% to 13% better using our fast heuristic (in all cases) compared to the SA optimization. However, the TAM results are much worse using our heuristics and the total cost improvements by the SA are in the range from 21% to 28%.

The results from experiments on Extended ASIC Z are in Table 21 (second group). Our heuristic our1 produced a solution after 1 second with a test time of 313 and a TAM cost of 720, resulting in a total cost of 1033. The solution produced after 4549 seconds by our SA (TI=TL=200, α=0.97) optimization has a test time of 270 and a TAM cost of 560. In this experiment, SA produced a better total cost (range 14% to 24%) as well as better cost

	Approach	*SA*	*Our1*	*Our2*	*Our 3*
ASIC Z	Test time	334	300	290	290
	Diff to SA	-	-10%	-13%	-13%
	TAM cost	180	360	360	360
	Diff to SA	-	100%	100%	100%
	Total Cost	514	660	650	650
	Diff to SA	-	28%	21%	21%
	Comp. cost	855 sec.	1 sec.	1 sec.	1 sec.
	Diff to SA	-	-85400%	-85400%	-85400%
Extended ASIC Z	Approach	SA	Our 1	Our 2	Our 3
	Test time	270	313	287	287
	Diff to SA	-	16%	6%	6%
	TAM cost	560	720	660	660
	Diff to SA	-	29%	18%	18%
	Total Cost	830	1033	947	947
	Diff to SA	-	24%	14%	14%
	Comp. cost	4549 sec.	1 sec.	1 sec.	1 sec.
	Diff to SA		-454800%	-454800%	-454800%
	Approach	*SA*	*Our1*	*Our2*	*Our3*
	Test time	33082	37336	34762	34762
	Diff to SA	-	11%	5%	5%
	TAM	6910	8245	9350	8520
Ericsson	Diff to SA	-	19%	35%	23%
	Total Cost	46902	53826	53462	51802
	Diff to SA	-	15%	14%	10%
	Comp. cost	15h	81 sec.	79 sec.	62 sec.
	Diff to SA		-66567%	-68254%	-86996%

Table 21. TAM and scheduling results.

regarding test time (range 6% to 16%) and TAM cost (range 18% to 29%). The results on the Ericsson design are collected in Table 21 (third group). For instance, our heuristic our1 results in a solution with a test time of 37336 and a TAM cost of 8245, which took 81 seconds to produce. The total cost is

53826 when using β_1=1 and β_2=2. The SA (TI=TL=200, α=0.95) optimization produced a solution with a test time of 33082 and a TAM cost of 6910 after 15 hours. In all cases, the SA produces better results. Regarding test time the SA improvement is in the range from 5% to 11%, for the TAM cost in the range from 19% to 35% and the total cost in the range from 10% to 15%.

For all experiments with the SA, the computational cost is extremely higher compared to our heuristics. A finer tuning of the SA parameters could reduce it, however, the extensive optimization is only used for the final design and therefore a high computational cost can be accepted.

8 CONCLUSIONS

For complex systems such as SOCs, it is a difficult problem for the test designer to develop an efficient test solution due to the large number of factors involved. The work-flow consists of two consecutive parts: an early design space exploration and an extensive optimization for the final solution. In this chapter we have proposed a framework suitable for these two parts where test scheduling, test access mechanism design, test set selection and test resource placement are considered in an integrated manner minimizing the test time and the size of the test access mechanism while satisfying test conflicts and test power constraint. For the early design space exploration, we have implemented an algorithm running at a low computational cost, which is suitable to be used iteratively many times. For the final solution, a more extensive optimization can be justified and we have proposed and implemented a technique based on Simulated Annealing for test scheduling and test access mechanism design. We have performed several experiments on benchmarks and industrial designs to show the efficiency and usefulness of our approach.

Chapter 12

EFFICIENT TEST SOLUTIONS FOR CORE-BASED DESIGNS[1]

1 INTRODUCTION[2]

A test solution for a complex system requires the design of a test access mechanism (TAM), which is used for the test data transportation, and a test schedule of the test data transportation on the designed TAM. An extensive TAM will lead to lower test application time at the expense of higher routing costs, compared to a simple TAM with low routing cost but long testing time. It is also possible to reduce the testing time of a testable unit by loading the test vectors in parallel, thus increasing the parallelization of a test. However, such a test time reduction often leads to higher power consumption, which must be kept under control since exceeding the power budget could damage the system under test. Furthermore, the execution of a test requires resources and concurrent execution of tests may not be possible due to resource or other conflicts. In this chapter, we propose an integrated technique for test scheduling, test parallelization and TAM design, where the test application time and the TAM routing are minimized while considering test conflicts and power constraints. The main features of our technique are the efficiency in terms of computation time and the flexibility to model the system's test behaviour, as well as the support for the testing of interconnections, unwrapped cores and user-defined logic. We have implemented our approach and made several experiments on benchmarks as well as industrial designs in order to demonstrate that our approach produces high quality solution at low computational cost.

1.1 Introduction

The advance in design methodologies and semiconductor process technologies has led to the development of systems with excessive functionality implemented on a single die, called system-on-chip (SOC). In a core-based design approach, a set of cores, *i.e.* pre-defined and pre-verified design modules, is integrated into a system using user-defined logic (UDL) and interconnections. In this way, complex systems can be efficiently developed;

[1]. The chapter is based on the following papers Asian Test Symposium (ATS), 2002 [166], and Transactions on Coputer-Aided Design for Integrated Circuits, 2004 [176].
[2]. The research is partially supported by the Swedish National Program STRINGENT.

however, the complexity in the systems leads to high test data volumes and the development of a test solution must therefore consider the following inter-dependent problems:

- how to design an infrastructure for the transportation of test data in the system, a test access mechanism (TAM), and
- how to design a test schedule to minimize test time, considering test conflicts and power constraints.

The testable units in an SOC design are the cores, the UDL and the interconnections. The cores are usually delivered with pre-defined test methods and test sets; while the test sets for UDL and interconnections are to be generated prior to test scheduling and TAM design. The test vectors, forming the test sets for each testable unit, are stored or created in some test source, and their test responses are stored or analyzed in some test sink. The TAM is the connection between the test sources, the testable units and the test sinks. The test application time can be minimized by applying several test sets concurrently; however, test conflicts, limitations and test power consumption must be considered.

The work-flow when developing an SOC test solution can mainly be divided into two consecutive parts; an early design space exploration followed by an extensive optimization for the final solution. For the former, we have proposed a technique for integrated test scheduling and TAM design to minimize test time and TAM cost [158, 165]. The advantage of the technique is its low computational cost making it useful for iteratively use in the early design space exploration phase. For extensive optimization of the final solution, we have proposed a technique based on Simulated Annealing, which is used only a few times, justifying its high computational cost [160,165]. We have also proposed an integrated test scheduling and scan chain partitioning (test parallelization) technique under power constraints [161]. The test parallelization problem is, for a testable unit with variable test time such as scan-tested cores, to determine the number of scan-chains to be loaded concurrently, *i.e.* to determine the test time for each testable unit in such a way that the system's total test time is minimized while considering test power limitations.

In this chapter, we propose a technique to integrate test scheduling, test parallelization (scan-chain partitioning) and TAM design with the objective to minimize the test application time and the TAM routing cost while considering test conflicts and power constraints. The aim with our approach is to reduce the gap between the design space exploration and the extensive optimization, *i.e.* to produce a high quality solution in respect to test time and TAM cost at a relatively low computational cost.

The features of our proposed approach are that we support:

- the testing of interconnections,
- the testing of user-defined logic,
- the testing of unwrapped cores,
- the consideration of memory limitations at test sources,
- the consideration of bandwidth limitations on test sources and test sinks, and
- embedding cores in core.

We have implemented our technique and performed experiments on several benchmarks including a large industrial design called Ericsson, which is tested by 170 test sets. The experimental results demonstrate that we can deal with systems tested with different test methods; our approach is not limited to scan-based systems.

The organization of the chapter is as follows. An introduction to the background and an overview of related work are given in Section 2. The considered test problems are discussed and described in Section 3. The system model is defined in Section 3.7, and our integrated test scheduling, test parallelization and TAM design technique is presented in Section 4. The paper is concluded with experimental results in Section 5 and conclusions in Section 6.

2 BACKGROUND AND RELATED WORK

The test application time when testing a system can be minimized by scheduling the execution of the test sets as concurrently as possible. The basic idea in test scheduling is to determine when each test set should be executed, and the main objective is to minimize the test application time. However, various conflicts and limitations must be considered. For instance, only one test set can be applied at any time to each testable unit. Power constraints must also be carefully considered otherwise the system under test can be damaged.

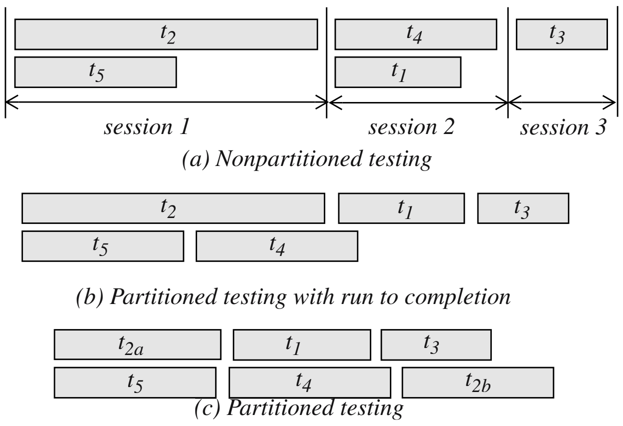


Figure 196. Scheduling approaches.

The scheduling techniques can be classified using a scheme by Craig *et al.* [47] into:

- nonpartitioned testing,
- partitioned testing with run to completion, and
- partitioned testing.

The differences among the techniques are illustrated with five test sets (t_1,..., t_5) in Figure 196, where the length of the rectangles corresponds to the test time of respective test sets. In *nonpartitioned testing*, Figure 196(a), test sets are grouped into sessions and new tests are allowed to start only when all test sets in the preceding session are completely executed. A test scheduling approach based on *partitioned testing with run to completion* does not group tests into sessions, and new tests are therefore allowed to start at any time (Figure 196(b)). And finally in *partitioned testing* or *preemptive testing* a test can be interrupted and resumed at a later point, as test t_2 in Figure 196(c), which is split into two partitions.

A set of test vectors is called a test set, and a system is usually tested by applying a number of test sets. For every test set, one test source and one test sink are required. The test source is where the test sets are stored or produced. A test source can be placed either on-chip or off-chip. The test sink is where the test response, produced by the testable unit when a test vector is applied, is stored or analyzed. Test sinks can, as test sources, be placed either on-chip or off-chip. If both the test source and the test sink for a particular testable unit are placed on-chip, it is common to refer to it as Built-In Self-Test (BIST). An example of an on-chip test source is a Linear-Feedback Shift-Register (LFSR) or a memory; and an example of an off-chip test source is an Automatic Test Equipment (ATE). The main advantage of using an ATE as test source and test sink is that a relatively small test set can be used for each testable unit. However, among the disadvantages are the slow speed of an ATE and its limited memory capacity [92]. An on-chip test source such as an LFSR, on the other hand, does not require an extensive global test infrastructure, which is especially true if each testable unit has its dedicated LFSR. The disdvantage with an LFSR is that usually a relatively large test set is required, which leads to long testing times and also more activity (power consumption) in the system.

The test sources and the sinks can be shared among several testable units. And every testable unit is tested by one or more test sets. A system may contain several test sources and test sinks. A test infrastructure, TAM, is used to connect the test sources, the testable units and the test sinks. The TAM is used for the transportation of test vectors from a test source to a testable unit and test responses from a testable unit to a test sink.

Zorian has proposed a test scheduling technique based on nonpartition testing (see Figure 196(a)) for systems where each testable unit has its dedicated on-chip test source and on-chip test sink [287]. In the approach, a test set is assigned a fixed test time and a fixed test power consumption value. The objective is to minimize the total test application time and the routing of control lines while making sure that the total test power consumption at any time is below a given limit. The minimization of control lines is achieved by grouping tests based on the floor-plan in such a way that testing of neighbouring cores are scheduled in the same test session. The advantage of the grouping is that the control lines can be shared among all test sets executed in the same session. Recently Wang *et al.* proposed a test scheduling technique based on *partitioned testing with run to completion* for memories with dedicated BIST [275].

An analytic test scheduling approach, also for nonpartitioned testing, was proposed by Chou *et al.* [41] where as in Zorian's approach each test set is assigned a fixed test time and a fixed test power value. Test conflicts are modeled in a general way using a resource graph. Based on the resource graph, a test compatibility graph is generated and a covering table is used to determine the tests scheduled in the same test session. Muresan *et al.* [202] have proposed a test scheduling technique with the same assumptions as Chou *et al.* and to allow a higher degree of flexibility in the scheduling process *partitioned testing with run to completion* is used.

In all the above approaches, each testable unit has one dedicated test set with a fixed test time. Sugihara *et al.* proposed a technique for the selection of test sets for each testable unit where each testable unit can be tested by one test set using an off-chip test source and an off-chip test sink as well as one test set using a dedicated on-chip test source and a dedicated on-chip test sink [256]. The objective is to find a trade-off between the number of test vectors in on-chip resources (test source and test sink) and off-chip resources (test source and test sink). The sharing of test resources may introduce conflicts if a test resource can only generate test patterns for a testable unit at a time. Chakrabarty also proposed a test scheduling technique for systems tested by two test sets, one BIST test set and one stored at the ATE [28,25]. Chakrabarty also considered the conflicts that appears when sharing the test bus for test data transportation. Furthermore, the sharing of BIST resources among testable units is considered.

A test infrastructure is used for the transportation of test data, that is test vectors and test responses. The Advanced Microcontroller Bus Architecture (AMBA) is an approach where all test sets are scheduled in a sequence on a single bus [88]. Another bus approach is proposed by Varma and Bhatia [267]. Instead of having all TAM wires in a single bus, Varma and Bhatia propose a technique where several set of wires form several test buses. The tests on each test bus are, as in the case with AMBA, scheduled in a sequence. However, tests on different buses are executed concurrently. Aerts and Marinissen have also proposed three architectures, *multiplexing* where all tests are scheduled in a sequence, *distributed* where all tests are scheduled concurrently on their dedicated TAM wires, and *daisy-chain* where all tests are scheduled concurrently and as soon as a core is finished with its testing, the core is by-passed using a clocked buffer [5]. A common draw-back with scheduling tests in a sequence is that the testing of interconnection is a bit cumbersome.

The advantage of scheduling the tests in a sequence is that only one test set is active at a time and there is only switching in one testable unit at a time, which reduces the test power consumption. In an concurrent approach the testing of several cores can be performed at the same time. In the distributed architecture the testing of all cores are started at the same time, which means that all cores are activated simultaneously, resulting in a high test power consumption. In the daisy-chain approach, all cores are also scheduled to start at the same time and the test vectors are pipelined through the cores. The idea of daisy-chain tests is efficient from a test time perspective, however, from a test power consumption perspective it results in a high switching activity. Saxena *et al.* have proposed a technique to reduce the test power consumption in scan-based designs by using a gated sub-chain scheme [241]. Experimental results indicate that comparing an original design and a design with gated sub-chains, the test time remains the same but the test power consumption is reduced by the number of gated sub-chains.

A core test wrapper is the interface between a core and the TAM. A standard core wrapper such as the proposed P1500 or the TestShell can be in four modes, normal operation, internal core test, external test, and bypass [104, 188]. The tests in a system can be grouped into *wrapped core tests* and *unwrapped core tests*. A *wrapped core test* is a test at a core equipped with a dedicated interface (a wrapper) to the TAM and an *unwrapped core test* is a test at a core that does not have a dedicated wrapper. A new conflict appears, namely test wrapper conflict. Several approaches have been proposed for wrapped core tests. Iyengar *et al.* proposed a technique for core tests where a fixed TAM bandwidth is assumed to have been partitioned into a set of fixed TAMs and the problem is to assign cores to the TAMs in such a way that the total test application time is minimized [113]. The tests on each TAM are scheduled in a sequence and the tests can be assigned to any of the TAMs in the system. In order to make the approach applicable to large industrial designs, Iyengar *et al.* have proposed the use of an heuristic instead of Integer Linear Programming (ILP) [112].

Several approaches by Iyengar *et al.* [114], Goel and Marinissen [73], Goel and Marinissen [74], Goel and Marinissen [75], Huang *et al.* [97], Koranne [139], and Koranne and Iyengar [141] have been proposed for the assignment of TAM wires to each core test. Hsu *et al.* [96] and Huang *et al.* [100] proposed also techniques for TAM wire assignment under power constraints for tests with fixed power consumption and variable test times. Iyengar *et al.* proposed a technique where hierarchical conflicts are considered [118]. An approach for TAM design and test scheduling is proposed by Cota *et al.* [43]. The test data can be transported on dedicated TAM as well as on the functional bus. To further explore the design space, the approach allows redesign and extensions of the functional bus.

3 TEST PROBLEMS

In this section we describe the test problems we are considering and their modeling.

3.1 Test Time

In this chapter we use a test time model that assumes within a given range a linear dependency between the test time and the number of TAM wires assigned to a testable unit. In our model we assume that the designer specifies a bandwidth range for each core. It means that for each testable unit a bandwidth is to be selected, which is within the minimal bandwidth and the

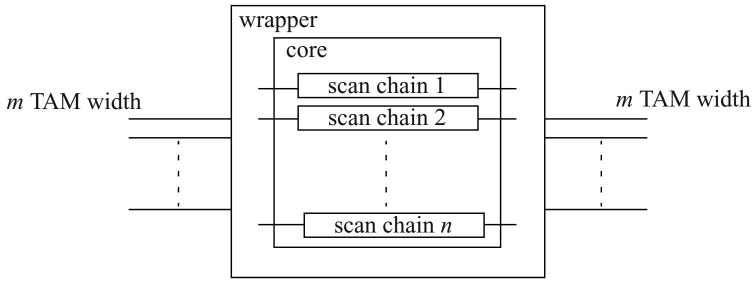


*Figure 197.*Scan-chains design at a core.

maximal bandwidth range and the test time contra TAM bandwidth within the given range is linear.

The test time for executing a test at a testable unit is defined by the time required for applying the test vectors. For some test methods, the test time is fixed, while for other test methods, such as scan-based testing, the testing time can be modified. A modification is achieved by test parallelization. For instance assume a core with a number of scan chains which form a single chain connected to a single TAM wire. The testing of the core is performed by shifting in a test vector and when the scan chains are filled, a capture cycle is applied, and then the test response is shifted out. A major part of the testing time is therefore consumed by the shift process. In order to reduce the shifting time, a new test vector can be shifted in at the same time as the test response from the previous test vector is shifted out. To further reduce the time consumed due to shifting, the scan chains can be connected into several wrapper chains where each wrapper chain is connected to a TAM wire. For instance if the *n* scan chains in Figure 197 are connected to *m* wrapper chains ($n \geq m$) the loading of a new test vector can be performed in the *m* wrapper chains concurrently.

The problem of forming wrapper chains has been addressed by Iyengar *et al.* [113] and the test application time for a core is given by:

$$\tau_e = (1 + \max\{s_i, s_o\}) \times p + \min\{s_i, s_o\} \tag{12.1}$$

where s_i is the longest wrapper chain for scan in, s_o is the longest wrapper chain for scan out and p is the number of test vectors.

The computation of an exact test time requires an algorithm such as the one proposed by Iyengar *et al.* to determine the number of wrapper chains at a core. It is important to note that even if the test time is computed exactly for each testable unit, the above formula does not consider the effect at the system level introduced by the clocked bypass structures, which is used in the TestShell when the TAM wires are becoming longer. Since the application of the formula at the system level does not lead to exact testing time and in order to reduce the computational cost, we suggest an approximation of the test time:

$$\tau_a = \left\lceil \frac{\tau_{e1}}{m} \right\rceil \tag{12.2}$$

where τ_{e1} is the test time when a single wrapper chain is assumed and *m* is the number of TAM wires assigned to the core and *m* is in the designer specified range.

We have analyzed the correlation between the exact test time (τ_e) computation and the approximated test time (τ_a). We have used one of the largest industrial designs, the P93791, in the ITC'02 benchmarks [189] to illustrate our analysis. We have extracted the twelve scan-based cores in the P93791. Key data for the design is in Table 22. The aim of the analysis is to check the correlation between the exact computation of the test time and our proposed approximation of the test time, and also to identify possible reasons for a nonlinear dependency between test time and the number of wrapper chains.

The analysis results for cores 1, 6 and 11 are collected in Table 23, the results for cores 12, 13, and 14 in Table 24, the results for cores 17, 19 and 20 in Table 25, and the results for core 20, core 23 and core 27 in Table 26. For each core, we have computed the exact test time using the wrapper chain partitioning algorithm presented by Iyengar *et al.* [113] for different cases of wrapper chains (TAM width) 1 to 16. For each width, we have also computed the approximate test time (τ_a) and the difference between the exact test time

Core	*Test vectors*	*Inputs*	*Outputs*	*Bidirs*	*Scan chains*	*Shortest scan-chain*	*Longest scan-chain*
1	409	109	32	72	46	1	168
6	218	417	324	72	46	500	521
11	187	146	68	72	11	17	82
12	391	289	8	72	46	92	93
13	194	111	31	72	46	173	219
14	194	111	31	72	46	173	219
17	216	144	67	72	43	145	150
19	210	466	365	72	44	97	100
20	416	136	12	72	44	181	132
23	234	105	28	72	46	1	175
27	916	30	7	72	46	50	68
29	172	117	50	0	35	185	189

Table 22. Characteristics for the scan-based cores in design P93791.

	Core 1			*Core 6*			*Core 11*		
TAM width	*Test time,* τ_e	*Test time,* τ_a	*Diff (%)*	*Test time,* τ_e	*Test time,* τ_a	*Diff (%)*	*Test time,* τ_e	*Test time,* τ_a	*Diff (%)*
1	2862952	2862952	0%	5317007	5317007	0%	149381	149381	0%
2	1431714	1431476	0.02%	2658613	2658504	0.004%	74784	74691	0.12%
3	954862	954317	0.06%	1809815	1772336	2.1%	49981	49794	0.4%
4	740459	715738	3.3%	1358456	1329252	2.1%	37580	37345	0.6%
5	573163	572590	0.1%	1126316	1063401	5.6%	32513	29876	8.1%
6	494049	477159	3.4%	907097	886168	2.3%	25177	24897	1.1%
7	431729	408993	5.3%	793217	759572	4.2%	21608	21340	1.2%
8	370639	357869	3.4%	679337	664626	2.2%	18977	18673	1.6%
9	318561	318106	0.14%	674957	590779	12.5%	17105	16598	3.0%
10	308319	286295	7.1%	565457	531701	6.0%	16538	14938	9.7%
11	305449	260268	14.8%	561077	483364	13.9%	15603	13580	13.0%
12	248049	238579	10.6%	455738	443084	2.8%	15603	12448	20.2%
13	246409	220227	10.6%	451577	409001	9.4%	15603	11491	26.4%
14	244359	204497	16.3%	451358	379786	15.9%	15603	10670	31.6%
15	191464	190863	0.3%	447197	354467	20.8%	15603	9959	36.2%
16	186549	178935	4.1%	341858	332313	2.8%	15603	9336	40.2%

Table 23. Test time comparison between an exact method and an approximation for cores 1, 6, and 11 in P93791. Diff (%) is computed as $|\tau_e-\tau_a|/\tau_e\times100$.

and the approximated test time. From the results in Tables 23, 24, 25 and 26 we observe that the difference between τ_e and τ_a is extremely low for low TAM width. However, as the TAM width increases, the difference between τ_e and τ_a also increases. Among the cores, the case is worst for core 11. We have therefore made an investigation of core 11, and we made two observations. The number of scan chains is 11 while the TAM bandwidth has been in the range from 1 to 16, and the length of the scan chains is rather unbalanced. The shortest scan chain is 17 flip-flops long while the longest scan chain consists of 82 flip-flops. We have made three new scan chain partitions of core 11 (Table 27), namely

- *balanced.11* - where the 11 scan chains are redesigned to be as balanced as possible,
- *balanced.22* - where the number of scan chains is increased to 22 and the partitions are made as balanced as possible,

	Core 12			*Core 13*			*Core 14*		
TAM width	*Test time, τ_e*	*Test time, τ_a*	*Diff (%)*	*Test time, τ_e*	*Test time, τ_a*	*Diff (%)*	*Test time, τ_e*	*Test time, τ_a*	*Diff (%)*
1	1813502	1813502	0%	1893564	1893564	0%	1893564	1893564	0%
2	906947	906751	0.02%	946880	946782	0.01%	946880	946782	0.01%
3	604799	604501	0.05%	639989	631188	1.4%	639989	631188	1.4%
4	453892	453376	0.1%	480479	473391	1.5%	480479	473391	1.5%
5	363774	362700	0.3%	395654	378713	4.3%	395654	378713	4.3%
6	302596	302250	0.11%	320969	315594	1.7%	320969	315594	1.7%
7	259492	259072	0.16%	278654	270509	2.9%	278654	270509	2.9%
8	227337	226688	0.29%	242384	236696	2.3%	242384	236696	2.3%
9	217951	201500	7.5%	235754	210396	10.8%	235754	210396	10.8%
10	182278	181350	0.51%	200069	189356	5.4%	200069	189356	5.4%
11	181887	164864	9.4%	193439	172142	11.0%	193439	172142	11.0%
12	151689	151125	0.4%	165749	157797	4.8%	165749	157797	4.8%
13	145823	139500	4.3%	159509	145659	8.7%	159509	145659	8.7%
14	145431	129536	10.9%	153269	135255	11.8%	153269	135255	11.8%
15	145431	120900	16.9%	152879	126238	17.4%	152879	126238	17.4%
16	114060	113344	0.6%	123434	118348	4.1%	123434	118348	4.1%

Table 24. Test time comparison between an exact method and an approximation for cores 12, 13, and 14 in P93791. Diff (%) is computed as $|\tau_e-\tau_a|/\tau_e\times100$.

- *balanced.44* - where the number of scan chains is increased to 44 and the partitions are made as balanced as possible,
- *balanced.88* - where the number of scan chains is increased to 88 and the partitions are made as balanced as possible.

The results from the experiments on the original core 11 and the four versions of the balanced design are collected in Table 28. We made experiments with the TAM width in the range from 1 to 16, and for each of the versions of core 11, at each TAM width we computed the exact test time, the approximated test time and the difference between the exact time and the approximated time. On average, the approximated test time is 12.1% from the exact test time on the original design. For the balanced.11, which is the balanced version of the original one, the average is down to 5.7%. If the number of scan chains are increased to 22 as in balanced.22 the average difference is only 2.7% and if the number of scan chains are increased to 44 the average

	Core 17			*Core 19*			*Core 20*		
TAM width	*Test time, τ_e*	*Test time, τ_a*	*Diff (%)*	*Test time, τ_e*	*Test time, τ_a*	*Diff (%)*	*Test time, τ_e*	*Test time, τ_a*	*Diff (%)*
1	1433858	1433858	0%	1031266	1031266	0%	3193678	3193678	0
2	717181	716929	0.04%	515843	515633	0.04%	1597047	1596839	0.01%
3	483258	477953	1.1%	343904	343755	0.04%	1065003	1064559	0.04%
4	358699	358465	0.07%	258027	257816	0.08%	798940	798419	0.07%
5	290128	286772	1.2%	206553	206253	0.15%	639256	638736	0.08%
6	257361	238976	7.1%	179524	171878	4.3%	551690	532280	3.5%
7	225028	204837	9.0%	156728	147324	6.0%	477047	456240	4.4%
8	192912	179232	7.1%	134801	128908	4.4%	416582	399210	4.2%
9	161664	159318	1.5%	114775	114585	0.17%	361121	354853	1.7%
10	160579	143386	10.7%	111164	103127	7.2%	346109	319368	7.7%
11	130849	130350	0.4%	94095	93751	0.4%	291064	290334	0.25%
12	129331	119488	7.6%	90077	85939	4.6%	286061	266140	7.0%
13	128680	110297	14.3%	85444	79328	7.2%	271466	245668	9.5%
14	128029	102418	20.0%	83133	73662	11.4%	261458	228120	12.8%
15	97215	95591	1.7%	68992	68751	0.35%	216005	212912	1.4%
16	97215	89616	7.8%	67506	64454	4.5%	215588	199605	7.4%

Table 25. Test time comparison between an exact method and an approximation for cores 17, 19 and 20 in P93791. Diff (%) is computed as $|\tau_e - \tau_a|/\tau_e \times 100$.

difference is down to 1.5%. A further increase of the number of scan chains to 88, balanced.88, will not give a lower difference at the TAM bandwidth we did experiments with. The analysis indicates that designing the scan chains at a core in a balanced way with a relatively high number of scan chains will result in a near linear dependency between test time and TAM width. It should be noted that we used TAM bandwidth in the range from 1 to 16. Obviously, the linear dependency does not hold for small cores with a few scanned element. However, our modeling assumes a linear dependency within a range specified by the designer.

3.2 Test Power Consumption

The power consumption is usually higher during testing compared to that during normal operation. The reason is that a high switching activity is desired in order to detect as many faults as possible for each test vector. Detecting a high number of faults per vector minimizes the number of test

	Core 23			*Core 27*			*Core 29*		
TAM width	*Test time,* τ_e	*Test time,* τ_a	*Diff (%)*	*Test time,* τ_e	*Test time,* τ_a	*Diff (%)*	*Test time,* τ_e	*Test time,* τ_a	*Diff (%)*
1	1836917	1836917	0%	2869269	2869269	0%	1161619	1161619	0%
2	918609	918459	0.02%	1435093	1434635	0.03%	584201	580810	0.6%
3	612618	612306	0.05%	957346	956423	0.1%	389582	387206	0.6%
4	475404	459229	3.4%	718007	717317	0.1%	292187	290405	0.6%
5	367758	367383	0.1%	588713	573854	2.5%	232497	232324	0.07%
6	317719	306153	3.6%	480507	478212	0.5%	194963	193603	0.7%
7	277534	262417	5.4%	418151	409896	2.0%	166241	165946	0.2%
8	240874	229615	4.7%	365882	358659	2.0%	161235	145202	9.9%
9	204440	204102	0.16%	341123	318808	6.5%	130090	129069	0.8%
10	200689	183692	8.5%	303526	286927	5.5%	129057	116162	10.0%
11	194579	166992	14.2%	279684	260843	6.7%	128884	105602	18.1%
12	160739	153076	4.8%	248506	239106	3.8%	97568	96802	0.8%
13	160504	141301	12.0%	248506	220713	11.2%	96879	89355	7.8%
14	156979	131208	16.4%	232000	204948	11.7%	96879	82973	14.4%
15	122898	122461	0.35%	217328	191285	12.0%	96706	77441	19.9%
16	120554	114807	4.8%	187067	179329	4.1%	96533	72601	24.8%

Table 26. Test time comparison between an exact method and an approximation for cores 23, 27 and 29 in P93791. Diff (%) is computed as $|\tau_e - \tau_a| / \tau_e \times 100$.

	# scan chains	*Scan chains*
Original	11	82 82 82 81 81 81 18 18 17 17 17
Balanced.11	11	53 53 53 53 52 52 52 52 52 52 52
Balanced.22	22	27 27 27 27 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26 26
Balanced.44	44	14 14 14 14 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
Balanced.88	88	7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 66 6 6 6 6 6 6 6

Table 27. Modified scan-chain partitioning on core 11.

vectors and therefore also the test time. The problem is that a high power consumption might damage the system. It is therefore needed to schedule the tests in such a way that the total test power consumption is kept under control. To be able to analyze the test power consumption, a model of the power con-

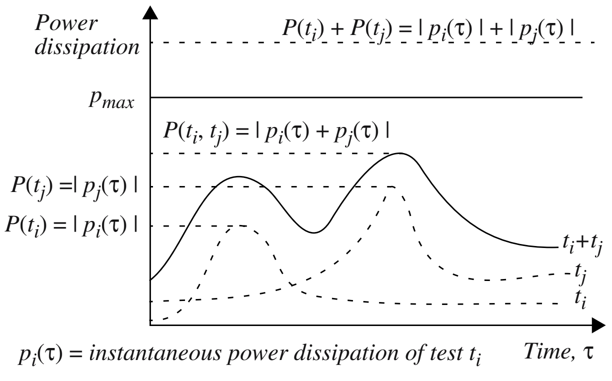
	Original			*Balanced.11*		*Balanced.22*		*Balanced.44*		*Balanced.88*	
TAM width	τ_a	τ_e	*Diff (%)*	τ_e	*Diff (%)*	τ_e	*Diff(%)*	τ_e	*Diff(%)*	τ_e	*Diff(%)*
1	149381	149381	0	149381	0	149381	0	149381	0	149381	0
2	74691	74784	0.1	74784	0.1	74784	0.1	74784	0.1	74784	0.1
3	49794	49981	0.4	49981	0.4	49981	0.4	49981	0.4	49981	0.4
4	37345	37580	0.6	37579	0.6	37579	0.6	37579	0.6	37579	0.6
5	29876	32513	8.1	32135	7.0	30065	0.6	30064	0.6	30064	0.6
6	24897	25177	1.1	25177	1.1	25177	1.1	25177	1.1	25177	1.1
7	21340	21608	1.2	21806	2.1	21618	1.3	21608	1.2	21608	1.2
8	18673	18977	1.6	20487	8.9	18976	1.6	18976	1.6	18976	1.6
9	16598	17105	3.0	19739	15.9	16905	1.8	16909	1.8	16909	1.8
10	14938	16538	9.7	19739	24.3	16161	7.6	15219	1.8	15219	1.8
11	13580	15603	13.0	13903	2.3	13903	2.3	13903	2.3	13903	2.3
12	12448	15603	20.2	12775	2.6	12775	2.6	12775	2.6	12775	2.6
13	11491	15603	26.4	11839	2.9	11831	2.9	11835	2.9	11836	2.9
14	10670	15603	31.6	11090	3.8	10903	2.1	10903	2.1	10898	2.1
15	9959	15603	36.2	11086	10.2	10899	8.6	10147	1.9	10147	1.9
16	9336	15603	40.2	10338	9.7	10337	9.7	9582	2.6	9582	2.6
	Average:		12.1		5.7		2.7		1.5		1.5

Table 28. Test time comparison between an exact model (τ_e) and our approximation (τ_a) for variations of core 11 (see Table 27). Diff (%) is computed as $|\tau_e-\tau_a|/\tau_e\times100$.

sumption is needed. Chou *et al.* [41] have introduced a test power model by denoting a fixed test power value to each test set. The motivation for the model is as follows. Figure 198 shows an example of the test power dissipation variation over time τ when the two tests t_i and t_j are executed. If $p_i(\tau)$ and $p_j(\tau)$ are the instantaneous power dissipation of the two compatible tests t_i and t_j, respectively, and $P(t_i)$ and $P(t_j)$ are the corresponding maximal power dissipation. In the case where $p_i(\tau) + p_j(\tau) < p_{max}$, the two tests can be scheduled at the same time. However, instantaneous power of each test vector is hard to obtain. To simplify the analysis, a fixed value $p_{test}(t_i)$ is assigned for all test vectors in a test t_i such that when the test is performed the power dissipation is no more than $p_{test}(t_i)$ at any moment. The $p_{test}(t_i)$ can be assigned as the average power dissipation over all test vectors in t_i or as the maximum power dissipation over all test vectors in t_i. The former approach could be too optimistic, leading to an undesirable test schedule, which exceeds the test power constraints. The latter could be too pessimistic; however, it guarantees that the power dissipation will always satisfy the constraints. Usually, in a test environment the difference between the average and the maximal power dissipation for each test is often small since the objective is to maximize the circuit activity so that it can be tested in the shortest possible time [41]. Therefore, the definition of power dissipation $p_{test}(t_i)$ for a test t_i is usually assigned to the maximal test power dissipation ($P(t_i)$) when test t_i alone is applied to the device. This simplification was introduced by Chou *et al.* [41] and has been used by Zorian [287] and by Muresan *et al.* [202].We will also use this assumption in our approach.

3.3 Test Power Consumption at Test Parallelization

The test power consumption depends on the switching activity. During testing in scan-based systems, switches appear not only during the application of test vectors, at the capture cycles, but also in the shift process when a new test vector is shifted in while the test response from the previous test vector is shifted out. Saxena *et al.* [241] proposed a gating scheme to reduce the test power dissipation during the shift process. Given a set of scan-chains as in Figure 199 where the three scan-chains are forming a single chain. During the shift process, all scan flip-flops are active and it leads to high switch activity in the system and high power consumption. However, if a gated sub-chain scheme as proposed by Saxena *et al.* is introduced (Figure 200), only one of the three chains is active at a time during the shift process while the others are switched off and as a result no switching activity is taking place in them. The test time in the examples (Figure 199 and Figure 200) are the same while the switch activity is reduced in the gated example and also the activity in the clock tree distribution is reduced [241]. The experimental results presented by Saxena *et al.* [241] indicate that the test power consumption can be reduced to



$p_i(\tau)$ = *instantaneous power dissipation of test* t_i

$P(t_i) = | p_i(\tau) |$ = *maximum power dissipation of test* t_i

*Figure 198.*Power dissipation as a function of time [41].

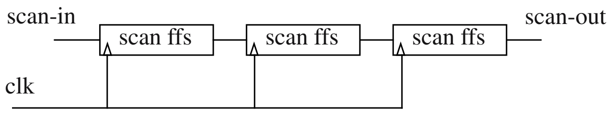


*Figure 199.*Original scan chain [241].

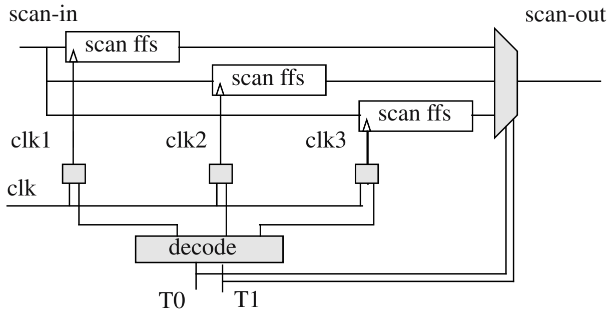


*Figure 200.*Scan chain with gated sub-chains[241].

a third using a gated scheme with three sub-chains as in Figure 200 compared to the original scheme in Figure 199. We use a generalized model based on the experimental results presented by Saxena *et al.*, which shows that there is a linear dependency between the test time and the test power consumption. If *x* scan chains exist at a core, it is possible to form *x* number of wrapper chains. In such a case, the test time will be reduced since the shift process is minimized, however, the test power consumption will be maximized since all of the *x* scan chains are active at the same time. On the other hand, if a single wrapper chain is assumed where all scan chains are connected into a single wrapper chain, the test time will increase but the test power consumption can be reduced by gating the *x* scan chains. For the power modeling, we do as with the test time. We assign one test time value and one test power consumption value at a single wrapper chain. As the number of assigned TAM wires changes, we assume that there is a linear change in test time and the test power within the specified range.

3.4 Test Resource Limitations

A test source usually has a limited bandwidth. For instance, an external tester only supports a limited number of scan chains at a time [92]. The memory limitation at a test source may also put a limitation on the testing [92] and there could also be a limitation in the number of available pins. In our model, we use a fixed value for the maximal bandwidth at a test source and a test sink. We also use a fixed value to indicate the memory size used for test vector storage at a test source.

3.5 Test Conflicts

We have discussed conflicts due to test power consumption, bandwidth limitations, memory limitations and sharing of TAM wires. These are conflicts that are to be considered during the test scheduling. There are also conflicts that are known in advance. In this section we will discuss such test conflicts. These conflicts are due to interference during testing and also the testing of interconnections and UDL. It is also possible that conflicts appear when the system contains cores that are embedded in cores [118].

In general, in order to execute a test, a set of testable units might be required. This set can often be specified in advance. The advantage of specifying the test conflicts explicitly is that it gives the designer the flexibility to explore different design possibilities. It also makes our technique more flexible compared to an approach where test conflicts are built in to the tool.

We will use the example in Figure 201 to illustrate unwrapped core testing and test interference. The example consists of only one test source and one test sink and three cores (c_1, c_2, and c_3) where core c_1 consists of two testable units, b_{11} and b_{12}, core c_2 consists of two testable units, b_{21} and b_{22}, and core c_3 consists of two testable units, b_{31} and b_{32}. Core c_1 and core c_2 have interfaces to the TAM, *i.e.* they are placed in wrappers. As discussed above, we call such cores *wrapped cores* while cores such as core c_3, which do not have a dedicated interface to the TAM, are called *unwrapped cores*. Tests performed at wrapped cores are called *wrapped core tests,* and tests at unwrapped cores are called *unwrapped core test.* The testing of the UDL at the testable unit b_{32} is an unwrapped core test. In order to perform testing of b_{32} the test vectors are transported from the test source r_1 using the TAM to the wrapped core c_1. The wrapper at c_1 is placed in the external test mode, which means that no core tests can be applied at c_1 as long as the wrapper is in external test mode. The test vectors are transported from the wrapper at c_1 to the testable unit b_{32}. The test responses are captured at the wrapper at core c_2, which, as core c_1, is placed in the external test mode. From the wrapper at core c_2 the test response is transported via the TAM to the test sink s_1. The testing of the testable units b_{11} and b_{12} at core c_1 and b_{21} and b_{22} core c_2 cannot be performed at the same time as the testing of b_{32}. In our model we specify this as a conflict list: $\{b_{11}, b_{12}, b_{21}, b_{23}\}$.

The testing of b_{32} *could, but does not have to,* have an impact on the testing of some other testable units such as b_{31}. In our approach we list all testable units that are required in order to execute a test explicitly. If b_{31} is interfered during the testing of b_{32}, b_{31} is also included in the conflict list.

An advantage of listing all the test conflicts explicitly is that it makes it possible to model hierarchy where for instance cores are embedded in cores. An hierarchical modeling technique usually has an implicit way to model conflicts. In our approach, such implicit modeling does not exist, and hence longer conflict lists are required.

3.6 Test Access Mechanism Design

In our design flow, we assume that there are initially no TAM wires in the system. TAM wires are added when the transportation of test data in the system requests them. In general, all test sources are to be connected with wrapped cores and the wrapped cores have to be connected with the test sinks.

In our modeling of the TAM wires, we assume that each TAM wire is independent of other TAM wires. It means we are not partitioning the TAM wires into subsets. We also assume that the delay of data transportation on the TAM wires is negligible. We also assume, as discussed above, that the time impact from eventual by-pass structures introduced on long time wires is not considered.

For the placement modeling of cores and test resources, we assume a single point assignment given by (x,y)-coordinates for each. The placement model could be more elaborate than a single point assignment, however, a more advanced model would lead to higher computational complexity. To illustrate that further, consider a conceptional view of a P1500 compliant core given in Figure 202. It is important to note that it is only a conceptional view. For instance, the inputs do not always have to be placed on the left hand side and the outputs on the right hand side. A more elaborate placement model than a single point model would need a way to determine where to connect the wires. A more elaborate wiring model would also have to consider the size of the core since the wiring inside the wrapper has a cost.

Furthermore, a more elaborate model should handle the wiring due to connecting the scan-chains into a set of wrapper chains. It means that the model should consider exactly where on the core each of the scan inputs and each of the the scan outputs are placed. On top of this, it is also needed to have a model for different types of cores. A core can be equipped with a built-in bypass structure or a special transparency mode, which can reduce the required

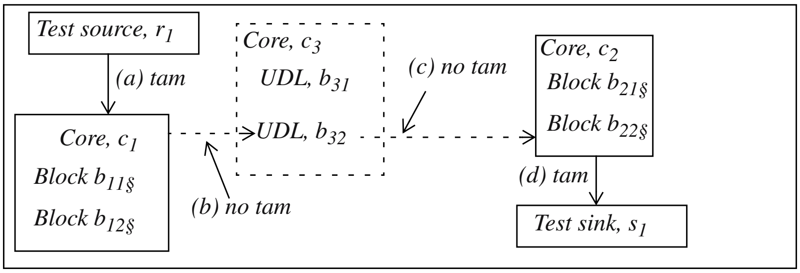


*Figure 201.*Illustration of unwrapped core testing and test interference.

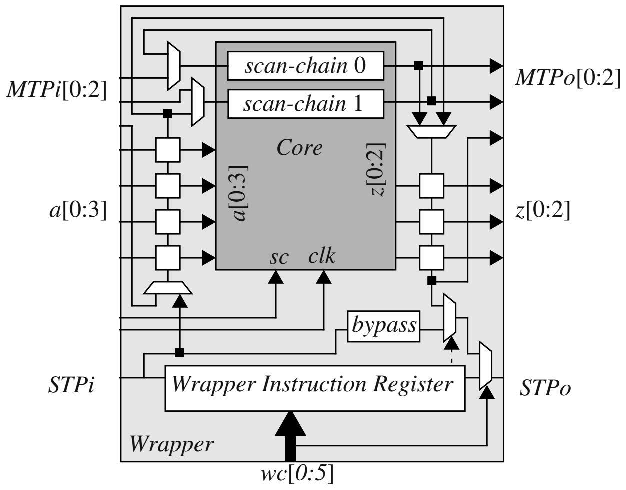


*Figure 202.*Conceptual view of a P1500 compliant core [186].

wiring but will require consideration during the test time modeling since the transparency mode can require several clock cycles in order to transport test data from the core input to its outputs. There might also exist several possibilities for transparency where each such mode requires a certain number of clock cycles and each mode consumes a certain amount of power. We have therefore decided to use a single point model for the placement.

3.7 System Modeling

We have in Section 3 discussed SOC test problems and their modeling. In this section, we describe our system model and the input specification to our test design tool. We illustrate the modeling and the input specification using an example.

We have developed a model, which is based on our previous system model [158, 160, 165], to model a SOC system with given test methods and added test resources, as illustrated in Figure 203, A *design with test*, *DT,* is represented as follows:

$DT = (C, T, R_{source}, R_{sink}, p_{max})$, where: $C = \{c_1, c_2,..., c_n\}$ is a finite set of cores; each core, $c_i \in C$, is characterized by:

(x_i, y_i): placement denoted by x and y coordinates and each core consists of a finite set of blocks $c_i = \{b_{i1}, b_{i2},..., b_{im_i}\}$ where $m_i > 0$ $\{i=1,2,...,n\}$. Each block, $b_{ij} \in B$ $\{i=1, 2,..., n, j=1, 2,..., m_i\}$, is characterized by:

$minbw_{ij}$: minimal bandwidth,

$maxbw_{ij}$: maximal bandwidth, which are the minimal possible bandwidth and maximal possible bandwidth at a block, respectively.

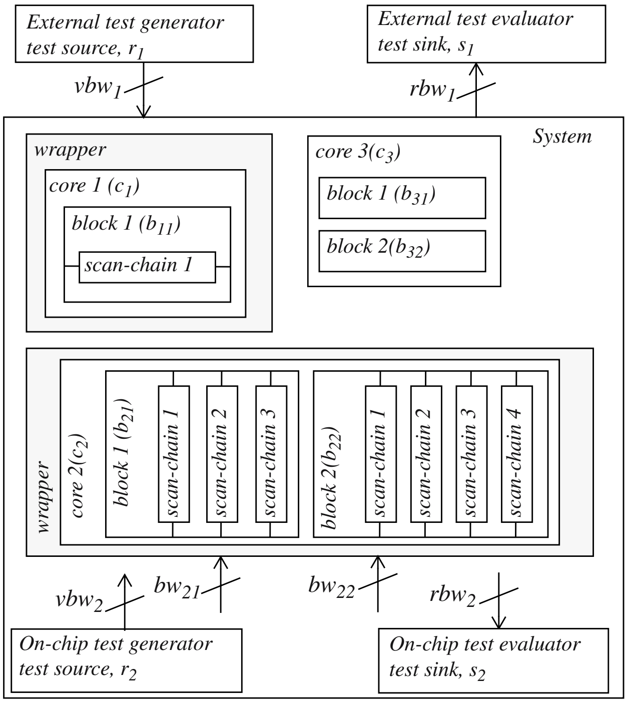


Figure 203. Modeling the example system.

Each block, b_{ij}, is attached with a finite set of tests, $b_{ij}=\{t_{ij1}, t_{ij2},..., t_{ijo_{ij}}\}$ where $o_{ij}>0$ $\{i=1, 2,..., n, j=1, 2,..., m_i\}$ and each test, $t_{ijk} \in T$ $\{i=1, 2,..., n, j=1, 2,..., m_i, k=1, 2,..., o_{ij}\}$, is characterized by:

τ_{ijk}: test time (at TAM bandwidth 1),

p_{ijk}: test power (at TAM bandwidth 1 using a gated sub-chain scheme (discussed above)),

mem_{ijk}: required memory for test pattern storage.

cl_{ijk}: constraint list with blocks required for the test.

$R_{source} = \{r_1, r_2,..., r_p\}$ is a finite set of test sources where each test source, $r_i \in R_{source}$, is characterized by:

(x_i, y_i): placement denoted by x and y coordinates,

vbw_i : vector bandwidth,

$vmem_i$: vector memory size.

$R_{sink} = \{s_1, s_2,..., s_q\}$ is a finite set of test sinks; where each test sink, $s_i \in R_{sink}$, is characterized by:

(x_i, y_i): placement denoted by x and y coordinates,

rbw_i: response bandwidth,

source: $T \rightarrow R_{\text{source}}$ defines the test sources for the tests;

sink: $T \rightarrow R_{sink}$ defines the test sinks for the tests;

p_{max}: maximal allowed power at any time.

The input specification for the example system (Figure 203) to our test design tool is outlined in Figure 204. The power limit for the system is given under [Global Constraints], and at [Cores] the placement (x, y) for each core is given and all blocks within each core are listed. The placement (x, y) for each test source, its possible bandwidth, and its test vector memory are given at [Generators]. At [Evaluators], the placement (x, y) and the maximal allowed bandwidth for each test sink is given. For each test the following is specified under [Tests], the test identifier, test power <pwr>, test time <time>, test source <tpg>, test sink <tre>, minimal <minbw> and maximal bandwidth <maxbw>, memory requirement <mem> and, optional, if the test is for testing of interconnections or UDL placed between another core <ict>. For instance, test t_2 is an *unwrapped core test* of UDL logic or interconnections between core c_1 and core c_2 (Figure 204). Test t_2 requires at least two TAM wires but not more than four. The test source for test t_2 is r_1 and the test sink is s_2. The test vectors for test t_2 requires 5 units for storage at r_1. The tests for each block are specified at [Blocks] and at [Constraints] the blocks required in order to apply a test are listed. Note that, the possibility to specify idle power for each block is implemented in our algorithm but, to simplify the discussion, it is excluded from the system model above.

The advantage of this model is that we can model a system with a wide range of tests (scan tests as well as non-scan tests such as delay, timing and cross-talk tests) and constraints. For instance we can model:

- unwrapped core test, which is for the testing of interconnections and UDL,
- any combination of test resources, for instance a test source can be on-chip while the test sink is off-chip, and vice versa,
- any number of tests per block (testable unit),
- memory requirements at test sources,
- bandwidth limitations at test resources,
- constraints among blocks, which allows the modeling of constraints such as cores embedded in cores.

Initially, it is assumed that no TAM exists in the system. Our technique adds a set of TAMs, tam_1, tam_2,..., tam_n where tam_i is a set of wires with a certain bandwidth. We assume that we can partition the TAM connected to a set of wrapped cores freely, which means we are not limited to assigning all wires in a TAM to one core at a time or dedicate TAM wires to a single core. We also assume that we can extend a subset of the TAM wires if required.

```
# Example design
[Global Constraints]
MaxPower = 25
[Cores] identifier x y {block1, block2, ..., block n}
        c1  10  30  {b11}
        c2  10  20  {b21, b22}
        c3  20  30  {b31, b32}
[Generators] identifier x y maxbw memory
            r1  10  40  3       100
            r2  10  10  1       100
[Evaluators] identifier  x y maxbw
            s1   20 40  4
            s2   20 10  4
[Tests] identifier pwr time tpg tre minbw maxbw mem ict
        t1      10  15   r1 s1  1       2       10    no
        t2      5   10   r1 s2  2       4       5     c2
    // for all tests in similar way
        t12   15  20  r1  s2   2       2      2     no
[Blocks] #Syntax: identifier idle pwr {test1, test2, ..., test n}
                b11   1     {t1, t2, t3}
                b21   1     {t4, t5}
    // for all blocks in similar way
                b32   2     {t11, t12}
[Constraints] Syntax: test {block1, block2,..., block n}
                   t1 {b11}
                   t2 {b11, b21, b22, b31, b32}
    // constraint for all tests in similar way
                  t12 {b11}
```

Figure 204. Test specification of the system in Figure 203.

The TAM is modeled as a directed graph, $G=(N, A)$, where a node, $n_i \in N$, corresponds to a member of C, R_{source} or R_{sink}. An arc, $a_{ij} \in A$, between two nodes n_i and n_j indicates the existence of a wire and a wire w_k consists of a set of arcs. For instance a wire w_1 from c_1 to c_3 passing c_2 is given by the two arcs: $\{a_{12}, a_{23}\}$.

The assignment of cores to TAM wires means connecting a test source, n_{source}, a set of cores, $n_1, n_2, ..., n_n$, and a test sink, n_{sink}, and it is denoted as:

$$n_{source} \rightarrow [n_1, n_2, \dots, n_n] \rightarrow n_{sink} \qquad (12.3)$$

where [] indicates that these nodes (wrapped cores) are included (assigned) to this TAM but not ordered.

The length, l_j, of a test wire, w_j, is given by:

$$\mathrm{adist}(n_i, n_1) + \sum_{k=2}^{n} \mathrm{adist}(n_{k-1}, n_k) + \mathrm{adist}(n_n, n_l) \qquad (12.4$$

where $n_i = n_{source}$ and $n_l = n_{sink}$ and the function *adist* gives the Manhattan distance between two nodes, *i.e*:

$$\mathrm{adist}(n_i, n_j) = |x_i - x_j| + |y_i - y_j| \qquad (12.5)$$

A set of wires form a tam_i and the routing cost is given by:

$$\mathrm{tamlength}_i = l_i \times \mathrm{bandwidth}_i \qquad (12.6)$$

where l_i is the length and $bandwidth_i$ is the width of the TAM, and the total TAM routing cost in the system is given by:

$$c_{tam} = \sum_{\forall i} \mathrm{tamlength}_i \qquad (12.7)$$

The total cost for a test solution is given by: $\alpha \times test\ time + \beta \times c_{tam}$ where *test time* is the total test application time, c_{tam} (defined above) is the total wiring cost, and α and β are two designer-specified constants determining the relative importance of the test time and the TAM cost.

4 OUR APPROACH

In this section we describe our approach to integrate test scheduling, test parallelization and TAM design. For a given floor-planned system with tests, modeled as in Section 3.7, we have to:

- determine the start time for all tests,
- determine the bandwidth for each test,
- assign each test to TAM wires,
- determine the number of TAMs,
- determine the bandwidth of each TAM, and
- route each TAM,

while minimizing the test time and the TAM cost, and considering constraints and power limitations. Note that, when the start time and the bandwidth for a test are determined, the end time is implicitly given. Compared to previous approaches [158,160,161,165], we have the following improvements:

- *Test scheduling*. In [158, 165] when a test was selected and all constraints were fulfilled, a TAM was designed. The technique there always minimized test time at the expense of the TAM cost. In the proposed approach, a cost function including both test time and TAM cost guides the test scheduling process.
- *Test parallelization*. The technique described in [161] maximized the bandwidth for each test, which resulted in low test time. However, its

draw-back is a higher TAM cost. In our proposed approach an elaborate cost function guides the algorithm.

- *TAM design*. In [158,165] when a test was considered and a free TAM existed, the TAM was selected. If an extension was required, an extension was made to minimize the additional TAM. A disadvantage of the approach is illustrated in Figure 205 where *D* is to be connected using the dashed line (Figure 205(a)). A re-routing as *A, C, D, B* would include *D* at no additional cost (Figure 205(b)).

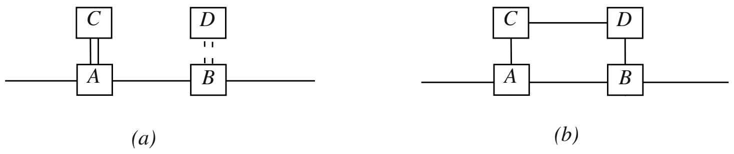


Figure 205. Illustration of (a) a published and (b) our improved TAM design approach.

The cost function for a test solution was defined above as: $\alpha\times$*test time* + $\beta\times c_{tam}$ where *test time* is the total test time, c_{tam} is the cost of the TAM and, α and β are user-defined constants used to determine the relative importance between the test time and the TAM cost. We also use the cost function for the guidance at each step in our algorithm based on each test and the TAM design. The cost function guiding our algorithm is for a test t_{ijk} using tam_l:

$$c = \alpha \times \tau_{start} + \beta \times tamlength_l \quad (12.8)$$

where: τ_{start}: is the time when t_{ijk} can start, $tamlength_l$: is the cost of the tam wiring (Eq. 12.6), and the designer specified factors α and β are used to set the relative importance between test time and TAM cost. Eq. 12.8 is used in the test scheduling algorithm for the selection of start time and TAM for each test.

4.1 Bandwidth Assignment

Test parallelization allows a flexible bandwidth assignment for each test depending on the bandwidth limitations at the block under test and the bandwidth limitations at the test resources.

The test time (see Section 3.1) for a test t_{ijk} at block b_{ij} at core c_i is given by:

$$\tau'_{ijk} = \lceil \tau_{ijk}/bw_{ij} \rceil \quad (12.9)$$

and the test power (see Section 3.2):

$$p'_{ijk} = p_{ijk} \times bw_{ij} \quad (12.10)$$

where bw_{ij} is the bandwidth at block b_{ij} at core c_i [161].

Combining the TAM cost and the test time (Equation 12.9), we get for each block b_{ij} and its tests t_{ijk}:

$$cost(b_{ij}) = \sum_{\forall k} l_l \times bw_{ij} \times \beta + \tau_{ijk} / (bw_{ij}) \times \alpha \qquad (12.11)$$

where: $l_l = [source(t_{ijk}) \rightarrow c_i \rightarrow sink(t_{ijk})]$ and k is the index of all tests at the block. To find the minimum cost of Equation 12.11, the derivative in respect to bw gives the bandwidth bw_{ij} at a block b_{ij}:

$$\sqrt{\frac{\alpha}{\beta} \times \left(\sum_{\forall k} \tau_{ijk}\right) / \left(\sum l_l\right)} \qquad (12.12)$$

Naturally, when selecting bw_{ij}, we also consider the bandwidth limitations at each block.

4.2 Test Scheduling

Our test scheduling algorithm is outlined in Figure 206. First, the bandwidth is determined for all blocks (Section 4.1) and the tests are sorted based on a key (time, power or time×power). The outmost loop terminates when all tests are scheduled. In the inner loop, the first test is picked and after calling *create_tamplan* (Section 4.3) , the required number of TAM wires are selected or designed for the test based on the cost function. If the TAM factor is important, a test can be *delayed* in order to use an existing TAM. This is determined by the cost function. If all constraints are fulfilled, the test is scheduled and the TAM assignment is performed using the technique in section 4.3. Finally, all TAMs are optimized according to the technique discussed in section 4.5.

```
for all blocks bandwidth = bandwidth(block)
sort the tests descending based on time, power or time×power
τ=0
until all tests are scheduled begin
    until a test is scheduled begin
        tamplan = create_tamplan(τ,test) // see Figure 207 //
        τ' = τ+delay(tamplan)
        if all constraints are fulfilled then
            schedule(τ')
            execute(tam plan) // see Figure 208 //
            remove test from list
        end if
    end until
    τ=first time the next test can be scheduled
end until
order (tam) // see Figure 210 //
```

Figure 206. Test scheduling algorithm.

4.3 TAM Planning

In the TAM planning phase, our algorithm:

- creates the TAMs,
- determines the bandwidth of each TAM,
- assigns tests to the TAMs, and
- determines the start time and the end time for each test.

The difference compared to the published approaches is that in the planning phase we only determine the existence of the TAMs but not their routing.

For a selected test, the cost function is used to evaluate all options (*create_tamplan*(τ', *test*)) (Figure 207). The time (τ') when the test can be scheduled to start and its TAM is determined using the cost function and if all constraints are fulfilled, the TAM floor plan is determined (*execute* (*tamplan*)) (Figure 208).

To compute the cost of extending a TAM wire with a node, the length of the required additional wires is computed. Since the order of the cores on a TAM is not decided, we need an estimation technique for the wire length. For most TAMs, the largest wiring contribution comes from connecting the nodes with the largest distance from each other. The rest of the nodes can be added on the TAM at a limited additional cost (extra routing). However, for TAMs with a high number of nodes, the number of nodes becomes important.

Our estimation of the wire length considers both of these cases. We assume that the nodes (test sources, test sinks and the wrapped cores) in the system are evenly distributed over the area, *i.e.* A = width×height = $(N_x \times \Delta) \times (N_y \times \Delta) = N_x \times N_y \times \Delta^2$, where N_x and N_y are the number of cores on the x and y axis, respectively. Therefore Δ, the average distance between two nodes, is computed as:

$$\Delta = \sqrt{A/(N_x \times N_y)} \tag{12.13}$$

The estimated length, el_i, of a wire, w_i, with k nodes is:

$$el_i = \max_{1 \le j \le k}\{l(n_{source} \to n_j \to n_{sink}), \Delta \times (k+1)\} \tag{12.14}$$

It means that we compute the maximum between the length of the longest created wire and the sum of the average distances for all needed arcs (wire parts). For example, let $n_{furthest}$ be the node creating the longest wire, and

```
for all tams connecting the test source and test sink used by the
test, select the one with lowest total cost
  tam cost=0;
  demanded bandwidth=bandwidth(test)
  if bandwidth(test)>max bandwidth selected tam then
    demanded bandwidth=max bandwidth(tam)
    tam cost=tam cost+cost for increasing bandwith of tam;
  end if
  time=first free time(demanded bandwidth)
  sort tams ascending according to extension (τ, test)
  while more demanded bandwidth
    tam=next tam wire in this tam;
    tam cost=tam cost+cost(bus, demanded bandwidth)
    update demanded bandwidth accordingly;
  end while
  total cost=costfunction(tam cost, time, test);
```

*Figure 207.*TAM estimation, *i.e. create_tamplan(τ, test)*.

```
demanded bandwidth = bandwidth(test)
if bandwidth(test)>max bandwidth selected virtual tam then
  add a new tam with the exceeding bandwidth
  decrease demanded bandwidth accordingly
end if
time=first time the demanded bandwidth is free sufficient long
sort tams in the tam ascending on extension (test)
while more demanded bandwidth
  tam=next tam in this tam;
  use the tam by adding node(test) to it, and mark it busy
  update demanded bandwidth accordingly;
end while
```

*Figure 208.*TAM modifications based on *create_tamplan* (Figure 207), i.e. *execute (tamplan)*.

n_{new} the node to be added, the estimated wiring length after inserting n_{new} is given by (Eq. 12.13):

$$el'_i = \max\left\{\begin{array}{l} \min\left\{\begin{array}{l} l(n_{source} \rightarrow n_{new} \rightarrow n_{furthest} \rightarrow n_{sink}) \\ l(n_{source} \rightarrow n_{furthest} \rightarrow n_{new} \rightarrow n_{sink}) \end{array}\right\} \\ \Delta \times (k+2) \end{array}\right\} \qquad (12.15)$$

For a TAM, the extension is given as the summation of all extensions of the wires included in the TAM that are needed in order to achieve the required bandwidth. The TAM selection for a test t_{ijk} is based on the TAM with the lowest cost according to:

$$(el'_l - el_l) \times \quad + \text{delay}(tam_l, t_{ijk}) \times \alpha. \qquad (12.16)$$

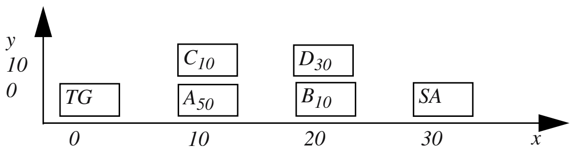
Using this cost function, we get a trade-off between adding a new TAM and delaying a test on an existing TAM. For a newly created TAM, the delay for a test is 0 (since no other test is scheduled on the TAM and the test can start at time 0): $newcost(t_{ijk}) = l(source(t_j) \rightarrow c_i \rightarrow sink(t_j)) \times \beta$.

4.4 An Example

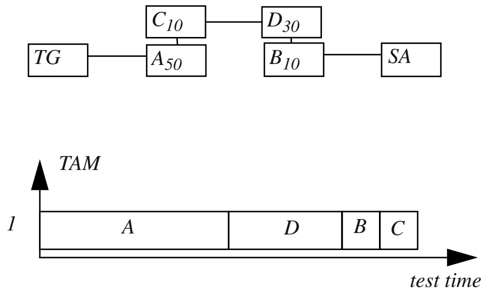
We illustrate the TAM assignment by using an example with four cores each with one block (testable units). The four cores are placed as in Figure 209(a). We assume that there is one test per block and that the test time is attached to each block. In the example, all tests are making use of the same test resources (test source and test sink) and there are no bandwidth limitations. Assuming an initial sorting of the tests based on test time, *i.e.* A, D, B, C, and success for the schedule at first attempt for each test (there are no limiting constraints which means that when a test is selected it can be scheduled) and the cost function (α:β) is in Figure 209 (b) set to 1:3 and in Figure 209 (c) to 2:3. The TAM design algorithm is illustrated for the two cases in Table 29 and the results are in Figure 209(b) and Figure 209(c). In the case when α=1 and β=3 (top Table 29), at step 1, A is selected (first in the list). No TAM exists in the design and the cost for a new TAM to be created is 90, which comes from the distance connecting TG with A and A with SA (TG→[A]→SA), 10+10+10=30, times the tam factor (β) which is 3. It is a new TAM, which means that there is no delay for the test; test A is scheduled at time 0 to 50. In the second step D is considered. The TAM created in step 1 can be extended or alternatively a new TAM can be created. Both options are estimated. The cost of a new TAM, TG→[A]→SA, is 150 while the cost of an extension of T1, TG→[A, D]→SA, is 110, computed as TAM extension 20×3 and delay on TAM 50×1. The delay on the TAM is due to that A occupies the TAM during 0 to 50. The algorithm selects to make use of the existing TAM. When all tests are assigned to TAMs the resulting TAM and test schedule are as in Figure 209(b).

For Figure 209(c), a different solution is created due to a different cost function (α=2 and β=3)(see algorithm flow bottom Table 29). The example illustrates the importance of considering the test time and the TAM design in an integrated way. In Figure 209(b) the result is a single TAM, which implies higher testing time while in Figure 209(c) two TAMs are created which makes it possible to reduce the testing time.

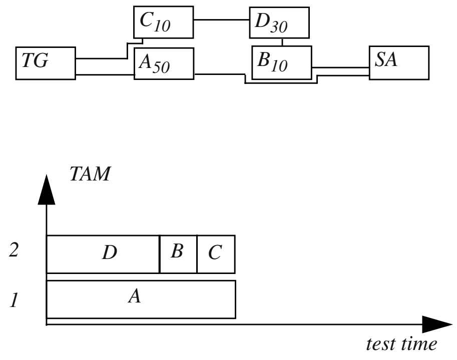
In the example, only a single TAM wire is assumed. In complex systems a higher number of TAM wires exists. In our approach, we handle that and we treat each wire independently, and when the question comes to select TAM wires for a test we explore all possibilities.



(a) Before TAM design.



(b) TAM and corresponding test schedule with α:β *factor 1:3.*



(c) TAM and corresponding test schedule with α:β *factor 2:3*

Figure 209. An example to illustrate TAM assignment.

Cost function	*Step*	*Test/ block*	*Length*	*TAM options*	*Cost*	*Schedule on selected TAM*	*Selected TAM*
α=1 and β=3	1	A	50	New:TG→A→SA	30×3+0×1=90	0-50	New:TG→A→SA
	2	D	30	New:TG→D→SA T1:TG→[A,D]→SA	50×3+0×1=150 20×3+50×1=110	0-50-80	T1:TG→[A,D]→SA,
	3	B	10	New:TG→B→SA T1:TG→[A,D,B]→SA	30×3+0×1=90 0×3+80×1=80	0-50-80	T1:TG→[A,D,B]→SA
	4	C	10	New:TG→C→SA T1:TG→[A,D,B,C]→SA	50×3+0×1=150 0×3+90×1=80	0-50-80	T1:TG→[A,D,B,C]→SA
α=2 and β=3	1	A	50	New:TG→A→SA	30×3+0×2=90	0-50	T1:TG→A→SA
	2	D	30	New:TG→D→SA T1:TG→[A,D]→SA	50×3+0×2=150 20×3+50×2=160	0-30	T2:TG→D→SA
	3	B	10	New:TG→B→SA T1:TG→[A,B]→SA T2:TG→[D,B]→SA	30×3+0×2=90 0×3+50×2=100 0×3+30×2=60	0-30-40	T2:TG→[D,B]→SA
	4	C	10	New:TG→C→SA T1:TG→[A,C]→SA T2:TG→[D,B,C]→SA	50×3+0×2=150 20×3+50×2=110 0×3+40×2=80	0-30-40-50	T2:TG→[D,B,C]→SA

Table 29. Illustration of TAM assignment. On top with a cost function where α=1 and β=3 and below with a cost function where α=2 and β=3.

4.5 TAM Optimization

Above we created the TAMs for the system, assigned each test to a TAM, determined the bandwidth of the TAMs and every test was given a start time and an end time in such a way that no conflicts and no limitations were violated. In this section, we discuss the routing of the TAMs, *order*(*tam*) in Figure 206. Our approach is based on a simplification of an algorithm presented by Caseau and Laburthe [38]. The notation TG→[A,D]→SA was above used to indicate that core A and D were assigned to the same TAM, however, the order of [A,D] was not determined (Equation 12.3), which is the objective in this section. We use:

$$n_{\text{source}} \rightarrow n_1 \rightarrow n_2 \ldots \rightarrow n_n \rightarrow n_{sink} \tag{12.17}$$

to denote that a TAM from n_{source} (the test source) to n_{sink} (the test sink) connects the cores in the order n_{source}, n_1, n_2,..., n_n, n_{sink}.

The TAM routing algorithm is outlined in Figure 210. The algorithm is applied for each of the TAMs and initially in each case, the nodes (test sources, wrapped cores, and test sinks) of a TAM are sorted in descending order according to:

$$\text{dist}(n_{\text{source}}, n_i) + \text{dist}(n_i, n_{sink}) \tag{12.18}$$

where the function *dist* gives the distance between two cores, or between a test source and a core, or between a core and a test sink, *i.e*:

$$\mathrm{dist}(n_i, n_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{12.19}$$

First the test source and the test sink are connected (Figure 210). In the loop over the list of nodes to be connected, each node is removed and added to the final list in such a way that the wiring distance is minimized according to:

$$\min\{\mathrm{dist}(n_i, n_{new}) + \mathrm{dist}(n_{new}, n_{i+1}) - \mathrm{dist}(n_i, n_{i+1})\} \tag{12.20}$$

where $1 \le i < n$ (all nodes on the TAM).

We use the TAM, TG→[C, D, A, B]→SA, from the example in Figure 209(a) to illustrate the algorithm, see Table 30. At step 0, the nodes are ordered, C, D, A, B, and a connection is added between TG and SA. At step 1, in the loop over the sorted list, C is picked and inserted between TG and SA and the TAM is modified accordingly, TG→C→SA. Node C can obviously only be inserted in one way, that is between the test source and the test sink. At step 2, node D is to be inserted. D can be inserted as TG→D→C→SA or as TG→C→D→SA. Eq. 12.20 determines which of the alternatives to select and in this example TG→C→D→SA is selected. The algorithm continues until all nodes are inserted, resulting in a TAM: TG→A→C→D→B→SA.

```
add test source and test sink to a final list
sort all cores descending according to Eq. 12.20
while cores left in the list
  remove first node from list and insert in the final list
  insert direct after the position where Eq. 12.18 is fulfilled
end while
```

Figure 210. Routing optimization of all TAMs.

Step	*Selected*	*Order before selection*	*Length of each TAM partition*	*Order after selection*	*TAM length*	*Remaining in list*
0		-		TG→SA	30	C,D,A,B
1	C	TG→SA	0	TG→C→S	14+22=36	D,A,B
2	D	TG→C→SA	18,2	TG→C→D→SA	14+10+14=38	A,B
3	A	TG→C→D→SA	6,14,20	TG→A→C→D→SA	10+10+10+14=44	B
4	B	TG→A→C→D→SA	20,14,14,6	TG→A→C→D→B→SA	10+10+10+10+10=50	-

Table 30. Illustration of TAM routing.

4.6 Complexity

The worst case complexity for the test scheduling when the TAM design is excluded is of $O(|T|^3)$ where T is the set of tests. In the TAM design there are two sequential steps; assignment and ordering (optimization). The assignment can be done in $O(|T| \times log(|T|))$ and the optimization in $O(|n|^2)$ for a TAM with n cores. If we assume that each core consists of one block (testable unit) tested by one test set, the optimization is of $O(|T|^2)$. The test scheduling and the TAM assignment are integrated while the TAM optimization is performed as a final step. The total complexity is therefore $O(|T|^4 \times log(|T|))$.

5 EXPERIMENTAL RESULTS

We have implemented our technique and compared it with previously proposed approaches using the following designs: Ericsson [160], System L [158], System S [25], ASIC Z [287], an extended version of ASIC Z, a design called Kime [132] and one design named Muresan [202]. Detailed information about all benchmarks can be found at [178] and in Appendix 1.

When referring to our technique, unless stated, the reported results are produced based on an initial sorting of the test based on the key $t \times p$ (test time×test power), and our previous techniques are referred to as SA (Simulated Annealing) [160,165] and DATE [158,165]. The final cost and our algorithm are guided by the cost function: $\alpha \times$*test time*$+\beta \times$*TAM cost* where α and β are designer specified constants determining the relative importance of the test time and the TAM cost (we use $\alpha=1$ and $\beta=1$ unless stated), see Section 4.

For the experiments we have used a Pentium II 350 MHz processor with 128 Mb RAM. Our previous results referred to as DATE and SA were performed on a Sun Ultra Sparc 10 with 450 MHz processor and 256 Mb RAM [158, 160, 165].

5.1 Test Scheduling

We have performed experiments comparing our test scheduling technique with previously proposed techniques. The results are given in Table 31. For design Muresan, the optimal test time can be computed to 25 time units. The approach proposed by Muresan *et al.* produces a solution with a test time of 29 time units. The DATE approach finds a solution using 26 time units at a computational cost of only 1 second. The SA optimization finds after 90 seconds the optimal solution while our technique finds it within 1 second. For the designs Kime, System S, and System L our approach finds the optimal solution at a low computational cost. For the larger Ericsson design, our approach finds the optimal solution after 5 seconds. The SA approach also finds the optimal solution, however, the computational cost is high. The DATE solution computes the solution after 3 seconds but the solution is 12.5% from the optimal solution.

5.2 Integrated Test Scheduling and TAM Design

We have made experiments where we integrate the TAM and the test scheduling. For each of the benchmarks (ASIC Z, Extended ASIC Z, System S, and Ericsson), we applied our algorithm using the initial sorting of the tests based on the key, *tp* (time×power), *t* (time), and *p* (power). The results from the experiments are collected in Table 32. On ASIC Z when not considering idle power, the SA produces a solution with a test time of 326 time units and 180 as the TAM cost.The total cost of the solution is 506. The DATE approach produces a solution with a better test time than the SA approach, however, the TAM cost is higher, leading to a higher total cost. Our approach produces solutions with a test time in the same range as the DATE approach. The TAM cost using our approach is better than the DATE approach leading to better total cost. The computational cost of our approach is in the same range as the DATE approach.

For the extended ASIC Z example, our approach produces test solutions with a lower test time in all cases compared to SA and the DATE approach. Furthermore, the TAM cost using our approach is lower than compared to the SA and the DATE approach, which leads to a lower total cost.

In the experiments on System S, the efficiency of our TAM design algorithm is shown. In the SA and DATE approaches the external tester supported several tests at the same time [160]. For our approach, we now assume that the external tester can support 2 tests concurrently, *i.e.* we have more limitation. The SA approach produced a solution with a TAM cost of 160 and the DATE produced a solution at a TAM cost of 320. Our approach results in a TAM cost of 100. The total cost is evaluated to 1462180 for our approach, which is to be compared to 1492194 using SA (α=1 and β=3100).

For the experiments on the Ericsson design we have used α=1 and β=2 in the cost function. In the previous approaches no bandwidth limitations were given on the external tester [158,160,165], however, here we assume a bandwidth limitation of 12 (there are 12 wires or lines to distribute).

The experiments indicate that our approach produces test solutions at a low computational cost. The test time for the test solutions are in the range of the DATE solutions, however, the TAM costs are reduced leading to lower total costs. In many cases, our approach produces solutions near the SA solution at a low computational cost.

Design	*Approach*	*Test time*	*Difference to optimum (%)*	*CPU (s)*
Muresan [202]	Optimal	25	-	-
	SA [160]	25	0	90
	Muresan [202]	29	16.0	-
	DATE [158]	26	4.0	1
	Our (p)	25	0	1
Kime [132]	Optimal	318	-	-
	Kime and Saluja[132]	349	9.7	-
	DATE [158]	318	0	1
	Our	318	0	1
System S [25]	Optimal	1152180		
	Chakrabarty SJF[28]	1204630	4.5	-
	DATE [158]	1152180	0	1
	Our	1152180	0	1
System L [158]	Optimal	1077	-	-
	DATE [158]	1077	0	1
	Our	1077	0	1
	Designer	1592	47.8	-
Ericsson [160]	Optimal	30899	-	-
	SA [160]	30899	0	3260
	DATE [158] (t)	34762	12.5%	3
	Our	30899	0	5

Table 31. Test scheduling results.

The advantage of a low computational cost is that it gives the designer a possibility to explore the design space since each iteration consumes only a low computational cost. We have computed the cost for a set of design alternatives for the Ericsson design (Table 33), which are plotted in Figure 211. The results show that when test time decreases the TAM cost increases. Typically, the designer starts with the extreme points α=0, β=1 (only TAM design is important) and α=1, β=0 (only test time is important). And then creating new solutions at different values of α and β. The designer tries to find an approperiate balance of α and β based on the previous runs and inspection of the computed testing times and the TAM costs.

Design	*Approach*	*Test Application Time*		*Test Access Mechanism*		*Total*		*CPU*
		Test time (τ)	*Difference to SA (%)*	*TAM cost (tam)*	*Difference to SA (%)*	*Total cost* $\alpha \times \tau + \beta \times tam$	*Difference to SA(%)*	
ASIC Z	SA [160]	326	-	180	-	506		865
	DATE[158]	262	-19.6	300	66.7	562	11.1	<1
	Our tp	262	-19.6	280	55.6	542	7.1	<1
	Our t	262	-19.6	280	55.6	542	7.1	<1
	Our p	305	-6.4	240	33.3	545	7.7	<1
Extended ASIC Z	SA [160]	270	-	560	-	830	-	4549
	DATE[158]	287	6.3	660	17.9	947	14.1	<1
	Our tp	264	-2.2	480	-14.3	744	-10.4	<1
	Our t	264	-2.2	460	-17.9	724	-12.8	<1
	Our p	264	-2.2	480	-14.3	744	-10.4	<1
System S	SA [160]	996194	0	160	-	1492194	-	1004s
	DATE[158] t	996194	0	320	100	1988194	33.2	<1
	Our tp	1152180	15.7%	100	-37.5	1462180	-2.0	<1
	Our t	1152180	15.7%	100	-37.5	1462180	-2.0	<1
	Our p	1152180	15.7%	100	-37.5	1462180	-2.0	<1
Ericsson	SA [160]	33082	-	6910	-	46902	-	15h
	DATE[158] tp	34762	5.1	8520	23.3	51802	10.4	62s
	Our tp	30899	-6.6	6015	-13.0	42929	-8.5	10s
	Our t	30899	-6.6	6265	-9.3	43429	-7.4	10s
	Our p	30899	-6.6	6205	-10.2	43309	-7.7	10s

Table 32. Experimental results on integrated test scheduling and TAM design.

5.3 Test Scheduling, Test Parallelization and TAM Design

In the experiments above we assumed a fixed test time for each test set. In this experiment we allow modifications of the test time at each test set. It means we have performed experiments combining test scheduling, TAM design and test parallelization using the System L design. The results are collected in Table 34. The results using SA and DATE approaches do not support TAM of higher bandwidth than 1 and therefore only test time is

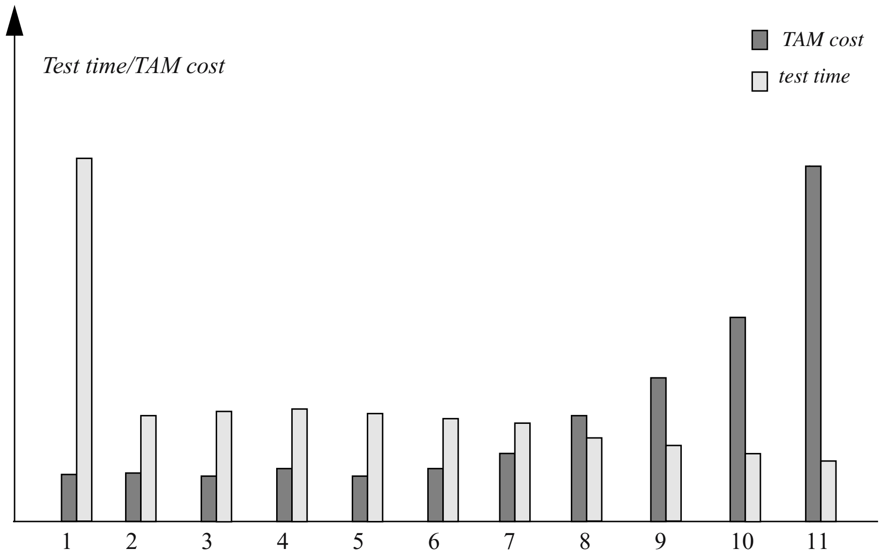
Design alternative	*a*	*b*	*test time*	*TAM cost*
1	1	0	30899	19030
2	1	15	31973	5745
3	1	30	31973	5865
4	1	45	32901	5975
5	1	60	30899	5705
6	1	90	34332	5445
7	1	125	44488	5355
8	1	250	71765	4235
9	1	500	99016	4015
10	1	1000	134706	3635
11	0	1	235084	3175

Table 33. Design alternatives for the Ericsson design.

reported (the experiments were performed ignoring TAM design). In Our* we have forced the bandwidth to 1.

6 CONCLUSIONS

Test time minimization and efficient test access mechanism (TAM) design are becoming increasingly important due to the high amount of test data to be transported in a System-on-Chip (SOC) design. In the process of developing an efficient test solution, both test time and TAM design must be considered simultaneously. A simple TAM leads to long test application time while a more extensive TAM would reduce the test time at the cost of more wiring. However, an extensive TAM might not automatically lead to lower testing times since test resource conflicts and power limitations might limit the solution. We have proposed an integrated technique for test scheduling, test parallelization (scan chain partitioning) and TAM design that minimizes test time and TAM cost while considering test conflicts and power limitations. With our approach it is possible to model a variety of tests as well as tests of wrapped cores, unwrapped cores and user-defined logic. We have implemented the technique and performed several experiments to demonstrate the efficiency of our approach.



Design alternatives.

Figure 211. Variation of test time and TAM cost for the design alternatives in Table 33 of the Ericsson design.

Approach	*Test Application Time*	*Test Access Mechanism cost*	*CPU*
SA [160]-flexible test time	316	N.A.	38s
DATE[158]-flexible test time	316	N.A.	<1
Our (α=15500:1)	316	18500	<1
Our (α=3000:β=1)	318	9490	<1
Our (α=500:β=1)	322	5140	<1
Our (α=100:β=1)	343	2420	<1
Our (α=50:β=1)	360	1750	<1
Our (α=10:β=1)	399	1030	<1
Our (α=5:β=1)	463	710	<1
Our (α=2:β=1)	593	510	<1
Our (α=1:β=1)	710	490	<1
Our (α=1:β=5)	923	380	<1
Our*	1077	240	<1

Table 34. Experimental results on System L on combined test scheduling, TAM design and test parallelization.

Chapter 13

Core Selection in the SOC Test Design-Flow[1]

1 INTRODUCTION[2]

Testing is performed to ensure the manufacturing of fault-free chips. As the number of possible faults in a chip is increasing dramatically due to the technology development, the testing process and the test design are becoming complicated and very expensive, especially in the case of SOCs. It is therefore important to take test design into consideration as early as possible in the SOC design flow in order to develop an efficient test solution. In this chapter, we propose a technique to integrate test design in the early design exploration process. The technique can, in contrast to previous approaches, be used already in the core selection process to evaluate the impact on the system's final test solution imposed by different design decisions, regarding the selection of cores and their test characteristics. The proposed technique considers the interdependent problems of core selection, test scheduling, TAM (test access mechanism) design, test set selection, and test resource floor-planning. It minimizes a weighted cost-function based on test time and TAM routing cost while considering test conflicts and test power limitations. The power consumed during testing is often higher than that during normal operation since during testing hyper-activity is desired in order to maximize the number of tested faults at a minimal time. A system under test can actually be damaged during testing, and therefore power constraints must be considered. However, power consumption is complicated to model, and often simplistic models that focus only on the global system power limit have been proposed and used. We therefore include a novel three-level power model: system, power-grid, and core. The advantage is that system-level power budget is met, and hot-spots can be avoided both at a specific core and at certain areas in the chip. We have implemented and compared the proposed technique with an estimation-based approach and a computational expensive pseudo-exhaustive method. The results from the experiments show that the pseudo-exhaustive technique cannot produce results within reasonable computational time and the estimation-based technique cannot produce solutions with a high quality. Our the proposed technique, on the other hand, produces results close to the ones produced by the pseudo-exhaustive technique at computational cost

1. The chapter is based on the papers presented at International Test Resource Workshop (TRP), 2004, [175], and International Test Conference (ITC), 2004 [177].
2. The research is partially supported by the Swedish National Program STRINGENT.

close to the cost of the estimation-based technique, i.e it produces high-quality solutions at low computational cost.

The rest of the chapter is organized as follows. A background is in Section 2 and an overview of prior work is in Section 3. The problem formulation and the problem complexity discussion are in Section 4, and the test problems and their modeling are in Section 5. The algorithm and an illustrative example are in Section 6. The experimental results are in Section 7, and the conclusions are in Section 8.

2 BACKGROUND

Technology development has made it possible to design a chip where a complete system is placed on a single die, so called system chip or System-on-Chip (SOC). These system chips have to be tested to ensure fault-free operation. The growing complexity of such chips, device size miniaturization, increasing transistor count, and high clock frequencies have led to the dramatic increase of the number of possible fault sites and the fault types, and therefore a high test data volume is needed for high-quality testing. The high test data volume leads to long testing times, and therefore the planning and organization of the testing become a challenge that has to be tackled.

EDA (Electronic Design Automation) tools are developed to reduce the design productivity gap, *i.e.* the gap between what technology allows to be designed, and what a design team can produce within a reasonable time. A way to handle the increasing complexity of systems is to model the systems at higher abstraction levels. However, modeling at higher abstraction levels means that less implementation specific details are visible. The problem is that device size miniaturization has made implementation specific details highly important. The core-based design approach is therefore proposed to design complex systems, in reasonable time, and at the same time handle implementation specific details [191, 84]. The idea is that pre-designed and pre-verified blocks of logic, so called cores, are integrated by the *core integrator* to a SOC. The cores, provided by *core vendors*, may each have different origin, such as from various companies, reuse from previous designs, or the cores can be completely new in-house designs. The *test designer* is then responsible for the design of the system's test solution, which includes decision on the organization and the application of test data (test stimulus and test responses) for each core in the system. Test application time minimization is often one of the main objective since it is highly related to the cost of test, but the added over-head, such as additional wiring is also important to minimize, while constraints and conflicts should be considered.

The design flow has traditionally been sequential where system design is followed by test design, and as a final step the chip is produced and tested. A core-based SOC design methodology consists of two major steps: a core selection step where the core integrator selects the appropriate cores for the system, and the core test design step where the test solution for the system is created, which includes test scheduling and the design of the infrastructure for test data transportation, the TAM (Test Access Mechanism). These two steps are traditionally performed in sequence, one after the other (see Figure 212(a)). For such a SOC design flow, it is important to note that, the core integrator can, in the initial design step (core selection), select among several different cores often from several core vendors to implement a certain functionality in the system. The core integrator selects, based on each core's design characteristics given in its specification, the cores that fits the system best. Each possible core may not only have different design characteristics, but may also have different test characteristics (for instance test sets and power consumption). For example, one core may require a large ATE (Automatic Test Equipment) stored test set, while another core, implementing the same functionality, requires a combination of a limited ATE test set and a BIST (Built-In Self-Test) test set. The decision on which core to select has therefore an impact on the global test solution. Selecting the optimal core only based on its functionality will lead to local optimum, which is not necessarily the global optimum when the total cost of the system including test cost is considered. In other words, the selection (of cores and/or test partitioning) must be considered with a system perspective in order to find a globally optimized solution. This characteristics means that there is need for a test solution design tool that can be used in the early core selection process to explore and optimize the system's test solution (see Figure 212(b)). Such a tool could help the test designer to answer the following question from a core integrator "For this SOC-design, which out of these cores is most suitable for the systems test solution?".

We have previously proposed a technique for integrated test scheduling and TAM design where a weighted cost-function based on test time and TAM wiring cost is minimized while considering test conflicts and test power consumption [176]. We then assumed that the tests for each testable unit was fixed, and the main objective was, for a given system, to define a test solution. In this chapter, on the other hand, we assume that for each testable unit several alternatives may exist. We propose a technique for integrated *core selection*, *test set selection*, *test resource floor-planning*, *TAM design,* and *test scheduling.* The core selection, test set selection, test resource floor-planning, TAM design, and test scheduling are highly interdependent. The test time can be minimized by scheduling the tests as concurrent as possible, however, the

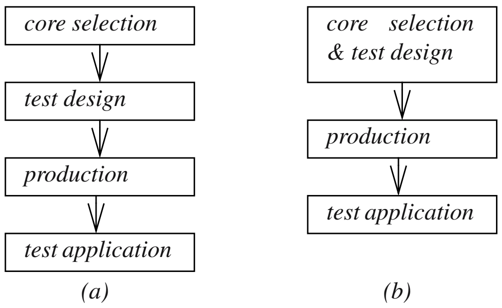


Figure 212. Design flow in a core-based design environment (a) traditional and (b) proposed.

possibility of concurrent testing depends on the size of the TAM connecting the test resources (test sources and test sinks). The placement of the test resources has a direct impact on the length of the TAM wires. And finally, the selected test sets for each testable unit are partitioned over the test resources and impacts the TAM design and the test schedule. Therefore, these problems must be considered in an integrated manner.

Test power consumption is becoming a severe problem. In order to reduce testing times, concurrent execution of test is explored. However, this may lead to too high power consumption. The proposed technique includes an improved power model that consider (1) global system-level limitations, (2) local limitations on power-grid level (hot spots), as well as (3) core-level limitations. The motivation for this more elaborate power model is that the system is designed to operate in normal mode, however, during testing mode the testable units are activated in a way that would not usually occur during normal operation. It can lead to that (1) the systems power budget is exceeded, or (2) hot spots appear and damage a certain part in the system, or (3) a core is activated in such a way that the core is damaged.

The proposed technique can be used to explore alternative cores for a SOC, different test alternatives for each testable unit, as well as the placement of test resources. As the design alternatives increases, we make use of Gantt charts to limit the search space. We have implemented the proposed technique, an estimation-based technique and a pseudo-exhaustive technique. The experiments using the estimation-based technique shows that it is difficult to produce high-quality solutions and the experiments with the pseudo-exhaustive technique demonstrates that the search space is enormous. The proposed technique, on the other hand, produces solutions with a cost close to the pseudo-exhaustive technique but at a computational cost close to the estimation-based technique.

3 RELATED WORK

The technology development has, as discussed above, enforced the introduction of the core-based design environment [84]. Blocks of reusable logic blocks, so called cores, are combined to a system that is placed on a single die. A core-based design flow is typically a sequential sequence that starts with core selection, followed by test solution design, and after production, the system is tested (Figure 212(a)). In the core selection stage, the core integrator selects appropriate cores to implement the intended functionality of the system. For each function there is usually a number of possible cores to select from, and where each candidate core has its specification. The specification include, for instance, performance, power consumption, area, and test characteristics. The core integrator explores the design space (search and combines cores) in order to optimize the performance of the SOC. Once the system is fixed (the cores are selected) the core test designer designs the TAM and schedules the tests based on the test specification for each core. In such a design flow (illustrated in Figure 212(a)), the test solution design is a consecutive step to core selection. It means that even if each core's specification is highly optimized, when integrated as a system, the system's global test solution is not highly optimized.

A design flow such as the one in Figure 212(b), on the other hand, integrates the core selection step and the test solution design step. The advantage is that it makes it possible to consider the impact of core selection when designing the test solution. In such a design flow (Figure 1(b)), the global system impact on core selection is considered, and the advantage is that it is possible to develop a more optimized test solution. The design flow in Figure 1(b) can be viewed as in Figure 213, where the core type is floorplanned in the system but there is not yet a design decision on which core to select. For each position, several cores are possible. For instance, for the cpu_x core there are in Figure 213 three alternative processor cores (cpu1, cpu2 and cpu3).

In this chapter we make use of the concepts introduced by Zorian *et al.* [191], which are illustrated with an example in Figure 214. The example con-

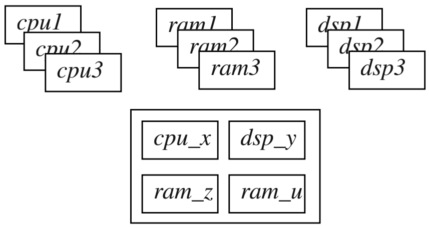


*Figure 213.*System design with different alternatives.

sists of three main blocks of logic, core A (CPU core), core B (DSP core), and core C (UDL (user-defined logic) block). A test source is where test stimulus is created or stored, and a test sink is where the test response for a test is stored or analyzed. The test resources (test source and test sink) can be placed on-chip or off-chip. In Figure 214 the ATE serves as an off-chip test source and off-chip test sink, while TS1, for instance, is an on-chip test source. The TAM is the infrastructure (1) for test stimulus transportation from a test source to the testable unit, and (2) for test response transportation from a testable unit to a test sink. A wrapper is the interface between a core and the TAM, and a core with a wrapper is *wrapped* while a core without wrapper is *unwrapped*. Core A is a wrapped core while Core C is unwrapped. The wrapper cells at each wrapper can be in one of the following modes at a time: *internal mode*, *external mode* and *normal operation mode*. In addition to the definitions by Zorian *et al*. [191], we assume that a testable unit is not a core, but a block at a core and that a core can consist of a set of blocks. For example, core A (Figure 214) consists of two blocks (A.1 and A.2).

For a fixed system where cores are selected and floor-planned, and for each testable unit the tests are fixed, the main tasks are to organize the testing and the transportation of test stimuli and test responses (as the example design in Figure 214). Several techniques have been proposed to solve different important problems under the assumption that the cores are already selected (design flow as in Figure 212(a)).

Zorian [287] proposed a test scheduling technique for fully BISTed system where each testable unit is tested by one test with a fixed test time, and

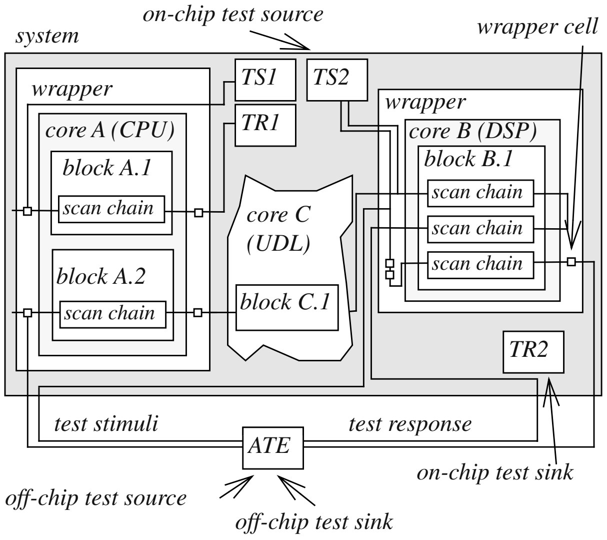


*Figure 214.*A system and the illustration of some test concepts.

each testable unit has its dedicated on-chip test source and its dedicated on-chip test sink. A fixed test power value is attached to each test and the aim is to organize the tests into sessions in such a way that the summation of the power consumed in a session is not above the system's power budget while minimizing the test application time. In a system where the testable units share test source and test sink, the test conflicts must be taken into account. Chou *et al.* proposed a test scheduling technique that minimizes the test time for systems where both the test time and power consumption for each test are fixed, and to handle general conflicts a conflict graph is used [41].

The approaches by Zorian and Chou *et al.* assume fixed testing times for each testable unit. The test time for a core can be fixed by the core provider. It can be due to that the core providers have optimized their cores in order to protect the IP-blocks, for instance. However, the test time at a core is not always fixed. For scan-tested cores, the scanned elements can be connected to any number of wrapper chains. If the scanned elements (scan-chains, inputs, and outputs) at a core are connected to a small number of wrapper chains, the testing time is higher compared to if the scan elements are connected into a higher number of wrapper chains. Iyengar *et al.* proposed a scheduling technique for systems where the testing time for all cores is flexible and the objective is to form a set of wrapper chains for each core in such a way that the testing time for the system is minimized [113].

In order to minimize the test times as many fault locations as possible are activated concurrently, which leads to high power consumption. A high-level of power can also be consumed at certain so called hot-spots. Zorian [287] and Chou *et al.* [41] assign a fixed power value to each test and make sure that the scheduling does not activate the tests in such a way that the system's power budget is exceeded at any time. Bonhomme *et al.* [18] and Saxena *et al.* [241] proposed clock-gating schemes intended to reduce the test power consumed during the scan-shift process. The advantage is that the test power can be reduced at a core with such a scheme, and hence a higher number of cores can be scheduled concurrently. The basic idea is if a wrapper chain consists of *n* scan-chains, the scan-chains can be loaded one at a time, which means that only one chain is active at a time, hence, lower power consumption.

There has been research on finding the most suitable ATE/BIST partition for each testable unit. Sugihara *et al.* investigated the partitioning of test sets where one part is on-chip test (BIST) and the other part is off-chip test using an ATE (Automatic Test Equipment) [257]. A similar approach was proposed by Jervan *et al.* [126], which later was extended to not only locally optimize the test set for a core but to consider the complete system by using an estimation technique to reduce the test generation complexity [127].

Hetherington *et al.* discussed several important test limitations such as ATE bandwidth and memory limitations [92]. These problems as well as the problems described above are important to include and consider in the search of a final test solution for the system.

The above addressed problems are all individually important to consider when designing the test solution for an SOC, however, it is important to consider them simultaneously from a system test perspective. We have previously proposed an integrated technique for test scheduling and TAM design where the test application time and the TAM design are minimized while considering test conflicts and power consumption [176]. The technique handles unwrapped as well as wrapped cores, and also cores with a fixed testing time as well as cores with a flexible testing time. The technique is also general in the test source and test sink usage. Each test can be defined to use any test source and any test sink. It is not needed that a test uses a test source and a test sink where both are placed on-chip, or both are placed off-chip. Furthermore, the technique allows an arbitrary number of tests per testable unit, which is important to handle testing such as timing faults and delay faults and not only stuck-at faults. However, the technique assumes that the tests for each testable unit are fixed and defined.

4 PROBLEM FORMULATION

The problem we address in this chapter can be illustrated with Figure 215 where a SOC with its floor-plan is given as the rectangle. Note, the core types are defined but the particular core is not yet selected. For example, at position CpuX, coreA and coreB are the alternatives. Each of the alternative cores may consist of a set of blocks (testable unit) where each block has multiple test alternatives. For instance, blockA1 at coreA can be tested by test t_1 or by t_2 and t_3. Each test is attached to one block and each test can have its combination of test source and test sink. For instance, t_1 makes use of r_1 and s_1. Since no other test in the system makes use of r_1 and s_1, r_1 and s_1 will most likely not limit the test time. On the other hand, since s_1 and r_1 are not used by any other test, the added TAM has a low utilization, which leads to the waste of resources.

An example of an input specification, the starting point in our approach, is given in Figure 216. The structure of the input specification is based on the specification we made use of in [176]. The major extension are (1) that for each block (testable unit) several lists of tests can be specified, instead of as before where it was only possible to assign one list per block, and (2) the improved power-grid model.

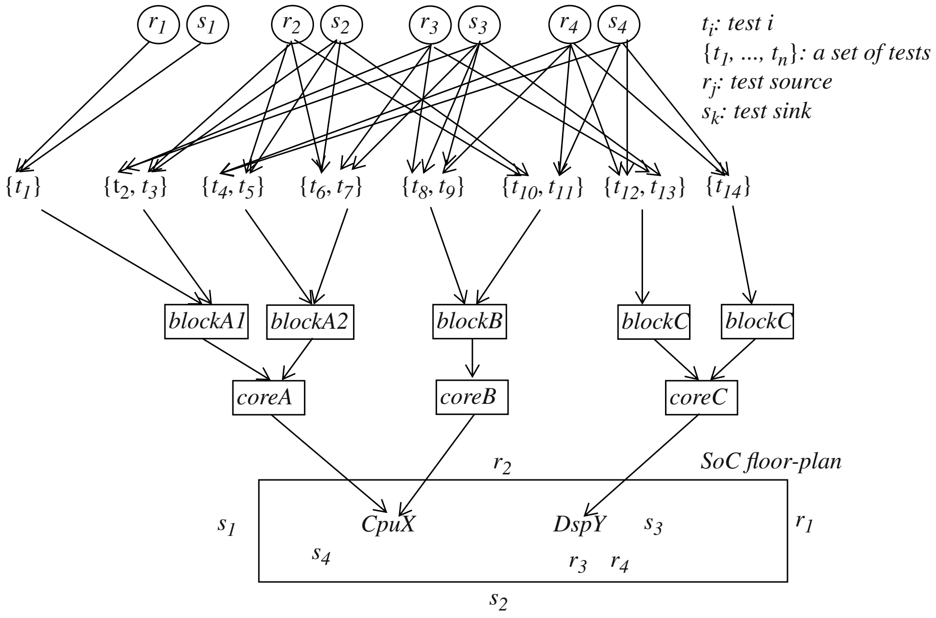


*Figure 215.*Illustration of design alternatives.

The advantage of the possibility to specify several lists of tests for each block (testable unit) where each test in a list make use of its specified resources (test source and test sink), and each test has its test characteristics, is that it makes it possible to explore different design alternatives (as illustrated in Figure 215).

The test problems that are considered in our technique and their modeling are discussed in Section 5. The cores are floor-planned, *i.e.* given (x,y) coordinates and each core consists of a set of blocks (testable units):

```
[Cores] name x   y   {list for Core1} {list for Core2}
        CpuX 10  20  {BlockA1 BlockA2} {BlockB1}
```

For each block, several sets of tests are available, where each set of tests is sufficient for the testing the block. For instance, to test a block bA three possible test sets are given:

```
[Blocks] name idle_pwr pwr_grid {test1, t2,..., tn} {t1,..tn}
         bA   0        grid1   {tA1, tA2} {tB1} {tC1 tC2 tC3}
```

{tA1, tA2} or {tB1} or {tC1 tC2 tC3} should be selected where each test has its resources and characteristics.

The problem is to select, cores and corresponding blocks and for each block, which set of tests to use in order to produce an optimized test solution

```
[Global Options]
MaxPower = 100
[Power Grid] #name      power_limit
             p_grid1    50
             p_grid2    60
[Cores]   #name    x    y    block_list
          coreA    20   10   { blockA1, blockA2 }
          coreB    40   10   { blockB1, blockB2 }
          coreC              {  blockC1 }
[Generators] #name   x    y    max_bw memory
             ATE     10   0    4          200
             TG1     30   0    1          50
             // the rest of the generators
             TG2     30   10   1          100
[Evaluators] #name   x    y    max_bw
             ATE     50   0    4
             TRE1    30   0    1
             // the rest of the evaluators
             TRE2    30   10   1
[Tests]   #name pwr time tpg    tre    min_bw  max_bw  ict
          tA1.1  60   60   TG1   TE1     1        1      no
          // more tests for coreA
          tB1.1  60   72   TG1   TE1     1        1      no
          // more tests for coreB
          tC1.1  70   80   TG1   TE2     1        4    coreB
          // more tests for coreC
[Blocks]  #name    idle_pwr pwr_grid   test_sets {}, {}, ...,{}
          blkA1      0       p_grd1   { tA1.1 }{ tA1.2, tA1.3}
          blkA2      0       p_grd1   { tA2.1 }{ tA2.2 }
          blkB1      5       p_grd2   { tB1.1 tB1.2 } { tB1.3 }
          blkB2      10      p_grd2   { tB2.1 }
          blkC1      0       p_grd1   { tC1.1}
[Constraint] #name   {block1, block2, ... , blockn}
          tA1.1      {}
          // constraints for each test
          tC1.1      {blkC1 blkA1 blkA2 blkB1 blkB2}
```

*Figure 216.*Input specification for the example system in Figure 214.

for the system. The cost of a test solution is given by the test application time and the amount of routed TAM wires:

$$cost = \alpha \times \tau_{\text{total}} + \beta \times \text{TAM} \qquad (13.1)$$

where τ_{total} is the total test application time (end time of the test with highest test time), *TAM* is the routing length of all TAM wires, and, α and β are two user-defined constants used to determine the importance of test time in relation to TAM cost. The user-defined constants α and β depends on each particular design, and it is therefore not possible to define universal values on α and β. The selection of a and b is based on the characteristics of the particular SOC.

The produced output from our technique is a test schedule where the cores are selected and for each block (testable unit) at the selected cores, the tests are selected and given a start time, and an end time in such a way that all conflicts and constraints are not violated, and a corresponding TAM layout where the cost (Eq. (13.1)) is minimized.

4.1 Problem Complexity

The number of design options when developing the test resource specification can be high. Assume that each block (testable unit) $b_{ij} \in B$ at a core c_i has $|T_{ij}|$ possible combinations of test sets where each test set can be placed at n_{ij} positions and each test set can be modified in m_{ij} ways where a high number of TAM wires reduces the test time and vice versa. The number of possibilities are:

$$\prod_{i=1, j=1}^{|B|} |T_{ij}| \times n_{ij} \times m_{ij}$$

For a small system consisting of only two cores; each with two test sets where each test set have two possible placements and each test set can be modified in two ways, the number of design alternatives are: $(2\times2\times2)^2=64$.

5 TEST PROBLEMS AND THEIR MODELING

In this section we discuss the test problems that have to be considered when developing a SOC test solution and the modeling of the problems.

5.1 Test Time

The testing time for a testable unit can be *fixed* or *non-fixed* prior to the design of the test solution. A core provider might protect the core and therefore optimize the core and its core wrapper prior to delivery, hence, having the testing time fixed. On the other hand, the testing time for a scan-tested core can also be non-fixed since the scanned elements (scan-chains and wrapper cells) can be connected into one or more wrapper-chains. The testing time for a test with flexible test time depends on the number of wrapper-chains. Important to note is that tests with fixed and non-fixed testing times can be mixed in the system. The test time model must handle systems where some cores have *fixed* test time while other cores have *non-fixed* testing time.

A higher number of wrapper-chains at a core results in lower testing time compared to if fewer wrapper-chains are used. The scan-chains at a core can be few and unbalanced (of unequal length), and the testing time might not linearly dependent on the number of wrapper chains. Therefore, we analyzed the linearity of the testing time (τ) versus the number of wrapper-chains (w) ($\tau \times w$ = *constant*) for the scan-tested cores in one of the largest ITC'02 designs, namely the P93791 design [176]. We observed that the testing time for core 11 was most non-linear (Core 11 - original in Figure 217). We noted that the 576 scanned elements were partitioned into 11 scan-chains (82 82 82 81 81 81 18 18 17 17 17). We re-designed core 11 into four new cores with 11, 22, 44, and 88 balanced scan-chains, respectively. We plotted $\tau \times w$ for all cores in Figure 217. As the number of scan-chains increases, the value $\tau \times w$ becomes more or less constant. The testing time at a single wrapper chain is 149381 (marked in Figure 217). For core 11 with 44 balanced scan-chains, the value $\tau \times w$ is always less than 5% from the constant theoretical value. Important to note is that for all cores, the value $\tau \times w$ is almost constant within a certain range. We assume that the core test designer optimizes the cores, hence, the number of scan-chains at a core is relatively high and of nearly equal length.

In our model [176], we specify the testing time for a testable unit at a single TAM wire and the bandwidth limitations. For instance a testA has a test time of 100 at a single wrapper chain and where the scanned elements can be arranged into wrapper-chains in the range 1 to 4:

```
[Tests] name   test time  minbw     maxbw
     testA    100        1       4
```

We assume, based on our experiments, that the test time is linear to the number of TAM wires within the bandwidth range. It means that given the test time at a single TAM wire (τ_1), the test time t_i can be computed by:

$$\tau_i = \frac{\tau_1}{i} \tag{13.2}$$

where i is in the range [*minbw, maxbw*]. If the testing time is fixed, *minbw* = *maxbw*.

5.2 Test Power Consumption

The testing time is reduced if a high number of cores are activated concurrently, but it leads to higher activity, and high power consumption can damage the system. The system-level power budget can be exceeded in such a situation. Furthermore, if cores that are physically close are activated during testing, a hot-spot can be created and damage the system. For instance, assume a memory organized as a bank of four where in normal operation only one bank is activated at a time. However, during testing, in order to shorten the test time, all banks are activated concurrently. The system's power limit might not be exceeded, however, a local hot spot is created, and the system may be damaged. Another problem is that a core during testing mode dissi-

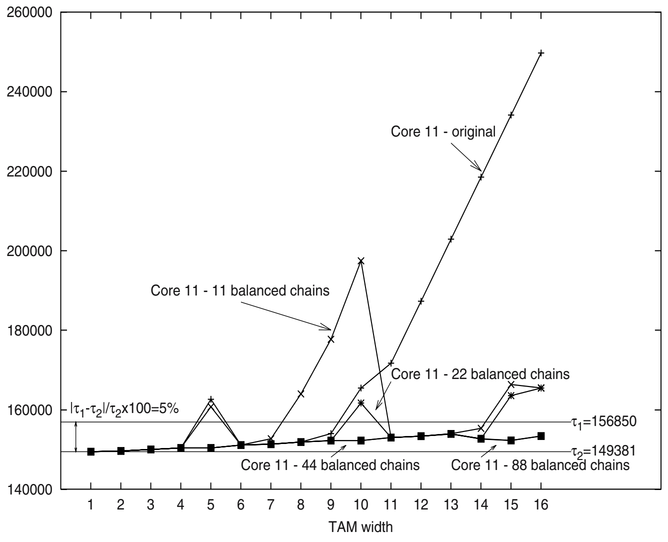


*Figure 217.*Test time analysis for core 11 in design P93791.

pates power above its specified limit due to the nature of the test stimuli and/ or the test clock frequency. We therefore make use of a three-level power model: *system-level*, *power-grid-level (local hot spot)*, and *core-level*.

For the system-level, we make use of the power model defined by Chou *et al*. where a fixed power value is attached to each test, and the tests are scheduled in such a way that at any time point, the summation of power values executed concurrently is below the power budget of the system [41].

As an example, we can specify the system budget as:

```
MaxPower = 100
```

and for each test we specify the power consumed when the test is activated:

```
[Tests] name  pwr time tpg tre minbw maxbw mem      ict
        testA 60  60   r1  s1     1    1     10       no
```

Additionally, the idle power, the power consumed when a block is not active is also specified at each block:

```
[Blocks] name idle pwr pwr_grid {t1, t2,...,tn} {t1,..tn}
          bA     0     grid1    {testA} {testA2}
```

For local hot spots, we introduce a power grid model, which has similarities to the approach proposed by Chou *et al.* [41], but in addition to it, our model also include local areas (power grids). We assume that each block (testable unit) is assigned to a power grid where the power grid has its power budget. The system can contain a number of power grids. Blocks assigned to a power grid cannot be activated in such a way that the power grid budget is exceeded at any time.

An example to illustrate the need of power grids is as follows, a memory can be organized as a bank of memory blocks (see Figure 218). Assume that the memory, during normal operation, never accesses more than a single memory block the power grid is designed accordingly.

As an example of a single grid is:

```
[PowerGrid] pwr_grid         limit
            grid1            30
```

and for each block the used power grid is given:

```
[Blocks] name idle_pwr pwr_grid {test1, t2,..., tn} {t1,..tn}
         bA   0        grid1    {testA} {testA2}
```

The motivation behind core-level adjustments is two-fold. First, by lowering the power consumption at a core, a higher number of cores can be activated concurrently without violating the system power budget. Second, since test power consumption often is higher than that during the normal operation, the power dissipation during test at a specific core can be higher than its own power budget, which can damage the core.

As discussed above, some tests have a fixed testing time while other tests allow flexible testing times. Regarding test power consumption, we have some tests where the power is fixed regardless of the number of assigned TAM wires, while other tests allow the power to be adjusted by clock-gating, for instance [241]. Clock-gating can be used to reduce the power consumption so that a higher number of tests can be executed concurrently, but more importantly, it can be use for testable units where its own power dissipation is higher than its allowed power consumption due to for instance a too high test clock frequency.

The power consumption for a test is given as a single value, for instance as in the following example:

```
[Tests] name    pwr   time    minbw    maxbw   flexible_pwr
        testA   50    60      1          4        yes
        testB   60    30      1          4        no
```

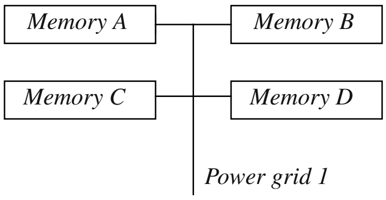


Figure 218. A memory organized as a bank of four blocks powered by a common grid.

Note that we include the possibility to specify if clock-gating can be used by setting *flexible_pwr* to yes or no. If power can be modified, we assume a linear dependency:

$$p_i = p_1 \times tam \tag{13.3}$$

where p_1 is the power at a single TAM wire, p_i is the power consumed when i number of TAM wires are used; i has to be in the specified range [*minbw*:*maxbw*].

5.3 Test Conflicts

During the test solution design there are a number of conflicts that have to be considered and modeled. Each test has its defined constraints depending on the type of test; stuck-at, functional, delay, timing, etc. For general conflicts we make use of the following notation [176]:

```
[Constraints] test  {block1, block2, ..., block n}
              tA    {bA bB}
```

The notation means that when testing tA both block bA and bB must be available since they are used by test tA or tA might interfere with one of the blocks. This modeling support general conflicts, which can be due to hierarchy where cores are embedded in cores or interference during testing. The model can also be used for designs where an existing functional bus is used as the TAM media. A functional bus can be modeled as a dummy blocks, where usually only one test can be active at a time.

A test source ([Generators]) may have limited bandwidth and memory. The bandwidth limitation and the memory limitation are especially critical for ATEs but are important if on-chip resources such as memories are used for test data storage. We model bandwidth limitation as an integer stating the highest allowed bandwidth for the test source. For memory limitations an integer is used as the maximal memory limit. A test sink ([Evaluators]) can also have a limited bandwidth and in a similar way as with test sources, we model it also as an integer. For each test we give a number of its memory requirement. An example with testA using test source r1 and test sink s1 [176] is given below:

```
[Generators]  name    x     y     maxbw   memory
              r1      10    20    1       100
[Evaluators]  name    x     y     maxbw
              s1      20    20    2
[Tests]     name      tg    tre   memory
            testA     r1    s1    10
```

The wrapper conflicts are slightly different compared to general conflicts because of the TAM routing. The testing of a wrapped core is different from the testing of an unwrapped one. The testing of the wrapped core A

```
Select tests for initial solution
Do {
    Create test schedule and TAM
    Find limiting resource with Gantt chart
    Modify tests (select alternative tests or modify
    TAM width) at limiting resource
    Select best modification
} Until no improvement is found
Return best solution.
```

Figure 219. The algorithm.

(Figure 214), for example, is performed by placing the wrapper in *internal test mode* and test stimuli are transported from the required test source using a set of TAM wires to the core and the test responses are transported from the core using a set of TAM wires to the test sink. In the case of an unwrapped testable unit such as the UDL block, the wrappers at core A and B are placed in *external test mode*. The test stimuli are transported from the required test source on the TAM via core A to the UDL block and the test responses are transported via core B to the TAM and to the test sink.

We model the wrapper conflict as in the following example with two blocks (bA and bB) and one test per block (tA and tB):

```
[Blocks] name  {test1, test2,..., test m} {test1,..., test n}
          bA       {tA}
          bB       {tB}
[Tests] name     tg     tre      ict
        tA         r1     s1       bB
        tB         r1     s1       no
```

Test tB is not an interconnection test, hence, ict (interconnection test) is marked as no. It means that there will be a connection between r1 to bB and from bB to s1. Test tA, on the other hand, is an interconnection test with bB. It means that r1 is connected to bA and bB is connected to s1.

6 TEST DESIGN ALGORITHM

In this section we describe the proposed test design algorithm (outlined in Figure 219, and detailed in Figure 221 and Figure 220). In order to evaluate the cost of a test solution, we make use of (Eq. 13.1). At a design modifications, the cost change before and after modification, is given by:

$$\Delta\tau \times \alpha + \Delta \mathrm{TAM} \times \beta \qquad (13.4)$$

where $\Delta\tau$ (ΔTAM) is the difference in test time (TAM cost), before and after the modification.

The *TAM* cost is given by the length l and its width w ($TAM=l\times w$), and by combining the cost function (Eq. (13.1)) considering only one testable unit,

```
sort the list of tests descending according to the cost
function.
repeat until the list is empty {
    select the first test in the list
    repeat until a test is scheduled or at end of list {
      repeat until selected test is scheduled or
      bandwidth cannot be decreased {
        try to schedule the test at current time
        if fail to schedule {
          if the bandwidth>1 then reduce bandwidth with 1
        }
      }
      if the selected test could not be scheduled {
        select the following test in the list
      }
    }
    if the test was scheduled {
      allocate TAM and remove the selected from the list
    } else {
        update current time to the nearest time in the
        future where possible to schedule the first
        test in the list
      }
}
```

*Figure 220.*Test scheduling and TAM design algorithm.

and the test time versus TAM cost (Eq. (13.2)), the optimal TAM bandwidth is given by [176]:

$$w = \sqrt{(\alpha \times \tau)/(\beta \times 1)} \qquad (13.5)$$

A detailed description of the algorithm (Figure 219) is in Figure 221 (test set selection outline) and Figure 220 (test scheduling and TAM design). The algorithm starts by the part given in Figure 221, where the list of test sets for each testable unit is sorted based on the cost function (Eq. (13.1)). The cost for each testable unit is locally optimized, however, there is at this point no global consideration on sharing of TAM wires or conflicts. For each testable unit, the first set of tests for each testable unit is selected, and the set is scheduled and the TAM is designed (Figure 220). From the test schedule, the test application time is given and from the TAM layout, the TAM cost for the solution is given. The algorithm checks the use of resources from a Gantt-chart for the solution (explained below in Section 6.1). For example, assume the a test solution generates a Gantt-chart as in Figure 222, where TG:r1 is the critical resource. For all tests that are using the critical (limiting) resource, we try to find alternative tests. We make use of Eq. (13.4) to evaluate the change in cost for each possible alternative (at the critical resource). Instead of trying all possible alternatives, we try a limited number of design modifications (given from the Gantt chart). And to reduce the TAM cost we try to make use of existing TAMs (a test can be delayed and applied later).

```
for each block (testable unit) {
    for each test set at a block {
      compute optimal bandwidth for each test (Eq. (13.5));
      compute cost for the full test set (Eq. (13.1));
    }
    place test sets sorted descending on cost (step (13.4));
    select first test set in the list as the active test set
}
repeat until no modification can be performed {
    create test schedule and TAM layout (see Figure 220)
    if the cost for schedule and TAM layout is best so far{
      remember this test schedule and the TAM layout
    }else {
      if last modification was a TAM width modification {
        undo the TAM width modification
      }
      if last modification was a test set modification {
        remove the test set from the blocks list of test sets
      }
    }
    for each block {
      if the active test set has a test resource limiting
      the solution {
        compute cost for increasing the TAM width with 1
        for every other test set for the block{
          compute the cost of changing this test set
          based on Eq. (13.4)
          if the cost is lower than lowest cost {
              remember this test set
          }
      }
      if lowest cost for the block < the total cost{
        remember block, TAM width and test set modification
      }
    }
    if any alternatives exists {
      perform TAM width modification or
      test set modification
    } else {
      for each block {
        for each test set after the active test set for the
        block{
          compute the cost of selecting it (Eq. (13.4))
          if cost is lowest then remember this test set
        }
        if best cost for the block < lowest total cost then {
            remember block change and test set change
        }
      }
      if lowest cost difference <0 {
        do the test set change
      }
    }
}
```

*Figure 221.*Test set selection algorithm.

6.1 Resource Utilization

We make use of a machine-oriented Gantt chart to track bottlenecks (the resource that limits the solution) [23]. We let the resources be the machines, and the tests be the jobs to show the allocation of jobs on machines. For example, a Gantt chart is given in Figure 222 where test B2 needs TG:r2 and TRE:s2. An inspection of Figure 222 shows that TG:r2 and TRE:s2 are not critical to the solution. On the other hand, test source TG:r1 is the most critical one. It means that testA, testB1, and testC are the obvious candidates for modification. The Gantt chart pin-points bottlenecks and therefore reduces the search for candidates for modification. Note that the Gantt chart does not show a valid schedule, only the usage of resources in the system.

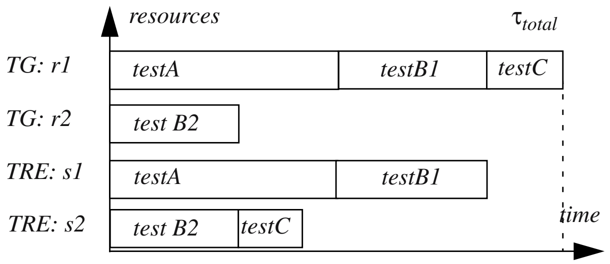


Figure 222. A machine-oriented Gantt chart [23].

6.2 Example

We use the design example in Figure 223 to illustrate the algorithm described above. The example (Figure 223), simplified by removing power grids, memory limitations and the list of general constraints, consists of two cores each with a single block (testable unit). Each block can be tested in two ways; there are two alternative test sets for each block. For instance, blockA can be tested by testA1 or by testA2. Each of the tests can be defined with its test time, combination of test sink and test source etc.

The algorithm proceeds as follows. Initial step: For each block, the test sets are ordered ascending according to the cost function (Eq. (13.1) assuming $\alpha=\beta=1$):

```
test    time    TAM     total cost
testA1: 60      40      100
testA2: 100     20      120
testB1: 72      40      112
testB2: 120     20      140
```

The evaluation results in the following sorted lists per block (first in the list is the best candidate):

```
BlockA: {{testA1}, {testA2}}
BlockB: {{testB1}, {testB2}}
```

The first set of tests are selected as active, that is for BlockA {testA1} and for BlockB {testB1}. The test scheduling algorithm sorts the tests based on test time, and starts with the longest test, making the test schedule: testB starting at time 0 followed by testA starting at time 72. The resulting total test application time is 132. The TAM design algorithm connects TG1, coreB, coreA, and TA1, and the Manhattan length is 20+20+20=60. The total cost (at α=β=1) for the test solution is then: 132 (test time)+60 (TAM cost)=192.

From the Gantt chart for this test solution, we observe that TG1 and TA1 both are used for 132 time units, while TG2 and TA2 are not used at all, and we note that TG1 and TA1 limits the solution. Based on the Gantt-chart, the algorithm tries to find an alternative that is not using TG1 and TA1. For each test that uses the limiting resources in the Gantt chart, in our example TG1 and TA1, the algorithm computes the alternative cost of using other resources. It is important to note, that in order to limit the number of possible options, we only try with the tests that depend on the resources critical to the solution (Gantt chart).

As the first alternative modification, we try using testA2 to test BlockA instead of using testA1. It means that testA1 will not be executed (one of the set of tests for each block is only required). We evaluate the impact of the test modification on the TAM layout, and we observe that we do not have to include coreA in the layout. Taking coreA out of the bus layout means that TAM corresponding to 20 units can be removed (testA2 is making use of different test resources compared to testA1). However, in order to execute testA2 we have to include wires from TG2 to coreA and from coreA to TA2. The additional required wiring corresponds to 20 units.

The difference in test time between testA1 and testA2 is (100-60=) 40. It means that the total cost difference is estimated to: -20 (gain by not including coreA for testA1)+20 (what we have to add to include TAM for TG2->coreA->TA2)+40=40.

For the second alternative modification, we try testB2 instead of testB1. It means that a TAM (length and width) corresponding to 20 units can be removed. The additional TAM cost of adding testB2 (its resources) is 20, and the difference in test time between testB2 and testB1 is 48 (120-72). The cost difference for this alternative is -20+20+48=48.

In this example we have two tests using the resources that are critical to the solution (Gantt chart), and we also had only one possible alternative per test. And, since the first alternative is better than the second, the first one is selected. A new test schedule and a TAM layout is created where both testA1 and testB1 are scheduled to start at time 0, and there are two TAMs, one from TG2->coreA->TA2 at length 20, and one from TG1->coreB->TA1 at length 40. The total cost is 60+72=132 (an improvement from 192 to 132).

7 EXPERIMENTAL RESULTS

The objective with the experiments is to check that the proposed technique produces a high quality solutions at a reasonable computational cost (CPU time). For comparision purpose, we have also implemented an estimation-based technique [170] and a pseudo-exhaustive algorithm. The estimation-based technique, that tries to predict the cost at a low computational cost, is used to demonstrate that finding a high quality test solution is not trivial, and the pseudo-exhaustive algorithm, that basically tries all possible solutions, is used to demonstrate that the search space is enormous in size.

We have created three designs, in increasing size: *design_small*, *design_medium* and *design_large*. Design_small contains 4 cores each with one testable unit and for each testable unit there are two alternative tests, corresponding to two different core alternatives. Design_medium contains 13 cores also with one testable unit per core and for each testable unit there are 5

```
[Global Options]
MaxPower = 100
[Cores]  #name    x    y    block_list
         coreA    20   10   { blockA }
         coreB    40   10   { blockB }
[Generators] #name    x    y    max_bw
             TG1      30   0    1
             TG2      30   10   1
[Evaluators] #name    x    y    max_bw
             TA1      30   0    1
             TA2      30   10   1
[Tests]  #name pwr  time TPG  TRE    min_bw   max_bw ict
         testA1 60   60   TG1  TA1    1        1      no
         testB1 60   72   TG1  TA1    1        1      no
         testA2 40   100  TG1  TA1    1        1      no
         testB2 40   120  TG1  TA1    1        1      no
[Blocks] #name    idle_power   test_sets
         bA       0            { testA1 }{ testA2 }
         bB       0            { testB1 }{ testB2 }
```

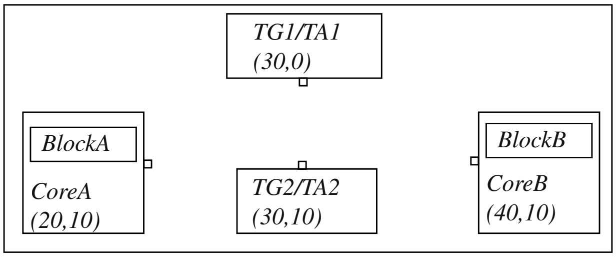


Figure 223. An illustrative example with a simplified input where power grids, memory limitations, and constraint list (general constraints) are not considered.

design alternatives. Design_large consists of 122 testable units distributed over 18 cores and 186 tests.

The experimental results with the three techniques on the three designs are collected in Table 35, Table 36, Table 37, and Table 38 where $\alpha=\beta=1$. Table 35 reports the computational cost (CPU time) for each of the three techniques. The estimation-based technique produces results very quickly (less than one second) while the pseudo-exhaustive approach does not terminate for the two larger designs, *i.e.* no results were produced within reasonable time. The proposed technique used CPU time that is in between the estimation-based and the pseudo-exhaustive approach, and results were produced for all designs at acceptable CPU times. For the largest design the CPU time was 4 seconds.

For the quality of the solutions we have collected the test application time, the TAM cost, and the total cost of the test solution for each of the three techniques at each of the three designs (reported in Table 36, Table 37, Table 38, respectively). The test application time for design_small of the solution produced by the estimation-based technique is 400, for the solution produced by both the pseudo-exhaustive technique and the technique two the test time is 320 (Table 36). The solution from the estimation-based technique is 25% worse than the solutions from the pseudo-exhaustive and the proposed technique. The experiment indicates that the proposed technique finds a solution of high quality (the same test time as the pseudo-exhaustive technique).

The TAM cost for the three techniques at the three designs are collected in Table 37. The proposed technique finds for design_small a test solution with the same TAM cost (120) as the pseudo-exhaustive technique. The results from estimation-based technique is 140, which 17% from the pseudo-exhaustive and the proposed techniques.

The total cost is computed with $\alpha=\beta=1$ making the total cost = test time + TAM cost (Eq. (13.1)). For instance, the total cost for the proposed technique at design_small is 440 and it comes from the summation of the test time 320 (Table 36) and the TAM cost 120 (Table 37). The proposed technique produces a solution at the same cost as the pseudo exhaustive technique for design_small, while the solution from the estimation-based technique is 23% worse. The proposed technique produces results for all three designs that are better than the results from the estimation-based technique.

Design	*Estimation* [170]	*Pseudo-exhaustive*	*Proposed technique*
Design_small	<1	<1	<1
Design_medium	<1	N.A	<1
Design_large	<1	N.A	4

Table 35. Computational cost (seconds).

Design	*Estimation* [170]	*Pseudo-exhaustive*	*Proposed technique*
Design_small	400	320	320
Design_medium	240	N.A	193
Design_large	215	N.A	220

Table 36. Test application time.

Design	*Estimation* [170]	*Pseudo-exhaustive*	*Proposed technique*
Design_small	140	120	120
Design_medium	810	N.A	690
Design_large	1072	N.A	962

Table 37. TAM routing cost.

Design	*Estimation* [170]	*Pseudo-exhaustive*	*Proposed technique*
Design_small	540	440	440
Design_medium	1050	N.A	883
Design_large	1287	N.A	1182

Table 38. Total cost (α=β=1).

8 CONCLUSIONS

Test design is traditionally considered as a final step in the system chip design flow, however, as testing is becoming a significant part in the design flow it is important to consider test design as early as possible in the design flow. The technology development has made it possible to design system chips that are shrinking in size but include an enormous number of transistors that are clocked at an immense clock frequency. The draw-back with such systems is that the number of fault sites increase drastically, and in order to test these system chips a high test data volume is required. It is therefore important to consider test planning as early as possible in the design flow. In this chapter we propose a technique where system test design is included as early as in the core selection phase. The advantage is that the technique makes it possible to explore the impact of test design taking a system's global perspective already when deciding on which cores to be used to implement the system. The proposed technique can be used to explore the impact of (1) the core selection on the test solution, (2) the test set partitioning (BIST size versus ATE size) on the test solution, and/or (3) the placement of test resources (test source and test sink) on the test solution.

Prior work assume a system where cores, tests and placement of test resources are already fixed when test planning is to be performed. It means that test scheduling and TAM design are the main problems. Our approach include the interdependent problems of test scheduling, TAM design, test set selection and test resource placement, together with core selection. Our technique defines a test solution where the test time and the TAM routing cost are minimized while test conflicts and power limitations are considered.

Test power consumption is becoming an important aspect to be considered, however, previously proposed power models have all been rather simplistic, and have only been focusing on the global power budget. We have improved test power modeling by introducing a three level power budget model: system-level, power-grid (local hot-spot) level, and core-level. The advantage is that with such a model it is possible to have more elaborate power constraints on where the power is consumed in the system, at cores, at certain hot-spot areas, and at the global level.

In this chapter we have proposed a technique for test solution design where core selection, test set selection, test resource placement and TAM design all are integrated. The design space is enormous when integrating these problems and in order to limit it, we make use of Gantt charts to find the limiting resources (bottlenecks). For validation of the proposed technique, we have implemented the proposed technique, an estimation-based technique and a pseudo-exhaustive technique. The experimental results show that the pseudo-exhaustive technique cannot produce solutions within a reasonable CPU time, and the estimation-based technique does not produce high-quality solutions. The proposed technique can, on the other hand, produce high-quality solutions at a reasonable CPU time.

Chapter 14

DEFECT-AWARE TEST SCHEDULING[1]

1 INTRODUCTION[2]

In this chapter we address the test scheduling problem for system-on-chip designs. Different from previous approaches where it is assumed that all tests will be performed until completion, we consider the cases where the test process will be terminated as soon as a defect is detected. This is common practice in production test of chips. The proposed technique takes into account the probability of defect-detection by a test in order to schedule the tests so that the expected total test time will be minimized. It supports different test bus structures, test scheduling strategies (sequential scheduling vs. concurrent scheduling), and test set assumptions (fixed test time vs. flexible test time). Several heuristic algorithms have been developed and experiments performed to demonstrate their efficiency.

The cost of developing electronic systems is increasing and a significant part of the cost is related to the testing of the systems. One efficient way to reduce this cost is therefore to reduce the testing cost. Test cost reduction can be achieved by minimizing the testing time of the system. An efficient ordering, test scheduling, of the execution of the test sets will minimize the total testing time.

The core-based design technique is another approach to reduce the increasing development costs. With such a technique, pre-designed and pre-verified blocks of logic, so called cores, are integrated to a complete system, which can be placed on a single die to form a system-on-chip (SOC). To test a SOC, a test bus is used for the transportation of test data in the system and its organization often has a great impact on the test schedule. SOC test scheduling can be performed assuming: *sequential scheduling*, *i.e.* only one test at a time, or *concurrent scheduling*, with a possibility to execute several tests at the same time. The testing time for the execution of each test set can be *fixed,* or *flexible* where it is possible to adjust it.

In a large volume production test for SOC, an *abort-on-fail* approach is usually used, which means that the test sequence is aborted as soon as a fault is detected. This approach is used to reduce the test application time. With the abort-on-fail assumption, the tests should be ordered in such a way that tests

1. The chapter is based on the papers presented at International Test Synthesis Workshop (ITSW'03) [168], 2003, Design and Diagnostics of Electronic Circuits & Systems (DDECS'03), 2003 [169], and VLSI Test Symposium (VTS'04), 2004 [174].

2. The research is partially supported by the Swedish National Program STRINGENT.

with a high probability to fail are scheduled before tests with a lower probability to fail since this will minimize the average testing time.

In this chapter we propose a test scheduling technique based on defect detection probability, collected from the production line or generated based on inductive fault analysis. We have defined models to compute the expected test time as well as scheduling heuristic taking the defect probabilities into account. We have performed experiments to show the efficiency of the proposed approach.

The rest of the chapter is organized as follows. An overview of related work is in Section 2. Sequential test scheduling is discussed in Section 3 and concurrent test scheduling is described in Section 4. The proposed algorithms are presented in Section 4.3 and the chapter is concluded with experimental results in Section 5 and conclusions in Section 6.

2 RELATED WORK

Test scheduling determines the execution order of the tests in a system. The most common objective is to minimize the test application time while considering test conflicts. In SOC systems, where each core is equipped with a wrapper, an interface to the test access mechanism (TAM), test conflict is due to the sharing of the TAM or the test bus. The TAM, used for the transportation of test data, is used to connect the test source, the cores and the test sink. The test source is where the test vectors are generated or stored and the test sink is where the test responses are analyzed or stored. An automatic test equipment (ATE) is a typical example of a test source and test sink.

A TAM can be organized in different ways, which impacts the test scheduling. An example is the AMBA test bus, which makes use of the existing functional bus, however, the tests have to be scheduled in a sequence [88]. An alternative is the approach proposed by Varma and Bhatia where several test buses are used. The tests on each bus are scheduled in a sequence, however, since several buses are allowed, testing can be performed concurrently [267]. Another approach is the TestRail, which allows a high degree of flexibility [184].

The TestRail approach has recently gained interest and several scheduling techniques for scan tested SOCs have been proposed [97,111,138]. The objective is to arrange the scan-chains into wrapper chains, which then are connected to TAM wires. Iyengar *et al.* made use of integer-linear programming [111] and Huang *et al.* used a bin-packing algorithm. Both these approaches assume that the tests will always be performed until completion. Koranne proposed an *abort-on-fail* technique to minimize the average-completion time by scheduling tests with short test time early [138]. For sequential testing, several abort-on-fail test scheduling techniques considering the defect probability have been proposed [102,124]. Huss and Gyurcsik made use of a dynamic programming algorithm to order the tests [102]. Milor and Sangiovanni-Vincentelli proposed a technique for the selection and ordering of the test sets [196], which is based on the dependencies between the test sets. For SOC testing with cores in wrappers, however, there is no dependency between the testing of different cores. In the approach proposed by Jiang and Vinnakota the actual fault coverage is extracted from the manufacturing line [124]. The technique minimizes the average completion time by ordering of the tests based on the failure probability.

3 SEQUENTIAL TEST SCHEDULING

In sequential testing, all tests are scheduled in a sequence, one test at a time. When the *abort-on-fail* approach is assumed, if a defect is detected, the testing should be terminated at once. In order to account for the case when the test responses are compacted into a single signature after a whole test set is applied, we assume that the test abortion occurs at the end of a test set even if the actual defect is detected in the middle of applying the test set. This assumption is also used in our formula to compute the expected test time. Note, this means that the computational results are pessimistic, or the actual test time will be smaller than the computed one, in the case when the tests are actually aborted as soon as the first defect is detected.

Given a core-based system with *n* cores, for each core *i* there is a test set t_i with a test time τ_i and a probability of passing p_i (*i.e.* the probability that test t_i will detect a defect at core *i* is 1-p_i). For a given schedule, the *expected test time* for sequential testing is given by:

$$\sum_{i=1}^{n}\left(\left(\sum_{j=1}^{i}\tau_j\right)\times\left(\prod_{j=1}^{i-1}p_j\right)\times(1-p_i)\right)+\left(\sum_{i=1}^{n}\tau_i\right)\times\prod_{i=1}^{n}p_i \qquad (14.1)$$

For illustration of the computation of the expected test time, we use an example with four tests (Table 39). The tests are scheduled in a sequence as in Figure 224. For test t_1, the expected test time is given by the test time τ_1 and the probability of success p_1, $\tau_1 \times p_1 = 2 \times 0.7 = 1.4$. Note if there is only one test in the system, our above formula will give the expected test time to be 2 since we assume that every test set has to be fully executed before we can determine if the test is a successful test or not.

Core i	*Test t_i*	*Test time τ_i*	*Probability to pass, p_i*	Cost ($\tau_i \times p_i$)
1	t_1	2	0.7	1.4
2	t_2	4	0.8	2.4
3	t_3	3	0.9	2.7
4	t_4	6	0.8	4.8

Table 39. Example data.

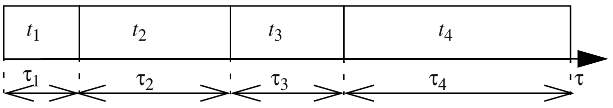


Figure 224. Sequential schedule of the example (Table 39).

The expected test time for the completion of the complete test schedule in Figure 224 is:

$\tau_1 \times (1-p_1)$ + // test t_1 fails
$(\tau_1+\tau_4) \times p_1 \times (1-p_4)$ + // test t_1 passes, test t_4 fails
$(\tau_1+\tau_4+\tau_3) \times p_1 \times p_4 \times (1-p_3)$ +// test t_1, t_4 pass, test t_3 fails
$(\tau_1+\tau_4+\tau_3+\tau_2) \times p_1 \times p_4 \times p_3 \times (1-p_2)$ + //test t_1, t_4, t_3 pass, test t_2 fails
$(\tau_1+\tau_4+\tau_3+\tau_2) \times p_1 \times p_4 \times p_3 \times p_2$ =// all tests run until completion,
// *i.e.* correct system.

$2 \times (1-0.7)$ +
$(2+6) \times 0.7 \times (1-0.6)$ +
$(2+6+3) \times 0.7 \times 0.6 \times (1-0.9)$ +
$(2+6+3+2) \times 0.7 \times 0.6 \times 0.9 \times (1-0.8)$ +
$(2+6+3+2) \times 0.7 \times 0.6 \times 0.9 \times 0.8 = 8.2$

As a comparison, for the worst schedule, where the test with highest passing probability is scheduled first, the order will be t_4, t_3, t_2, t_1, and the expected test time is 12.1. In the case of executing all tests until completion, the total test time does not depend on the order, and is $\tau_1+\tau_2+\tau_3+\tau_4=15$.

4 CONCURRENT TEST SCHEDULING

The total test time of a system can be reduced by executing several tests at the same time, *concurrent testing*. Concurrent testing is for instance possible in systems with several test buses. In this section, we analyze concurrent scheduling with fixed test time per test set and flexible test time per test set.

4.1 Test Sets with Fixed Test Times

A concurrent test schedule of the example system used in Section 3 with data as in Table 39 assuming 3 TAMs (test buses) is in Figure 225. The test schedule (Figure 225) consists of a set of sessions, S_1, S_2, S_3, and S_4. Test session S_1 consists of test t_1, t_2 and t_3; $S_1=\{t_1,t_2,t_3\}$. The length of a session S_k is given by l_k. For instance $l_1=2$. We assume now that the abortion of the test process can occur at any time during the application of the tests. To simplify the computation of expected test time, it is assumed that the test process will terminate at the end of a session (note this is again a pessimistic assumption). The probability to reach the end of a session depends in the concurrent test scheduling approach not only on a single test but on all tests in the session. For instance, the probability to complete session 1 depends on the tests in session 1 (S_1): t_1, t_2 and t_3. As can be observed in Figure 225, only test t_1 is fully completed at the end of session 1. For a test t_i that is not completed at the end of a session, the probability p_{ik} for it to pass all test vectors applied during session k is given by:

$$p_{ik} = p_i^{l_k/\tau_i} \tag{14.2}$$

It can be seen that for a test set t_i, which is divided into m sessions, the probability that the whole test set is passed is equal to:

$$\prod_{k=1}^{m} p_{ik} = p_i^{\frac{l_1}{\tau_i}} \times p_i^{\frac{l_2}{\tau_i}} \times \ldots \times p_i^{\frac{l_k}{\tau_i}} = p_i^{\sum_{k=1}^{m} \frac{l_k}{\tau_i}} = p_i \tag{14.3}$$

since:

$$\sum_{k=1}^{m} \frac{l_k}{\tau_i} = \frac{1}{\tau_i} \times \sum_{k=1}^{m} l_k = 1 \tag{14.4}$$

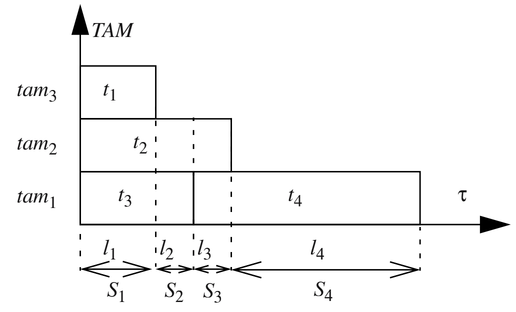


*Figure 225.*Concurrent schedule of the example (Table 39).

For example, the probability for the tests in session S_1 are (Figure 225):
$p_{11} = p_1 = 0.7$.
$p_{21} = 0.8^{2/4} = 0.89$.
$p_{31} = 0.9^{2/3} = 0.93$.
The formula for computing the expected test time for a complete concurrent test schedule is given as:

$$\sum_{i=1}^{n}\left(\left(\sum_{j=1}^{i} l_j\right)\times\left(\prod_{j=1}^{i-1}\prod_{\forall t_k \in S_j} p_{kj}\right)\times\left(1-\prod_{\forall t_k \in S_i} p_{ki}\right)\right)+\left(\sum_{i=1}^{n}\tau_i\right)\times\prod_{i=1}^{n}p_i \qquad (14.5)$$

As an example, the computation of the expected test time for the test schedule in Figure 225 is given below. First we compute the probability for each test set in each session.

The probabilities p_{11}, p_{21}, p_{31} are computed to 0.7, 0.89, and 0.93, respectively (see above).
$p_{22} = 0.7^{1/4} = 0.91$.
$p_{32} = 0.9^{1/3} = 0.96$.
$p_{23} = 0.8^{1/4} = 0.95$.
$p_{43} = 0.95^{1/6} = 0.99$.
$p_{44} = 0.95^{5/6} = 0.96$.
From the formula we get:
$l_1\times(1-p_{11}\times p_{21}\times p_{31})+$
$(l_1+l_2)\times p_{11}\times p_{21}\times p_{31}\times(1-p_{22}\times p_{32})+$
$(l_1+l_2+l_3)\times p_{11}\times p_{21}\times p_{31}\times p_{22}\times p_{32}\times (1-p_{23}\times p_{43})+$
$(l_1+l_2+l_3+l_4)\times p_{11}\times p_{21}\times p_{31}\times p_{22}\times p_{32}\times p_{23}\times p_{43}\times(1-p_{44})+$
$(l_1+l_2+l_3+l_4)\times p_{11}\times p_{21}\times p_{31}\times p_{22}\times p_{32}\times p_{23}\times p_{43}\times p_{44}=$
$2\times(1-0.7\times 0.89\times 0.93)+$
$(2+1)\times 0.7\times 0.89\times 0.93\times(1-0.91\times 0.96)+$
$(2+1+1)\times 0.7\times 0.89\times 0.93\times 0.91\times 0.96\times (1-0.95\times 0.99)+$
$(2+1+1+5)\times 0.7\times 0.89\times 0.93\times 0.91\times 0.96\times 0.95\times 0.99\times$
$(1-0.96) + (2+1+1+5)\times 0.7\times 0.8\times 0.9\times 0.95 = 5.66$.
As a comparison, if all tests are assumed to be executed until completion, the total test time will be 9.

4.2 Test Sets with Flexible Test Times

A way to further reduce the test application time is to modify, if possible, the test times of the individual test sets. For instance, in scan tested cores the test times at each core can be modified by loading several scan chains in parallel. The scan-chains and the wrapper cells are to form a set of wrapper chains. Each wrapper chain is then connected to a TAM wire. If a high number of wrapper chains are used, their length is shorter and the loading time of a new test vector is reduced. However, the higher number of wrapper chains require more TAM wires.

In Figure 226 the TAM bandwidth |W| is 4, there are four wires in W={w_1, w_2, w_3, w_4}. The testing of each core is performed by transporting test vectors on the assigned TAM wires to a core and the test response is also transported from the core to the test sink using the TAM. The testing of cores sharing TAM wires cannot be executed concurrently. For instance, the testing of core 1 and core 2 cannot be performed concurrently due to the sharing of TAM wire w_3 (Figure 226). A test schedule for the system is given in Figure 227 and the computation of the expected test time can be done using formula 14.3 in Section 4. In the case with flexible test times, the number of assigned TAM wires will affect the expected test time. The problem is to assign TAM wires to each core in such a way that the expected test time is minimized.

4.3 Test Scheduling Algorithms

The algorithm for test scheduling based on defect probability in the sequential case is straight forward, it sorts the tests in descending order based on $\tau_i \times (1-p_i)$ and schedule the tests in this order (Figure 228).

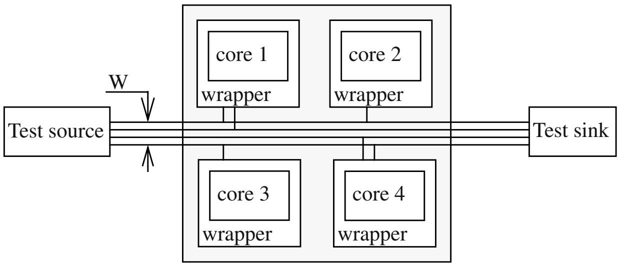


Figure 226. SoC example.

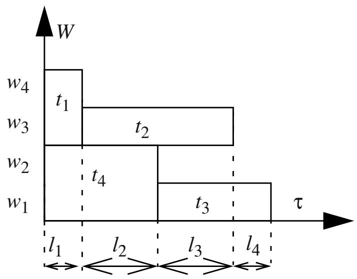


Figure 227. SOC test schedule of the example (Table 39).

```
Compute the cost $c_i = p_i \times \tau_i$ for all tests $t_i$
Sort the costs $c_i$ ascending in $L$
until $L$ is empty (all tests are scheduled) begin
  select, schedule and remove the first test in $L$
end
```

*Figure 228.*Sequential test scheduling algorithm.

In concurrent scheduling with fixed test times, we sort the tests based on $\tau_i \times (1-p_i)$ and select n tests for the n TAMs based on the sorted list. The selected tests are scheduled and removed from the list. As soon as a test terminates, a new test from the list of unscheduled tests is selected. The process continues until all tests are scheduled (sketch of the algorithm is given in Figure 229.)

For tests with flexible test times versus number of assigned TAM wires, the wrapper chain design algorithm configures the scanned elements (scan-chains, input wrapper cells, output wrapper cells and bidirectional wrapper cells) into a given number of wrapper chains and computes the testing time for the wrapper configuration.

The proposed wrapper design heuristic is illustrated in Figure 230. We use an internal chaining function aiming at balancing the scan chains in order to reduce the longest wrapper chain. The longest wrapper chain is the one that limits the solution (the testing time). The generated designs are memorized so that all the possible architectures for each core can be checked during the TAM building and the test scheduling steps.

The scheduling heuristic is outlined in Figure 230. First the tests are sorted in decreasing cost order. For each test, one Pareto optimal point is selected considering the maximal width use (*i.e.* the couple T_i, W_i with W_i being the closest to W_{max} (W_{max} is given)).

```
Compute the cost $c_i = \tau_i \times p_i$ for all tests $t_i$
Sort the costs $c_i$ ascending in $L$
$f$=number of TAMs
$\tau$=0 // current time,
**Until** $L$ is empty (all tests are scheduled) begin
   at time $\tau$ **Until** $f$=0 **Begin**
     select tests from list in order and reduce $f$ accordingly
   **End**
   $\tau$=time when first test terminates.
 **End**
```

*Figure 229.*Concurrent test scheduling algorithm for tests with fixed test times.

At step two, the *VirtualTime* test time is estimated in order to obtain a lower bound for the system test time. This bound is used in the scheduling heuristic during the selection of configurations for each core. The advantage is that points with a testing time higher than *VirtualTime* will not be selected since they will increase the total test time.

The main idea in the heuristic is to schedule the tests as soon as possible using the Pareto optimal points defined in the wrapper design heuristic. For each test, the heuristic tries to place each test in a session starting from time t=0, and also trying all the Pareto optimal points (*i.e.* changing the values of W_i and T_i) of the considered test with a cost loss lower or equal to the tolerance to fit in the constraints. The heuristic defines one schedule and one TAM configuration for each tolerance (*i.e.* 80 schedules and TAM configurations from 0% to 80%) and memorizes the solution with the smallest test time fitting into the limits imposed by the constraints.

For all the tests that are first sorted into a list L1, if one test can not be scheduled, it is placed in a auxiliary list L2 to be scheduled later. When L1 is

```
L1=list of sorted tests in decreasing cost $c_i = \tau_i \times p_i$ order
VirtualTime= $\sum W_i \times T_i / W_{limit}$
For tolerance=0 to tolerance=80
 While all tests are not scheduled
   While L1 not empty
     For each test T in L1
         For each time t defining the start of a test session
           Select the best Pareto optimal point such that
             a) it respects the tolerance;
             b) the width constraint is satisfied,
             c) the test time does not exceed VirtualTime, and
             d) precedence, power, incompatibilities
                 constraints are respected.
             If (the current total test time will not change
                 when T is scheduled to start at t)
               Schedule T at t with the selected Pareto point;
               remove T from L1.
             Else
               If T is the first test of L1
                 Schedule T at t with selected Pareto point;
                 remove T from L1.
               Else
                 Place the test T in L2; remove T from L1.
   L1<=L2
End
```

*Figure 230.*Our Test Scheduling Heuristic empty, *i.e.* all tests are scheduled or placed in L2, then L2 becomes L1 and the process is re iterated until all tests are scheduled.

5 EXPERIMENTAL RESULTS

We have compared three approaches to demonstrate the importance of considering the defect probability during test scheduling. We have implemented the three approaches; (1) sequential scheduling where the tests are ordered in a sequence, (2) one where a fixed test time is assigned to each test prior to scheduling [219], and (2) one where the testing time versus the number of used TAM wires is flexible [220], . We have used the ITC'02 designs [190] and for all experiments we have used an AMD 1800 machine (1.53 GHz, 512 MB RAM) and the computational cost is usually a few seconds, and never exceeds 15 seconds. The defect probability for each core is collected in Table 40.

The experimental results are collected in Table 41. We have for each of the benchmarks made experiments at various TAM bandwidths. We have also compared the three approaches. The results indicate that an efficient ordering taking the defect probabilities into account can reduce the testing times up to nearly 90% compared to sequential testing (also taking the defect probabilities into account).

6 CONCLUSIONS

In this chapter we have developed test scheduling techniques for system-on-chip (SOC) that take into account the defect probability of each test. The advantage of our approach is that by considering defect probabilities during the test scheduling process, the expected test time can be minimized, which is important in large volume production of SOC where the testing process is terminated as soon as a defect is detected (abort-on-fail).

We have analyzed several different test bus structures and scheduling approaches, and defined models to compute the expected test times and test scheduling algorithms for several types of test buses. We have also performed experiments to demonstrate the efficiency of our approach.

Core	*Design*						
	d695	*h953*	*g1023*	*t51250*	*p22810*	*p34392*	*p93791*
1	98	95	99	99	98	98	99
2	99	91	99	95	98	98	99
3	95	92	99	97	97	97	97
4	92	92	98	93	93	91	90
5	99	97	94	90	91	95	91
6	94	90	95	98	92	94	92
7	90	94	94	98	99	94	98
8	92	96	97	96	96	93	96
9	98		92	92	96	99	91
10	94		92	91	95	99	94
11			96	91	93	91	93
12			92	92	91	91	91
13			93	91	92	90	91
14			96	91	93	95	90
15				99	99	94	99
16				95	99	96	98
17				97	99	96	97
18				95	95	97	99
19				94	96	92	99
20				99	97	90	99
21				91	93	92	90
22				99	99	99	99
23				91	96	96	90
24				97	98	98	98
25				92	99		92
26				96	92		96
27				95	91		95
28				92	91		91
29				90	93		90
30				91	94		96
31				95			

Table 40. The pass probability in percentage for cores in systems d695, g1023, h953, t512505, p22810, p34392, p93791.

Design	*TAM Width*	*Expected Test Time*			*Comparison*	
		1 - Sequential Testing	*2 - Fixed Testing Times*	*3 - Flexible Testing Times*	*(3) vs (1)*	*(3) vs (2)*
g1023	128	41 807	24 878	17 904	- 57,2%	- 28,0%
	96	43 289	23 443	18 741	- 56,7%	- 20,1%
	80	44 395	23 112	18 229	- 58,9%	- 21,1%
	64	44 395	27 358	20 773	- 53,2%	- 24,1%
	48	46 303	27 997	21 501	- 53,6%	- 23,2%
	32	55 562	27 662	26 867	- 51,6%	- 2,9%
	24	56 711	29 410	28 795	- 49,2%	- 2,1%
	20	60 609	36 979	35 431	- 41,5%	- 4,2%
	16	75 100	44 728	44 657	- 40,5%	- 0,2%
	12	95 679	67 549	60 239	- 37,0%	- 10,8%
d695	128	31 113	10 884	9 468	- 69,6%	- 13,0%
	96	31 158	14 716	11 712	- 62,4%	- 20,4%
	80	31 158	14 881	14 509	- 53,4%	- 2,5%
	64	40 586	25 483	16 652	- 59,0%	- 34,7%
	48	40 692	27 388	23 983	- 41,1%	- 12,4%
	32	70 411	50 998	33 205	- 52,8%	- 32,9%
	24	70 598	62 367	42 165	- 40,3%	- 32,4%
	20	70 696	68 611	50 629	- 28,4%	- 26,2%
	16	131 178	123 164	61 473	- 53,1%	- 50,1%
	12	131 465	131 465	82 266	- 37,4%	- 37,4%
h953	128	104 382	87 339	82 358	- 21,1%	- 5,7%
	96	104 466	82 733	82 437	- 21,1%	- 0,4%
	80	104 466	85 307	82 448	- 21,1%	- 3,4%
	64	104 466	87 349	82 466	- 21,1%	- 5,6%
	48	104 508	87 443	82 495	- 21,1%	- 5,7%
	32	104 549	92 245	84 169	- 19,5%	- 8,8%
	24	104 591	99 888	104 290	- 0,3%	+ 4,4%
	20	104 633	125 262	92 159	- 11,9%	- 26,4%
	16	159 657	137 089	135 438	- 15,2%	- 1,2%
	12	189 740	185 016	183 359	- 3,4%	- 0,9%

Table 41. Experimental results comparing three approaches (1), (2) and (3).

Design	*TAM Width*	*Expected Test Time*			*Comparison*	
		1 - Sequential Testing	*2 - Fixed Testing Times*	*3 - Flexible Testing Times*	*(3) vs (1)*	*(3) vs (2)*
p22810	128	423 852	71 628	50 484	- 88,1%	- 29,5%
	96	423 968	93 921	59 177	- 86,0%	- 37,0%
	80	423 993	122 641	71 995	- 83,0%	- 41,3%
	64	443 459	141 999	92 218	- 79,2%	- 35,1%
	48	510 795	213 995	121 865	- 76,1%	- 43,1%
	32	535 586	355 646	160 237	- 70,1%	- 54,9%
	24	707 813	480 480	294 612	- 58,4%	- 38,7%
	20	836 491	756 138	328 270	- 60,8%	- 56,6%
	16	877 443	855 355	383 034	- 56,3%	- 55,2%
	12	1 341 549	1 336 251	401 720	- 70,1%	- 69,9%
t512505	128	9 724 227	1 073 413	889 677	- 90,9%	- 17,1%
	96	9 724 227	1 217 641	894 924	- 90,8%	- 26,5%
	80	9 724 227	1 269 333	928 499	- 90,5%	- 26,9%
	64	9 724 227	2 810 847	1 062 112	- 89,1%	- 62,2%
	48	14 883 557	8 938 649	1 828 281	- 87,7%	- 79,5%
	32	14 883 609	8 940 193	1 955 361	- 86,9%	- 78,1%
	24	25 202 194	16 090 266	2 891 241	- 88,5%	- 82,0%
	20	25 202 230	16 308 884	3 652 388	- 85,5%	- 77,6%
	16	25 202 298	21 716 978	3 961 341	- 84,3%	- 81,8%
	12	46 296 336	27 526 848	5 394 939	- 88,3%	- 80,4%
p34392	128	1 168 630	258 038	265 777	- 77,3%	+ 3,0%
	96	1 168 630	343 408	248 170	- 78,8%	- 27,7%
	80	1 212 761	374 916	262 563	- 78,3%	- 30,0%
	64	1 212 899	470 976	268 010	- 77,9%	- 43,1%
	48	1 232 116	627 558	389 813	- 68,4%	- 37,9%
	32	1 525 655	1 271 626	563 470	- 63,1%	- 55,7%
	24	1 559 706	1 389 689	823 435	- 47,2%	- 40,7%
	20	1 640 812	1 596 012	1 382 206	- 15,8%	- 13,4%
	16	2 888 061	2 677 967	1 604 297	- 44,5%	- 40,1%
	12	2 960 587	2 926 044	2 154 610	- 27,2%	- 26,4%

Table 41. Experimental results comparing three approaches (1), (2) and (3).

Design	*TAM Width*	*Expected Test Time*			*Comparison*	
		1 - Sequential Testing	*2 - Fixed Testing Times*	*3 - Flexible Testing Times*	*(3) vs (1)*	*(3) vs (2)*
p93791	128	491 279	431 628	124 278	- 74,4%	- 71,2%
	96	524 537	488 083	192 940	- 63,2%	- 60,5%
	80	942 900	852 477	197 393	- 79,1%	- 76,8%
	64	983 943	922 505	270 979	- 72,5%	- 70,6%
	48	1 072 900	1 003 672	360 045	- 66,4%	- 64,1%
	32	1 941 982	1 941 892	682 101	- 64,9%	- 64,9%
	24	2 125 118	2 125 118	826 441	- 61,1%	- 61,1%
	20	3 546 031	3 546 031	1 023 667	- 71,1%	- 71,1%
	16	3 854 386	3 854 386	1 353 034	- 64,9%	- 64,9%
	12	4 238 379	4 238 379	3 768 819	- 11,1%	- 11,1%

Table 41. Experimental results comparing three approaches (1), (2) and (3).

Chapter 15

An Integrated Technique for Test Vector Selection and Test Scheduling under ATE Memory Depth Constraint[1]

1 INTRODUCTION[2]

The technology development has made it possible to develop chips where a complete system with an enormous number of transistors, which are clocked at an immense frequency and partitioned into a number of clock-domains, is placed on a single die. As the technology development makes it possible to design these highly advanced system chips or SOC (system-on-chip), the EDA (Electronic Design Automation) tools are aiming at keeping up the productivity, making it possible to design a highly advanced system with a reasonable effort in a reasonable time. New design methodologies are under constant development. At the moment, a modular design approach where modules are integrated to a system is promising. The advantage with such an approach is that pre-designed and pre-verified modules, blocks of logic or cores, with technology specific details, can at a reasonable time and effort be integrated to a system. The core provider designs the cores and the system integrator selects the appropriate cores for the system where the cores may origin from previous in-house designs, or from different core vendors (companies). The cores can be delivered in various formats. They can in general be classified as soft cores, firm cores, and hard cores. Soft cores are general high-level specifications where the system integrator can, if necessary, apply modifications. Hard cores are gate-level specifications where, if any, only a few modifications are possible. Firm cores are somewhere between soft cores and hard cores. Soft cores allow more flexibility compared to hard cores. The advantage is that the system integrator can modify such a core. On the other hand, hard cores can be made highly protected by the core provider, which often is desirable by the core provider.

A produced chip is tested to determine if it is faulty or not. In the test process, a number of test vectors, stored in an ATE (Automatic Test Equipment), are applied to the chip under test. If the produced test response from the applied vectors corresponds to the expected response, the chip is considered to be fault-free and can be shipped. However, testing these complex chips is becoming a problem, and one major problem is the increasing test data vol-

1. The chapter is based on the work submitted to Asian Test Symposium (ATS), 2004, [57].

2. The research is partially supported by the Swedish National Program STRINGENT.

ume that has to be stored in the ATE. Currently, the test data volume increases faster than the number of transistors in a design [272]. The increasing test data volume is due to (1) high number of fault sites because of the high amount of transistors, (2) new defect types introduced with nanometer process technologies, and (3) faults related to timing and delay since systems have higher performance and make use of multiple-clock domains [272].

The high test data volume is a problem. It is known that the purchase of a new ATE with higher memory capabilities is costly; hence, it is desirable to make use of the existing ATE instead of investing in a new. Vranken *et al.* [272] discuss three alternatives to make the test data fit the ATE; (1) *test memory reload*, where the test data is divided into several partitions, is possible but not practical due to the high time involved, (2) *test data truncation*, the ATE is filled as much as possible and the test data that does not fit the ATE is simply not applied, leads to reduced test quality, and (3) *test data compression*, the test stimuli is compressed, however, it does not guarantee that the test data will fit the ATE. As, test memory reload is not practical, the alternatives are test data truncation and test data compression. This chapter focuses on test data truncation where the aim is a technique that maximizes test quality while making sure the test data volume fits the ATE memory.

The test data must also be organized or scheduled in the ATE. A recent industrial study showed that by using test scheduling the test data was made to fit the ATE [76]. The study demonstrated that the ATE memory limitation is a real and critical problem. The basic idea in test scheduling is to reduce the amount of idle bits to be stored in the ATE, and therefore scheduling must be considered in combination with the test data truncation scheme. Further, when discussing memory limitations, the ATE memory depth in bits is equal to the maximal test application time for the system in clock cycles [116]. Hence, the memory constraint must be seen as a time constraint.

In this chapter, we explore test data truncation. The aim is a technique that maximizes test quality while making sure that the selected test data fits the ATE. We assume that given is a core-based design and for each core the defect probability, the maximal fault coverage when all its test vectors have been applied, and the size of the test set (the number of test vectors) are given. We define for a core, a CTQ (core test quality) metric, and for the system, a STQ (system test quality) metric. The CTQ metric reflects that test data should be selected for a core (1) with high probability of having a defect, and (2) where it is possible to detect a fault using a minimal number of test vectors. For the fault coverage function we make use of an estimation function. Fault simulation can be used to extract the fault coverage at each test vector, however, it is a time consuming process and also it might not be applicable for all cores due to IP (Intellectual Property)-protection, for instance.

The test vectors in a test set can be applied in any order. However, regardless of the order, it is well-known in the test community that the first test vectors detects a higher number of faults compared to the last applied test vectors, and that the function fault coverage versus number of test vectors has an exponential/logarithmic behaviour. We therefore assume that the fault coverage over time (number of applied test vectors) for a core can be approximated to an exponential function.

We make use of CTQ metric to select test data volume for each core in such a way that the test quality for the system is maximized (STQ), and we integrate the test data selection with test scheduling in order to verify that the selected test data actually fits the ATE memory. We have implemented our technique and we have made experiments on several ITC'02 benchmarks to demonstrate that high test quality can be achieved by applying only a sub-set of the test stimuli. The results indicate that the test data volume and the test application time can be reduced to 50% while the test quality remains high. Furthermore, it is possible to turn the problem (and our solution), and view it as: for a certain test quality, which test data should be selected to minimize the test application time.

The advantage with our technique is that given a core-based system, a test set per core, a number on maximal fault coverage, and defect probability per core, we can select test data for the system and schedule the selected test data in such a way that the test quality is maximized and the selected test data fits the ATE memory. In the chapter, we assume a single test per core. However, the technique can easily be extended to allow multiple tests per core by introducing constraint considerations in the scheme.

The rest of the chapter is organized as follows. In Section 2 we present related work, and in Section 3 the problem definition is given. The test quality metric is defined in Section 4 and our test data selection and scheduling approach is described in Section 5. The experiments are presented in Section 6 and finally the chapter is concluded in Section 7.

2 RELATED WORK

Test scheduling and test data compression are examples of approaches proposed to reduce the high test data volumes that must be stored in the ATE in order to test SOCs. The basic principle in test scheduling is to organize the test bits in the ATE in such a way that the number of introduced so called idle bits (not useful bits) is minimized. The gain is reduced test application time and a reduced test data volume. A scheduling approach depends on the test architecture such as the AMBA test bus [88], the test bus [267] and the TestRail [184].

Iyengar *et al.* [111] proposed a technique to partition the set of scan chain elements (internal scan chains and wrapper cells) at each core into wrapper scan chains, which are connected to TAM wires in such a way that the total test time is minimized. Goel *et al.* [76] showed that ATE memory limitation is a critical problem. On an industrial design they showed that by using an effective test scheduling technique the test data can be made to fit the ATE.

There has also been scheduling techniques that make use of an abort-on-fail strategy that is the testing is terminated as soon as a fault is detected. The idea is that as soon as a fault is present, the chip is faulty and the testing can be terminated. Koranne minimizes the average-completion time by scheduling short tests early [139]. Other techniques have taken the defect probability for each testable unit into account [102,124,111]. Huss and Gyurcsik proposed a sequential technique making use of a dynamic programming algorithm for ordering the tests [102], while Milor and Sangiovanni-Vincentelli present a sequential technique based on selection and ordering of test sets [196]. Jiang and Vinnakota proposed a sequential technique, where the information about the fault coverages provided by the tests is extracted from the manufacturing line [124]. For SOC designs, Larsson *et al.* proposed a technique based on ordering of tests, considering different test bus structures, scheduling approaches (sequential vs. concurrent) and test set assumptions (fixed test time vs. flexible test time) [111]. The technique takes defect probability into account; however, the probability of detecting a fault remains constant through the application of a test.

Several compression schemes have been used to compress the test data. For instance, Ichihara *et al.* used statistical codes [103], Chandra and Chakrabarty made use of Golomb codes [34], Iyengar *et al.* explored the use of run-length codes [109], Chandra and Chakrabarty tried Frequency-directed run-length codes [33], and Volkerink *et al.* have investigated the use of Packet-based codes [270].

All approaches above (test scheduling and test data compression techniques) reduce the ATE memory requirement. In the case of test scheduling, the effective organization means that both the test time and the needed test data volume are reduced, and in the case of test data compression, less test data is required to be stored in the ATE. The main advantage with these two approaches is that the highest possible test quality is reached since the whole test data volume is applied. However, the main disadvantage is that these techniques do not guarantee that the test data volume fits the ATE. Hence, they might not be applicable in practice. It means that there is a need for a technique that in a systematic way defines the test data volume for a system in such a way that the test quality is maximized while the test data is guaranteed to fit the ATE memory.

3 PROBLEM FORMULATION

We assume that given is a core-based architecture with *n* cores denoted by *i,* and for each core *i* in the system, the following is given:

- $sc_{ij}=\{sc_{i1}, sc_{i2},..., sc_{im}\}$ - the length of the scanned elements at core *i* are given where *m* is the number of scanned elements,
- wi_i - the number of input wrapper cells,
- wo_i - the number of output wrapper cells,
- wb_i - the number of bidirectional wrapper cells,
- tv_i - the number of test vectors,
- fc_i - the fault coverage reached when all the tv_i test vectors are applied.
- pp_i - the pass probability per core and,
- dp_i - the defect probability per core (given as $1\text{-}pp_i$).

For the system, a maximal TAM bandwidth W_{tam}, a maximal number of *k* TAMs, and a upper-bound memory constraint M_{max} on the memory depth in the ATE are given.

The TAM bandwidth W_{tam} is to be partitioned into a set of *k* TAMs denoted by *j* each of width $W_{tam}=\{w_1, w_2, ..., w_k\}$ in such a way that:

$$W_{tam} = \sum_{j=1}^{k} w_j \tag{15.1}$$

and on each TAM, one core can be tested at a time.

Since the memory depth in the ATE (in bits) is equal to the test application time for the system (in clock cycles) [116], the memory constraint is actually a time constraint τ_{max}:

$$M_{max} = \tau_{max} \tag{15.2}$$

Our problem is to:

- for each core *i* select the number of test vectors (stv_i),
- partition the given TAM width W_{tam} into no more than *k* TAMs,
- determine the width of each TAM (w_j), j=1..k,

- assign each core to one TAM, and
- assign a start time for the testing of each core.

The selection of test data (stv_i for each core i) and the test scheduling should be done in such a way that the test quality of the system (defined in Section 4) is maximized while the memory constraint (M_{max}) (time constraint τ_{max}) is met.

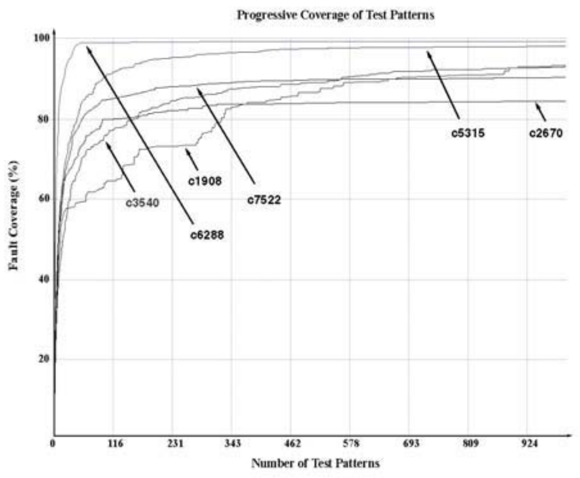
4 TEST QUALITY METRIC

For the truncation scheme we need a test quality metric to (1) select test data for each core and (2) to measure the final system test quality. In this section we describe the metric where we take the following parameters into account to measure test quality:

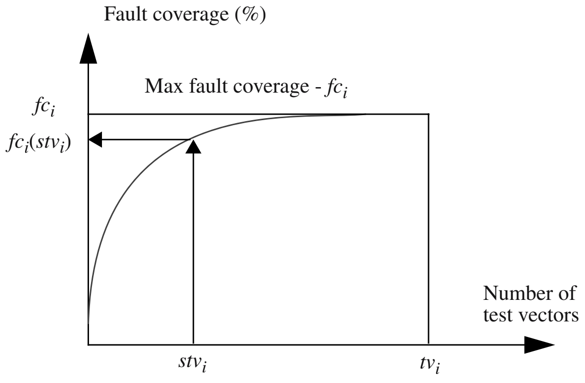
- defect probability,
- fault coverage, and
- number of applied test vectors.

The defect probability, the probability that a core is defect, can be collected from the production line or set by experience. Defect probability has to be taken into account since it is better to select test data for a core with a high defect probability than to select test data for a core with a low defect probability since the core with high defect probability it is more likely to hold a defect.

The possibility to detect faults depends on the fault coverage versus the number of applied test vectors; hence the fault coverage and the number of applied test vectors also have to be taken into account. Fault simulation can be used to extract which fault each test vector detects. However, in a complex core-based design with a high number of cores, fault simulation for each core is, if possible due to IP-protection, highly time consuming. A core provider may want to protect the core, which makes fault simulation impossible. We therefore make use of an estimation technique. It is known that the fault coverage does not increase linearly over the number of applied test vectors. For instance, Figure 231 (a) shows the fault coverage for a set of ISCAS benchmarks. The following observation can be made: the curves have an exponential/logarithmic behaviour (as in Figure 231 (a)). We, therefore,



(a)



(b)

*Figure 231.*Fault coverage versus number of test vectors (a) for a set of ISCAS designs (b) estimated as an exponential function.

assume that the fault coverage after applying stv_i test vectors for core i can be estimated to (Figure 2 (b)):

$$\mathrm{fc_i(stv_i)} = \frac{\log(\mathrm{stv_i} + 1)}{\mathrm{slopeConst}} \tag{15.3}$$

where the *slopeConst* is given as follows:

$$\mathrm{slopeConst} = \frac{\log(\mathrm{tv_i} + 1)}{\mathrm{fc_i}} \tag{15.4}$$

and the +1 is used to adjust the curve to passes the origin.

For a system we assume that the test quality can be estimated to:

$$\text{P(we find a defect}|\text{we have a defect in the SOC)} \tag{15.5}$$

The test quality describes the probability of finding a defect when we have the condition that the SOC has one defect. By introducing this probability, we find a way to measure the probability of finding a defect if a defect exist in the SOC and hence the test quality. However, it is important to note that our metric only describes the test quality and hence we are not introducing any assumptions about the number of defects in the SOC.

In order to derive an equation for the test quality using information about defect probability, fault coverage and the number of test vectors, we make use of definitions from basic probability theory [20]:

1. If A and B are independent events => $P(A \cap B) = P(A)P(B)$
2. If $A \cap B$ is the empty set $\varnothing$ => $P(A \cup B) = P(A) + P(B)$
3. $P(A \cap B) = P(A)P(B|A)$, where P(B|A) is the probability of B conditioned on A.

Furthermore, we assume (Section 3) that the quality of a test set (a set of test vectors) for a core *i* is composed by the following:

- fault coverage fc_i and
- probability of defect dp_i.

Since the number of applied test vectors indirectly has an impact on the fault coverage, we define for each core *i*:

- stv_i - selected number of test vectors, and
- $fc_i(stv_i)$ - fault coverage after stv_i test vectors have been applied.

We do the following assumption:

- dp_i and fc_i are independent events.

Since we assume one defect in the system when we introduced test quality (Equation 15.5), we can only have one defect in a core at a time in the system. Therefore we can say:

- The intersection of any of the events dp_i is the empty set $\varnothing$.

For a system with *n* cores, we can now derive *STQ* (system test quality) from Equation 15.5 by using Definition 1., 2. and 3.:

$$\mathrm{STQ} = P\langle \text{defect detected in the SOC}|\text{defect in the SOC}\rangle \Rightarrow$$
$$\frac{P\langle \text{defect detected in the SOC} \cap \text{defect in the SOC}\rangle}{P\langle \text{defect in the SOC}\rangle} \Rightarrow$$
$$\frac{\sum_{i=1}^{n} P\langle \text{defect detected in core } i \cap \text{defect in the core } i\rangle}{P\langle \text{defect in the SOC}\rangle} \Rightarrow$$
$$\frac{\sum_{i=1}^{n} P\langle \text{defect detected in core } i \times \text{defecet in the core } i\rangle}{P\langle \text{defect in the SOC}\rangle} \Rightarrow$$
$$\frac{\sum_{i=1}^{n} fc_i \times fc_i(stv_i)}{\sum_{i=1}^{n} dp_i} \qquad (15.6)$$

And for a single core *i*, the CTQ (core test quality) is:

$$\mathrm{CTQ}_i = dp_i \times fc_i(stv_i) \qquad (15.7)$$

5 TEST SCHEDULING AND TEST VECTOR SELECTION

In this section we describe our technique to optimize test quality by selecting test vectors for each core and schedule the selected vectors for an SOC under the time constraint given by the ATE memory depth (see Equation 15.2 and [116]). We assume that given is a system as described in Section 3 and we assume an architecture where the TAM wires can be grouped into several TAMs and the cores connected to the same TAM are tested sequentially one after the other [267]. We make use of the test quality metric defined in Section 4.

The scanned elements (scan-chains, input cells, output cells and bidirectional cells) at a core has to be configured into a set of wrapper chains, which are to be connected to a corresponding number of TAM wires. The wrapper scan chains, which are to be connected to the TAM wires w_j, should be as balanced as possible and we make use of the *Design_wrapper* algorithm proposed by Iyengar *et al.* [111]. For a wrapper chain configuration at a core *i* where si_i is the longest wrapper scan-in chain and so_i is the longest wrapper scan-out chain, the test time for core *i* is given by [111]:

$$\tau_i(w_j, tv_i) = (1 + \max(si_i(w), so_i(w))) \times tv + \min(si_i(w), so_i(w)) \qquad (15.8)$$

where tv_i is the number of applied test vectors for core *i* and *w* is the TAM width.

We need a technique to partition the given TAM width W_{tam} into a number of TAMs *k* and to determine which core that should be assigned to the designed which TAM. The number of different ways we can assign *n* cores to *k* TAMs grows with k^n, and therefore the number of possible alternatives will be huge. We need a technique to guide the assignment of cores to the TAMs. We make use of the fact that Iyengar *et al.* [111] made use of, which is that balancing the wrapper scan-in chain and wrapper scan-out chain introduces different number of ATE idle bits as the TAM bandwidth varies. We define TWU_i (TAM width utilization) for a core *i* at a TAM of width *w* as:

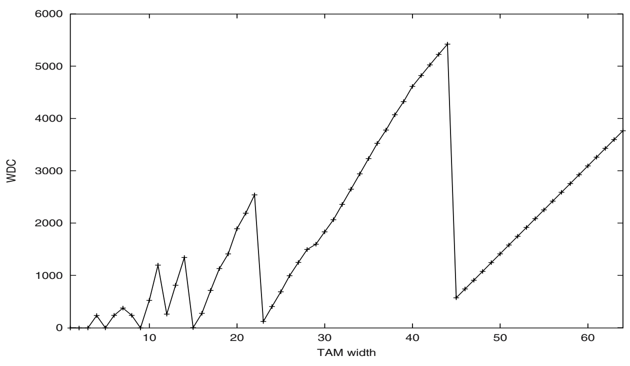
$$TWU_i(w) = \max(si_i(w), so_i(w)) \times w \qquad (15.9)$$

and we make use of a single wrapper-chain (one TAM wire) as a reference point to introduce *WDC* (wrapper design cost) that measure the imbalance (introduced number of idle bits) for a TAM width *w* relative to TAM width 1:

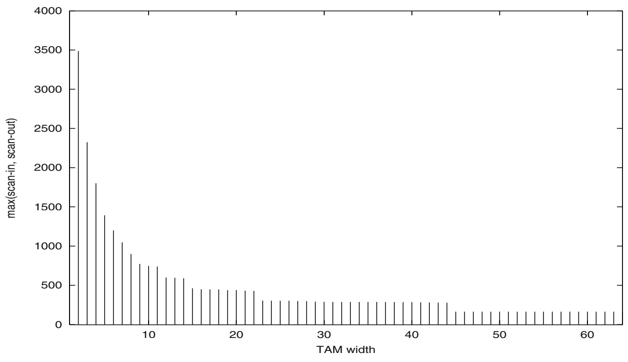
$$WDC_i = TWU_i(w) - TWU_i(1) \qquad (15.10)$$

For illustration of the variations in the number of ATE idle bits, we plot in Figure 232(a) the value of *WDC* for different TAM widths (number of wrapper chains), obtained by using core 1 of the ITC'02 benchmark p93791. We also plot the maximum value of the scan-in and scan-out lengths at various TAM widths for the previous design in Figure 232(b). In Figure 232(b) several TAM widths have the same test time. For a set of TAM widths with the same test time, a Pareto-optimal point is the one with lowest TAM [111]. We notice, we can notice that the TAM widths having a low value of the WDC, and hence a small number of idle bits, corresponds to the Pareto-optimal points. Hence, we make use of WDC to guide the selection of wrapper chains at a core.

The algorithm for our test truncation scheme is outlined in Figure 233. Given is a system, the upper bound on the test time (τ_{max}) and the TAM width (W_{tam}). Initially no test vectors are selected for any core (stv_i=0 for all *i*) and the test time for the test schedule is zero (TAT=0). The test vector that contributes most to improving STQ is selected, assigned to a TAM where WDC is minimal and scheduled on the selected TAM in order to make sure that the τ_{max} is not violated. Additional vectors are selected one by one in such a way that STQ is maximized, and after each selection the schedule is created to verify that the time constraint (ATE memory depth constraint) is not violated.



(a)



(b)

Figure 232. Variation of (a) WDC and (b) max(scan-in, scan-out) at different TAM widths at core 1 (p93791).

Note that the test vectors for a core might not be selected in order. For instance, in a system with two cores A and B, the first vector can be selected from core A, the second from core B, and the third from core A. However, at the scheduling, the test vectors for each core are grouped and scheduled as a single set. The algorithm (Figure 233) assumes a fixed TAM partition (number of TAMs and their width). We have therefore added an outer loop that makes sure that we explore all possible TAM configurations.

5.1 Illustrative Example

To illustrate the proposed technique for test scheduling and test vector selection, we make use of an example where the time constraint is set to 5%

```
Given:
$\tau_{max}$ - the upper test time limit for the system
$W_{tam}$ - number of TAM wires - distributed over $k$ TAMs $w_1$, $w_2$,
..., $w_k$ in such a way that Eq. 15.1 holds.
Variables:
$stv_i$ = 0   //selected number of test vectors for core $i$
$TAT$ = 0    // test application time of the system
Compute $WDC_i$ for all cores at all $k$ TAMs (Eq. 15.10)
Select best TAM for each core based on $WDC_i$
while $TAT$< $\tau_{max}$ at any TAM begin
     for $i$=1 to $n$ begin // For all cores
       Compute $\tau(w_j,1)$ (Eq. 15.8)
       Compute $CTQ_i$ assuming $stv_i$=$stv_i$+1 (Eq. 15.7)
     end
     for core with highest CTQ/$\tau(w_j,1)$ and $stv_i$<$tv_i$
       $stv_i$=$stv_i$+1
     for all cores where $stv_i$>0 begin// some selected vec-
tors
       Assign core to an available TAM with minimal $WDC_i$
       if a TAM is full (<$\tau_{max}$) - mark TAM as unavailable.
     end
Compute and return STQ (Eq. 15.7).
end
```

Figure 233. Test vector selection and test scheduling algorithm.

of the maximal test application time (the time when all available test vectors are applied). For the example, we make use of the ITC'02 benchmark [189, 190] d695 with the data presented in Table 42. As the maximal fault coverage for a core when all test vectors are applied and the pass probability per core are not given in the ITC'02 benchmarks, we have added these numbers. In order to show the importance of combining test scheduling and test vector selection, we compare our proposed technique to a naive approach where we order the tests and assign test vectors according to the initial sorted order until

	Core										
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Scan-chains	0	0	0	1	4	32	16	16	4	32	32
Inputs w_i	0	32	207	34	36	38	62	77	35	35	28
Outputs w_o	0	32	108	1	39	304	152	150	49	320	106
Test vectors tv_i	0	12	73	75	105	110	234	95	97	12	68
Pass probability pp_i	97	98	99	95	92	99	94	90	92	98	94
Max fault coverage fc_i (%)	95	93	99	98	96	96	99	94	99	95	96

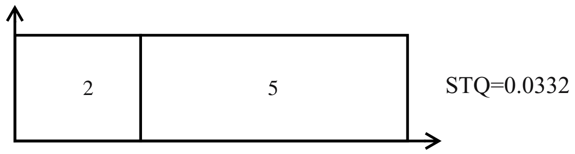
Table 42. Data for benchmark d695.

the time limit (ATE memory depth) is reached. For this naive approach we consider three different techniques.

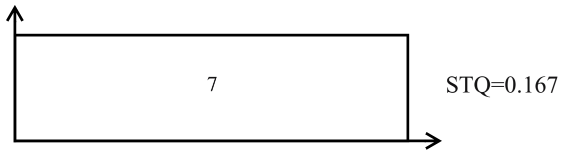
1. Sorting when not considering defect probability and fault coverage (Technique 1).
2. Sorting when considering defect probability but not fault coverage. The cores are sorted in descending order according to defect probability (Technique 2).
3. Sorting when considering defect probability in combination with fault coverage. In this technique, we make use of the *STQ* (Equation 15.6) equation to find a value of the test quality for each core. The cores are then sorted in descending order according to test quality per clock cycle. The sorting constant is described in Equation 15.11 (Technique 3).

$$\text{sortConst} = \frac{dp_i \times fc_i(tv_i)}{\tau(w, tv_i) \times \sum_{i=1}^{n} dp_i} \qquad (15.11)$$

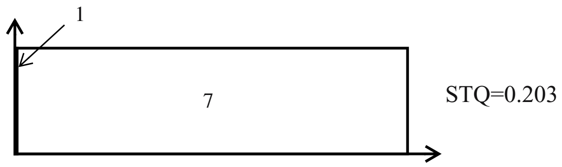
For our test vector selection and test scheduling technique, we consider three cases where we divide the TAM into one (Technique 4), two (Technique 5) or three test buses (Technique 6). The selected test data volume per core for each of the six scheduling techniques is reported in Table 43 and the test schedules with the corresponding *STQ* are presented in Figure 234. Figure 234 (a) illustrates the case when no information about defect probability and fault coverage is used in the test ordering. As seen in the figure, such technique produces a schedule with an extremely low system test quality (*STQ*). By making use of the information on defect probability (Figure 234 (b)), respective defect probability and fault coverage (Figure 234 (c)) in the ordering, we can improve the test quality significantly. Although it is possible to increase the *STQ* by using an efficient sorting technique, we are still not exploiting the fact that the first test vectors in a test set detect more faults than the last test vectors. In (Figure 234 (d) - (f)), we make use this information as we are using our proposed technique for test scheduling and test vector selection. We note that it is possible to further improve the *STQ* by dividing the TAM into several test buses (Figure 234 (e) - (f)).



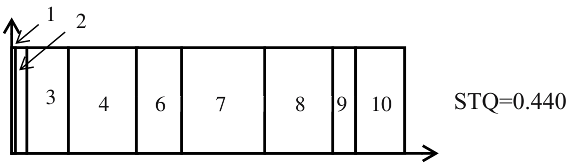
(a) Test scheduling without test vector selection when not considering defect probability and fault coverage.



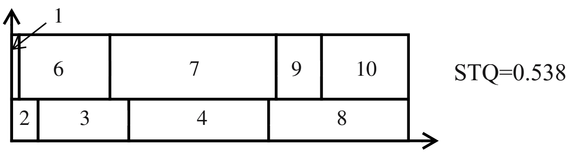
(b) Test scheduling considering defect probability.



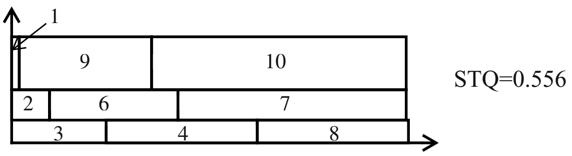
(c) Test scheduling considering defect probability and fault coverage.



(d) Test scheduling using test vector selection and one TAM.



(e) Test scheduling using test vector selection and two TAMs.



(f) Test scheduling using test vector selection and three TAMs.

*Figure 234.*Results for different scheduling techniques.

Technique	*Selected test data for each core (%)*										
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Technique 1	0	0	100	0	0	20	0	0	0	0	0
Technique 2	0	0	0	0	0	0	0	54.7	0	0	0
Technique 3	100	0	0	0	0	0	0	52.6	0	0	0
Technique 4	0	100	9.6	6.7	4.8	0	1.7	10.5	6.2	8.3	4.4
Technique 5	0	100	9.6	16.0	10.5	0	3.8	21.1	13.4	8.3	4.4
Technique 6	0	100	9.6	17.3	11.4	0	2.6	13.7	17.5	33.3	14.7

Table 43. Selected test vectors (%) for the cores in design d695 considering different scheduling techniques.

5.2 Optimal Solution For Single TAM

The algorithm above can easily be improved to produce an optimal solution in the case of a single TAM. The algorithm above aborts the assignment of test vectors immediately when the time constraint (memory constraint) is reached - a selected test vector cannot be assigned since it violates the constraint. However, test vectors from other cores (not from the core that violates the time constraint) could have been selected while making sure that they do not violate the ATE constraint.

Note, that the selection of test vectors is based on a monotonically decreasing function. The test vector that contributes most to the test quality is first selected. That process continous on an updated list until the constraint is reached. In the case of a single TAM, the scheme is optimal.

6 EXPERIMENTAL RESULTS

The aim with the experiments is to demonstrate that the test quality can be kept high by using the proposed ATE memory constrained test data truncation scheme. We have implemented the proposed technique described above, and we have in the experiments made use of five ITC'02 benchmarks [189, 190], d281, d695, p22810, p34392, and p93791. It is given for each core in these benchmarks, the number of test vectors, the number of scanned elements (number and length of the scan-chains), the number of input pins, bidirectional pins and output pins. The netlists for the ITC'02 benchmarks are not publicly available, and therefore we have, in order to perform experiments, added for each core a pass probability and a maximal fault coverage number when all its test vectors are applied (Table 44). And to get soft cores, we have assumed that when we need soft cores the scan-chains can be broken; only the number of flip-flops is given.

In order to have a memory (time) constraint from the ATE, we performed for each design a schedule where all vectors are applied and that test application time reefers to 100%. We have performed experiments at various ATE memory depths constraints (equal to time constraints (see Equation 15.2 and [116])) and these constraints are set as a percentage of the time it would take to apply all test vectors.

We identify six techniques:

1. Test scheduling when not considering defect probability nor fault coverage and testing is aborted at τ_{max} - technique 1.
2. Test scheduling when considering defect probability but not fault coverage and testing is aborted at τ_{max} - technique 2.
3. Test scheduling when considering defect probability as well as fault coverage and testing is aborted at τ_{max} - technique 3.
4. Test scheduling and test vector selection when considering defect probability and fault coverage, using one TAM - technique 4.
5. Test scheduling and test vector selection when considering defect probability and fault coverage, using up to two TAMs - technique 5.
6. Test scheduling and test vector selection when considering defect probability and fault coverage, using up to three TAMs - technique 6.

In the first experiment, we analyze the importance of TAM width. We have made experiments on benchmark p93791 at TAM width 16, 32 and 64 at time constraint 5%, 10%, 25%, 50%, 75%, and 100% of the test application time if all test data is applied. The results are collected in Table 45 and illustrated for technique 2, 4, and 6 in Figure 235. The results show that the produced results (STQ) are at a given time constraint, rather similar at various TAM widths. Therefore, for the rest of the experiments we assume a TAM bandwidth W_{tam} of 32.

All experimental results are collected in Table 46 for soft cores and in Table 48 for hard cores. The CPU times and TAM widths (where applicable) is found in Table 47 for soft cores, and in Table 49 for hard cores. The results collected in Table 46 and Table 48 are for each design plotted in Figure 236 (D281 - soft cores), Figure 237 (D281 - hard cores), Figure 238 (D695 - soft cores), Figure 239 (D695 - hard cores), Figure 240 (P22810 - soft cores), Figure 241 (P22810 - hard cores), Figure 242 (P34392 - soft cores), Figure 243 (P34392 - hard cores), Figure 244 (P93791 - soft cores), Figure 245 (P93791 - hard cores). In column 1 the design name is given, in column 2 the percentage of the test time is given, and in column 3 to 8 the produced STQ is reported for each technique (1 to 6). The computational cost for every experiment is in the range of a few seconds to a few minutes.

From the experimental results we learn that the STQ value increases with the time constraint (a larger ATE memory results in a higher STQ), which is obvious. It is also obvious that the STQ value for a design is the same at 100% test time, all test data is applied. From the results, we also see that test set selection improves the test quality when comparing STQ at the same test time limit. That is, technique 4, 5, 6 have significant higher STQ value compared to technique 1, 2 and 3. But also important, we note that we can achieve a high test quality at low testing times. Take design p93791, for example, where the STQ value (0.584) for technique 1 at 75% of the testing time is lower than the STQ value (0.748) at only 5% for technique 6. It means that it is possible, by integrating test set selection and test scheduling, to reduce the test application time while keeping the test quality high.Also, we have selected rather high pass probabilities and rather high fault coverage as these numbers are not publicly available. For designs with lower pass probabilities and lower fault coverage, and also, for designs where the variations in these numbers are higher, our technique becomes more important.

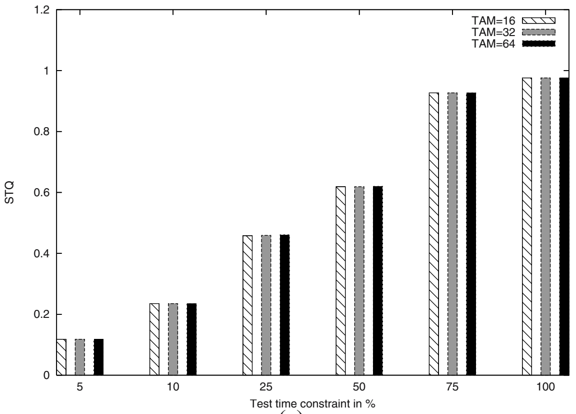
7 CONCLUSIONS

The technology development has made it possible to design extremely advanced chips where a complete system is placed on a single die. The requirement to test these system chips increase, and especially, the growing test data volume is becoming a problem. Several test scheduling techniques have been proposed to organize the test data in the ATE in such a way that the ATE memory limitation is not violated, and several test compression schemes have been proposed to reduce the test data volume. However, these techniques do not guarantee that the test data volume fits the ATE.

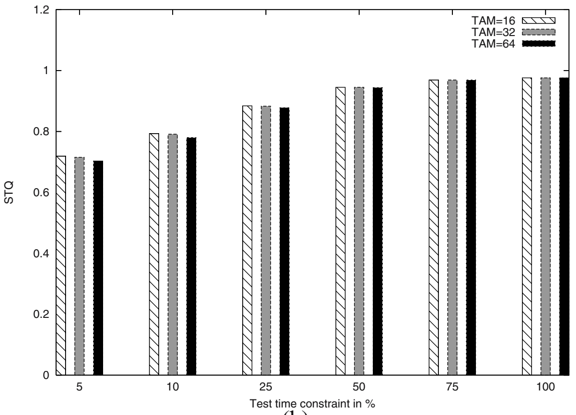
In this chapter we have therefore proposed a test data truncation scheme that systematically selects test vectors and schedules the selected test vectors for each core in a core-based system in such a way that the test quality is maximized while the constraint on ATE memory depth is met. We have defined a test quality metric based on defect probability, fault coverage and the number of applied vectors that is used in the proposed test data selection scheme. We have implemented our technique and the experiments on several ITC'02 benchmarks [189, 190] at reasonable CPU times show that high test quality can be achieved by careful selection of test data. Further, our technique can be used to shorten the test application time for a given test quality value.

Table 44. Pass probability (pp) and maximal fault coverage (fc).

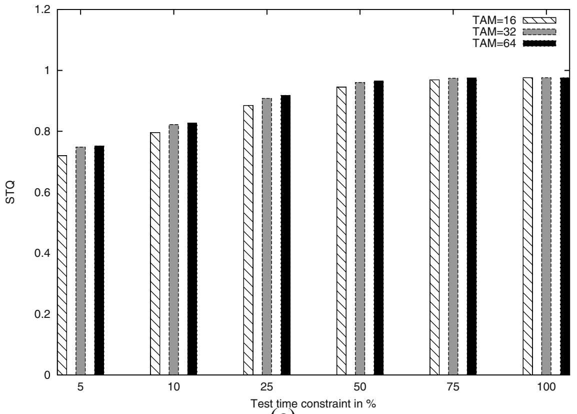
Core	*D281*		*D695*		*P22810*		*P34392*		*P93791*	
	pp (%)	*fc (%)*	*pp (%)*	*fc (%)*	*pp (%)*	*fc (%)*	*pp (%)*	*fc (%)*	*pp (%)*	*fc (%)*
0	98	93	97	95	98	95	98	97	99	99
1	98	98	98	93	98	99	98	97	99	99
2	99	97	99	99	97	97	97	99	99	95
3	95	95	95	98	93	98	91	98	97	98
4	92	98	92	96	91	94	95	99	90	98
5	99	98	99	96	92	99	94	99	91	99
6	94	96	94	99	99	99	94	97	92	97
7	90	99	90	94	96	97	93	98	98	99
8	92	97	92	99	96	95	99	94	96	95
9			98	95	95	97	99	96	91	96
10			94	96	93	97	91	98	94	97
11					91	99	91	98	93	99
12					92	99	90	99	91	99
13					93	94	95	94	91	94
14					99	97	94	97	90	98
15					99	94	96	95	99	94
16					99	99	96	98	98	97
17					95	98	97	98	97	97
18					96	94	92	95	99	95
19					97	95	90	95	99	95
21					93	99			99	99
22					99	99			90	98
23					96	95			99	96
24					98	98			90	98
25					99	95			98	94
26					92	99			92	99
27					91	99			96	99
28					91	97			95	98
29					93	98			91	99
30									90	97
31									96	98
32									99	99



(a)



(b)



(c)

*Figure 235.*Comparing STQ at TAM width 16, 32, and 64 for technique 2 (a), technique 4(b) and technique 6(c).

SOC	*% of max test time*	*Technique 1*	*Technique 2*	*Technique 3*	*Technique 4*	*Technique 5*	*Technique 6*
		STQ	*STQ*	*STQ*	*STQ*	*STQ*	*STQ*
p93791 TAM width 16	*5*	0.00542	0.118	0.560	0.719	0.720	0.720
	10	0.0248	0.235	0.618	0.793	0.796	0.796
	25	0.0507	0.458	0.747	0.884	0.885	0.885
	50	0.340	0.619	0.902	0.945	0.945	0.945
	75	0.588	0.927	0.958	0.969	0.969	0.969
	100	0.976	0.976	0.976	0.976	0.976	0.976
p93791 TAM width 32	*5*	0.00542	0.118	0.559	0.715	0.748	0.748
	10	0.0249	0.235	0.618	0.791	0.822	0.822
	25	0.0507	0.459	0.742	0.883	0.908	0.908
	50	0.340	0.619	0.902	0.945	0.960	0.960
	75	0.584	0.927	0.957	0.969	0.974	0.974
	100	0.976	0.976	0.976	0.976	0.976	0.976
p93791 TAM width 64	*5*	0.00535	0.118	0.499	0.703	0.752	0.752
	10	0.00606	0.235	0.567	0.780	0.827	0.827
	25	0.0356	0.461	0.739	0.878	0.918	0.918
	50	0.335	0.620	0.901	0.944	0.965	0.965
	75	0.566	0.927	0.961	0.969	0.975	0.975
	100	0.976	0.976	0.976	0.976	0.976	0.976

Table 45. Comparison of different TAM widths using ITC'02 benchmark p93791.

Table 46. Test quality (STQ) at different test time constraints where cores in the design are soft.

Design	*Percentage of maximal test time*	*Technique 1*	*Technique2*	*Technique3*	*Technique 4*	*Technique5*	*Technique 6*
		STQ	*STQ*	*STQ*	*STQ*	*STQ*	*STQ*
d281	5	0.0209	0.164	0.649	0.762	0.776	0.788
	10	0.0230	0.186	0.676	0.838	0.846	0.846
	25	0.209	0.215	0.884	0.905	0.910	0.911
	50	0.940	0.237	0.931	0.947	0.951	0.951
	75	0.965	0.944	0.949	0.967	0.968	0.968
	100	0.974	0.974	0.974	0.974	0.974	0.974
d695	5	0.0332	0.185	0.445	0.625	0.641	0.641
	10	0.0370	0.497	0.635	0.755	0.768	0.768
	25	0.306	0.629	0.823	0.904	0.904	0.904
	50	0.612	0.963	0.963	0.963	0.963	0.963
	75	0.966	0.966	0.966	0.966	0.966	0.966
	100	0.966	0.966	0.966	0.966	0.966	0.966
p22810	5	0.0526	0.179	0.672	0.822	0.822	0.834
	10	0.0788	0.189	0.762	0.884	0.884	0.895
	25	0.338	0.775	0.924	0.951	0.951	0.955
	50	0.973	0.973	0.973	0.973	0.973	0.973
	75	0.973	0.973	0.973	0.973	0.973	0.973
	100	0.973	0.973	0.973	0.973	0.973	0.973
p34392	5	0.0566	0.329	0.640	0.839	0.858	0.869
	10	0.0616	0.459	0.783	0.899	0.912	0.919
	25	0.542	0.767	0.934	0.959	0.961	0.962
	50	0.972	0.972	0.972	0.972	0.972	0.972
	75	0.972	0.972	0.972	0.972	0.972	0.972
	100	0.972	0.972	0.972	0.972	0.972	0.972
p93791	5	0.00572	0.199	0.566	0.748	0.755	0.755
	10	0.0294	0.278	0.656	0.823	0.828	0.828
	25	0.125	0.516	0.840	0.912	0.914	0.914
	50	0.453	0.882	0.949	0.964	0.965	0.965
	75	0.976	0.976	0.976	0.976	0.976	0.976
	100	0.976	0.976	0.976	0.976	0.976	0.976

Table 47. CPU time and TAM width assignment where the cores in the design are soft.

Design	*Percentage of maximal test time*	*Technique 1*	*Technique2*	*Technique3*	*Technique 4*	*Technique5*		*Technique 6*	
		CPU (s)	*CPU (s)*	*CPU (s)*	*CPU (s)*	*CPU (s)*	*TAMs*	*CPU (s)*	*TAMs*
d281	5	0.1	0.1	0.1	0.4	4.6	11/21	29.0	8/10/14
	10	0.1	0.1	0.1	0.4	4.8	11/21	29.9	11/21
	25	0.1	0.1	0.1	0.4	4.8	11/21	28.2	5/13/14
	50	0.1	0.1	0.1	0.4	5.4	11/21	34.0	11/21
	75	0.1	0.1	0.1	0.5	5.9	11/21	34.0	11/21
	100	0.1	0.1	0.1	0.5	5.9	32	36.6	32
d695	5	0.2	0.2	0.2	0.7	8.5	13/19	56.0	13/19
	10	0.2	0.2	0.3	0.7	8.7	13/19	55.1	13/19
	25	0.2	0.2	0.2	0.8	8.6	32	57.0	32
	50	0.2	0.2	0.3	0.7	8.7	32	55.2	32
	75	0.2	0.2	0.2	0.7	8.7	32	56.1	32
	100	0.2	0.3	0.2	0.7	8.7	32	56.4	32
p22810	5	0.8	0.8	0.9	2.5	32.8	32	210.2	7/12/13
	10	0.8	0.8	0.8	2.6	33.7	32	211.9	7/12/13
	25	0.8	0.8	0.8	3.2	39.3	32	235.8	7/12/13
	50	0.8	0.7	0.8	3.7	42.5	32	250.9	32
	75	0.8	0.7	0.8	3.7	45.5	32	264.0	8/12/12
	100	0.7	0.7	0.8	3.7	48.2	11/21	276.5	8/12/12
p34392	5	0.7	0.6	0.7	2.6	35.5	15/17	211.3	7/12/13
	10	0.7	0.6	0.6	3.4	42.0	15/17	245.7	7/12/13
	25	0.6	0.6	0.6	4.3	53.5	14/18	299.9	5/11/16
	50	0.8	0.6	0.7	4.4	61.1	32	336.7	32
	75	0.7	0.6	0.9	4.5	61.2	32	383.6	7/9/16
	100	0.8	0.6	0.7	4.4	63.4	32	379.6	7/9/16
p93791	5	3.3	2.6	2.9	8.4	108.1	14/18	688.5	14/18
	10	2.9	2.5	3.1	8.3	110.1	14/18	700.5	14/18
	25	3.0	2.5	2.6	9.2	118.1	14/18	737.5	14/18
	50	3.2	2.5	2.9	9.0	118.7	14/18	740.4	14/18
	75	3.1	2.5	2.9	9.2	120.2	32	752.8	32
	100	3.1	2.6	2.8	9.4	123.6	14/18	760.9	14/18

Table 48. Test quality (STQ) at different test time constraints where cores in the design are hard.

Design	*Percentage of maximal test time*	*Technique 1*	*Technique2*	*Technique3*	*Technique 4*	*Technique5*	*Technique 6*
		STQ	*STQ*	*STQ*	*STQ*	*STQ*	*STQ*
d281	5	0.0209	0.164	0.496	0.674	0.726	0.726
	10	0.0230	0.186	0.563	0.774	0.818	0.818
	25	0.198	0.215	0.834	0.879	0.905	0.912
	50	0.912	0.237	0.903	0.935	0.949	0.949
	75	0.956	0.870	0.923	0.960	0.968	0.968
	100	0.974	0.974	0.974	0.974	0.974	0.974
d695	5	0.0332	0.167	0.203	0.440	0.538	0.556
	10	0.0370	0.257	0.254	0.567	0.670	0.690
	25	0.208	0.405	0.510	0.743	0.849	0.863
	50	0.335	0.617	0.803	0.879	0.952	0.952
	75	0.602	0.821	0.937	0.946	0.965	0.965
	100	0.966	0.966	0.966	0.966	0.966	0.966
p22810	5	0.0333	0.174	0.450	0.659	0.691	0.759
	10	0.0347	0.186	0.608	0.764	0.796	0.856
	25	0.0544	0.398	0.769	0.885	0.900	0.940
	50	0.181	0.830	0.912	0.949	0.949	0.968
	75	0.600	0.916	0.964	0.969	0.969	0.973
	100	0.973	0.973	0.973	0.973	0.973	0.973
p34392	5	0.0307	0.312	0.683	0.798	0.843	0.859
	10	0.0341	0.331	0.766	0.857	0.893	0.898
	25	0.0602	0.470	0.846	0.919	0.940	0.942
	50	0.533	0.492	0.921	0.950	0.963	0.967
	75	0.547	0.906	0.943	0.965	0.972	0.972
	100	0.972	0.972	0.972	0.972	0.972	0.972
p93791	5	0.00542	0.118	0.559	0.715	0.748	0.748
	10	0.0249	0.235	0.618	0.791	0.822	0.822
	25	0.0507	0.459	0.742	0.883	0.908	0.908
	50	0.340	0.619	0.902	0.945	0.960	0.960
	75	0.584	0.927	0.957	0.969	0.974	0.974
	100	0.976	0.976	0.976	0.976	0.976	0.976

Table 49. CPU time and TAM width assignment where the cores in the design are hard.

Design	*Percentage of maximal test time*	*Technique 1*	*Technique2*	*Technique3*	*Technique 4*	*Technique5*		*Technique 6*	
		CPU (s)	*CPU (s)*	*CPU (s)*	*CPU (s)*	*CPU (s)*	*TAMs*	*CPU (s)*	*TAMs*
d281	5	0.2	0.1	0.1	0.4	3.1	11/21	19.7	11/21
	10	0.1	0.2	0.1	0.4	3.4	11/21	20.6	11/21
	25	0.1	0.1	0.1	0.4	3.6	11/21	21.3	5/13/14
	50	0.1	0.1	0.2	0.4	4.1	11/21	26.0	11/21
	75	0.1	0.1	0.1	0.4	4.4	11/21	27.0	11/21
	100	0.1	0.1	0.1	0.4	4.8	11/21	28.1	11/21
d695	5	0.09	0.09	0.1	0.2	1.9	13/19	11.9	7/9/16
	10	0.09	0.09	0.1	0.2	1.9	13/19	12.0	7/9/16
	25	0.08	0.09	0.09	0.2	2.0	13/19	12.1	7/9/16
	50	0.08	0.09	0.1	0.2	2.0	13/19	12.7	13/19
	75	0.08	0.09	0.1	0.2	2.0	13/19	12.7	13/19
	100	0.09	0.09	0.1	0.2	2.0	32	13.0	32
p22810	5	0.1	0.1	0.1	0.3	4.0	14/18	27.1	9/11/12
	10	0.1	0.1	0.1	0.4	5.2	14/18	34.7	9/11/12
	25	0.1	0.1	0.1	0.8	9.4	14/18	48.3	9/11/12
	50	0.1	0.1	0.1	1.0	12.6	32	78.6	9/11/12
	75	0.1	0.1	0.1	1.5	15.6	32	97.5	9/11/12
	100	0.1	0.1	0.1	1.6	18.4	32	115.9	9/11/12
p34392	5	0.06	0.06	0.06	1.0	12.6	11/21	64.4	9/10/13
	10	0.06	0.07	0.08	1.7	21.0	11/21	96.5	9/10/13
	25	0.06	0.06	0.06	2.4	31.0	11/21	151.4	9/10/13
	50	0.1	0.06	0.06	2.7	36.2	11/21	193.8	9/10/13
	75	0.08	0.06	0.08	2.8	39.3	11/21	217.2	9/10/13
	100	0.07	0.07	0.07	2.8	45.0	14/18	327.0	14/18
p93791	5	0.5	0.5	0.6	1.3	12.3	15/17	68.3	15/17
	10	0.3	0.4	0.4	1.4	14.5	15/17	78.7	15/17
	25	0.4	0.3	0.4	1.7	19.0	15/17	100.9	15/17
	50	0.4	0.8	0.4	1.9	22.5	14/18	117.7	14/18
	75	0.4	0.4	0.4	2.0	24.7	15/17	128.6	15/17
	100	0.4	0.4	0.4	2.3	26.7	14/18	140.5	14/18

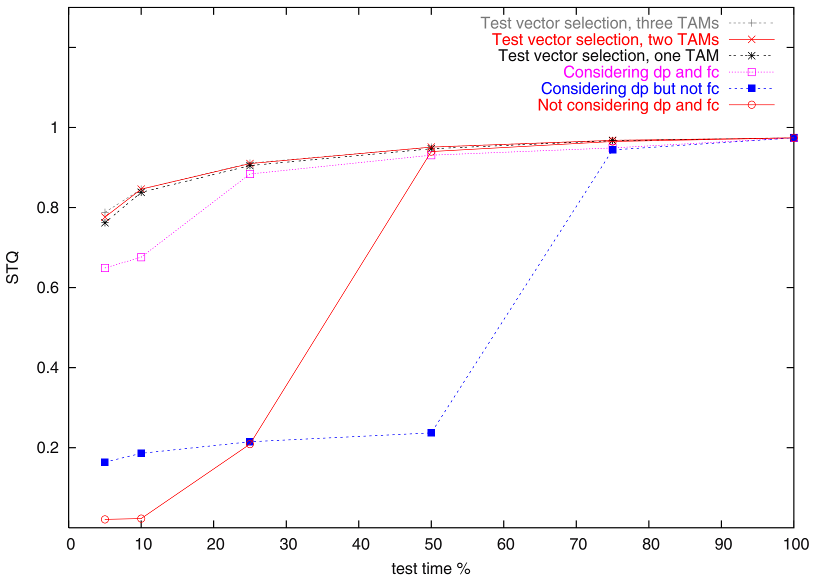


*Figure 236.*Design D281 - soft cores.

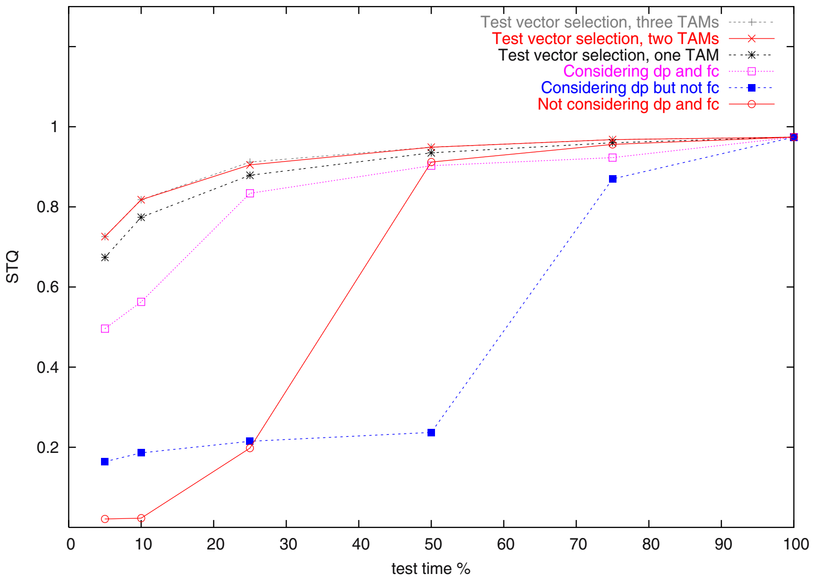


*Figure 237.*Design D281 - hard cores.

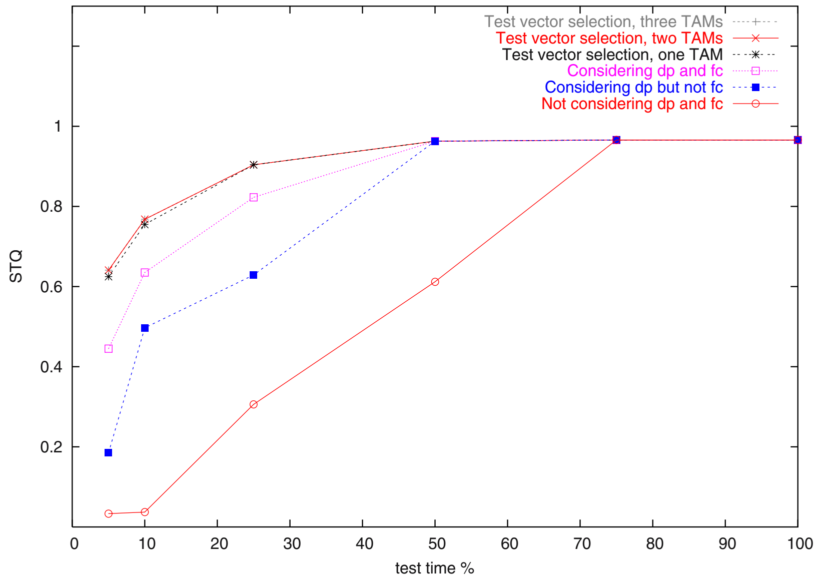


*Figure 238.*Design D695 - soft cores.

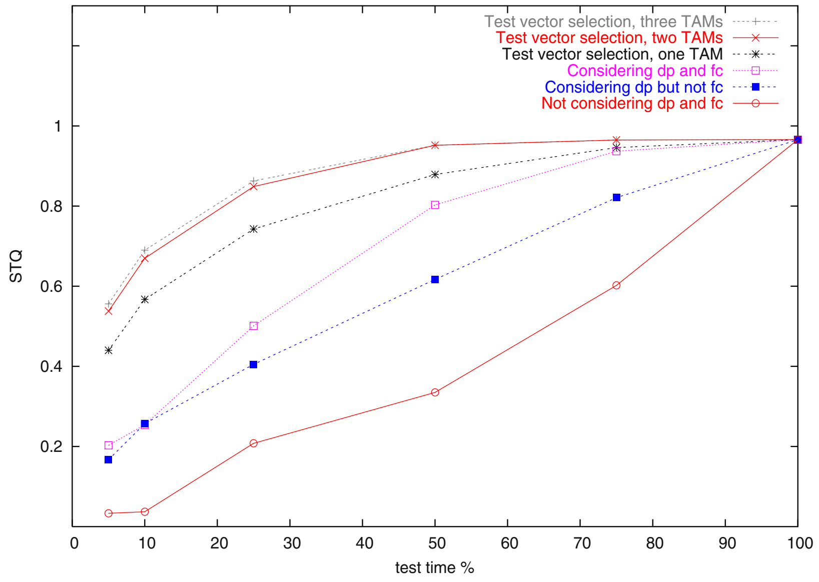


*Figure 239.*Design D695 - hard cores.

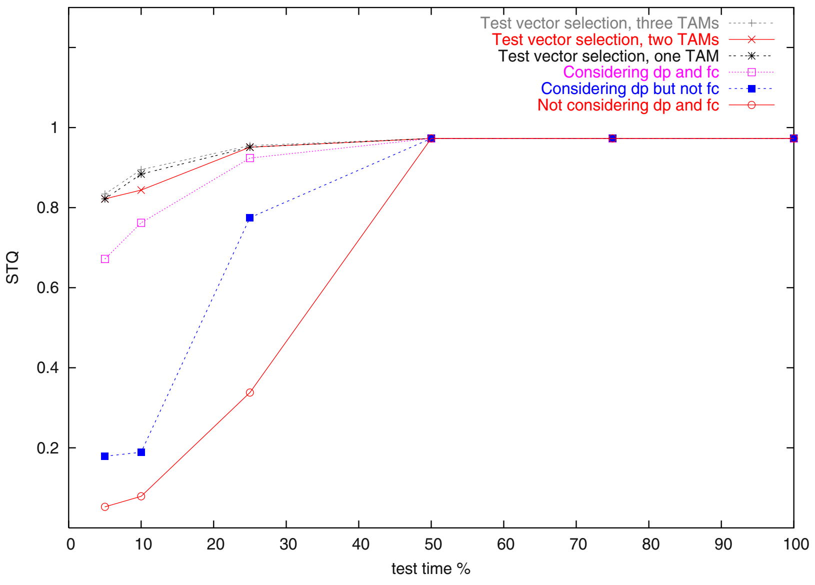


Figure 240. Design D22810 - soft cores.

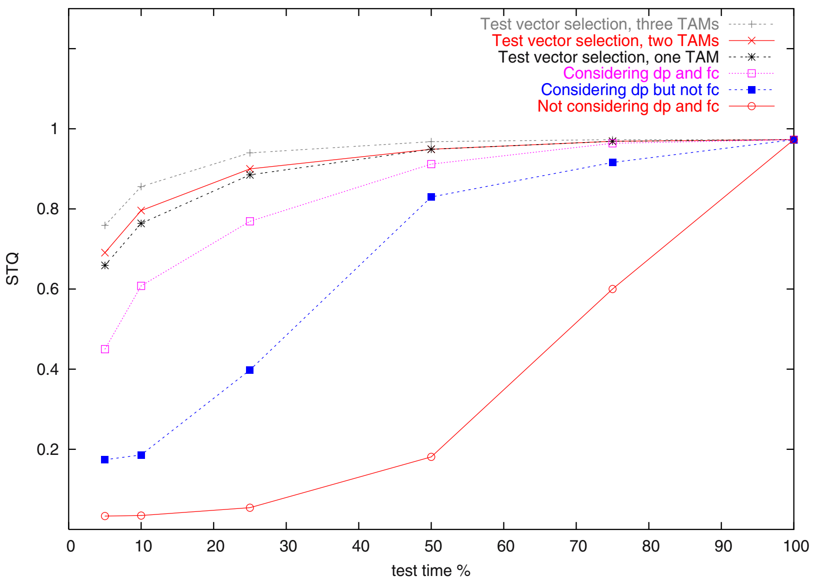


Figure 241. Design D22810 - hard cores.

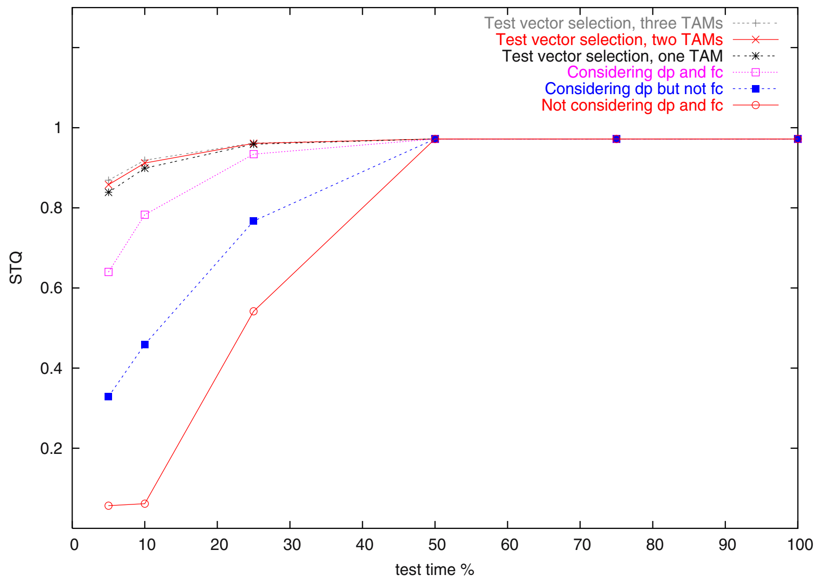


Figure 242. Design D34392 - soft cores.

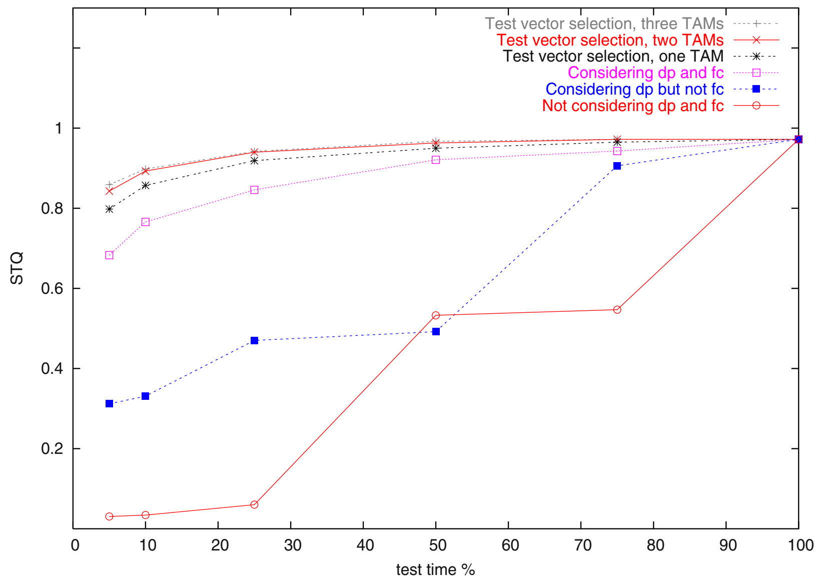


Figure 243. Design D34392 - hard cores.

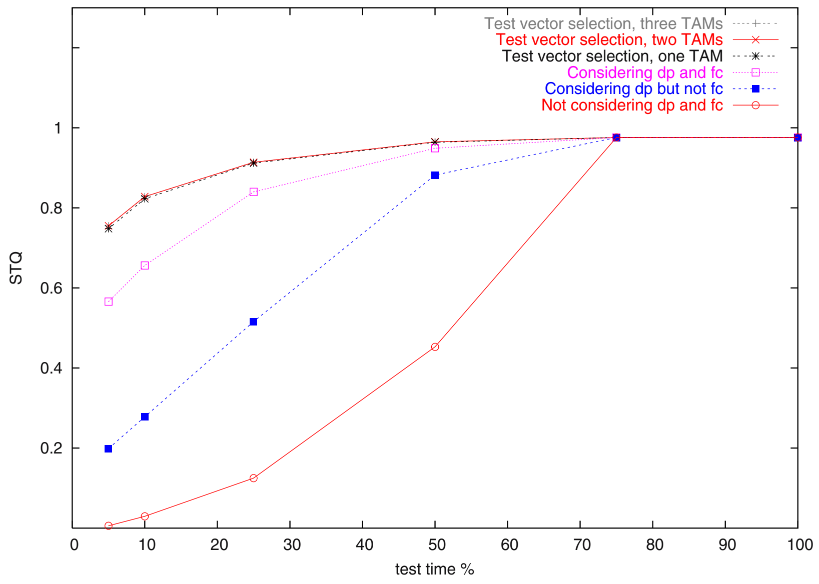


*Figure 244.*Design D93791 - soft cores.

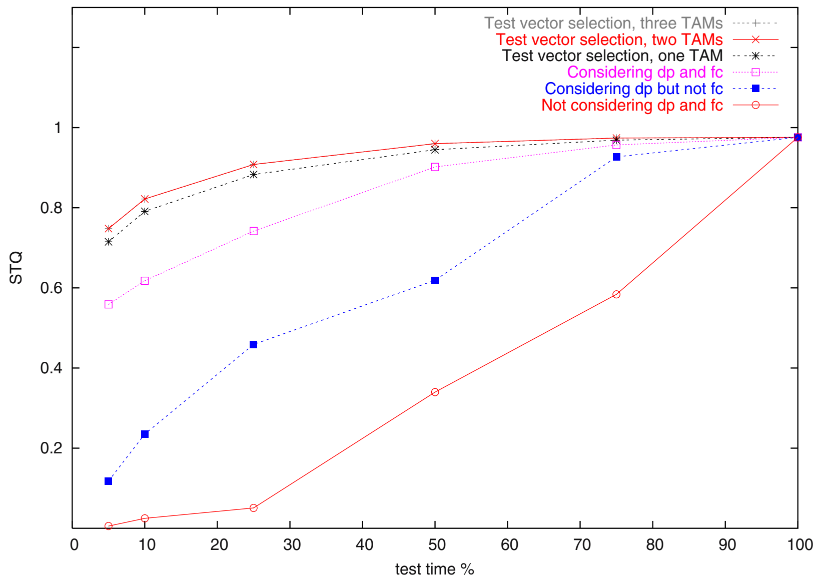


*Figure 245.*Design D93791 - hard cores.

Appendix 1

Benchmarks

1 INTRODUCTION

This is the inputfiles for the various designs used to get the results:

- Design Kime
- Design Muresan 10
- Design Muresan 20
- ASIC Z
- Extended ASIC Z
- Ericsson
- System S (and variants)
- System L

2 FORMAT OF THE INPUTFILE

The inputfile is divided in sections, in the hope to make it clear, and easy to extend. Each section is started with a *[SectionName]*, and is parsed row by row, and the content is interpreted according to each sections predefined format. Different values is separated by one or more spaces, lists within '{}', empty lines are ignored and everything it case-sensitive. The sign '#' causes the rest of the line to be ignored and can be used for comments.

The format of each section is described below, every single name identifying a core, block, test, etc. is required to be unique.

[Global Constraints]

Rowformat: `identifier = value`

Identifier	Value	Default	Description
MaxPower	Integer	0	Highest possible total powerconsumtion.
OptimalTime	Integer	Perfect	Optimal testtime. (Only purpose is for comparisions of results.)
CmpTestTest	Yes / No	Yes	Don't allow tests at the same block to execute concurrently.
CmpTestCts	Yes / No	No	Don't allow a test if it's block is among the concurrency constraints of an simultaneous test and vice vera.
CmpCtsCts	Yes / No	Yes	Don't allow concurrent scheduling of tests with common blocks in the concurrency constraints.
DoBusPlan	Yes / No	No	Create a buslayout.
BusFactor	Integer	1	Costfactor for buses.
TimeFactor	Integer	1	Costfactor for testtime.
AvgNodeDist	Integer	Auto	Average distance between nodes (cores, TPG, TRE).

[Cores]

Rowformat: name x y blocklist
`blocklist` specified as `{block1 block2 ... blockN}`

Value	*Description*
name	Name identifying the core.
x	Integer giving the x-coordinate of the cores position.
y	Integer giving the y-coordinate of the cores position.
blocklist	Names (at least one) of the blocks part of this core.

[Generators] and [Evaluators

Rowformat: `name x y bmin bmax mem movable`

Value	*Description*
name	Name identifying the generator or eveluator.
x	Integer giving the x-coordinate of the position.
y	Integer giving the y-coordinate of the position.
bmax	Integer giving the highest possible bandwidth (0 indicates infinity)
mem	This is only for generators and specifies the amount of memory available in the generator.
movable	Yes or No, Specifies if the position is allowed to be changed by the algorithm. Not implemented.

[Tests]

Rowformat: `name power time tpg tre bmin bmax mem icore`

Value	*Description*
name	Name identifying a test.
power	Integer giving the amount of power consumed by the test.
time	Integer giving the time a test needs to complete.
tpg	The name of the used generator.
tre	The name of the used evaluator.
bmin	Integer giving the (minimal) required bandwidth.
bmax	Integer giving the possible (maximal) bandwidth (0 indicates infinity).
mem	The amount of memory allocated in the generator.
icore	The name of a core to perform interconnection test to, else 'no' or left out.

[Blocks]

Rowformat: `block idle testlist`
`testlist` specified as `{test1 test2 ... testN}`

Value	*Description*
block	Name identifying a block.
idle	Integer giving the amount ow idle power used by the block.
testlist	The names of the tests supposed to run on the block.

[Concurrency Constraints]

Rowformat: `test blocklist`
`blocklist` specified as `{block1 block2 ... blockN}`

Value	*Description*
test	The name of a test.
blocklist	The names of the blocks part of this tests concurrency constraints.

3 DESIGN KIME

The test compatibility graph of a design with six tests is taken from Kime and Saluja [132], see Figure 1.1.. Test t_1 and t_6 may be scheduled concurrently since an arc exists between node t_1 and node t_6. On the other hand, test t_1 and t_2 may not be scheduled concurrently since no arc exists between the node t_1 and node t_2. Each node has its test time attached to it. For instance, test t_1 requires 255 time units.

```
# Design Kime
[Global Options]
MaxPower = 2      # Maximal allowed simultaneous power
[Cores]
#Syntax: name x y {block1 block2 ... blockN}
N1 10 10 {n1}
N2 20 10 {n2}
N3  0  0 {n3}
N4 10  0 {n4}
N5 20  0 {n5}
N6  0 10 {n6}
[Generators]
#Syntax: name x y max_bandw max_mem movable
TC  0 20 2 6 no
```

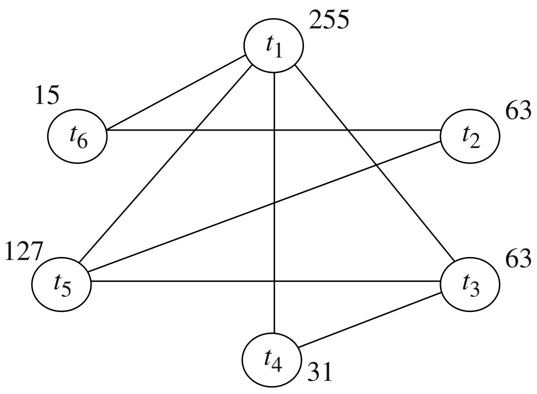


*Figure 1.1.*Test compatibility graph.

```
[Evaluators]
#Syntax: name x y max_bandw movable
SA 20 20 6 no
[Tests]
#Syntax:
#name power time tpg tre min_bandw max_bandw mem #ict_to_core
t1 1 255 TC SA 1 1 1
t2 1  63 TC SA 1 1 1
t3 1  63 TC SA 1 1 1
t4 1  31 TC SA 1 1 1
t5 1 127 TC SA 1 1 1
t6 1  15 TC SA 1 1 1
[Blocks]
#Syntax: name idle_power {test1 test2 ... testN}
n1 0 {t1}
n2 0 {t2}
n3 0 {t3}
n4 0 {t4}
n5 0 {t5}
n6 0 {t6}
[Concurrency Constraints]
#Syntax: test {block1 block2 ... blockN}
t1 {   n2}
t2 {n1    n3 n4}
t3 {   n2          n6}
t4 {   n2       n5 n6}
t5 {         n4    n6}
t6 {      n3 n4 n5}
```

4 DESIGN MURESAN 10

Muresan *et al.* present a design with the design data presented in Table 1.2. [202]. For instance, test t_2 requires 8 time units and 4 power units and it is test compatible with the following tests: $\{t_1, t_3, t_7, t_9\}$. For instance, it means that test t_2 can be scheduled at the same time as test t_3.

The power limit for the design is 12 power units.

Test	*Test time*	*Test power*	*Test Compatibility*
t_1	9	9	$t_2, t_3, t_5, t_6, t_8, t_9$
t_2	8	4	t_1, t_3, t_7, t_8
t_3	8	1	$t_1, t_2, t_4, t_7, t_9, t_{10}$
t_4	6	6	t_3, t_5, t_7, t_8
t_5	5	5	t_1, t_4, t_9, t_{10}
t_6	4	2	t_1, t_7, t_8, t_9
t_7	3	1	$t_2, t_3, t_4, t_6, t_8, t_9$
t_8	2	4	$t_1, t_2, t_4, t_6, t_7, t_9, t_{10}$
t_9	1	12	$t_1, t_3, t_5, t_6, t_7, t_8, t_{10}$
t_{10}	1	7	t_3, t_5, t_8, t_9

*Figure 1.2.*Design data for design Muresan.

```
[Global Options]
MaxPower = 12       # Maximal allowed simultaneous power
OptimalTime = 25
TimeFactor = 20
BusFactor  = 1
[Cores]
#Syntax: name x y {block1 block2 ... blockN}
C1  10   10   {B1}
C2  20   10   {B2}
C3  30   10   {B3}
C4  40   10   {B4}
C5  50   10   {B5}
C6  10   20   {B6}
C7  20   20   {B7}
C8  40   20   {B8}
C9  40   30   {B9}
C0  50   40   {B0}
[Generators]
#Syntax: name x y max_bandw max_mem movable
TCG   50   20   8   32   yes
[Evaluators]
#Syntax: name x y max_bandw movable
TCE   50   20   8   no
```

```
[Tests]
#Syntax:
# name power time tpg tre min_bandw max_bandw mem ict_to_core
t1  9  9    TCG TCE    1         1       1     no
t2  4  8    TCG TCE    1         1       1     no
t3  1  8    TCG TCE    1         1       1     no
t4  6  6    TCG TCE    1         1       1     no
t5  5  5    TCG TCE    1         1       1     no
t6  2  4    TCG TCE    1         1       1     no
t7  1  3    TCG TCE    1         1       1     no
t8  4  2    TCG TCE    1         1       1     no
t9 12  1    TCG TCE    1         1       1     no
t0  7  1    TCG TCE    1         1       1     no
[Blocks]
#Syntax: name idle_power {test1 test2 ... testN}
B1 0 {t1}
B2 0 {t2}
B3 0 {t3}
B4 0 {t4}
B5 0 {t5}
B6 0 {t6}
B7 0 {t7}
B8 0 {t8}
B9 0 {t9}
B0 0 {t0}
[Concurrency Constraints]
#Syntax: test {block1 block2 ... blockN}
# Note that the constraints are symmetric around the #diagonal,
and only one half is actually needed.
t1 {         B4       B7       B0}
t2 {         B4 B5 B6       B9 B0}
t3 {            B5 B6    B8      }
t4 {B1 B2             B6       B9 B0}
t5 {   B2 B3       B6 B7 B8      }
t6 {   B2 B3 B4 B5             B0}
t7 {B1          B5             B0}
t8 {      B3    B5               }
t9 {   B2    B4                  }
t0 {B1 B2    B4    B6 B7         }
```

5 DESIGN MURESAN 20

```
[Global Options]
MaxPower = 15      # Maximal allowed simultaneous power
[Cores]
#Syntax: name x y {block1 block2 ... blockN}
CORE 0 0 {b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15 b16
b17 b18 b19 b20}
[Generators]
#Syntax: name x y max_bandw max_mem movable
TG  0 0 20 20 no
[Evaluators]
#Syntax: name x y max_bandw movable
SA 0 0 20 no
[Tests]
#Syntax:
# name power time tpg tre min_bandw max_bandw mem ict_to_core
```

```
t1   3 12  TG SA 1 1 1
t2   5 11  TG SA 1 1 1
t3   9  9  TG SA 1 1 1
t4  12  8  TG SA 1 1 1
t5   4  8  TG SA 1 1 1
t6   2  8  TG SA 1 1 1
t7   1  8  TG SA 1 1 1
t8   7  6  TG SA 1 1 1
t9   6  6  TG SA 1 1 1
t10  7  5  TG SA 1 1 1
t11  5  5  TG SA 1 1 1
t12 11  4  TG SA 1 1 1
t13  2  4  TG SA 1 1 1
t14  3  3  TG SA 1 1 1
t15  1  3  TG SA 1 1 1
t16  5  2  TG SA 1 1 1
t17  4  2  TG SA 1 1 1
t18 12  1  TG SA 1 1 1
t19  8  1  TG SA 1 1 1
t20  7  1  TG SA 1 1 1
[Blocks]
#Syntax: name idle_power {test1 test2 ... testN}
b1 0 {t1}
b2 0 {t2}
b3 0 {t3}
b4 0 {t4}
b5 0 {t5}
b6 0 {t6}
b7 0 {t7}
b8 0 {t8}
b9 0 {t9}
b10 0 {t10}
b11 0 {t11}
b12 0 {t12}
b13 0 {t13}
b14 0 {t14}
b15 0 {t15}
b16 0 {t16}
b17 0 {t17}
b18 0 {t18}
b19 0 {t19}
b20 0 {t20}
[Concurrency Constraints]
#Syntax: test {block1 block2 ... blockN}
t1  { b2 b3 b6 b7 b11 b13 b14 b18 }
t2  { b6 b7 b8 b10 b11 b15 b16 b18 }
t3  { b4 b6 b8 b9 b15 b16 b19 b20 }
t4  { b5 b6 b8 b10 b12 b13 b16 b18 b20 }
t5  { b9 b10 b11 b13 b14 b16 b19 }
t6  { b8 b10 b12 b13 b15 b16 b18 b19 }
t7  { b8 b10 b11 b13 b17 }
t8  { b12 b13 b15 b18 }
t9  { b10 b13 b14 b16 b18 b20 }
t10 { b12 b13 b14 b19 b20 }
t11 { b12 b13 b15 b17 b19 }
t12 { b15 b17 b18 b20 }
t13 { b14 b20 }
t14 { b15 b17 b19 }
t15 { b19 b20 }
t16 { b18 }
```

```
t17 { }
t18 { }
t19 { }
t20 { }
```

6 ASIC Z

The ASIC Z design presented by Zorian [287] with the estimations on test length made by Chou *et al*. is in Figure 1.3. and Table 1.4. The power consumption for each block when it is in idle mode and for each test when it is in test mode is given by Zorian. The test lengths for each test is computed by Chou *et al*. with an assumption of linear dependency between test length and block size, see Table 1.4. [40,41].

The design originally consists of 10 cores. However, no data is available for one block therefore it is excluded from the design. The maximal allowed power dissipation of the system is 900 mW. All blocks have their own dedicated BIST which means that all tests can be scheduled concurrently.

We have added the placement, see Table 1.4., where each block is given an x-placement and a y-placement.

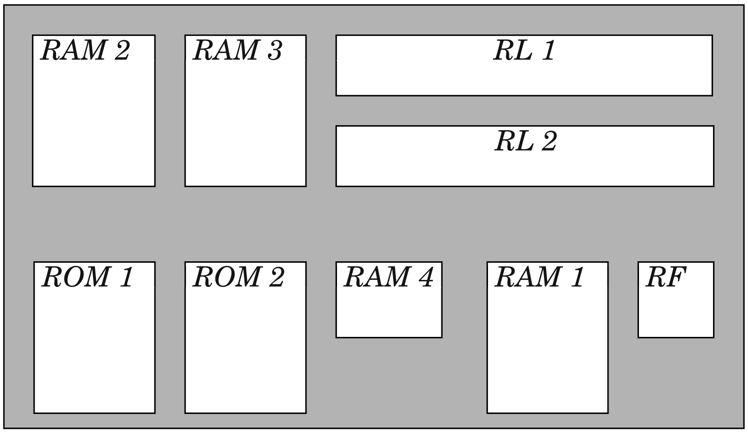


*Figure 1.3.*ASIC Z floor-plan.

```
[Global Options]
MaxPower = 900     # Maximal allowed simultaneous power
[Cores]
#Syntax: name x y {block1 block2 ... blockN}
ROM1  10   10   {rom1_1}
ROM2  20   10   {rom2_1}
RAM4  30   10   {ram4_1}
RAM1  40   10   {ram1_1}
RF    50   10   {rf_1}
RAM2  10   20   {ram2_1}
```

Block	*Size*	*Test Time*	*Idle Power*	*Test Power*	*Placement*	
					x	*y*
RL1	13400 gates	134	0	295	40	30
RL2	16000 gates	160	0	352	40	20
RF	64 × 17 bits	10	19	95	50	10
RAM1	768 × 9 bits	69	20	282	40	10
RAM2	768 × 8 bits	61	17	241	10	20
RAM3	768 × 5 bits	38	11	213	20	20
RAM4	768 × 3 bits	23	7	96	30	10
ROM1	1024 × 10 bits	102	23	279	10	10
ROM2	1024 × 10 bits	102	23	279	20	10

Figure 1.4. ASIC Z characteristics.

```
RAM3  20   20   {ram3_1}
RL2   40   20   {rl2_1}
RL1   40   30   {rl1_1}
[Generators]
#Syntax: name x y max_bandw max_mem movable
TCG   50   20   9   32   yes
[Evaluators]
#Syntax: name x y max_bandw movable
TCE   50   20   9   no
[Tests]
#Syntax:
# name power time tpg tre min_bandw max_bandw mem ict_to_core
tROM1   279  102  TCG TCE    1         1       4     no
tROM2   279  102  TCG TCE    1         1       2     no
tRAM4    96   23  TCG TCE    1         1       2     no
tRAM1   282   69  TCG TCE    1         1       4     no
tRF      95   10  TCG TCE    1         1       8     no
tRAM2   241   61  TCG TCE    1         1       2     no
tRAM3   213   38  TCG TCE    1         1       2     no
tRL2    352  160  TCG TCE    1         1       4     no
tRL1    295  134  TCG TCE    1         1       4     no
[Blocks]
#Syntax: name idle_power {test1 test2 ... testN}
  rom1_1 23 {tROM1}
  rom2_1 23 {tROM2}
  ram4_1  7 {tRAM4}
  ram1_1 20 {tRAM1}
  rf_1   19 {tRF}
  ram2_1 17 {tRAM2}
  ram3_1 11 {tRAM3}
  rl2_1   0 {tRL2}
  rl1_1   0 {tRL1}
[Concurrency Constraints]
#Syntax: test {block1 block2 ... blockN}
tRAM1  {ram1_1 ram2_1 ram3_1 ram4_1}
tRAM2  {ram1_1 ram2_1 ram3_1 ram4_1}
```

```
tRAM3  {ram1_1 ram2_1 ram3_1 ram4_1}
tRAM4  {ram1_1 ram2_1 ram3_1 ram4_1}
```

7 EXTENDED ASIC Z

The Extended ASIC Z design is an extended version of ASIC Z, see section 6. For each core three tests are defined:

- an interconnection test,
- an BIST test, and
- an external test.

In total there are 27 tests spread over the 9 cores. The maximal power consumption and placement is assumed to be the same as for ASIC Z. The characteristics for Extended ASIC Z are in Table 1.5. For instance, a BIST test at RL1 require test generator TG_{rl1} and test analyser TA_{rl1}. The test takes 67 time units and consumes 295 mW and when it is applied no other tests at RL1 can be performed.

The interconnection tests are performed between two cores. For instance core RL1 performs an interconnection test with RL2 which requires 10 time units and 10 mW. When this test is applied it is assumed that no other test can be performed at RL1 and RL2 (specified under block constraint in Table 1.5.).

In this design the BIST resources are shared and each BIST resources can be used by one test at a time. For instance when RAM1 is tested using TG_{ram} and TA_{ram} no other tests can be performed using these test resources. The external tests are connected through TAP and several tests can be applied concurrently using the external tester. For Extended ASIC Z all tests at a core are at one block which means that the BIST and the external test may not be scheduled concurrently.

Core	*Test time*	*Test power*	*Test source*	*Test sink*	*Block constraint*
RL1	67	295	TAP	TAP	RL1
	67	295	TG_{rl1}	TA_{rl1}	RL1
	10	10	TAP	TAP	RL1, RL2
RL2	80	352	TAP	TAP	RL2
	80	352	TG_{rl2}	TA_{rl2}	RL2
	10	10	TAP	TAP	RL2, RAM3
RF	5	95	TAP	TAP	RF
	5	95	TG_{rf}	TA_{rf}	RF
	10	10	TAP	TAP	RF,RL1
RAM1	35	282	TAP	TAP	RAM1
	35	282	TG_{ram}	TA_{ram}	RAM1
	10	10	TAP	TAP	RAM1,RF
RAM2	31	241	TAP	TAP	RAM2
	31	241	TG_{ram}	TA_{ram}	RAM2
	10	10	TAP	TAP	RAM2, ROM1
RAM3	19	213	TAP	TAP	RAM3
	19	213	TG_{ram}	TA_{ram}	RAM3
	10	10	TAP	TAP	RAM3, RAM2
RAM4	12	96	TAP	TAP	RAM4
	12	96	TG_{ram}	TA_{ram}	RAM4
	10	10	TAP	TAP	RAM4, RAM1
ROM1	51	279	TAP	TAP	ROM1
	51	279	TG_{rom}	TA_{rom}	ROM1
	10	10	TAP	TAP	ROM1, ROM2
ROM2	51	279	TAP	TAP	ROM2
	51	279	TG_{rom}	TA_{rom}	ROM2
	10	10	TAP	TAP	ROM2, RAM4

*Figure 1.5.*Extended ASIC Z characteristics.

8 SYSTEM L

System L is an industrial design consisting of 14 cores that are named A through N, see Table 50. The system is tested by 17 tests distributed over the system as block-level tests and top-level tests. The block-level tests and the top-level tests can not be executed simultaneously. Furthermore, all block-level using the test bus can not be executed concurrently. The top-level tests are using the functional pins which makes concurrent scheduling among them impossible.

All tests are using external test resources and the total power limit for the system is 1200 mW.

Test	*Block*	*Test*	*Test time*	*Idle power*	*Test power*	*Test port*
Block-level tests	A	Test A	515	1	379	scan
	B	Test B	160	1	205	test-bus
	C	Test C	110	1	23	test-bus
	D	Test D	Tested as part of other top-level test			
	E	Test E	61	1	57	test-bus
	F	Test F	38	1	27	test-bus
	G	Test G	Tested as part of other top-level test			
	H	Test H	Tested as part of other top-level test			
	I	Test I	29	1	120	test-bus
	J	Test J	6	1	13	test-bus
	K	Test K	3	1	9	test-bus
	L	Test L	3	1	9	test-bus
	M	Test M	218	1	5	test-bus
Top-level tests	A	Test N	232	1	379	functional pins
	N	Test O	41	1	50	functional pins
	B	Test P	72	1	205	functional pins
	D	Test Q	104	1	39	functional pins

Table 50. System L characteristics.

```
# System L
[Global Options]
MaxPower = 1200    # Maximal allowed simultaneous power
[Cores]
#Syntax: name x y {block1 block2 ... blockN}
A 10   0   {a}
B 20   0   {b}
C 30   0   {c}
D 40   0   {d}
E  0  10   {e}
F 10  10   {f}
G 20  10   {g}
H 30  10   {h}
I 40  10   {i}
J 50  10   {j}
K 10  20   {k}
L 20  20   {l}
M 30  20   {m}
N 40  20   {n}
[Generators]
#Syntax: name x y max_bandw max_mem movable
scanG  10   0   1   32   no
busG    0   0   1   32   no
pinsG   0  20   1   32   no
[Evaluators]
#Syntax: name x y max_bandw movable
scanE  10   0   1   no
busE   50  20   1   no
pinsE  50   0   1   no
[Tests]
#Syntax:
# name power time tpg tre min_bandw max_bandw mem i_t_c
ta  379  515   scanG scanE    111     no
tb  205  160   busG busE 1 11     no
tc   23  110   busG busE    111     no
td    0    0   busG busE    111     no
te   57   61   busG busE    111     no
tf   27   38   busG busE    111     no
tg    0    0   busG busE    111     no
th    0    0   busG busE    111     no
ti  120   29   busG busE    111     no
tj   13    6   busG busE    111     no
tkl   9    3   busG busE    111     no
tm    5  218   busG busE    111     no
tn   50   41   pinsG pinsE    111     no
to  379  232   pinsG pinsE    111     no
tp  205   72   pinsG pinsE    111     no
tq   39  104   pinsG pinsE 111     no
[Blocks]
#Syntax: name idle_power {test1 test2 ... testN}
a 1 {ta to}
b 1 {tb tp}
c 1 {tc}
d 1 {   tq}
e 1 {te}
f 1 {tf}
g 1 {}
h 1 {}
i 1 {ti}
j 1 {tj}
```

```
k 1 {tkl}
l 1 {tkl}
m 1 {tm}
n 1 {tn}
[Concurrency Constraints]
#Syntax: test {block1 block2 ... blockN}
ta {a}
tb {  b c d e f g h i j k l m n}
tc {  b c d e f g h i j k l m n}
td {}
te {  b c d e f g h i j k l m n}
tf {  b c d e f g h i j k l m n}
tg {}
th {}
ti {  b c d e f g h i j k l m n}
tj {  b c d e f g h i j k l m n}
tkl{  b c d e f g h i j k l m n}
tm {  b c d e f g h i j k l m n}
tn {a b c d e f g h i j k l m n}
to {a b c d e f g h i j k l m n}
tp {a b c d e f g h i j k l m n}
tq {a b c d e f g h i j k l m n}
```

9 ERICSSON DESIGN

The Ericsson design, see Figure 1.6., consists of 8 digital signal processor (DSP) cores: a block for DSP control (DSPIOC); 2 memory banks, a common program memory (CPM) and common data memory (CDM); a control unit for each memory bank, common data memory controller (CDMC) and common program memory controller (CPMC); and five other blocks, RX1C, RX0C, DMAIOC, CKReg and TXC. In total there are 18 cores.

Each of the DSP cores in the Ericsson design in Figure 1.6. consists of four banks of local data memory (LDM), one bank of local program memory and two banks of other memory (LZM) and five logic blocks, see Figure 1.7. The memory banks of the CPM block and the CDM block in Figure 1.6. are shown in Figure 1.8. respectively in Figure 1.9.

The characteristics for each of the blocks in the design are in Table 1.1 where the test time, test power and test resource is specified for each block in the system. The idle power is zero for all blocks. The DSPs are numbered by n in range 0 to 7 which results in total 170 ($17 \times 7 + 51$) tests.

The maximal allowed power consumption is limited to 5125 mW. For each logic block two test sets are applied. One using an external tester and one on-chip tester. These tests can not be applied at the same time since they test the same logic. All logic blocks within a DSP core share one test source and test sink for the on-chip test. The connection to the external tester is named TAP and several tests may use the external tester concurrently.

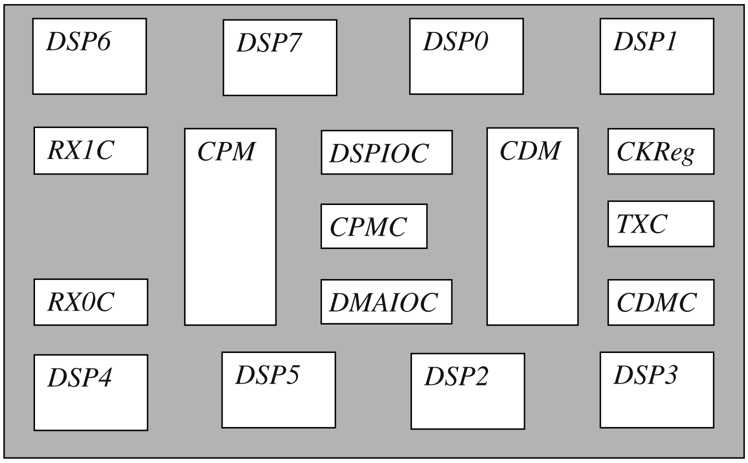


Figure 1.6. The Ericsson design.

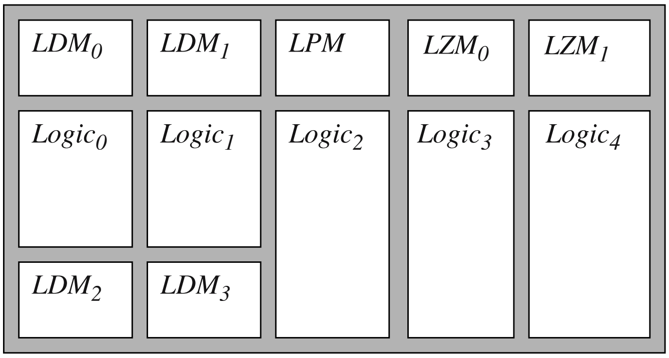


Figure 1.7. The blocks within each DSP_n.

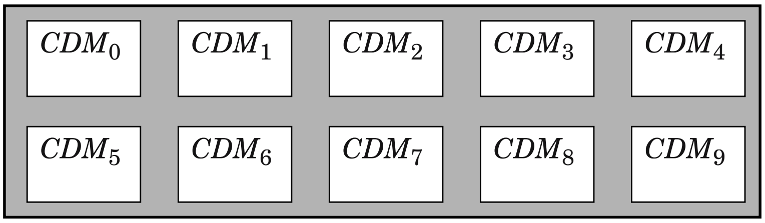


Figure 1.9. The common data memory bank.

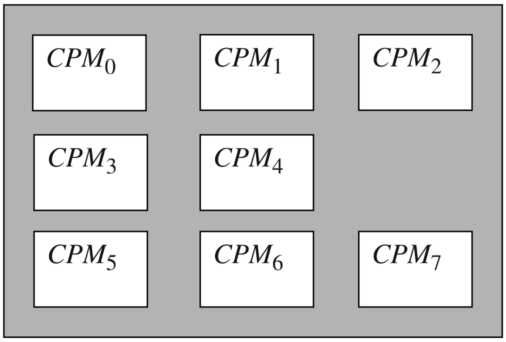


*Figure 1.8.*The blocks within CPM.

All memory blocks of the same type have their own test resources. For instance, the blocks within the CPM have one test generator and one test response analyser. The placement of all blocks are in Table 1.2.

Block	*Test*	*Test time*	*Test power*	*Test source*	*Test sink*
RX0C	1	970	375	TAP	TAP
	2	970	375	TG_0	TRA_0
RX1C	3	970	375	TAP	TAP
	4	970	375	TG_0	TRA_0
DSPIOC	5	1592	710	TAP	TAP
	6	1592	710	TG_0	TRA_0
CPMC	7	480	172	TAP	TAP
	8	480	172	TG_0	TRA_0
DMAIOC	9	3325	207	TAP	TAP
	10	3325	207	TG_0	TRA_0
CKReg	11	505	118	TAP	TAP
	12	505	118	TG_0	TRA_0
CDMC	13	224	86	TAP	TAP
	14	224	86	TG_0	TRA_0
TXC	15	364	140	TAP	TAP
	16	364	140	TG_0	TRA_0
CPM_i	17+i	239	80	TG_1	TRA_1
CDM_j	25+j	369	64	TG_1	TRA_1

Table 1.1. Design characteristics Ericsson.

Block		*Test*	*Test time*	*Test power*	*Test source*	*Test sink*
DSP_n	LPM	17×n+35	46	16	$TG_{n,0}$	$TRA_{n,0}$
	LDM_l	17×n+l+36	92	8	$TG_{n,0}$	$TRA_{n,0}$
	LZM_m	17×n+m+40	23	2	$TG_{n,0}$	$TRA_{n,0}$
	$Logic_0$	17×n+42	4435	152	TAP	TAP
		17×n+43	4435	152	$TG_{n,1}$	$TRA_{n,1}$
	$Logic_1$	17×n+44	4435	152	TAP	TAP
		17×n+45	4435	152	$TG_{n,1}$	$TRA_{n,1}$
	$Logic_2$	17×n+46	7009	230	TAP	TAP
		17×n+47	7009	230	$TG_{n,1}$	$TRA_{n,1}$
	$Logic_3$	17×n+48	7224	250	TAP	TAP
		17×n+49	7224	250	$TG_{n,1}$	$TRA_{n,1}$
	$Logic_4$	17×n+50	7796	270	TAP	TAP
		17×n+51	7796	270	$TG_{n,1}$	$TRA_{n,1}$

Table 1.1. Design characteristics Ericsson.

Block	*X*	*Y*	*Block*	*X*	*Y*
TG6	0	0	TG0	80	0
TG6L	10	0	TG0L	90	0
DSP6LDM1	20	0	DSP0LDM1	100	0
DSP6LDM2	30	0	DSP0LDM2	110	0
DSP6LDM3	0	10	DSP0LDM3	80	10
DSP6LDM4	10	10	DSP0LDM4	90	10
DSP6LPM	20	10	DSP0LPM	100	10
DSP6LZM1	30	10	DSP0LZM1	110	10
DSP6LZM2	0	20	DSP0LZM2	80	20
DSP6L1	10	20	DSP0L1	90	20
DSP6L2	20	20	DSP0L2	100	20
DSP6L3	30	20	DSP0L3	110	20
DSP6L4	0	30	DSP0L4	80	30
DSP6L5	10	30	DSP0L5	90	30
SA6	20	30	SA0	100	30
SA6L	30	30	SA0L	110	30
TG7	40	0	TG1	120	0
TG7L	50	0	TG1L	130	0
DSP7LDM1	60	0	DSP1LDM1	140	0
DSP7LDM2	70	0	DSP1LDM2	150	0
DSP7LDM3	40	10	DSP1LDM3	120	10
DSP7LDM4	50	10	DSP1LDM4	130	10
DS7LPM	60	10	DSP1LPM	140	10
DSP7LZM1	70	10	DSP1LZM1	150	10
DSP7LZM2	40	20	DSP1LZM2	120	20
DSP7L1	50	20	DSP1L1	130	20
DSP7L2	60	20	DSP1L2	140	20
DSP7L3	70	20	DSP1L3	150	20
DSP7L4	40	30	DSP1L4	120	30

Table 1.2. Placement characteristics Ericsson.

Block	*X*	*Y*	*Block*	*X*	*Y*
DSP7L5	50	30	DSP1L5	130	30
SA7	60	30	SA1	140	30
SA7L	70	30	SA1L	150	30
TG4	0	60	TG2	80	60
TG4L	10	60	TG2L	90	60
DSP4LDM1	20	60	DSP2LDM1	100	60
DSP4LDM2	30	60	DSP2LDM2	110	60
DSP4LDM3	0	70	DSP2LDM3	80	70
DSP4LDM4	10	70	DSP2LDM4	90	70
DSP4LPM	20	70	DSP2LPM	100	70
DSP4LZM1	30	70	DSP2LZM1	110	70
DSP4LZM2	0	80	DSP2LZM2	80	80
DSP4L1	10	80	DSP2L1	90	80
DSP4L2	20	80	DSP2L2	100	80
DSP4L3	30	80	DSP2L3	110	80
DSP4L4	0	90	DSP2L4	80	90
DSP4L5	10	90	DSP2L5	90	90
SA4	20	90	SA2	100	90
SA4L	30	90	SA2L	110	90
TG5	40	60	TG3	120	60
TG5L	50	60	TG3L	130	60
DSP5LDM1	60	60	DSP3LDM1	140	60
DSP5LDM2	70	60	DSP3LDM2	150	60
DSP5LDM3	40	70	DSP3LDM3	120	70
DSP5LDM4	50	70	DSP3LDM4	130	70
DS5LPM	60	70	DSP3LPM	140	70
DSP5LZM1	70	70	DSP3LZM1	150	70
DSP5LZM2	40	80	DSP3LZM2	120	80
DSP5L1	50	80	DSP3L1	130	80

Table 1.2. Placement characteristics Ericsson.

Block	*X*	*Y*	*Block*	*X*	*Y*
DSP5L2	60	80	DSP3L2	140	80
DSP5L3	70	80	DSP3L3	150	80
DSP5L4	40	90	DSP3L4	120	90
DSP5L5	50	90	DSP3L5	130	90
SA5	60	90	SA3	140	90
SA5L	70	90	SA3L	150	90
TG8b	0	40	CDM4	0	50
TG9b	10	40	CDM5	10	50
TG10	20	40	CDM6	20	50
CPM0	30	40	CDM7	30	50
CPM1	40	40	CDM8	40	50
CPM2	50	40	RX0C	50	50
CPM3	60	40	RX1C	60	50
CPM4	70	40	CPMC	70	50
CPM5	80	40	DSPIOC	80	50
CPM6	90	40	DMAIOC	90	50
CPM7	100	40	CDMC	100	50
CPM8	110	40	TXC	110	50
CPM9	120	40	CKREG	120	50
CDM1	130	40	SA8b	130	50
CDM2	140	40	SA10	140	50
CDM3	150	40	TAP	150	50

Table 1.2. Placement characteristics Ericsson.

```
#
# Ericsson
[Global Options]
DesignName = "Ericsson"
MaxPower = 5125
[Cores]
DSP0ldm0 100 0 {ldm00}
DSP0ldm1 110 0 {ldm10}
DSP0ldm2  80 10 {ldm20}
DSP0ldm3  90 10 {ldm30}
DSP0lpm  100 10 {lpm0}
DSP0lzm0 110 10 {lzm00}
DSP0lzm1  80 20 {lzm10}
DSP0logic0  90 20 {logic00}
DSP0logic1 100 20 {logic10}
DSP0logic2 110 20 {logic20}
DSP0logic3  80 30 {logic30}
DSP0logic4  90 30 {logic40}
DSP1ldm0 140 0 {ldm01}
DSP1ldm1 150 0 {ldm11}
DSP1ldm2 120 10 {ldm21}
DSP1ldm3 130 10 {ldm31}
DSP1lpm  140 10 {lpm1}
DSP1lzm0 150 10 {lzm01}
DSP1lzm1 120 20 {lzm11}
DSP1logic0 130 20 {logic01}
DSP1logic1 140 20 {logic11}
DSP1logic2 150 20 {logic21}
DSP1logic3 120 30 {logic31}
DSP1logic4 130 30 {logic41}
DSP2ldm0 100  60 {ldm02}
DSP2ldm1 110  60 {ldm12}
DSP2ldm2  80  70 {ldm22}
DSP2ldm3  90  70 {ldm32}
DSP2lpm  100  70 {lpm2}
DSP2lzm0 110  70 {lzm02}
DSP2lzm1  80  80 {lzm12}
DSP2logic0  90  80 {logic02}
DSP2logic1 100  80 {logic12}
DSP2logic2 110  80 {logic22}
DSP2logic3  80  90 {logic32}
DSP2logic4  90  90 {logic42}
DSP3ldm0 140  60 {ldm03}
DSP3ldm1 150  60 {ldm13}
DSP3ldm2 120  70 {ldm23}
DSP3ldm3 130  70 {ldm33}
DSP3lpm  140  70 {lpm3}
DSP3lzm0 150  70 {lzm03}
DSP3lzm1 120  80 {lzm13}
DSP3logic0 130  80 {logic03}
DSP3logic1 140  80 {logic13}
DSP3logic2 150  80 {logic23}
DSP3logic3 120  90 {logic33}
DSP3logic4 130  90 {logic43}
DSP4ldm0 20  60 {ldm04}
DSP4ldm1 30  60 {ldm14}
DSP4ldm2  0  70 {ldm24}
DSP4ldm3 10  70 {ldm34}
DSP4lpm  20  70 {lpm4}
DSP4lzm0 30  70 {lzm04}
```

```
DSP4lzm1  0  80 {lzm14}
DSP4logic0 10  80 {logic04}
DSP4logic1 20  80 {logic14}
DSP4logic2 30  80 {logic24}
DSP4logic3  0  90 {logic34}
DSP4logic4 10  90 {logic44}
DSP5ldm0 60  60 {ldm05}
DSP5ldm1 70  60 {ldm15}
DSP5ldm2 40  70 {ldm25}
DSP5ldm3 50  70 {ldm35}
DSP5lpm  60  70 {lpm5}
DSP5lzm0 70  70 {lzm05}
DSP5lzm1 40  80 {lzm15}
DSP5logic0 50  80 {logic05}
DSP5logic1 60  80 {logic15}
DSP5logic2 70  80 {logic25}
DSP5logic3 40  90 {logic35}
DSP5logic4 50  90 {logic45}
DSP6ldm0 20  0 {ldm06}
DSP6ldm1 30  0 {ldm16}
DSP6ldm2  0 10 {ldm26}
DSP6ldm3 10 10 {ldm36}
DSP6lpm  20 10 {lpm6}
DSP6lzm0 30 10 {lzm06}
DSP6lzm1  0 20 {lzm16}
DSP6logic0 10 20 {logic06}
DSP6logic1 20 20 {logic16}
DSP6logic2 30 20 {logic26}
DSP6logic3  0 30 {logic36}
DSP6logic4 10 30 {logic46}
DSP7ldm0 60  0 {ldm07}
DSP7ldm1 70  0 {ldm17}
DSP7ldm2 40 10 {ldm27}
DSP7ldm3 50 10 {ldm37}
DSP7lpm  60 10 {lpm7}
DSP7lzm0 70 10 {lzm07}
DSP7lzm1 40 20 {lzm17}
DSP7logic0 50 20 {logic07}
DSP7logic1 60 20 {logic17}
DSP7logic2 70 20 {logic27}
DSP7logic3 40 30 {logic37}
DSP7logic4 50 30 {logic47}
CPM0 30 40 {cpm0}
CPM1 40 40 {cpm1}
CPM2 50 40 {cpm2}
CPM3 60 40 {cpm3}
CPM4 70 40 {cpm4}
CPM5 80 40 {cpm5}
CPM6 90 40 {cpm6}
CPM7 100 40 {cpm7}
CPM8 110 40 {cpm8}
CPM9 120 40 {cpm9}
CDM0 130 40 {cdm0}
CDM1 140 40 {cdm1}
CDM2 150 40 {cdm2}
CDM3 0  50 {cdm3}
CDM4 10 50 {cdm4}
CDM5 20 50 {cdm5}
CDM6 30 50 {cdm6}
CDM7 40 50 {cdm7}
```

```
CDMC  165 40 {cdmc}
CPMC   90 60 {cpmc}
RX0C   15 40 {rx0c}
RX1C   15 80 {rx1c}
DSPIOC 90 80 {dspioc}
DMAIOC 90 40 {dmaioc}
CKREG 165 80 {ckreg}
TXC   165 60 {txc}
[Generators]
ETG    150 50 12 0 no
BTG     20 40 1 0 no
CPMG     0 40 1 0 no
CDMG    10 40 1 0 no
DSPLG0  90  0 1 0 no
DSPMG0  80  0 1 0 no
DSPLG1 130  0 1 0 no
DSPMG1 120  0 1 0 no
DSPLG2  90 60 1 0 no
DSPMG2  80 60 1 0 no
DSPLG3 130 60 1 0 no
DSPMG3 120 60 1 0 no
DSPLG4  10 60 1 0 no
DSPMG4   0 60 1 0 no
DSPLG5  50 60 1 0 no
DSPMG5  40 60 1 0 no
DSPLG6  10  0 1 0 no
DSPMG6   0  0 1 0 no
DSPLG7  50  0 1 0 no
DSPMG7  40  0 1 0 no
[Evaluators]
ERE    150 50 12 no
BRE    145 50 1 no
CPME   130 50 1 no
CDME   140 50 1 no
DSPLE0 110 30 1 no
DSPME0 100 30 1 no
DSPLE1 150 30 1 no
DSPME1 140 30 1 no
DSPLE2 110 90 1 no
DSPME2 100 90 1 no
DSPLE3 150 90 1 no
DSPME3 140 90 1 no
DSPLE4  30 90 1 no
DSPME4  20 90 1 no
DSPLE5  70 90 1 no
DSPME5  60 90 1 no
DSPLE6  30 30 1 no
DSPME6  20 30 1 no
DSPLE7  70 30 1 no
DSPME7  60 30 1 no
[Tests]
etrx     375  970 ETG  ERE 1 1 0
btrx     375  970 BTG  BRE 1 1 0
etdspioc 710 1592 ETG  ERE 1 1 0
btdspioc 710 1592 BTG  BRE 1 1 0
etcpmc   172  480 ETG  ERE 1 1 0
btcpmc   172  480 BTG  BRE 1 1 0
etdmaioc 207 3325 ETG  ERE 1 1 0
btdmaioc 207 3325 BTG  BRE 1 1 0
etckreg  118  505 ETG  ERE 1 1 0
```

```
btckreg  118  505 BTG  BRE 1 1 0
etcdmc    86  224 ETG  ERE 1 1 0
btcdmc    86  224 BTG  BRE 1 1 0
ettxc    140  364 ETG  ERE 1 1 0
bttxc    140  364 BTG  BRE 1 1 0
btcpm     80  239 CPMG CPME 1 1 0
btcdm     64  369 CDMG CDME 1 1 0
etlogic0 152 4435 ETG  ERE 1 1 0
etlogic1 152 4435 ETG  ERE 1 1 0
etlogic2 230 7009 ETG  ERE 1 1 0
etlogic3 250 7224 ETG  ERE 1 1 0
etlogic4 270 7796 ETG  ERE 1 1 0
btlogic00 152 4435 DSPLG0 DSPLE0 1 1 0
btlogic10 152 4435 DSPLG0 DSPLE0 1 1 0
btlogic20 230 7009 DSPLG0 DSPLE0 1 1 0
btlogic30 250 7224 DSPLG0 DSPLE0 1 1 0
btlogic40 270 7796 DSPLG0 DSPLE0 1 1 0
btlogic01 152 4435 DSPLG1 DSPLE1 1 1 0
btlogic11 152 4435 DSPLG1 DSPLE1 1 1 0
btlogic21 230 7009 DSPLG1 DSPLE1 1 1 0
btlogic31 250 7224 DSPLG1 DSPLE1 1 1 0
btlogic41 270 7796 DSPLG1 DSPLE1 1 1 0
btlogic02 152 4435 DSPLG2 DSPLE2 1 1 0
btlogic12 152 4435 DSPLG2 DSPLE2 1 1 0
btlogic22 230 7009 DSPLG2 DSPLE2 1 1 0
btlogic32 250 7224 DSPLG2 DSPLE2 1 1 0
btlogic42 270 7796 DSPLG2 DSPLE2 1 1 0
btlogic03 152 4435 DSPLG3 DSPLE3 1 1 0
btlogic13 152 4435 DSPLG3 DSPLE3 1 1 0
btlogic23 230 7009 DSPLG3 DSPLE3 1 1 0
btlogic33 250 7224 DSPLG3 DSPLE3 1 1 0
btlogic43 270 7796 DSPLG3 DSPLE3 1 1 0
btlogic04 152 4435 DSPLG4 DSPLE4 1 1 0
btlogic14 152 4435 DSPLG4 DSPLE4 1 1 0
btlogic24 230 7009 DSPLG4 DSPLE4 1 1 0
btlogic34 250 7224 DSPLG4 DSPLE4 1 1 0
btlogic44 270 7796 DSPLG4 DSPLE4 1 1 0
btlogic05 152 4435 DSPLG5 DSPLE5 1 1 0
btlogic15 152 4435 DSPLG5 DSPLE5 1 1 0
btlogic25 230 7009 DSPLG5 DSPLE5 1 1 0
btlogic35 250 7224 DSPLG5 DSPLE5 1 1 0
btlogic45 270 7796 DSPLG5 DSPLE5 1 1 0
btlogic06 152 4435 DSPLG6 DSPLE6 1 1 0
btlogic16 152 4435 DSPLG6 DSPLE6 1 1 0
btlogic26 230 7009 DSPLG6 DSPLE6 1 1 0
btlogic36 250 7224 DSPLG6 DSPLE6 1 1 0
btlogic46 270 7796 DSPLG6 DSPLE6 1 1 0
btlogic07 152 4435 DSPLG7 DSPLE7 1 1 0
btlogic17 152 4435 DSPLG7 DSPLE7 1 1 0
btlogic27 230 7009 DSPLG7 DSPLE7 1 1 0
btlogic37 250 7224 DSPLG7 DSPLE7 1 1 0
btlogic47 270 7796 DSPLG7 DSPLE7 1 1 0
btlpm0     16   46 DSPMG0 DSPME0 1 1 0
btlpm1     16   46 DSPMG1 DSPME1 1 1 0
btlpm2     16   46 DSPMG2 DSPME2 1 1 0
btlpm3     16   46 DSPMG3 DSPME3 1 1 0
btlpm4     16   46 DSPMG4 DSPME4 1 1 0
btlpm5     16   46 DSPMG5 DSPME5 1 1 0
btlpm6     16   46 DSPMG6 DSPME6 1 1 0
btlpm7     16   46 DSPMG7 DSPME7 1 1 0
```

```
btldm0      8   92 DSPMG0 DSPME0 1 1 0
btldm1      8   92 DSPMG1 DSPME1 1 1 0
btldm2      8   92 DSPMG2 DSPME2 1 1 0
btldm3      8   92 DSPMG3 DSPME3 1 1 0
btldm4      8   92 DSPMG4 DSPME4 1 1 0
btldm5      8   92 DSPMG5 DSPME5 1 1 0
btldm6      8   92 DSPMG6 DSPME6 1 1 0
btldm7      8   92 DSPMG7 DSPME7 1 1 0
btlzm0      2   23 DSPMG0 DSPME0 1 1 0
btlzm1      2   23 DSPMG1 DSPME1 1 1 0
btlzm2      2   23 DSPMG2 DSPME2 1 1 0
btlzm3      2   23 DSPMG3 DSPME3 1 1 0
btlzm4      2   23 DSPMG4 DSPME4 1 1 0
btlzm5      2   23 DSPMG5 DSPME5 1 1 0
btlzm6      2   23 DSPMG6 DSPME6 1 1 0
btlzm7      2   23 DSPMG7 DSPME7 1 1 0
[Blocks]
rx0c   0 {etrx btrx}
rx1c   0 {etrx btrx}
dspioc 0 {etdspioc btdspioc}
cpmc   0 {etcpmc btcpmc}
dmaioc 0 {etdmaioc btdmaioc}
ckreg  0 {etckreg btckreg}
cdmc   0 {etcdmc btcdmc}
txc    0 {ettxc bttxc}
cpm0   0 {btcpm}
cpm1   0 {btcpm}
cpm2   0 {btcpm}
cpm3   0 {btcpm}
cpm4   0 {btcpm}
cpm5   0 {btcpm}
cpm6   0 {btcpm}
cpm7   0 {btcpm}
cpm8   0 {btcpm}
cpm9   0 {btcpm}
cdm0   0 {btcdm}
cdm1   0 {btcdm}
cdm2   0 {btcdm}
cdm3   0 {btcdm}
cdm4   0 {btcdm}
cdm5   0 {btcdm}
cdm6   0 {btcdm}
cdm7   0 {btcdm}
# DSP0
logic00 0 {btlogic00 etlogic0}
logic10 0 {btlogic10 etlogic1}
logic20 0 {btlogic20 etlogic2}
logic30 0 {btlogic30 etlogic3}
logic40 0 {btlogic40 etlogic4}
lpm0   0 {btlpm0}
ldm00  0 {btldm0}
ldm10  0 {btldm0}
ldm20  0 {btldm0}
ldm30  0 {btldm0}
lzm00  0 {btlzm0}
lzm10  0 {btlzm0}
# DSP1
logic01 0 {btlogic01 etlogic0}
logic11 0 {btlogic11 etlogic1}
logic21 0 {btlogic21 etlogic2}
```

```
logic31 0 {btlogic31 etlogic3}
logic41 0 {btlogic41 etlogic4}
lpm1   0 {btlpm1}
ldm01  0 {btldm1}
ldm11  0 {btldm1}
ldm21  0 {btldm1}
ldm31  0 {btldm1}
lzm01  0 {btlzm1}
lzm11  0 {btlzm1}
# DSP2
logic02 0 {btlogic02 etlogic0}
logic12 0 {btlogic12 etlogic1}
logic22 0 {btlogic22 etlogic2}
logic32 0 {btlogic32 etlogic3}
logic42 0 {btlogic42 etlogic4}
lpm2   0 {btlpm2}
ldm02  0 {btldm2}
ldm12  0 {btldm2}
ldm22  0 {btldm2}
ldm32  0 {btldm2}
lzm02  0 {btlzm2}
lzm12  0 {btlzm2}
# DSP3
logic03 0 {btlogic03 etlogic0}
logic13 0 {btlogic13 etlogic1}
logic23 0 {btlogic23 etlogic2}
logic33 0 {btlogic33 etlogic3}
logic43 0 {btlogic43 etlogic4}
lpm3   0 {btlpm3}
ldm03  0 {btldm3}
ldm13  0 {btldm3}
ldm23  0 {btldm3}
ldm33  0 {btldm3}
lzm03  0 {btlzm3}
lzm13  0 {btlzm3}
# DSP4
logic04 0 {btlogic04 etlogic0}
logic14 0 {btlogic14 etlogic1}
logic24 0 {btlogic24 etlogic2}
logic34 0 {btlogic34 etlogic3}
logic44 0 {btlogic44 etlogic4}
lpm4   0 {btlpm4}
ldm04  0 {btldm4}
ldm14  0 {btldm4}
ldm24  0 {btldm4}
ldm34  0 {btldm4}
lzm04  0 {btlzm4}
lzm14  0 {btlzm4}
# DSP5
logic05 0 {btlogic05 etlogic0}
logic15 0 {btlogic15 etlogic1}
logic25 0 {btlogic25 etlogic2}
logic35 0 {btlogic35 etlogic3}
logic45 0 {btlogic45 etlogic4}
lpm5   0 {btlpm5}
ldm05  0 {btldm5}
ldm15  0 {btldm5}
ldm25  0 {btldm5}
ldm35  0 {btldm5}
lzm05  0 {btlzm5}
```

```
lzm15  0 {btlzm5}
# DSP6
logic06 0 {btlogic06 etlogic0}
logic16 0 {btlogic16 etlogic1}
logic26 0 {btlogic26 etlogic2}
logic36 0 {btlogic36 etlogic3}
logic46 0 {btlogic46 etlogic4}
lpm6   0 {btlpm6}
ldm06  0 {btldm6}
ldm16  0 {btldm6}
ldm26  0 {btldm6}
ldm36  0 {btldm6}
lzm06  0 {btlzm6}
lzm16  0 {btlzm6}
# DSP7
logic07 0 {btlogic07 etlogic0}
logic17 0 {btlogic17 etlogic1}
logic27 0 {btlogic27 etlogic2}
logic37 0 {btlogic37 etlogic3}
logic47 0 {btlogic47 etlogic4}
lpm7   0 {btlpm7}
ldm07  0 {btldm7}
ldm17  0 {btldm7}
ldm27  0 {btldm7}
ldm37  0 {btldm7}
lzm07  0 {btlzm7}
lzm17  0 {btlzm7}
```

10 SYSTEM S

System S is defined by Chakrabaraty [25] and it consists of six cores where each core is an ISCAS benchmark (core), see Figure 1.10.. Data for the system is given in Table 1.11. where *i* is the core and for each core *i* the number of external test cycles, e_i, and number of BIST cycles, b_i are specified.

Each core is tested by two test sets, one BIST test set and one deterministic test set. The deterministic test vector set is applied using an external tester and the test bus. Only one core at the time can use the test bus and the external tester. The BIST patterns take one clock cycle to apply while the external tester is ten times slower.

We have added placement for the cores in the system, see Table 1.11.

Circuit	Core *i*	*Number of external test cycles, e_i.*	*Number of BIST cycles, b_i.*	*Placement*	
				x	*y*
c880	1	377	4096	10	10
c2670	2	15958	64000	20	10

*Figure 1.11.*Test data for the cores in System S.

Circuit	Core *i*	*Number of external test cycles, e_i.*	*Number of BIST cycles, b_i.*	*Placement*	
				x	*y*
c7552	3	8448	64000	10	30
s953	4	28959	217140	20	30
s5378	5	60698	389210	30	30
s1196	6	778	135200	30	10

Figure 1.11. Test data for the cores in System S.

```
# System S
[Global Options]
MaxPower = 14      # Maximal allowed simultaneous power
TimeFactor = 1
BusFactor  = 15000
[Cores]
#Syntax: name x y {block1 block2 ... blockN}
C1  10   10   {c880}
C2  20   10   {c2670}
C3  10   30   {c7552}
C4  20   30   {s953}
C5  30   30   {s5378}
C6  30   10   {s1196}
C7   0    0   {s13207} # extra BIST-tested core (variant IV)
[Generators]
#Syntax: name x y max_bandw max_mem movable
BG1   10   10   1   32   yes
BG2   20   10   1   32   yes
BG3   10   30   1   32   yes
BG4   20   30   1   32   yes
BG5   30   30   1   32   yes
BG6   30   10   1   32   yes
```

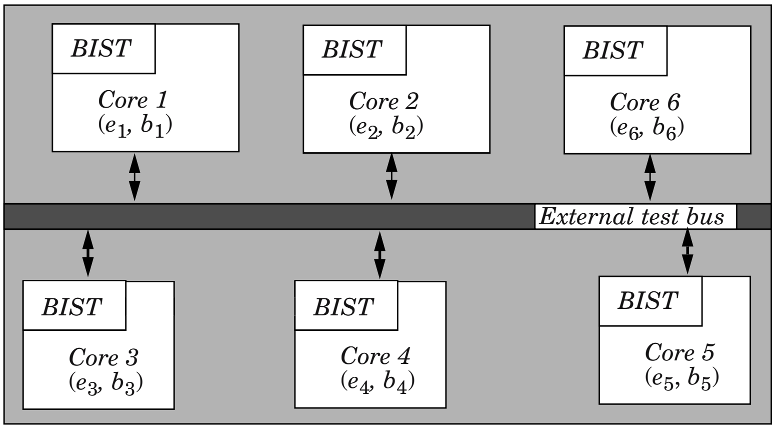


Figure 1.10. System S.

```
EG     0   20   1   32   yes
[Evaluators]
#Syntax: name x y max_bandw movable
BE1   10   10   1      yes
BE2   20   10   2      yes
BE3   10   30   1      yes
BE4   20   30   2      yes
BE5   30   30   3      yes
BE6   30   10   1      yes
EE    40   20   3      yes
[Tests]
#Syntax:
# name power time tpg tre min_bandw max_bandw mem ict_to_core
e1  1    3770    EG   EE     1        1       1     no
e2  1  159580    EG   EE     1        2       1     no
e3  1   84480    EG   EE     1        1       1     no
e4  1  289590    EG   EE     1        2       1     no
e5  1  606980    EG   EE     1        3       1     no
e6  1    7780    EG   EE     1        1       1     no
b1  1    4096    BG1  BE1    1        1       1     no
b2  1   64000    BG2  BE2    1        2       1     no
b3  1   64000    BG3  BE3    1        1       1     no
## dedicated BIST (variant I)
b4  1  217140    BG4  BE4    1        2       1     no
b5  1  389214    BG5  BE5    1        3       1     no
b6  1  135200    BG6  BE6    1        1       1     no
## (s1196, c7552) and (s953, s5378) share BIST (variant II)
#b4  1  217140    BG4  BE4    1        2       1     no
#b5  1  389214    BG4  BE4    1        3       1     no
#b6  1  135200    BG3  BE3    1        1       1     no
## (s1196, c7552, s953, s5378) share BIST (variant III)
#b4  1  217140    BG3  BE3    1        2       1     no
#b5  1  389214    BG3  BE3    1        3       1     no
#b6  1  135200    BG3  BE3    1        1       1     no
## (s13207, c7552, s953, s5378) share BIST (variant IV)
#b4  1  217140    BG3  BE3    1        2       1     no
#b5  1  389214    BG3  BE3    1        3       1     no
#b6  1  135200    BG4  BE4    1        1       1     no
#b7  1  512000    BG3  BE3    1        1       1     no
[Blocks]
#Syntax: name idle_power {test1 test2 ... testN}
c880  0 {e1 b1}
c2670 0 {e2 b2}
c7552 0 {e3 b3}
s953  0 {e4 b4}
s5378 0 {e5 b5}
s1196 0 {e6 b6}
#s13207 0 {  b7} ## only for variant IV
s13207 0 {} ## not when variant IV
```

References

[1] M.S. Abadir and H. K. Reghabati, "Functional Testing of Semiconductor Random Access Memories", Computer Survey, Vol. 15, No. 3, 1983, pages 175-198.

[2] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design, IEEE Press, ISBN 0-7803-1062-4, 1990.

[3] R. D. Adams, "High Performance Memory Testing - Design Principles, Fault Modeling and Self-Test", Kluwer Academic Publisher, ISBN 1-4020-7255-4, 2003.

[4] Advantest, http://www.advantest.com/

[5] J. Aerts and E. J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs", *Proceedings of International Test Conference (ITC)*, Washington, DC, USA, October 1998, pages 448-457.

[6] A. V. Aho, J. E. Hopcroft and J. D. Ullman, Data Structures and Algorithms, Addison-Wesley, 1983, ISBN 0-201-00023-7.

[7] Agilent (HP), http://www.agilent.com/

[8] P.H. Bardell, W. H. McAnney, and J. Savir, "Built-In Test for VLSI Pseudorandom Techniques", *John Wiley and Sons*, 1987.

[9] J. Barwise and J. Etchemendy, The Language of First-Order Logic, *CSLI Publications,* ISBN 0-937073-99-7*,* 1993.

[10] I. Bayraktarolgu and A. Orailoglu, "Test Volume And Application Time Reduction Through Scan Chain Concealment", *Proceedings of Design Automation Conference (DAC)*, Las Vegas, NV, USA, June 2001, pages 151-155.

[11] F. Beenker, B. Bennets, and L. Thijssen, "Testability Concepts for Digital ICs - The Macro Test Approach", *Frontiers in Electronic Testing*, vol. 3, Boston, Kluwer, 1995.

[12] M. Benabdenbi, W. Maroufi, and M. Marzouki, "CAS-BUS: A Test Access Mechanism and a Toolbox Environment for Core-Based System Chip Testing", *Journal of Electronic Testing; Theory and Applications (JETTA)*, Vol. 18, Nos. 4/5, August/October 2002, pages 455-472.

[13] M. J. Bending, "Hitest: A Knowledge-Based Test Generation System", *Design & Test of Computers*, Vol. 1, May 1984, pages 83-92.

[14] L. Benini and G. De Micheli, "Networks on Chips: a New SoC Paradigm", *Computer,* Volume: 35, Issue: 1, January 2002, pages 70-80.

[15] A. Benso, S. Cataldo, S. Chiusano, P. Prinetto, and Y. Zorian, "A High-Level EDA Environment for Automatic Insertion of HD-BIST Structures", *Journal of Electronic Testing: Theory and Applications,* Vol. 16, No. 3, June 2000, pages 179-184.

[16] A. Benso and P. Prinetto, "Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation", *Kluwer Academic Publisher,* ISBN 1-4020-7589-8, 2003.

[17] A. Benso, S. Di Carlo, P. Prinetto, and Y. Zorian, "A Hierarchical Infrastructure for SoC Test Management", *Design & Test of Computers*, July-August 2003, pages 32-39.

[18] Y. Bonhomme, P. Girard, L. Guiller, C. Landrault, and S. Pravossoudovitch, "A Gated Clock Scheme for Low Power Scan Testing of Logic ICs or Embedded Cores", *Proceedings of Asian Test Symposium (ATS)*, November 2001, pages 253-258.

[19] H. Bleeker, P. van den Eijnden and F. de Jong, Boundary-Scan Test: A Practical Approach, *Kluwer Academic Publishers,* ISBN 0-7923-9296-5, 1993.

[20] G. Blom, "Sannolikhetsteori och statistikteori med tillämpningar", Studentlitteratur, 1989.

[21] D. Brahme and J. A. Abraham, “Functional Testing of Microprocessors”, *Transactions on Computers*, Vol. C-33, June 1984, pages 475-485.

[22] M. A. Breuer and A. D. Friedman, “Diagnosis and Reliable Design of Digital Systems”, Computer Science Press, 1976.

[23] P. Brucker, “Scheduling Algorithms”, *Springer-Verlag*, ISBN 3-540-64105-X, 1998.

[24] M. L. Bushnell and V. D. Agrawal, “Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits”, *Kluwer Academic Publisher*, ISBN 0-7923-7991-8.

[25] K. Chakrabarty, “Test Scheduling for Core-Based Systems”, *Proceedings of International Conference on Computer Aided Design (ICCAD),* San Jose, CA, USA, November 1999, pages 391-394.

[26] K. Chakrabarty, “Design of System-on-a-Chip Test Access Architectures Using Integer Linear Programming”, *Proceedings of VLSI Test Symposium (VTS),* Montreal, Canada, April 2000, pages 127-134.

[27] K. Chakrabarty, “Design of System-on-a-Chip Test Access Architecture under Place-and-Route and Power Constraints”, *Proceedings of the Design Automation Conference*, 2000*,* pages 432-437.

[28] K. Chakrabarty, “Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming”, *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 10, October 2000, pages 1163-1174.

[29] K. Chakrabarty, R. Mukherjee, and A. Exnicios, “Synthesis of Transparent Circuits for Hierarchical and System-on-a-Chip Test”, *Proceedings of International Conference on VLSI Design (VLSID)*, Bangalore, India, January 2001, pages 431-436.

[30] K. Chakrabarty, “Optimal test access architectures for system-on-a-chip”, *ACM Transactions on Design Automation of Electronic Systems*, Vol. 6, January 2001, pages 26-49.

[31] K. Chakrabarty, V. Iyengar, and A. Chandra, "Test Resource Partitioning for System-on-a-Chip", *Kluwer Academic Publisher,* 2002, ISBN-1-4020-7119-1.

[32] K. Chakrabarty, V. Iyengar, and M. D. Krasniewski, “Test Planning for Modular Testing of Hierarchical SOCs”, *Transactions on Computer-Aided Design of Iintegrated Circuits and Systems*, 2004.

[33] A. Chandra and K. Chakrabarty, “Frequency-Directed-Run-Length (FDR) Codes with Applications to System-on-a-Chip Test Data Compression”, *Proceedings of VLSI Test Symposium (VTS)*, Marina Del Rey, CA, USA, April 2001, pages 42-47.

[34] A. Chandra and K. Chakrabarty, “System-on-a-Chip Test Data Compression and Decompression Architectures Based on Golomb Codes”, *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 20, No. 3, 2001, pages 355-367.

[35] A. Chandra and K. Chakrabarty, “Reduction of SOC Test Data Volume, Scan Power and Testing Time Using Alternating Run-length Codes”, *Proceedings of Design Automation Conference (DAC)*, New Orleans, LA, USA, June 2002, pages 673-678.

[36] D. Chapiro, “Globally-Asynchronous Locally-Synchronous Systems”, Department of Computer Science, Ph. D. Thesis, Stanford University, Stanford, CA, USA, October 1984.

[37] W. T. Chen, “The BACK Algorithm for Sequential Test Generation”, *Proceedings of International Conference on Computer Design (ICCD)*, Rye Brook, NY, USA, October 1988, pages 66-69.

[38] Y. Caseau and F. Laburthe, “Heuristics for large constrained vehicle routing problems”, October 1999, *Journal of Heuristics*, vol.5, no.3, pages 281-303.

[39] K.-T. Cheng and V. D. Agrawal, “Unified Methods for VLSI Simulation and Test Generation”, Kluwer Academic Publisher, ISBN 0-7923-9025-3, 1989.

[40] R. M. Chou, K. K. Saluja, and V. D. Agrawal, ”Power Constraint Scheduling of Tests”, *Proceedings of International Conference on VLSI Design*, Calcutta, India, January 1994, pages 271-274.

[41] R. M. Chou, K. K. Saluja, and V. D. Agrawal, “Scheduling Tests for VLSI Systems Under Power Constraints”, *IEEE Transactions on VLSI Systems*, Vol. 5, No. 2, June 1997, pages 175-185.

[42] T. Cormen, C. Leiserson, and R. Rivest, “Introduction To Algorithms”, *The MIT Press*, ISBN 0-262-03141-8, 1989.

[43] E. Cota, L. Carro, M. Lubaszewski, and A. Orailoglu, “Test Planning and Design Space Exploration in a Core-based Environment”, *Proceedings of the Design, Automation and Test in Europe Conference (DATE),* Paris, France, March 2002, pages 478-485.

[44] E. Cota, M. Kreutz, C.A. Zeferino, L. Carro, M. Lubaszewski, A. Susin, “The Impact of NoC Reuse on the Testing of Core-based Systems”, *Proceedings VLSI Test Symposium (VTS)*, Napa, CA, April 2003, pages 128-133.

[45] E. Cota, L. Carro, F. Wagner, and M. Lubaszewski, “Power-Aware NoC Reuse on the Testing of Core-Based Systems”, *Digest of Papers of European Test Workshop (ETW)*, Maastricht, The Netherlands, May 2003, pages 123-128.

[46] E. Cota, L. Carro, F. Wagner, and M. Lubaszewski, “Power-Aware NoC Reuse on the Testing of Core-Based Systems”, *Proceedings of International Test Conference (ITC)*, Charlotte, NC, USA, September 2003, pages 612-621.

[47] G. L. Craig, C. R. Kime, and K. K. Saluja, “Test Scheduling and Control for VLSI built-in-self-test”, *Transactions on Computers*, Vol. 37, No. 9, September 1988, pages 1099-1109.

[48] Credence, http://www.credence.com

[49] A. L. Crouch, "Design-for-Test for Digital IC's and Embedded Core Systems", *Prentice Hall*, 1999, ISBN 0-13-084827-1.

[50] H. Date, T. Hosokawa, and M. Muraoka, “A SoC Test Strategy Based on a Non-Scan DFT Method”, *Proceedings of Asian Test Symposium (ATS)*, Tamuning, Guam, USA, November 2002, pages 305-310.

[51] F. DaSilva, Y. Zorian, L. Whetsel, K. Arabi, and R. Kapur, “Overview of the IEEE P1500 Standard”, *Proceedings of International Test Conference (ITC)*, Charlotte, NC, USA, September 2003, pages 988-997.

[52] B. L. Dervisoglu and G. E. Strong, “Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement”, *Proceedings of International Test Conference (ITC)*, Nashville, TN, USA, October 1991, pages 365-374.

[53] S. Dey, E. J. Marinissen, and Y. Zorian, “Testing System Chips: Methodologies and Experiences” *Integrated System Design*, Vol. 11(No. 123), September 1999, pages 36-48.

[54] R. Dorsch and H. -J. Wunderlich, “Tailored ATPG For Embedded Testing”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2001, pages 530-537.

[55] R. Dorsch, R. Huerta Rivera, H. -J. Wunderlich, and M. Fischer, “Adapting an SoC to ATE Concurrent Test Capabilities”. *Proceedings International Test Conference (ITC)*, Baltimore, MD, USA, October 2002, pages 1169-1175.

[56] Z. S. Ebadi and A. Ivanov, “Design of an Optimal Test Access Architecture Using a Genetic Algorithm”, *Proceedings of Asian Test Symposium (ATS)*, Kyoto, Japan, November 2001, pages 205-210.

[57] S. Edbom and E. Larsson, “An Integrated Technique for Test Vector Selection and Test Scheduling under Test Time Constraint”, *Proceedings of Asian Test Symposium (ATS)*, Taiwan, November 2004, pages 254-257.

[58] Ericsson, *Design document,* 2000.

[59] B. H. Fang, Q. Xu, and N. Nicolici, “Hardware/Software Co-testing of Embedded Memories in Complex SOCs”, *Proceedings of International Conference on Computer-Aided Design* (*ICCAD*), San Jose, CA, USA, November 2003, pages 599-605.

[60] J. Ferguson and J. Shen, “A CMOS Fault Extractor for Inductive Fault Analysis”, *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, November 1988, Vol. 7, No. 11, pages 1181-1194.

[61] M. -L. Flottes, J. Pouget, and B. Rouzeyre, “Sessionless Test Scheme: Power-Constrained Test Scheduling for System-on-a-Chip” *Proceedings of International Conference on Very Large Scale Integration (VLSI-SOC)*, Montpellier, France, December 2001, pages 105-110.

[62] M. -L. Flottes, J. Pouget, and B. Rouzeyre, "Power-Constrained Test Scheduling for SOCs Under a ``No Session" Scheme", In Michel Robert, Bruno Rouzeyre, Christian Piguet, and Marie-Lise Flottes, editors, *SOC Design Methodologies*, Kluwer Academic Publishers, 2002, pages 401-412.

[63] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms", *Transactions on Computers*, Vol. 32, No. 12, December 1983, pages 1137-1144.

[64] D.D. Gajski, N. D. Dutt, and A. C. Wu, "High-Level Synthesis: Introduction and System Design, *Kluwer Academic Publisher*, 1992, ISBN 0-7923-9194-2.

[65] M. Garg, A. Basu, T.C. Wilson, D.K. Banerji, J.C. Majithia, "A New Test Scheduling Algorithm for VLSI Systems", *Proceedings of the Symposium on VLSI Design*, New Delhi, India, January 1991, pages 148-153.

[66] M. R. Garey and, D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", *W. H. Freeman and Company*, San Fransisco, 1979, ISBN 0-716-71045-5 .

[67] S. H. Gerez, "Algorithms for VLSI Design Automation", John Wiley and Sons Ltd, ISBN 0-471-98489-2, 1998.

[68] S. Gerstendörfer and H.-J. Wunderlich, "Minimized Power Consumption for Scan-Based BIST", *Proceedings of International Test Conference (ITC)*, Atlantic City, NJ, USA, September 1999, pages 77-84.

[69] I. Ghosh, S. Dey, and N. K. Jha, "A Fast and Low Cost Testing Technique for Core-based System-ob-Chip", *Proceedings of Design Automation Conference (DAC)*, San Francisco, CA, USA, June 1998, pages 542-547.

[70] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac, Reducing Power Consumption During Test Application By Test Vector Ordering, *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, Vol. 2, Austin, May/June 1998, pages 296-299.

[71] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computer and Operations Research.,* Vol. 13, No. 5, May 1986, pages 533-549.

[72] P. Goel, “An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits”, *Transactions on Computers*, Vol. 30, No. 3, March 1981, pages 215-222.

[73] S. K. Goel and E. J. Marinissen, “Cluster-Based Test Architecture Design for System-On-Chip, *Proceedings of VLSI Test Symposium (VTS)*, Monterey, CA, USA, April 2002, pages 259-264.

[74] S. K. Goel and E.J. Mariniseen, “A Novel Test Time Reduction Algorithm for Test Architecture Design for core-Based System Chips”, *Formal Proceedings of European Test Workshop (ETW)*, pp 7-12, Corfu, Greece, May 2002.

[75] S. K. Goel and E. J. Marinissen, “Effective and Efficient Test Architecture Design for SOCs”, *Proceedings of International Test Conference (ITC)*, Baltimore, Maryland, USA, October 2002, pages 529-538.

[76] S. K. Goel and E. J. Marinissen, “Layout-Driven SOC Test Architecture Design for Test Time and Wire Length Minimization.”, *Proceedings Design, Automation, and Test in Europe (DATE)*, Munich, Germany, March 2003, pages 738-743.

[77] S. K. Goel and E. J. Marinissen, “Control-Aware Test Architecture Design for Modular SOC Testing”, *Informal Proceedings of European Test Workshop (ETW)*, Maastricht, The Netherlands, May 2003, pages 129-134.

[78] S. K. Goel and E. J. Marinissen, “SOC Test Architecture Design for Efficient Utilization of Test Bandwidth”, *Transactions on Design Automation of Electronic Systems, Special Issue on VLSI Testing,* Vol. 8, No. 4, October 2003, pages 399-429.

[79] S. K. Goel, K. Chiu, E. J. Marinissen, T. Nguyen, and S. Oosrdijk, “Test Infrastructure Design for the Nexperia^{TM}Home Platform PNX8550 System Chip”, *Proceedings of Design Automation and Test in Europe (DATE)*, Paris, France, February 2004, pages 1530-1591.

[80] S. K. Goel, “A Novel Wrapper Cell Design for Efficient Testing of Hierarchical Cores in System Chips”, *Proceedings of European Test Symposium (ETS)*, Ajaccio, France, May 2004, pages 147-152.

[81] T. Gonzales and S. Sahni, "Open Shop Scheduling to Minimize Finish Time", *Journal of the ACM,* Vol. 23, October 1976, pages 665-679.

[82] A. J. van de Goor, "Testing Semiconductor Memories: Theory and Practice, Comtex Publishing, Gouda, The Netherlands, 1998.

[83] X. Gu, E. Larsson, K. Kuchcinski, and Z. Peng, "A Controller Testability Analysis and Enhancement Technique", *Proceedings of European Design and Test Conference (ED&TC)*, Paris, March 1997, pages 153-157.

[84] R. K. Gupta and Y. Zorian, "Introduction to Core-Based System Design", *Design and Test of Computers*, Vol. 14, No. 4, October 1997, pages 15-25.

[85] R. Gupta, R. Gupta, and M. A. Breuer, "The BALLAST Methodology for Structured Partial Scan Design", *Transactions on Computers,* Vol. 39, No. 4, April 1990, pages 538-544.

[86] S. Hamdioui, "Testing Static Random Access Memories Defects, Fault Models and Test Patterns", Kluwer Academic Publisher, 2004, ISBN 1-4020-7752-1.

[87] I. Hamazaoglu and J. H. Patel, "Reducing Test Application Time For Full Scan Embedded Cores", *Proceedings of International Symposium on Fault-Tolerant Computing (FTCS)*, Madison, WI, USA, June 1999, pages 260-267.

[88] P. Harrod, "Testing Reusable IP-a Case Study", *Proceedings of International Test Conference* (*ITC*), Atlantic City, NJ, USA, September 1999*,* pages 493-498.

[89] J. P. Hayes, "Transition Count Testing of Combinational Logic Circuits", *Transactions on Computers*, Vo. 25, No. 6, June 1976, pages 613-620.

[90] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkatraman, and B. Courtois, "Built-In Test For Circuits With Scan Based Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", *Transactions on Computers*, Vol. 44, No. 2, 1995, pages 223-233.

[91] S. Hellebrand and A. Wurtenberger, "Alternating Run-Length Coding - A Technique for Improved Test Data Compression", *Test Resource Partitioning Workshop*, Baltimore, MD, USA October 2002, pages 4.3.1-4.3.10.

[92] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies", *Proceedings of International Test Conference (ITC)*, Atlantic City, NJ, USA, September 1999*,* pages 358-367.

[93] A. Hertwig and H-J Wunderlich, "Low Power Serial Built-In Self-Test", *Compendium of Papers of European Test Workshop,* , Sitges, Spain, May 1998, pages 49-53.

[94] Hewlett-Packard Corp., "A Designer's Guide to Signature Analysis", Application note 222, April 1977.

[95] F. J. Hill and B. Huey, "A Design Language Based Approach to Test Sequence Generation", Computer, Vol. 10, June 1977, pages 28-33.

[96] H.-S. Hsu, J.-R. Huang, K.-L. Cheng, C.-W. Wang, C.-T. Huang, and C.-W. Wu, "Test Scheduling and Test Access Architecture Optimization for System-on-Chip", *Proceedings of Asian Test Symposium (ATS)*, Tamuning, Guam, USA, November 2002, pages 411-416.

[97] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S. M. Reddy, "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design", *Proceedings of Asian Test Symposium (ATS)*, Kyoto, Japan, November 2001, pages 265-270.

[98] Y. Huang, N. Mukherjee, C. -C. Tsai, O Samman, Y. Zaidan, Y. Zhang, W.-T. Cheng, and S.M. Reddy, "Constraint Driven Pin-mapping for Concurrent SOC Testing", *Proceedings of the Asia and South Pacific Design Automation Conference & International Conference on VLSI Design*, Bangalore India, January 2002, pages 511-516.

[99] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S. M. Reddy, "On Concurrent Test of Core-Based SOC Designs", *Journal of Electronic Testing: Theory and Applications (JETTA)*, Volume 18, Nos. 4/5, August/October 2002, pages 401-414.

[100]Y. Huang, S. M. Reddy, W.-T. Cheng, P. Reuter, N. Mukherjee, C.-C. Tsai, O. Samman, and Y. Zaidan, "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm", *Proceedings of International Test Conference (ITC)*, Baltimore, Maryland, USA, October 2002, pages 74-82.

[101]Y. Huang, W.-T. Cheng, C. -C. Tsai, N. Mukherjee, and S.M Reddy, "Static Pin Mapping and SOC Test Scheduling for Cores with Multiple Test Sets", *Proceedings of International Symposium on Quality Electronic Design (ISQED)*, March 2003, pages 99-104.

[102]S. D. Huss and R. S. Gyurcsik, "Optimal Ordering of Analog Integrated Circuit Tests to Minimize Test Time", *Proceedings of Design Automation Conference (DAC)*, San Francisco, CA, USA, June 1991, pages 494-499.

[103]H. Ichihara, A. Ogawa, T. Inoue, and A. Tamura, "Dynamic Test Compression Using Statistical Coding", *Proceedings of Asian Test Symposium (ATS)*, Kyoto, Japan, November 2001, pages 143-148.

[104]IEEE P1500 Standard for Embedded Core Test, web site: *http://grouper.ieee.org/groups/1500/* .

[105]V. Immaneni and S. Raman, "Direct Access Test Scheme - Design of Blocks and Core Cells for Embedded ASICs", *Proceedings of International Test conference (ITC)*, Washington, DC, USA, September 1990, pages 488-492.

[106]International Technology Roadmap for Semiconductors (ITRS), 2003, *http://public.itrs.net/*

[107]Intel, 2003, *http://www.intel.com/research/silicon/mooreslaw.htm*

[108]H. Ichihara, A. Ogawa, T. Inoue, and A. Tamura, "Dynamic Test Compression Using Statistical Coding", *Proceedings of Asian Test Symposium (ATS)*, Kyoto, Japan, November 2001, pages 143-148.

[109]V. Iyengar, K. Chakrabarty, and B. T. Murray, "Built-In Self-Testing of Sequential Circuits Using Precomputed Test Sets", *Proceedings of VLSI Test Symposium (VTS)*, Princeton, NJ, USA, April 1998, pages 418-423.

[110]V. Iyengar and K. Chakrabarty, "Precedence-based, Preemptive, and Power-constrained Test Scheduling for System-on-a-Chip", *Proceedings of VLSI Test Symposium (VTS)*, Marina del Rey, CA, USA, May 2001, pages 368-374.

[111]V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip", *Proceedings of International Test Conference (ITC)*, Baltimore, Maryland, November 2001*,* pages 1023-1032.

[112]V. Iyengar K. Chakrabarty, and E. J. Marinissen, "Efficient Wrapper/TAM Co-Optimization for Large SOCs", *Proceedings of Design and Test in Europe (DATE)*, Paris, France, March 2002, pages 491-498.

[113]V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip", *Journal of Electronic Testing; Theory and Applications (JETTA)*, April 2002, pages 213-230.

[114]V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization", *Proceedings of VLSI Test Symposium (VTS)*, Monterey, California, USA, April 2002, pages 253-258.

[115]V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Wrapper/Optimization, Test Scheduling, TAM Co-Optimization, Constraint-Driven Test Scheduling, and Test Data Volume Reduction for SOCs", *Proceedings of Design Automation Conference (DAC)*, New Orleans, Louisiana, June 2002, pages 685-690.

[116]V. Iyengar, S. K. Goel, E. J. Marinissen and K. Chakrabarty, "Test Resource Optimization for Multi-Site Testing of SOCs under ATE Memory Depth Constraints", *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2002, pages 1159-1168.

[117]V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Recent Advances in test Planning for Modular Testing of Core-Based SOCs", *Proceedings of Asian Test Symposium (ATS)*, November 2002, Guam, USA.

[118]V. Iyengar, K. Chakrabarty, M. D. Krasniewski, and G. N. Kuma, "Design and Optimization of Multi-level TAM Architectures for Hierarchical SOCs", *Proceedings of VLSI Test Symposium (VTS)*, pages 299-304, 2003.

[119]V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Test Access Mechanism, Test Scheduling, and Test Data Reduction for System-on-Chip", *Transactions on Computers*, Vol. 52, No. 12, December 2003, pages 1-14.

[120]D. Jansen *et al.* "The Electronic Design Automation Handbook", Kluwer Academic Publisher, 2003, ISBN-1-4020-7502-2.

[121]A. Jantsch and H. Tenhunen (editors), "Networks on Chip", Kluwer Academic Publishers, 2003, ISBN 1-4020-7392-5.

[122]A. Jas, J. Ghosh-Dastidar, and N. A. Touba, "Scan Vector Compression/ Decompression Using Statistical Coding", Proceedings of VLSI Test Symposium (VTS), San Diego, CA, USA, April 1999, pages 114-120.

[123]A. Jas, B. Pouya, and N. A. Touba, "Virtual Scan Chains: A Means For Reducing Scan Length In Cores", *Proceedings of VLSI Test Symposium (VTS)*, Montreal, Canada, April 2000, pages 73-78.

[124]W. J. Jiang and B. Vinnakota, "Defect-Oriented Test Scheduling", *Transactions on Very-Large Scale Integration (VLSI) Systems, Vol. 9, No. 3*, June 2001, pages 427-438.

[125]G. Jervan, Z. Peng and R. Ubar, "Test Cost Minimization for Hybrid BIST", *Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI (DFT)*, Yamanashi, Japan, October 2000, pages 283-291.

[126]G. Jervan, Z. Peng, R. Ubar, and H. Kruus, "A Hybrid BIST Architecture and its Optimization for SoC Testing", *Proceedings of International Symposium on Quality Electronic Design (ISQED)*, San Jose, CA, USA, March 2002, pages 273-279.

[127]G. Jervan, P. Eles, Z. Peng, R. Ubar, and M. Jenihhin, "Test Time Minimization for Hybrid BIST of Core-Based Systems", *Proceedings of Asian Test Symposium (ATS03)*, Xian, China, November 2003, pages 318-323.

[128]W. B. Jone, C. A. Papachrisou, and M. Perieria, "A Scheme for Overlaying Concurrent Testing of VLSI Circuits", *Proceedings of the Design Automation Conference (DAC)*, 1989*,* pages 531-536.

[129]A. Khoche, E. Volkerink, J. Rivoir, and S. Mitra, "Test Vector Compression Using EDA-ATE Synergies", *Proceedings of VLSI Test Symposium (VTS)*, Monterey, CA, USA, April 2002, pages 97-102.

[130]T. Kim, "Scheduling and Allocation Problems in High-Level Synthesis", *Ph.D. Dissertation*, Department of Computer Science, University of Illinois at Urbana-Champaign, USA, 1993.

[131]S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, No. 4598, May 1983, pages 671-680.

[132]C. R. Kime and K. K. Saluja, "Test Scheduling in Testable VLSI Circuits", *Proceedings of the International Symposium on Fault-Tolerant Computing*, pp. 406-412, 1982.

[133]J. Knaizuk Jr. and C. R. P. Hartmann, "An Algorithm for Testing Random Access Memories", *Transactions on Computers*, Vol. C-26, No. 4, 1977, pages 414-416.

[134]A. Kobayashi et al., "A Flip-Flop Circuit Suitable for FLT", *Annual Meeting of the Institute of Electronics, Information and Communication Engineers*, Manuscript 892, page 962, 1963.

[135]B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs", *Proceedings of European Test Conference (ETC)*, 1991, Munchen, Germany, pages 237-242.

[136]B. Koenemann, J. Mucha, and G. Zwiehoff, "Built-In Logic Block Observation Techniques", *Proceedings of International Test Conference (ITC),* Washington, DC, USA, October 1988, pages 37-41.

[137]B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST Variant With Guaranteed Encoding", *Proceedings of Asian Test Symposium (ATS)*, Kyoto, Japan, November 2001, pages 325-330.

[138]S. Koranne, "On Test Scheduling for Core-based. SOCs", *Proceedings of International Conference on VLSI Design*, Bangalore, India, January 2002, pages 505-510.

[139]S. Koranne, “Design of Reconfigurable Access Wrappers for Embedded Core Based SOC Test”, *Proceedings of International Symposium on Quality Electronic Design (ISQED)*, San Jose, CA, USA, March 2002, pages 106-111.

[140]S. Koranne, “A Novel Reconfigurable Wrapper for Testing Embedded Core-Based and its Associated Scheduling”, *Journal of Electronic Testing; Theory and Applications (JETTA)*, August 2002, pages 415-434.

[141]S. Koranne and V. Iyengar, “On The Use of k - tuples for SOC Test Schedule Representation”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2002, pages 539- 548.

[142]S. Koranne, “Formulating SoC test scheduling as a network transportation problem”, *Transactions on Computer-Aided Design of Integrated Circuits and System*, Volume: 21 , Issue: 12 , December 2002, pages:1517 - 1525.

[143]S. Koranne, “Design of Reconfigurable Access Wrappers for Embedded Core Based SoC Test”, *Transactions on Very Large Scale Integration (VLSI) Systems,* Vol. 11, Issue: 5 , October 2003, pages 955 - 960.

[144]A. Krasniewski and S. Pilarski, “Circular Self Test Path: A Low Cost BIST Technique”, *Proceedings of Design Automation Conference (DAC)*, Miami Beach, FL, USA, June 1987, pages 407-415.

[145]A. Krasniewski and S. Pilarski, “Circular self-test path: A low-cost BIST technique for VLSI circuits,” *Transactions on Computer-Aided Design*, Vol. 8, No. 1, 1989, pages 46-55.

[146]C. V. Krishna, A. Jas, and N. A. Touba, “Test Vector Encoding Using Partial LFSR Reseeding”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2001, pages 885-893.

[147]C. V. Krishna and N. A. Touba, “Reducing Test Data Volume Using LFSR Reseeding With Seed Compression”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2002, pages 321-330.

[148]A. Krstic and K.-T. Chen, “Dealy Fault Testing for VLSI Circuits”, Kluwer Academic Publisher, ISBN 0-7923-8295-1, 1998.

[149]L. Krundel, S. K. Goel, E. J. Marinissen, M.-L. Flottes, and B. Rouzeyre, “User-Constrained Test Architecture Design for Modular SOC Testing”, *Proceedings of European Test Symposium (ETS)*, Ajaccio, France, May 2004, pages 153-158.

[150]H. Kubo, “A Procedure for Generating Test Sequences to Detect Sequential Circuit Failures”, *NEC Journal on Research and Development*, October 1968, pages 1968.

[151]P. K. Lala, "Digital Circuit Testing and Testability", Academic Press, 1997, ISBN-0-12-434330-9.

[152]A. Larsson, E. Larsson, P. Eles, and Z. Peng, “Buffer and Controller Minimisation for Time-Constrained Testing of System-On-Chip”, *Proceedings of International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03)*, Cambridge, MA, USA, November, 2003, pages 385-392.

[153]A. Larsson, E. Larsson, P. Eles, and Z. Peng, “A Technique for Optimization of System-on-Chip Test Data Transportation”, *Proceedings of European Test Symposium (ETS)*, Corsica, France, May 23-26, 2004.

[154]E. Larsson and Z. Peng, “An Estimation-based Technique for Test Scheduling”, *Proceedings of Electronic Circuits and Systems Conference (ECS)*, Bratislava, Slovakia, September 1999, pages 25-28.

[155]E. Larsson and Z. Peng, “A Technique for Test Infrastructure Design and Test Scheduling”, *Proceedings of Design and Diagnostic of Electronic Circuits and Systems Workshop (DDECS)*, Smolenice Castle, Slovakia, April 2000, pages 26-29.

[156]E. Larsson and Z. Peng, “Test Infrastructure Design and Test Scheduling Optimization”, *Informal Proceedings of European Test Workshop (ETW)*, Cascais, Portugal, May 2000.

[157]E. Larsson, “An Integrated System-Level Design for Testability Methodology”, Ph. D. Thesis No. 660, Department of Computer and Information Science, Linköpings Universitet, Sweden, December 2000.

[158]E. Larsson and Z. Peng, "An Integrated System-On-Chip Test Framework", *Proceedings of Design, Automation and Test in Europe (DATE) Conference* , Munchen, Germany, March 2001, pages 138-144.

[159]E. Larsson and Z. Peng, "System-on-Chip Test Parallelization Under Power Constraints", *Informal Proceedings of European Test Workshop (ETW)*, Stockholm, Sweden, May/June 2001.

[160]E. Larsson, Z. Peng, and G. Carlsson, "The Design and Optimization of SOC Test Solutions", *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, November 2001, pages 523-530.

[161]E. Larsson and Z. Peng, "Test Scheduling and Scan-Chain Division Under Power Constraint", *Proceedings of Asian Test Symposium (ATS)*, Kyoto, Japan, November 2001, pages 259-264.

[162]E. Larsson and H. Fujiwara, "Power Constrained Preemptive TAM Scheduling", *Informal Proceedings of European Test Workshop 2002 (ETW)*, Corfu, Greece, May 2002, pages 411-416.

[163]E. Larsson and H. Fujiwara, "Power Constrained Preemptive TAM Scheduling", *Formal Proceedings of European Test Workshop (ETW)*, Corfu, Greece, May 2002, pages 119-126.

[164]E. Larsson and Z. Peng, "An Integrated Framework for the Design and Optimization of SOC Test Solutions", *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation.Book Series: FRONTIERS IN ELECTRONIC TESTING, Volume 21, Krishnendu Chakrabarty (editor), Kluwer Academic Publisher*, ISBN 1-4020-7205-8, 2002, pages 21-36.

[165]E. Larsson and Z. Peng, "An Integrated Framework for the Design and Optimization of SOC Test Solutions", *Journal of Electronic Testing; Theory and Applications (JETTA), Special Issue on Plug-and-Play Test Automation for System-on-a-Chip*, August/October 2002 issue (vol. 18, no. 4/5), pages 385-400.

[166]E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, "Integrated Test Scheduling, Test Parallelization and TAM Design", *Proceedings of Asian Test Symposium (ATS)*, Tamuning, Guam, USA, November 2002, pages 397-404.

[167]E. Larsson and H. Fujiwara, "Optimal Test Access Mechanism Scheduling using Preemption and Reconfigurable Wrappers", *Proceedings of Workshop on RTL and High Level Testing (WRTLT)*, Guam, USA, November 21-22, 2002.

[168]E. Larsson, J. Pouget and Z. Peng, System-on-Chip Test Scheduling based on Defect Probability, *International Test Synthesis Workshop (ITSW)*, Santa Barbara, CA, USA, March 31-April 2, 2003.

[169]E. Larsson, J. Pouget and Z. Peng, "Defect Probability-based System-On-Chip Test Scheduling", *Proceedings of International Workshop on Design and Diagnostics of Electronics Circuits and Systems (DDECS)*, Poznan, Poland, April 2003, pages 25-32.

[170]E. Larsson and H. Fujiwara, "Test Resource Partitioning and Optimization for SOC Designs", *Proceedings of VLSI Test Symposium (VTS)*, Napa, USA, 27 April/May 2003, pages 319-324.

[171]E. Larsson and Z. Peng, "A Reconfigurable Power-conscious Core Wrapper and its Application to SOC Test Scheduling", *Proceedings of International Test Conference (ITC)*, Charlotte, NC, USA, September/October 2003, pp. 1135-1144.

[172]E. Larsson and H. Fujiwara, "Optimal System-on-Chip Test Scheduling", *Proceedings of Asian Test Symposium (ATS)*, Xian, China, November 2003, pages 306-311.

[173]E. Larsson and H. Fujiwara, "Preemptive system-on-chip test scheduling," *IEICE Transactions on Information and Systems*, Vol. E87-D, No. 3, March 2004, pages 620-629.

[174]E. Larsson, J. Pouget, and Z. Peng, "Defect-Aware SOC Test Scheduling", *Proceedings of VLSI Test Symposium (VTS)*, Napa Valley, Ca, USA, April 2004.

[175]E. Larsson, "Core Selection Integrated in the SOC TestSolution Design-Flow", *International Workshop on Test Resource Partitioning (TRP)*, Napa Valley California USA, April 2004.

[176]E. Larsson, K. Arvidsson, H. Fujiwara, Z. Peng, “Efficient Test Solutions for Core-based Designs”, *Transaction on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 5, May 2004, pages 758-775.

[177]E. Larsson, “Integrating Core Selection in the SOC Test Solution Design-Flow”, *Proceedings of International Test Conference (ITC)*, Charlotte, NC, USA, October 2004, paper 48.1, pages 1349-1358.

[178]E. Larsson, A. Larsson, and Z. Peng, “Linkoping University SOC Test Site“, http://www.ida.liu.se/labs/eslab/soctest/ .

[179]J. J. LeBlanc, “LOCST: A Built-In-Self-Test Technique”, *Design and Test of Computers*, November 1984, pages 42-52.

[180]H.-G. Liang, S. Hellebrand, and H.-J. Wunderlich, “Two-Dimensional Test Data Compression For Scan-Based Deterministic BIST”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2001, pages 894-902.

[181]LTX, http://www.ltx.com/

[182]S. C. Ma, P. Franco, and E. J. McCluskey, “An Experimental Chip to Evaluate Test Techniques: Experiment Results”, *Proceedings of International Test Conference (ITC)*, October 1995, Washington, DC, USA, pages 663-772.

[183]S. Mallela and S. Wu, “A Sequential Circuit Test Generation System”, *Proceedings of International Test Conference (ITC)*, Philadelphia, PA, USA, November 1985, pages 57-61.

[184]E. J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, “A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores”, *Proceedings of the International Test Conference (ITC)*, Washington, DC, USA, October 1998, pages 284-293.

[185]E. J. Marinissen and Y. Zorian, “Challenges in Testing Core-Based System ICs”, *IEEE Communications Magazine*, 37(6), June 1999, pages 104-109.

[186]E. J.Marinissen, Y. Zorian, R. Kapur, T. Taylor, and L. Whetsel, “Towards a Standard for Embedded Core Test: An Example”, *Proceedings of International Test Conference (ITC)*, pp. 616-627, Atlantic City, NJ, USA, September 1999.

[187]E. J. Marinissen, S. K. Goel, and M. Lousberg, "Wrapper Design for Embedded Core Test", *Proceedings of International Test Conference (ITC)*, Atlantic City, NJ, USA, October 2000*,* pages 911-920.

[188]E. J.Marinissen, R. Kapur, and Y. Zorian, "On Using IEEE P1500 SECT for Test Plug-n-play", *Proceedings of International Test Conference (ITC)*, Atlantic City, NJ, USA, October 2000, pages 770-777.

[189]E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "ITC'02 SOC Test Benchmarks Web Site", http://www.extra.research.philips.com/itc02socbenchm/.

[190]E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SOCs", *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2002, pages 519-528.

[191]E. J. Marinissen, R. Kapur, M. Lousberg, T. McLaurin, M. Ricchetti, and Y. Zorian, "On IEEE P1500's Standard for Embedded Core Test", *Journal of Electronic Testing: Theory and Applications, (JETTA)*, Vol. 18, Nos. 4 & 5, August/October 2002, pages 365-383.

[192]E. J. Marinissen, "The Role of Test Protocols in Automated Test Generation for Embedded-Core Based ICs", *Journal of Electronic Testing: Theory and Applications, (JETTA)*, Vol. 18, Nos. 4 & 5, August/October 2002, pages 435-454.

[193]R. A. Marlett, "EBT: A Comprehensive Test Generation Technique for Highly Sequential Circuits", *Proceedings of Design Automation Conference (DAC)*, Las Vegas, NV, USA, June 1978, pages 250-256.

[194]R. A. Marlett, "An Effective Test Generation System for Sequential Circuits", Proceedings of Design Automation Conference (DAC), Las Vegas, NV, June 1986, pages 250-256.

[195]J. F. McDonald and C. Benmehrez, "Test Set Reduction Using the Subscripted D_Algorithm", *Proceedings of International Test Conference (ITC)*, October 1983, Philadelphia, PA, USA, pages 115-121.

[196]L. Milor and A. L. Sangiovanni-Vincentelli, "Minimizing Production Test Time to Detect Faults in Analog Circuits", *Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 13, No. 6, June 1994, pages 796-813.

[197]Z. Michalewicz, "Genetic Algorithm + Data Structure = Evolutionary Programs, 3d Edition, Springer Verlag, Berlin 1996.

[198]G. E. Moore, "*Cramming more components onto integrated circuits"*, Electronics, April 19, 1965, pages 114-117.

[199]S. Mourad and Y. Zorian, "Principles of Testing Electronic Systems", John Wiley & Sons, Inc., ISBN 0-471-31931-7, 2000.

[200]H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by Sequence-Pair", *Transactions on Computer-Aided Design of Integrated Circuits and Systems*," Volume: 15, Issue: 12, December 1996, pages 1518-1524.

[201]V. Muresan, X. Wang, V. Muresan, and M. Vladutiu, "Greedy Tree Growing Heuristics on Block-Test Scheduling Under Power Constraints", *Journal of Electronic Testing: Theory and Applications (JETTA)*, 20, 2004, pages 61-78.

[202]V. Muresan, X. Wang, V. Muresan, and M. Vladutiu, "A Comparison of Classical Scheduling Approaches in Power-Constrained Block-Test Scheduling", *Proceedings of International Test Conference (ITC*), Atlantic City, NJ, October 2000, pages 882-891.

[203]P. Muth, "A Nine-Valued Circuit Model for Test Generation", Transactions on Computers, Vol. C-25, June 1976, pages 630-636.

[204]B. Nadeau-Dostie (Editor), "Design for At-Speed Test, Diagnosis and Measurement", Kluwer Academic Publisher, ISBN 0-7923-8669-8, 2000.

[205]R. Nair, "Comments on an Optimal Algorithm for Testing Stuck-At Faults in Random Access Memories", *Transactions on Computers*, March 1979, pages 258-261.

[206]R. Nair, "Efficient Algorithms for Testing Semiconductor Random-Access Memories", *Transactions on Computers,* Vol. C-28, No. 6, 1979, pages 672-676.

[207]N. Nicolici and B. M. Al-Hashimi, "Power Conscious Test Synthesis and Scheduling for BIST RTL Data Paths", *Proceedings of International Test Conference (ITC)*, Atlantic City, NJ, October 2000, pages 662-671.

[208]N. Nicolici and B. Al-Hashimi, "Multiple Scan Chains for Power Minimization During Test Application in Sequential Circuits", *Transactions on Computers*, Vol. 51, No. 6, June 2002, pages 721-734.

[209]N. Nicolici and B. M. Al-Hashimi, "Power-Constrained Testing of VLSI Circuits", Kluwer Academic Publisher, ISBN 1-4020-7235-X, 2003.

[210]N. Nicolici and B. M. Al-Hashimi, "Power-Conscious Test Synthesis and Scheduling", *Design & Test of Computers*, July/August 2003, pages 48-55.

[211]R. B. Norwood and E. J. McCluskey, "Synthesis-for-Scan and Scan Chain Ordering", *Proceedings of the VLSI Test Symposium (VTS),* Princeton, NJ, USA, April 1996, pages 87-92.

[212]M. Nourani and C. Papachristou, "An ILP Formulation to Optimize Test Access Mechanisms in System-On-A-Chip Testing", *Proceedings of International Test Conference (ITC),* pp 902-910, Atlantic City, NJ, USA, October 2000.

[213]M. Nourani and J. Chin, "Power-Time Trade Off in Test Scheduling for SoCs", *Proceedings of International Conference on Computer Design (ICCD),* San Jose, CA, USA, October 2003, pages 548-553.

[214]K. P. Parker and S. Oresjo, "A Language for Describing Boundary-Scan Devices", *Proceedings of International Test Conference (ITC)*, Washington, DC, USA, September 1990, pages 222-234.

[215]K. P. Parker and S. Oresjo, "A Language for Describing Boundary-Scan Devices", *Journal on Electronic Testing: Theory and Application (JETTA)*, Vol. 2, No. 1, pages 43-74, 1991.

[216]K. P. Parker, "The Boundary-Scan Handbook", Boston: Kluwer Academic Publishers, second edition, 1998.

[217]J. L. Peterson, Petri net theory and the modeling of systems, *Prentice-Hall, Inc.,* ISBN 0-13-661983-5, 1981.

[218]J. Pouget, E. Larsson, Z. Peng, M.-L. Flottes, and B. Rouzeyre, “An Efficient Approach to SoC Wrapper Design, TAM Configuration and Test Scheduling”, *Informal Proceedings of European Test Workshop (ETW)*, Maastricht, The Netherlands, May 2003, pages 117-122.

[219]J. Pouget, E. Larsson, Z. Peng, M.-L. Flottes, and B. Rouzeyre, “An Efficient Approach to SoC Wrapper Design, TAM Configuration and Test Scheduling”, *Formal Proceedings of European Test Workshop (ETW)*, Maastricht, The Netherlands, May 2003, pages 51-56.

[220]J. Pouget, E. Larsson, and Z. Peng, “SOC Test Time Minimization Under Multiple Constraints”, *Proceedings of Asian Test Symposium (ATS03),* Xian, China, November 2003, pages 312-317.

[221]G. R. Putzolu and J. P. Roth, “A Heuristic Algorithm for the Testing of Asynchronous Circuits”, Transactions on Computers, Vol. C-20, June 1971, pages 639-647.

[222]J. Rajski, J. Tyszer, and N. Zacharia, “Test Data Compression For Multiple Scan Designs With Boundary Scan”, *Transactions on Computers*, Vol. 47, No. 11, 1998, pages 1188-1200.

[223]J. Rajski, J. Tyzer, M. Kassab, N. Mukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian “Embeeded Deterministic Test For Low Cost Manufacturing Test”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2002, pages 301-310.

[224]R. Rajsuman, “Testing a System-on-a-Chip with Embedded Microprocessor”, *Proceedings of International Test Conference (ITC)*, Atlantic City, NJ, USA, September 1999, pages 499-508.

[225]R. Rajsuman, “System-on-a-Chip: Design and Test”, Artech House Publishers, 2000, ISBN 1-58053-107-5

[226]S. Ravi, G Lakshminarayana, and N.K. Jha, ”Testing of Core-based Systems-on-a-Chip”, *Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Volume: 20 , Issue: 3, March 2001, pages 426 - 439.

[227]C. P. Ravikumar, G. Chandra, and A. Verma, “Simultaneous Module Selection and Scheduling for Power-constrained Testing of Core Based Systerms”, *Proceedings of International Conference on VLSI Design*, Calcutta, India, January 2000, pages 462-467.

[228]C. P. Ravikumar, A. Verma, and G. Chandra, “A Polynomial-time Algorithm for Power Constrained Testing of Core Based Systems”, *Proceedings of Asian Test Symposium (ATS)*, Shanghai, China, November 1999, pages 107-112.

[229]C. R. Reeves, Modern Heuristic Techniques for Combinatorial Problems, *Blackwell Scientific Publications,* ISBN 0-632-03238-3, 1993.

[230]P. M. Rosinger, B. M. Al-Hashimi, and N. Nicolici, “Scan Architecture for Shift and Capture Cycle Power Reduction”, *Proceedings of International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, Vancouver, Canada, November 2002, pages 129-137.

[231]P. M. Rosinger, B. M. Al-Hashimi, and N. Nicolici, “Power Profile Manipulation: A New Approach for Reducing Test Application Time under Power Constraints”, *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume 21, Issue 10, October 2002, pages 1217-1225.

[232]P. Rosinger, B. Al-Hashimi, and N. Nicolici, “Power Constrained Test Scheduling Using Power Profile Manipulation”, *Proceedings of International Symposium on Circuits and Systems (ISCAS)*, Sydney, Australia, May 2001, Volume 5, pages (V)251-(V)254.

[233]J. P. Roth, “Diagnosis of Automata Failures: A Calculus and a Method”, *IBM Journal of Research and Development, 10:4,* July 1966, pages 278-291.

[234]J. Savir, “Syndrome-Testable Design of Combinational Circuits”, *Transactions on Computers*, Volume 29, Number 6, June 1980, pages 442-451.

[235]J. Savir, “Skewed-Load Transition Test: Part 1, Calculus”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, September 1992, pages 705-713.

[236]S. Patil and J. Savir, “Skewed-Load Transition Test: Part 2, Coverage”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, September 1992, pages 714-722.

[237]A. Sangiovanni-Vincentelli, “Defining Platform-based Design”, *EEDesign of EETimes*, February 2002.

[238]J. Savir and S. Patil, “Scan-Based Transition Test”, *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 8, August 1993, pages 1232-1241.

[239]J. Savir and S. Patil, ”Broad-Side Delay Test”, *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 8, August 1994, pages 1057-1064.

[240]J. Savir and S. Patil, ”On Broad-Side Delay Test”, *Transactions on VLSI*, Vol. 2, No. 3, September 1994, pages 368-372.

[241]J. Saxena, K. M. Butler, and L. Whetsel, “An Analysis of Power Reduction Techniques in Scan Testing”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2001, pages 670-677.

[242]A. Sehgal, V. Iyengar, M. D. Krasniewski, and K. Chakrabarty, “Test Cost Reduction for SOCs Using Virtual TAMs and Lagrange Multipliers”, *Proceedings Design Automation Conference (DAC)*, Anaheim, CA, USA, June 2003, pages 738-743.

[243]A. Sehgal and K. Chakrabarty, “Efficient Modular Testing of SoCs Using Dual-Speed Tam Architectures”, *Proceedings of Design and Test in Europe (DATE)*, Paris, France, 2004, pages 422-427.

[244]A. Sehgal, V. Iyengar and K. Chakrabarty, "SOC Test Planning Using Virtual Test Access Architectures", *Transactions on VLSI Systems*, vol. 12, September 2004.

[245]A. Sehgal, S. K. Goel, E. J. Marinissen and K. Chakrabarty, “IEEE P1500-Compliant Test Wrapper Design for Hierarchical Cores”, *Proceedings of International Test Conference (ITC)*, Charlotte, NC, USA, October 2004.

[246]F. F. Sellers, M. Y. Hsiao, and C. L Bearnson, “Analyzing Errors with the Boolean Difference”, *Transactions on Computers*, Volume C-17, July 1968, pages 676-683.

[247]A. K. Sharma, “Semiconductor Memories: Technology, Testing, and Reliability”, IEEE Press, Piscataway, NJ, USA.

[248]N. Singh, “An Artificial Intelligence Approach to Test Generation”, Kluwer Academic Publisher, Norwell, MA, 1987.

[249]T. J. Snethen, “Simulator Oriented Fault Test Generator”, *Proceedings of Design Automation Conference (DAC)*, New Orleans, LA, USA, June 1977, pages 88-93.

[250]M. Soma, “Automatic Test Generation Algorithms for Analog Amplifiers”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 1993, pages 566-573.

[251]T. Sridhar and J. P. Hayes, “A Functional Approach to Testing Bit-Sliced Microprocessors”, *Transactions on Computers*, Vol. 1, June 1984, pages 475-485.

[252]Standard Delay Format Specification, Version 2.1, Open Verilog International: www.eda.org/sdf.

[253]A. Steiningerl, “Testing and Built-In Self-Test - A Survey”, *Journal of System Architecture,* ISSN-1383-7621, Vol. 46, No. 9, July 2000, pages 721-747.

[254]C. E. Stroud, “A Designer’s Guide to Built-In Self-Test”, Kluwer Academic Publisher, ISBN 1-4020-7050-0, 2002.

[255]M. Sugihara, H. Date, and H. Yasuura, “A Novel Test Methodology for Core-Based System LSIs and a Testing Time Minimization Problem”, *Proceedings of International Test Conference (ITC)*, Washington, DC, USA, October 1998, pages 465-472.

[256]M. Sugihara, H. Date, and H. Yasuura, “A Test Methodology for Core-Based System LSIs”, *IEICE Transactions on Fundamentals,* Vol. E81-A, No. 12, December 1998, pages 2640-2645.

[257]M. Sugihara, H. Date, and H. Yasuura, “Analysis and Minimization of Test Time in a Combined BIST and External Test Approach”, *Proceedings of Design and Test in Europe (DATE),* March 2000, pages 134-140.

[258]C. - P. Su and C. -W. Wu, “A Graph-Based Approach to Power-Constrained SOC Test Scheduling”, *Journal of Electronic Testing: Theory and Applications (JETTA)*, 20(1), February 2004, pages 45-60.

[259]Y. Takamatsu and K. Kinoshita, “CONT: A Concurrent Test Generation Algorithm”, *Digest of Papers of Fault-Tolerant Computing Symposium (FTCS-17)*, Pittsburg, PA, USA, July 1987, pages 22-27.

[260]Teradyne, http://www.teradyne.com

[261]N. Touba and B. Pouya, “Using Partial Isolation Rings to Test Core-Based Designs”, *Design and Test of Computers*, 14(4), December 1997, pages 52-59.

[262]N. Touba and B. Pouya, “Testing Embedded Cores Using Partial Isolation Rings”, *Proceedings of VLSI Test Symposium (VTS)*, May 1997, pages 10-16.

[263]J. L. Turino, "Design to test: a Definitive Guide for Electronic Design, Manufactoring, and Service", Van Nostrand Reinhold, 1990, ISBN-0-442-00170-3.

[264]F. F. Tsui, LSI/VLSI Testability Design, *McGraw-Hill Book Company,* ISBN 0-07-100356-8, 1988.

[265]Tuinenga, P. W., “SPICE, A Guide to Circuit Simulation & Analysis Using PSPICE”, Englewood Cliffs, NJ, Prentice Hall, 1992.

[266]J. Turley, “Embedded Processors by the Numbers”, *Embedded Systems Programming*, vol. 12, May 1999, pages 13-14.

[267]P. Varma and S. Bhatia, “A Structured Test Re-Use Methodology for Core-Based System Chips”, *Proceedings of International Test Conference (ITC)*, Washington, DC, USA, October 1998, pages 294-302.

[268]“VHDL Language Reference Manual”, IEEE Std 1076-1993 (ISBN 1-55937-376-8).

[269]B. Vinnakota, editor, Analog and Mixed-Signal Test, Upper Saddle River, New Jersey:Prentice-Hall, 1998.

[270]E. H. Volkernik, A. Khoche, and S. Mitra, “Packet-based Input Test Data Compression Techniques”, *Proceedings of International Test Conference (ITC)*, Baltimore, MD, USA, October 2002, pages 154-163.

[271]E. H. Volkernik, A. Khoche, J. Rivoir, and K. D. Hilliges, "Modern Techniques: Tradeoffs, Synergies, and Scalable Benefits", *Journal of Electronic Testing: Theory and Applications (JETTA)*, Vol. 19, 2003, pages 125-135.

[272]H. Vranken, F. Hapke, S. Rogge, D. Chindamo, and E. Volkerink, "ATPG Padding And ATE Vector Repeat Per Port For Reducing Test Data Volume", *Proceedings of International Test Conferenence (ITC),* Charlotte, NC, USA, October 2003, pages 1069-1078.

[273]T. Waayers, "An Improved Test Control Architecture for Core-Based System Chips", *Informal Proceedings of European Test Workshop (ETW)*, Maastricht, The Netherlands, May 2003, pages 333-338.

[274]T. Waayers, "An Improved Test Control Architecture and Test Control Expansion for Core-Based System Chips", *Proceedings of International Test Conference (ITC)*, Charlotte, NC, USA, October 2003, pages 1145-1154.

[275]C.-W. Wang, J.-R. Huang, Y.-F. Lin, K.-L. Cheng, C.-T. Huang, and C.-W. Wu, "Test Scheduling of BISTed Memory Cores for SOC", *Proceedings of Asian Test Symposium (ATS)*, Tamuning, Guam, USA, November 2002, pages 356-361.

[276]C.-W. Wang, J.-R. Huang, K.-L. Cheng, H. - S. Hsu, C.-T. Huang, and C.-W. Wu, "A Test Access Control and Test Integration System for System-on-Chip", *Workshop on Testing Embedded Core-Based System-Chips (TECS)*, Monterey, CA, USA, May 2002, pages P2.1-P2.8.

[277]N. H. E.Weste and K. Eshraghian, Principles of CMOS VLSI Design, *Addison-Wesley*, ISBN 0-201-53376-6, 1992.

[278]L. Whetsel, "An IEEE 1149.1 Based Test Access Architecture for ICs with Embedded Cores", *Proceedings of International Test Conference (ITC)*, Washington, DC, USA, November 1997, pages 69-78.

[279]Y. Xia, M. Chranowska-Jeske, B. Wang, and M. Jeske, "Using a Distributed Rectangle Bin-Packing Approach for Core-based SoC Test Scheduling with Power Constraints", *Proceedings of International Conference on Computer-Aided Design* (*ICCAD*), San Jose, CA, USA, November 2003, pages 100-105.

[280]Q. Xu and N. Nicolici, “Delay Fault Testing of Core-Based System-on-a-Chip”, *Proceedings of Design Automation and Test in Europe (DATE)*, Munchen, Germany, March 2003, pages 744-749.

[281]Q. Xu and N. Nicolici, “On Reducing Wrapper Boundary Register Cells in Modular SOC Testing”, *Proceedings of International Test conference (ITC)*, Charlotte, NC, USA, September 2003, pages 622-631.

[282]Q. Xu and N. Nicolici, “Wrapper Design for Testing IP Cores with Multiple Clock Domains”, *Proceedings of Design Automation and Test in Europe (DATE)*, Paris, France, February 2004, pages 416-421.

[283]T. Yoneda and H. Fujiwara, “A DFT Method for Core-Based Systems-on-a-Chip based on Consecutive Testability,” *Proceedings of Asian Test Symposium (ATS)*, Kyoto, Japan, November 2001, pages 193-198.

[284]T. Yoneda and H. Fujiwara, “Design for Consecutive Testability of System-on-a-Chip with Built-In Self Testable Cores,” *Journal of Electronic Testing: Theory and Applications (JETTA) Special Issue on Plug-and-Play Test Automation for System-on-a-Chip*, Vol. 18, No. 4/5, August/October 2002, pages 487-501.

[285]T. Yoneda and H. Fujiwara, “Design for ConsecutiveTransparency of Cores in System-on-a-Chip”, *Proceedings of VLSI Test Symposium (VTS)*, Napa Valley, CA, USA, April 2003, pages 287-292.

[286]T. Yoneda, T. Uchiyama and H. Fujiwara, "Area and Time Co-Optimization for System-on-a-Chip based on Consecutive Testability," *Proceedings of International Test Conference (ITC)*, Charlotte, NC, USA, September 2003, pages 415-422.

[287]Y. Zorian, “A Distributed BIST Control Scheme for Complex VLSI Devices”, *Proceedings of VLSI Test Symposium (VTS)*, Atlantic City, NJ, USA, April 1993, pages 4-9.

[288]Y. Zorian, “Test Requirements for Embedded Core-Based Systems and IEEE P1500”, *Proceedings of International Test Conference (ITC)*, Washington, DC, USA, November 1997, pages 191-199.

[289]Y. Zorian, E. J. Marinissen, and S. Dey, “Testing Embedded-Core Based System Chip”, *Proceedings of International Test Conference* (*ITC*), Washington, DC, October 1998, pages 130-143.

[290]D. Zhao and S. Upadhyaya, “Adaptive Test Scheduling in SoCs by Dynamic Partitioning”, *Proceedings of International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, Vancouver, Canada, November 2002, pages 334-342.

[291]D. Zhao and S. Upadhyaya, “Power Constrained Test Scheduling with Dynamically Varied TAM”, *Proceedings of VLSI Test Symposium (VTS)*, Napa Valley, CA, USA, April 2003, pages 273-278.

[292]W. Zou, S. R. Reddy, I. Pomerance, Y. Huang, “SOC Test Scheduling Using Simulated Annealing”, *Proceedings of VLSI Test Symposium* (*VTS*), Napa Valley, CA, USA, April 2003, pages 325-330.

Index