
**SILICON IMPLEMENTATION OF
PULSE CODED NEURAL NETWORKS**

**THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE**

SILICON IMPLEMENTATION OF PULSE CODED NEURAL NETWORKS

edited by

Mona E. Zaghoul
The George Washington University

Jack L. Meador
Washington State University

Robert W. Newcomb
University of Maryland



SPRINGER SCIENCE+BUSINESS MEDIA, LLC

Library of Congress Cataloging-in-Publication Data

Silicon implementation of pulse coded neural networks / edited by Mona E. Zaghoul, Jack L. Meador, Robert W. Newcomb.

p. cm. -- (The Kluwer international series in engineering and computer science)

Includes bibliographical references and index.

ISBN 978-1-4613-6152-7 ISBN 978-1-4615-2680-3 (eBook)

DOI 10.1007/978-1-4615-2680-3

1. Neural networks (Computer science) 2. Semiconductors.

I. Zaghoul, M. E. (Mona Elwakkad) II. Meador, Jack L.

III. Newcomb, Robert W. IV. Series.

QA76.87.S55 1994

006.3'3--dc20

93-48181

CIP

Copyright © 1994 by Springer Science+Business Media New York

Originally published by Kluwer Academic Publishers in 1994

Softcover reprint of the hardcover 1st edition 1994

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, mechanical, photo-copying, recording, or otherwise, without the prior written permission of the publisher, Springer Science+Business Media, LLC.

Printed on acid-free paper.

TABLE OF CONTENTS

PREFACE	vii
1. Some Historical Perspectives on Early Pulse Coded Neural Network Circuits <i>R. W. Newcomb</i>	1
2. Pulse Techniques in Neural VLSI: A Review <i>A. F. Murray</i>	9
3. Silicon Dendritic Trees <i>J. G. Elias</i>	39
4. Silicon Neurons for Phase and Frequency Detection and Pattern Generation <i>M. DeYong and C. Fields</i>	65
5. Pulse Coded Winner-Take-All Networks <i>J. L. Meador and P. D. Hylander</i>	79
6. Realization of Boolean Functions Using a Pulse Coded Neuron <i>M. de Savigny and R. W. Newcomb</i>	101
7. Design of Pulse Coded Neural Processing Element Using Modified Neural Type Cells <i>G. Moon and M. E. Zaghloul</i>	113
8. Low-Power Silicon Neurons, Axons and Synapses <i>J. Lazzaro and J. Wawrzynek</i>	153
9. Synchronous Pulse Density Modulation in Neural Network Implementation <i>J. Tomberg</i>	165

10.	CMOS Analog Neural Network Systems Based on Oscillatory Neurons <i>B. Linares-Barranco, E. Sanchez-Sinencio, A. Rodriguez-Vazquez and J. L. Huertas</i>	199
11.	A Digital Neural Network Architecture Using Random Pulse Trains <i>G. R. Salam and R. M. Goodman</i>	249
12.	An Unsupervised Neural Processor <i>J. Donald and L. A. Akers</i>	263
	INDEX	291

Preface

When confronted with the hows and whys of nature's computational engines, some prefer to focus upon neural function: addressing issues of neural system behavior and its relation to natural intelligence. Then there are those who prefer the study of the "mechanics" of neural systems: the nuts and bolts of the "wetware": the neurons and synapses. Those who investigate pulse coded implementations of artificial neural networks know what it means to stand at the boundary which lies between these two worlds: not just asking why natural neural systems behave as they do, but also how they achieve their marvelous feats. The research results presented in this book not only address more conventional abstract notions of neural-like processing, but also the more specific details of neural-like *processors*.

It has been established for some time that natural neural systems perform a great deal of information processing via electrochemical pulses. Accordingly, pulse coded neural network concepts are receiving increased attention in artificial neural network research. This increased interest is compounded by continuing advances in the field of VLSI circuit design. This is the first time in history in which it is practical to construct networks of neuron-like circuits of reasonable complexity that can be applied to real problems. We believe that the pioneering work in artificial neural systems presented in this book will lead to further advances that will not only be useful in some practical sense, but may also provide some additional insight into the operation of their natural counterparts.

The idea of creating a book dedicated to pulse coding in neural VLSI was first conceived at the International Conference on Circuits and Systems by Mona Zaghloul in the spring of 1992. A special session had been organized by Jack Meador and Robert Newcomb for the conference which was the first of its kind and had attracted a number of researchers from both Europe and the US who are working in the area. A poll of authors presenting at that session showed unanimous support for the development of a book which would showcase recent progress in pulse coded implementations. This book represents the combined contributions of both those who were present at that first special session as well as several others who are currently conducting research in this fascinating area.

This volume seeks to cover many of the relevant contemporary studies coming out of this newly emerging area. It is essentially a selection, reorganization, and expansion of recent publications. In addition, Prof. Robert

Newcomb provides a historical perspective on early pulse coded neural network circuits in Chapter 1. This is followed by a review of more recent pulsed neural VLSI techniques given by Alan Murray in Chapter 2. In that chapter, Dr. Murray reviews the variety of pulse coding strategies available and discusses pulse implementation techniques being investigated at the University of Edinburgh. Silicon dendritic trees which are intended to emulate the behavior of spatially extensive biological neurons are then discussed by John Elias in Chapter 3. A model for a typical spiking neuron is described by Mark DeYong and Chris Fields in Chapter 4 and is shown to be useful for phase and frequency detection, filtering, and pattern encoding. In Chapter 5, Jack Meador and Paul Hylander present a pulse coded winner-take-all network which encodes decision strength as a variable rate pulse train generated by the winning unit. The Neural Type Cell which is a special circuit that accepts analog inputs and produces pulse coded outputs is used in Chapter 6 by Marc De Savigny and Robert Newcomb to implement Boolean functions as well as by Gyu Moon and Mona Zaghoul in Chapter 7 as part of a unique processing element that can be used as a basic cell for artificial neural network implementation. Low power silicon implementation of Neurons, Axons and Synapses are introduced by John Lazzaro, and John Wawrzynek in Chapter 8. These new implementations improve upon the power dissipation characteristics of the self-resetting neuron and silicon axon described by Carver Mead in his popular 1989 book. Synchronous pulse density modulation techniques are presented in Chapter 9 by Jouni Tomberg. Both switched capacitor and digital implementations of synchronous pulse coded arithmetic are discussed in the context of established network architectures here. The transconductance mode technique is presented in Chapter 10 by Bernabe Linares-Barranco, Edgar Sanchez-Sinencio, Angel Rodriquez-Vazquez, and Jose Huertas to implement an equivalent circuit for an oscillatory neuron based on the simplified operation of the living neuron. This oscillatory neuron is used to implement and test established neural architectures. In Chapter 11 a digital architecture that utilizes random bit sequence properties is described by Gamze Salam and Rodney Goodman. The extension of previously established stochastic pulse coded implementation techniques are presented and demonstrated. Finally in Chapter 12, James Donald and Lex Akers present experimental results obtained with two adaptive pulse coded signal processing chips for real-time control. These chips use pulse coded signals for inter-chip communication and analog weights for information storage. In all of this work, novel circuits are presented and innovative techniques for implementing conventional neural network architectures are introduced.

By way of acknowledgment we would first like to thank all of the contributing authors for their prompt assistance with the details, some requiring their attention on very short notice. We would also like to express special thanks to Paul Hylander and Ioana Pirvulescu for their suggestions and assistance with the early drafts of the book. We would also like to express our gratitude to Holly Snyder for her assistance with the details of final draft preparation.

Mona Zaghoul

Jack Meador

Robert Newcomb

SOME HISTORICAL PERSPECTIVES ON EARLY PULSE CODED NEURAL NETWORK CIRCUITS

Robert W. Newcomb

*Electrical Engineering Department
University of Maryland, College Park MD, 20742*

PERSPECTIVES

From the beginnings of mankind the means of brain activity must have fascinated man. And although Galvani had shown in the late 1700s that muscles were excited by electrical activity of the nerves [Galvani 1791, Brazier 61] it was not known through most of the 1800's what was the basis for activity of the brain - indeed it is still unknown how a person thinks. In any event the publication by the Polish neurophysiologist Adolf Beck in the *Centralblatt für Physiologie* [Beck 1890], concerning his measurements of electrical activity in the brain [Beck 1888], caused considerable controversy as to whom was the first one to achieve such an accomplishment. After almost all sides were heard from, the controversy was settled by a further letter to the *Centralblatt* by Richard Caton calling attention to the measurements he had reported to the August 24, 1875, meeting of the British Medical Association and recorded in the report of the meeting [Caton 1875]. Among statements in Caton's original report is the following: "When any part of the grey matter is in a state of functional activity, its electric current usually exhibits negative variation" [Brazier 61] where by "negative variation" at the time was meant action potentials. Thus, we see that measurements were made on the pulse coded electrical activity in the brain as early as 1875.

Nevertheless it was well into the 1930's before really significant measurements were begun. Once such measurements were initiated more and more sophisticated measurements were needed and for them more elaborate electronic circuits were developed for that purpose. Among those making such measurements in the 1930's was Dr. Otto H. Schmitt, who devised a means of solving the equations proposed in theories of biological impulse propagation via vacuum tube circuits [Schmitt 37a].

From my search of the literature it appears that Dr. Schmitt should be given the credit for the first electronic circuit specifically designed as a pulse coded neural circuit. This occurs in his April 1937 paper "An Electrical Theory of Nerve Impulse Propagation" [Schmitt 37b] for which only the abstract survives in the open literature. From the two abstracts, [Schmitt 37a, Schmitt 37b], it is clear that an electrical circuit was built to test the theory of "impulse propagation" as the following is stated in [Schmitt 37b]: "The validity of the theory is tested by comparing the behavior of this artificial "nerve" with that of real nerve." It is also clear that Dr. Schmitt had in mind the use of such circuits for the simulation of live neuron behavior since the above quote continues: "If the theory is valid and the electrical model is a suitable equivalent, then certain of the unmeasurable electrical characteristics of nerve can be evaluated in terms of the constants in the electrical "nerve" required to make its performance duplicate that of real nerve." This viewpoint is further strengthened by private correspondence of Dr. Schmitt to the author (dated June 9, 1993) in which he states that his 1937 Ph.D. was "in Physics, Mathematics and Zoology with the topic of simulating the nerve axon with computer available components." Finally it is also clear that the purposes of these circuits was to better study neural function, rather than to make better computers or controllers, since the abstract closes with: "The way would then be open for a study of the mechanisms of the effects of abnormal agents such as drugs, ion imbalances, and the like."

Dr. Otto H. Schmitt's position is borne out further by the following quote from the 1979 article of Pellionisz [Pellionisz 79] for which it should be noted that [Schmitt 37a] follows [Schmitt 37b] on the original printed page and that [Schmitt 37a] is an abstract for a demonstration of the working circuits: "The third approach was the physical representation of highly simplified neurons by small electrical circuits (so called neuromeme: see Harmon, 1959; McGrogan, 1961; Jenik, 1962; Lewis, 1964). Although this approach received a great impetus from the widespread acceptance of the McCulloch-Pitts concept, it is actually rooted in an idea of Schmitt (1937), who created the binary-output electrical circuit that realizes threshold function (the Schmitt trigger)." These references are [Harmon 59], [McGrogan 61], [Jenik 62], [Lewis 64], and

[Schmitt 37a] for Schmitt. It should be noted that it is after Dr. Schmitt that the Schmitt trigger is named, this being the regenerative comparator which gives hysteresis familiar to almost all undergraduate students studying electronics [Millman 79, p. 623]. And, although the Schmitt trigger was published as a circuit in its own right [Schmitt 38], it is clear that the work on pulsing in neural circuits had a considerable influence on its development, if not being the primary reason for its existence.

As we can see from the references of Pellionisz cited above, there was a gap of roughly 15 years before further serious work on electronic circuits occurred after that cited of Schmitt. In the meantime the fundamental paper of Hodgkin and Huxley [Hodgkin 52] was published. Having given a mathematical treatment for the generation and processing of action potentials, this paper [Hodgkin 52] spawned a large number of circuits for simulating these equations and, as a consequence, stimulated the development of many aspects in the modeling of neural behavior [Eccles 57]. A typical circuit of the era used a large number of bipolar transistors, both npn and pnp, incorporated relays and many transistors and capacitors, see for example Figure 11 of [Jenik 64]. The references cited by Pellionisz give other representative circuits with the book of MacGregor and Lewis [MacGregor 77] somewhat giving the state of the art of the ideas surrounding circuits for neural modeling in the mid 1970's. However, to anyone versed in present day VLSI technology it is clear that it would not be practical to build VLSI circuits of any size using the neurons of most of the circuits published into the early 1970s.

With this last comment in mind it is worth noting that in the early and mid 1960s another philosophy emerged. This was the idea that rather than simulating the actual behavior of neurons, as set up in equations like those of Hodgkin and Huxley it might be expedient to mimic their behavior through more concise abstractions. Although a move in this direction can be seen in the work of the Applied Research Department team at RCA, [Putzrath 61, Martin 61, McGrogan 61], and of Harmon [Harmon 59] at Bell Laboratories, in my mind the most significant idea of the 1960s toward this philosophy is contained in the "neuristor" introduced by H. Crane in his 1960 doctoral dissertation at Stanford [Crane 60]. This was published two years later in the *Proceedings of the IRE* [Crane 62] where in the same issue appeared a very interesting circuit by Nagumo, et al [Nagumo 62], for mimicking nerve axon propagation, in essence giving a circuit realization of the neuristor, albeit quite impractical for VLSI due to its use of tunnel diodes (a similar type of circuit was actually given earlier by Cote [Cote 61] but the paper by Nagumo, et al, has received more recognition, probably due to its rather elegant mathematical development of axon mimicking equations). The idea of the neuristor was to

abstract the four key axon properties of threshold of excitation, refractory period, attenuationless and uniform speed of propagation from which new classes of computers could be conceived following upon the structure of neural systems. Although in some sense impractical for circuit realizations due to the need for lines and circulation of pulses upon the lines, the idea of abstracting the neural system properties into ones that are tractable for electronic realization is one that appears to be paying off. In any event the neuristor and its derivatives led to a large number of circuits being proposed for neural system realizations in the mid 1960s through the mid 1970s, in Europe, Japan and the US. Many of the pertinent references to that era are listed in [Newcomb 79], a paper appearing originally in Spanish out of friendship for a Latin American colleague interested in the ideas.

Although there were a number of isolated studies throughout the world at the time, one of the most interesting group undertakings was the Polish-USA neural-type microsystems studies funded in the early 1970s by the US National Science Foundation (under grants 42178 and 75-03227) using Polish wheat purchase debt funds to finance the Polish side of the research. This program had as its goal the development of bipolar and MOS circuits suitable for integrated circuit construction that would mimic the behavior of neural systems, and, hence, called "neural-type," a word coined by Professor N. DeClaris. On the Polish side this was directed by Dr. M. Bialko of Politecnica Gdansk, with involvement of C. Czarnul, B. Wilamowski, and J. Zurada, among others, and on the US side was jointly directed by Dr. N. DeClaris and myself, both of the University of Maryland, with involvement primarily of C. Kohli. The main results to come out of this research in the pulse coded hardware area were the development (primarily by B. Wilamowski) on the Polish side of a bipolar circuit that acted as an action-potential-like-pulse generator [Wilamowski 75] and a companion MOS circuit [Kulkarni-Kohli 76] developed on the US side (primarily by C. Kohli) as well as a pulse processing circuit jointly developed [Czarnul 76] (primarily by C. Czarnul). Probably it is fair to say that of all the circuits developed during the historical period of 1930 - 1980, only the MOS circuit of [Kulkarni-Kohli 76] has survived with any type of vitality and that in a form very much modified and improved upon to take advantage of present day technology; see the paper by G. Moon and that by M. de Savigny which follow in this volume. But all of the historical circuits, which were primarily of the pulse processing type well into the 1980s, have served the purpose of leading us step by step to better artificial neural networks. It is also clear from the historical record that the lack of funding during the 1975-1985 decade led to a new breed of researchers and a new set of ideas, most of which were not founded upon the pulse processing

philosophy. One important change of emphasis that has come about has been the concentration of interest upon synaptic influences, as incorporated in modern theories via the weights. In the research into the 1970s most of the circuits concentrated upon developing axon behavior with the synaptic junctions somewhat overlooked. In any event, it is refreshing for me to see an upsurge of interest in the pulse coding techniques since there are clear advantages, as biological systems discovered eons ago.

Because of the intrinsic interest in neural networks and the evident significance to the future of hardware developments to the field of artificial neural networks, it seems to me of considerable importance to have a well documented history of the hardware developments, especially of pulse coded circuits. Thus, I have tried to concisely put down some of what I am aware. But this is only a start and in the end I would emphasize that only an outline of what I consider to be the historically significant activities into the mid 1970s have been given here. And probably of that some has been missed as I discovered by recently locating a paper on a pulse code optoelectronic-magnetic neuron by Bray in 1961 [Bray 61]. In my mind this is a field of which we who are alive today can take great pride when we come to the point of being able to sit as elders on our porch swings relating the developments of the times to our grandchildren, as historically elders have done through the ages. Thus, it is with considerable anticipation that I await a thorough treatment by a competent historian of this fascinating field of modern history.

References

- [Beck 1888] A. Beck, "O pobudliwosci roznych miejsc tego samego nerwu," *Rozprawy Wydzsal matematyczno-przyrodniczy Polska Akademia Umiejelnosci*, Vol. 15, 1888, pp. 165-195 (cited in [Brazier 61, p. 49]).
- [Beck 1890] A. Beck, "Die Bestimmung der Localisation der Gehirn- und Rückenmarkfunctionen vermittelst der electrischen Erscheinungen," *Centralblatt für Physiologie*, Vol. 4, 1890, pp. 473-476 (cited in [Brazier 61, p. 49]).
- [Bray 61] T. E. Bray, "An Optoelectronic-Magnetic Neuron Component," Proceedings of the National Electronics Conference, Chicago, IL, Vol. 17, October 1961, pp. 322 - 326.

- [Brazier 61] M. A. B. Brazier, "A History of the Electrical Activity of the Brain - The First Half-Century," Pitman Medical Publishing Co., Ltd., London, 1961.
- [Caton 1875] R. Caton, "The Electric Currents of the Brain," *British Medical Journal* Vol. 2, 1875, p. 278 (cited in [Brazier 61, p. 25])
- [Crane 60] H. D. Crane, "Neuristor Studies," Technical Report No. 1506-2, Stanford Electronics Laboratory, July 11, 1960.
- [Crane 62] H. D. Crane, "Neuristor - A Novel Device and System Concept," *Proceedings of the IRE*, Vol. 50, No. 10, October, 1962, pp. 2048-2060.
- [Cote 61] A. J. Cote, Jr., "A Neuristor Prototype," *Proceedings of the IRE*, Vol. 49, No. 9, September 1961, pp. 1430-1431.
- [Czarnul 76] C. Czarnul, M. Bialko, and R. W. Newcomb, "Neuristor-Line Pulse-Train Selector," *Electronics Letters*, Vol. 12, No. 8, April 15, 1976, pp 207-208.
- [Eccles 57] J. C. Eccles, "The Physiology of Nerve Cells," The Johns Hopkins Press, Baltimore, 1957.
- [Galvani 1791] L. Galvani, "De Viribus Electricitatis in Motu Musculari," *Commentarius*, Bolgna, 1791 (cited in [Brazier 61,p. 2]).
- [Harmon 59] L. D. Harmon, Artificial Neuron, *Science*, Vol. 129, No. 3354, April 10, 1959, pp. 962-963.
- [Hodgkin 52] A. L. Hodgkin and A. F. Huxley, "A Quatitative Description of Membrane Current and Its application to Conduction and Excitation in Nerve," *The Journal of Physiology*, Vol 117, No. 4, August 28, 1952, pp. 500 - 544.
- [Jenik 62] F. Jenik, "Electronic Neuron Models as an Aid to Neurophysiological Research, " *Ergebnisse der Biologie.*, Vol. 25, 1962, pp. 206-245.

- [Jenik 64] F. Jenik, "Pulse Processing by Neuron Models," in "Neural Theory and Modeling" edited by R. F. Reiss, Stanford University Press, Stanford, CA, 1964, pp. 190-212.
- [Kulkarni-Kohli 76] C. Kulkarni-Kohli and R. W. Newcomb, "An Integrable MOS Neuristor Line," *Proceedings of the IEEE*, Vol. 65, No. 11, November 1976, pp. 1630-1632.
- [Lewis 64] E. R. Lewis, "An Electronic Model of the Neuron Based on the Dynamics of Potassium and Sodium Ion Fluxes," in *Neural Theory and Modeling*, R. F. Reiss, Ed., Stanford University Press, Stanford, CA, 1964, pp. 154-189.
- [MacGregor 77] R. J. MacGregor and E. R. Lewis, "Neural Modeling," Plenum Press, New York, 1977.
- [Martin 61] T. B. Martin, "Analog Signal Processing by Neural Networks," *Proceedings of the National Electronics Conference*, Chicago, IL Vol. 17, October 1961, pp. 317-321.
- [McGrogan 61] E. P. McGrogan, "Improved Transistor Neuron Models," *Proceedings of the National Electronics Conference*, Chicago, Vol. 17, October 1961, pp. 302-310.
- [Millman 79] J. Millman, "Microelectronics," McGraw-Hill book Co., New York, 1979.
- [Nagumo 62] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An Active Pulse Transmission Line Simulating Nerve Axon," *Proceedings of the IRE*, Vol. 50, No. 10, October, 1962, pp. 2061-2070
- [Newcomb 79] R. W. Newcomb, "MOS Neuristor Lines," in "Constructive Approaches to Mathematical Models," Coffman & Fix, Editors, Academic Press, 1979, pp. 87-111.
- [Pellionisz 79] A. Pellionisz, "Modeling of Neurons and Neuronal Networks," in *The Neurosciences, Fourth Study Program*, edited by F. O. Schmitt and F. G. Worden, The MIT Press, Cambridge, MA, 1979, pp. 525 - 546.

- [Putzrath 61] F. L. Putzrath and T. M. Martin, "Speech Recognition by Neural Networks," Proceedings of the National Electronics Conference, Chicago, IL Vol. 17, October 1961, pp. 311-316.
- [Schmitt 37a] O. H. Schmitt, "Mechanical Solution of the Equations of Nerve Impulse Propagation," abstract of a demonstration in the Proceedings of the American Physiological Society Forty-Ninth Annual Meeting, Memphis, TN, April, 1937, *The American Journal of Physiology*, Vol. 119, No. 2, June 1, 1937, pp. 399-400.
- [Schmitt 37b] O. H. Schmitt, "An Electrical Theory of Nerve Impulse Propagation," abstract in the Proceedings of the American Physiological Society Forty-Ninth Annual Meeting, Memphis TN, April, 1937, *The American Journal of Physiology*, Vol. 119, No. 2, June 1, 1937, p. 399.
- [Schmitt 38] O. H. Schmitt, "A Thermionic Trigger," *Journal of Scientific Instruments*, Vol 15, No. 1, January, 1938, pp. 24-26.
- [Wilamowski 75] B. M. Wilamowski, C. Czarnul, and M. B. Bialko, "Novel Inductorless neuristor Line," *Electronics Letters*, Vol. 11, No. 15, July 24, 1975, pp. 355-356.

PULSE TECHNIQUES IN NEURAL VLSI: A REVIEW

Alan F. Murray

*Department of Electrical Engineering
University of Edinburgh, Edinburgh EH9 3JL, UK*

INTRODUCTION TO PULSE CODING

The coding of signals as the characteristics of pulse trains, or of individual pulses is not a new idea. Many areas of the nervous system are known to perform processing via electrochemical pulsed processes. Furthermore, pulse-modulation schemes are widespread in electronic signal transmission - any textbook on electronics will provide a review of these techniques (see, for example [Horowitz 89]). for some readable background.

The use of pulses in neural VLSI, which is perhaps an obvious application area for the technique, is of more recent origin. Since the advent of the thermionic valve (tube) circuit models of "neurons" and "synapses" have been built, in an attempt to better understand the functionality of these neural building blocks. In 1987 we reported the first serious attempts to build a connected, pulse-firing VLSI network [Murray 87ab]. This was, to be sure, a crude and inelegant piece of design, but it did demonstrate that it was possible to use pulses not only as a communications strategy, but as a computational medium, in the neural context. Since then, more elegant designs have emerged from this author's and from many other research groups, examples of which can be found in this book. The technique may almost be said to have "entered the mainstream". No longer is it simply a perverse and quirky technique, adopted by academics for esoteric reasons. Chips based on pulse-coding are boasting specifications to rival those of better-established technologies (purely digital, and purely analogue, for example), and there are several advantages in pulse-coding which will become clear throughout this book.

Not all of these advantages have yet been fully exploited, and we believe that pulse-coding has much yet to offer, as might be expected from an emerging technique.

However, it is not the purpose of this chapter, this book, or any of the authors therein to claim that pulse-coding is **the best** way forward for neural integration. Some of the problems with pulse coding will be outlined in this and other chapters (after the advantages have been highlighted, of course).

In this chapter I will attempt to provide a grounding in pulse techniques and technologies, followed by a description of the "*Pulse-Stream*" work done by the Edinburgh neural VLSI group. The EPSILON chip represents the culmination of the first phase of that work, and will be described, warts and all.

In a final section, this chapter describes future directions in pulsed neural VLSI - in the author's group in particular.

Motivations

There is an attitude "spectrum" within the neural pulse fraternity, which is well-represented by the chapters in this book. At the "biological" end, Lazzaro (Chapter 8) builds silicon models of the spiking behaviour of the axon hillock. Lazzaro works very much within the Mead philosophy - building analog VLSI models of early neurological processing functions such as hearing and vision. His pulse-firing, or "spiking" circuits form building blocks in a silicon model of the auditory canal, in which the silicon "hair cells" cause spiking patterns whose temporal and spatial properties are important. In a similar vein, but with different aims, John Elias (Chapter 2) builds biologically-inspired silicon models of dendritic trees. These form delay lines, where pulses may be injected at different times and places. This is not a straightforward attempt to model biological forms, however, as the artificial structures may be viewed as generic programmable processors, with no conventional "weights", or learning algorithm. They are trainable by genetic algorithms, to perform a variety of functions - such as distinguishing between symmetric and asymmetric images. If the left hand side of the image is used to provide pulse inputs to the upper half of a symmetric dendritic tree, and the right hand side feeds the lower half, then the integrated output from the tree will have a different temporal shape for symmetric and asymmetric images. It is possible, therefore to conceive of a generic dendritic tree, where the **architecture** of the tree is adapted, rather than individual weights, to allow a common underlying structure to perform a variety of functions. The combination of a biologically-inspired structure and computational medium (pulses) with a genetic algorithm is seductive, although the neurobiological connection must always be welcomed cautiously.

At perhaps the opposite end of the spectrum alluded to above is the work of Tomberg and Kaski (Chapter 9). In their encoding scheme, "pulses" are synchronous pseudo-random bit streams, where the relative density of logical 1's and 0s in a time-window represents a (quantised) analog value. Such stochastic coding methods are in themselves not new [Mars 81] but have not been used in the neural context by other groups as yet. While the encoding scheme requires weights and states to be stored as digital pseudorandom "words", the computational elements reduce to simple logic gates. Thus a trade-off is made between an increase in weight/state storage area, and a reduction in the complexity (and thus area) of the arithmetic elements.

It can be seen, therefore, that pulsed methods are versatile. They can be more or less "analog", according to the application and the prejudices of the designer. What all pulsed methods have in common, however, is a more overt use of the time axis as a coding medium than in conventional analog and digital systems. It is this use of the time axis that brings about many of the advantages of pulsed techniques, and the next section begins with a discussion of the ways in which it may be approached.

Methods

A time-varying analog state signal S_i can be encoded in each of the following ways:-

PAM : Pulse Amplitude Modulation:

PWM : Pulse Width Modulation:

PFM : Pulse Frequency Modulation (average repetition rate of pulses)

PPM : Pulse Phase Modulation (delay between two pulses on different lines)

In addition, further variants exist - **PCM** : Pulse Code Modulation (weighted bits) and **PDM** : Pulse Delay Modulation (delay between a pulse pair on the same line).

PWM, PFM, PPM (and PDM) encode information **in the time domain**, and are here seen as variants of pulse rate. In other words, PDM and PPM are essentially equivalent to PFM with no averaging over several pulse intervals. Such non-averaged pulse modulation techniques are used in various natural systems, particularly where spatial or temporal information is derived from the time of arrival of individual pulses or pulse pairs.

Pulse Amplitude Modulation (PAM). Here, the signal amplitude A_i ($V_i = A_i \times \text{constant frequency pulsed signal}$) is modulated in time, reflecting the variation in S_i . This technique, useful when signals are to be multiplexed on to a single line, and can be interleaved, is not particularly satisfactory in neural nets. It incurs the disadvantages in robustness and susceptibility to processing variations inherent in a purely analog system, as information is transmitted as analog (pulsed) voltage levels.

Pulse Width Modulation (PWM). This technique is similarly straightforward, representing the instantaneous value of the state S_i as the width of individual digital pulses in V_i . The advantages of a hybrid scheme now become apparent, as no analog voltage is present in the signal, with information coded as described along the time axis. This signal is therefore robust, and furthermore can be decoded to an analog value by integration. The constant frequency of signalling means that either the leading or trailing edges of neural state signals all occur simultaneously. In massively parallel neural VLSI, this synchronism represents a drawback, as current will be drawn on the supply lines by all the neurons (and synapses) simultaneously, with no averaging effect. Power supply lines must therefore be oversized to cope with the high instantaneous currents involved.

Pulse Frequency Modulation (PFM). Here, the instantaneous value of the state S_i is represented as the instantaneous frequency of digital pulses in V_i whose widths are equal. Again, the hybrid scheme shows its value, for the same reasons as described above for PWM. The variable signalling frequency skews both the leading and trailing edges of neural state signals, and avoids the massive transient demand on supply lines. The power requirement is therefore averaged in time.

Phase and Delay Modulation. In this final example, two signals are used to represent each neural state, and the instantaneous value of S_i is represented as the *phase difference* between the two waveforms - in other words by modulating the delay between the occurrence of two pulses on one or two wires. In biological neural systems, decisions are often made on a timescale comparable with the inter-pulse time, thus implying that a time-domain, rather than frequency-domain process is at work [Simmon 89]. While this does not of itself provide justification for any technique, it is an interesting parallel. This technique enjoys many of the advantages of the PWM and PCM variants described above, but it does imply the use of two wires for signalling, unless one of the signals is a globally-distributed reference pulse waveform. If this choice is made, however, signal skew becomes a problem, distorting the phase information across a large device, and between devices.

In summary, pulsed techniques can code information across several pulses or within a single pulse. The former enjoys an advantage in terms of accuracy, while the second sacrifices accuracy for increased bandwidth.

Issues - Noise, Robustness, Accuracy and Speed

The four techniques described above share some attributes, but differ in several respects from each other and in most respects from other techniques. All the techniques except PAM are only really susceptible to FM (or edge-

jitter) noise, which will be less significant in a conventionally noisy environment. For instance, a small degradation in the voltage representing a logical 1 will only have a small effect on a system using time as the information coding axis for analog state, and such signals can survive transfer between devices in a multi-chip network much better than analog (or PAM) signals. Furthermore, signals which are purely digital (albeit asynchronous) are easily regenerated or "firmed up" by a digital buffer, which restores solid logic levels without altering pulse widths or frequencies.

Accuracy is, as in all analog systems, limited by noise, rather than by the choice of word length, as it is in a digital system. We have been designing systems with 1% accuracy in synapse weights, for instance, but there is no *a priori* reason why higher or lower accuracy cannot be achieved, at the expense of more or less area. In our own pulsed systems, a feedback scheme part-compensates for circuit inaccuracies (e.g. spread of transistor threshold). Without such a scheme, networks rely on the precision of weights programmed into the silicon memories, or must perform an elaborate pre-processing of the weights before applying them to the silicon.

Speed of operation is perhaps the most difficult and contentious "figure of merit" to quantify in all essentially analog, asynchronous systems. We can clarify the issues involved by looking at a PFM system. The most crucial parameter is the minimum pulse width. This defines the minimum time in which anything at all can happen. The maximum pulse frequency is then chosen to give the desired dynamic range in the multiplication process. This then defines the overall speed at which "calculations" are made. The final factor is the number of pulses over which averaging is performed, to arrive at an individual product $T_{ij}S_j$, and thus at the accumulated activity $\sum T_{ij}S_j$. Clearly, the technique used for multiplication, the dynamic range of the weights, and the level of activity in the network can all affect a speed calculation. We present below the results only of an attempt to estimate the speed of computation in one of the networks developed in Edinburgh, where we have been working to date on systems which require 100 pulses to drive a silicon neuron fully "on" or "off". We do not present this as a serious figure of merit for our networks, but only as a guide to the sort of speed that can be achieved, basing the calculation around a modest process speed. With a pulse frequency conservatively set at 0.5MHz, a two-layer multilayer perceptron with an average level of activity, and a typical distribution of synaptic weights will settle in around 3ms, **regardless of the number of synapses on the chip**. The average computational speed may therefore be estimated as $\frac{N_s}{3 \times 10^{-3}}$ operations/sec, where N_s is the number of synapses. For a typical 10,000 synapse network, this equates to over 3×10^6 operations (multiply/adds) per

second - a fearsome number for a self-contained single chip! We treat this sort of calculation with the greatest scepticism, however, masking as it does the data dependence of the result, and the asynchronoussness it implies. This is purely an averaged result, and individual calculations may take longer or shorter times.

What the speed calculation above does imply, however, is that pulse stream VLSI neural systems could perform many forms of computation (such as speech recognition) *in real time*. In truth, the speed bottleneck is likely to be shifted from pure computational speed to its normal location in analog systems - inter-chip communication. To this end, we are developing smart inter-communication schemes that address this new limitation directly.

"PULSE-STREAM" NEURAL VLSI

In this section I look, rather selfishly, at the history of pulsed methods within my own group. We have now a considerable body of experience in the use of pulsed techniques, and I feel we have generated both elegant and inelegant designs. For this reason, a brief resume should be of interest, offering as it does the Edinburgh Pulse Experience - warts and all.

History

We first struck upon the idea of pulse coding in the neural context in 1985, and reported our first working circuits in 1987 [Murray 87a]. At that stage, our reasons for adopting a pulsed method were, in retrospect, odd. We had already experimented with digital, bit-serial neural VLSI [Murray 87c] and identified that the analog option was more interesting for reasons of density and speed. We did not (and do not), however, have access to an "analog" process - one incorporating controlled resistor and capacitor structures, and providing well-specified MOSFETs. We also observed:-

- a) That biological neural networks use pulses.
- b) That oscillators were easy to create in CMOS.

Circuits

This section describes the different circuit forms and approaches that we have used to date, culminating in the microscopically analog, macroscopically digital form used for the EPSILON chip, which ends this chapter.

Almost Digital. Fig. 1 shows the form of a neuron in this system. Excitatory and inhibitory pulses are signalled on separate lines, and used to dump or remove charge packets from an activity capacitor, with excitatory pulses necessarily inverted. The resultant slowly varying analog voltage X_i is used to control a Voltage Controlled Oscillator (VCO), designed to output short pulses rather than a simple square wave.

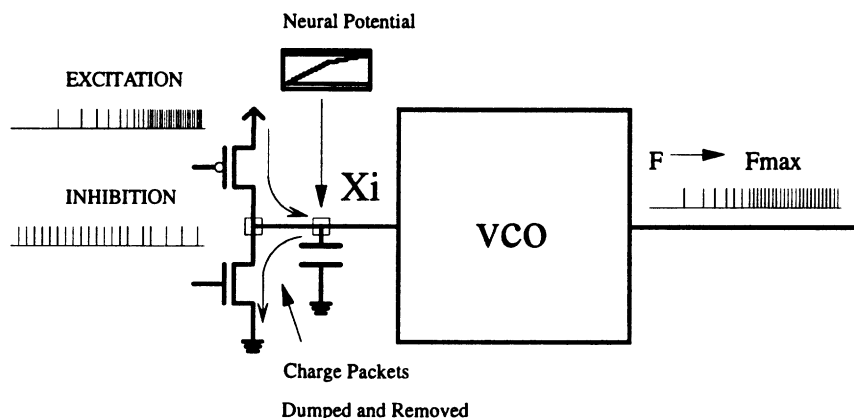


Figure 1. Early Pulse-Stream Neuron, with an inelegant "hard-limit" threshold characteristic.

Synaptic gating is achieved by dividing time artificially into periods representing $\frac{1}{2}$, $\frac{1}{4}$,..... of the time, by means of "chopping clocks", synchronous to one another, but asynchronous to neural activities. In other words, clocks are introduced with mark-space ratios of 1:1, 1:2, 1:4, etc., to define time intervals during which pulses may be passed or blocked. These chopping clocks therefore represent binary weighted bursts of pulses. They are then "enabled" by the appropriate bits of the synapse weights stored in digital RAM local to the synapse, to gate the appropriate *proportion* (i.e. $\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{4}$,.....) of pulses S_j to either the excitatory or inhibitory accumulator column, as shown in Fig. 2. Multiplication takes place when the presynaptic pulse stream S_j is logically ANDed with each of the chopping clocks enabled by the bits of T_{ij} , and the resultant pulse bursts (which will not overlap one another for a single synapse) Ored together. The result is an irregular succession of aggregated pulses at the foot of each column, in precisely the form required to control the neuron circuit of Fig. 1.

A small ≈ 10 - neuron network was been built around the $3\mu\text{m}$ CMOS synapse chip developed using this technique [Murray 87ab, 88b, Smith 88]. The small network, while not of intrinsic value, owing to its size, served to prove that the pulse stream technique was viable, and could be used to implement networks that behaved similarly to their simulated counterparts.

Digital Circuits for Analog Purposes. The "almost digital" system described above proved the viability of the pulse stream technique. However, the area occupied by the digital weight storage memory is unacceptably large. Furthermore, the use of pseudo-clocks in an analog circuit is both aesthetically unsatisfactory and detrimental to smooth dynamical behaviour, and

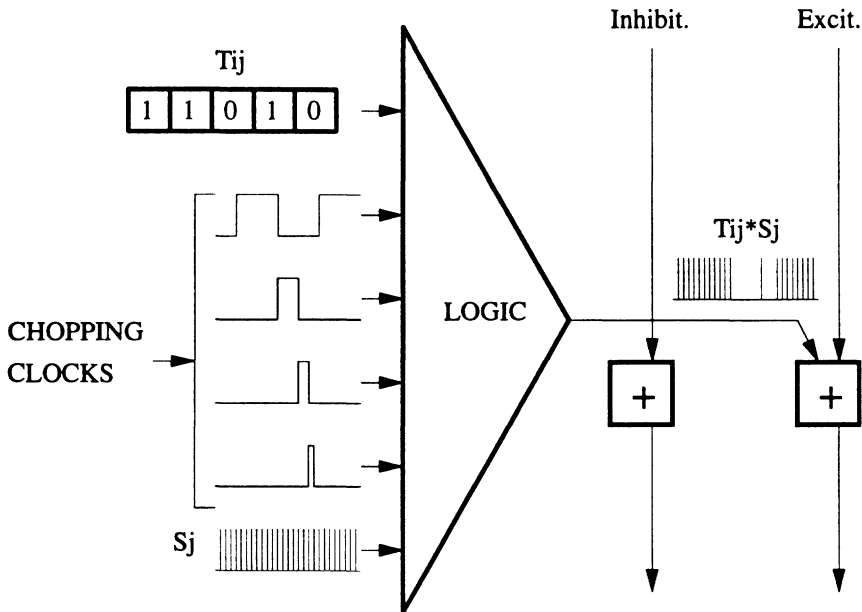
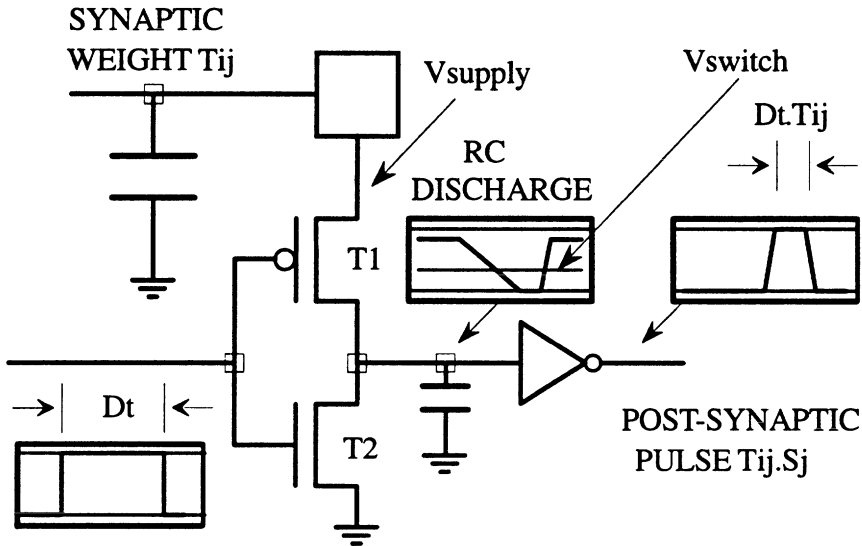


Figure 2. Early, near-digital pulse stream synapse, using synchronous "chopping clocks" to implement multiplication by digital weights.

using separate signal paths for excitation and inhibition is both clumsy and inefficient. Accordingly, we developed a fully programmable, fully analog synapse using dynamic weight storage, and operating on individual pulses to perform arithmetic. Further, the "dump and remove" transistors in the neuron of Fig. 1 have been distributed amongst the synapses, as has the activity capacitor, reducing the neuron to a straightforward VCO [Murray 88bc, 89].

Fig. 3 shows the fundamentals of the synapse circuit, with some details concealed at this stage, to aid explanation. The synapse weight T_{ij} is stored as a voltage on a capacitor. The viable storage time is determined by the size of the capacitor, the temperature at the chip surface, and the leakage characteristic of the CMOS process used. At lowered temperatures, dynamic storage of the voltage representing T_{ij} for a number of seconds, with leakage below 1% of the correct value would be possible. At room temperature, refresh of dynamically stored values is necessary. This dynamically stored voltage controls the positive supply V_{supply} to a two-transistor CMOS inverter T1/T2. Increasing T_{ij} lowers this supply voltage.

Presynaptic input pulses $\{ S_j \}$ are presented asynchronously at the input to this inverter, with a constant width Dt , and a frequency determined by the state of neuron k as described earlier in this section. The inverter T1/T2 is ratioed (i.e. transistor widths and lengths are chosen) such that the inverter's



INPUT PULSE S_j

Figure 3. Time-modulation (pulse-width) synapse - schematic.

ability to *discharge* its output node is weaker than its charging ability. The output of the inverter, on receiving an input pulse S_j is therefore as shown, discharging from an initial value of V_{supply} to 0, at a rate determined by the effective "on" resistance of T2. The discharge is almost exactly linear, as the transistor T2 is operating in saturated mode, and is therefore equivalent to a constant current sink for almost all of the discharge. At the end of an input pulse, a rapid charge back to V_{supply} occurs via T1.

The second inverter restores the sense of the pulse, and also performs a thresholding operation via its *switching threshold* V_{switch} , the voltage at which the inverter switches between output high and low. Putting the two inverters together, the length of the output pulse is determined by *how long the discharge node spends below the switching threshold* of the second inverter. This is determined by the first inverter's supply voltage V_{supply} , which is in turn proportional to T_{ij} .

The net effect is that pulses appear at the postsynaptic node with a *frequency* given by the firing rate of neuron k , and a *width* equal to a maximum possible value Dt , multiplied by a factor $0 \leq T_{ij} \leq 1$. The multiplication is only linear, however, over a restricted range of T_{ij} . We have determined the range over which the product $T_{ij}S_j$ is linear by (SPICE) simulation [Murray 88b] and Fig. 4 shows typical results with $T_{ij} \approx 0$ and ≈ 1 . Clearly, there is potential noise problem when $T_{ij} \approx 1$ ($V_{supply} \approx V_{switch}$) in that a small

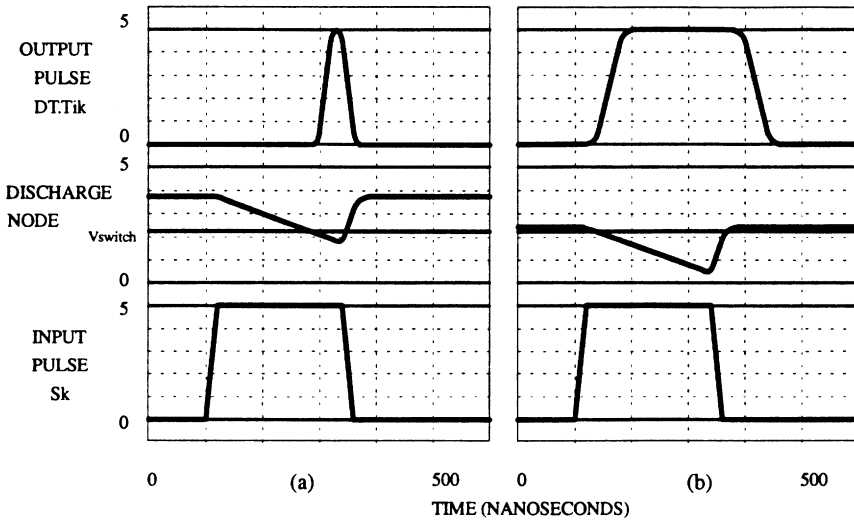


Figure 4. Operation of time-modulation synapse - schematic.

negative noise impulse on the discharge node or a positive disturbance to T_{ij} itself will cause a spurious output pulse. This can be avoided by leaving some headroom in the dynamic range of T_{ij} and by restricting the switching speed of the second inverter, to filter the pulse, and thus remove undesirably high frequency effects.

With this caveat, the useful dynamic range is shown in Fig. 5 to be $1V \leq T_{ij} \leq 3V$. In comparison with networks based on subthreshold operation, this is a wide range. It is therefore possible to "split" the range, to allow the upper half ($2V \leq T_{ij} \leq 3V$) to represent *excitation* and the lower half ($1V \leq T_{ij} \leq 2V$) to represent *inhibition*. Fig. 6 shows the entire analog synapse, which uses this technique to implement both excitation and inhibition in one circuit.

The circuit shown in Fig. 3 can be seen as T1/T2 and the first inverter following. Transistor T5 merely allows an additional control to be applied electronically to the discharge rate. The dynamic synapse voltage is buffered from the inverter T1/T2 by a simple active load amplifier circuit T3/T4. Transistors T6/T7 are distributed versions of the "front end" of the neuron circuit shown in Fig. 1, with an inverter on the input of T6 to get the sense of the gate signal correct. The method of implementing inhibition requires more explanation.

If we arrange that the transistors T6/T7 are always either fully open circuit or saturated, then they are a switched current source and sink respectively, whose associated currents are controlled by the transistor widths and

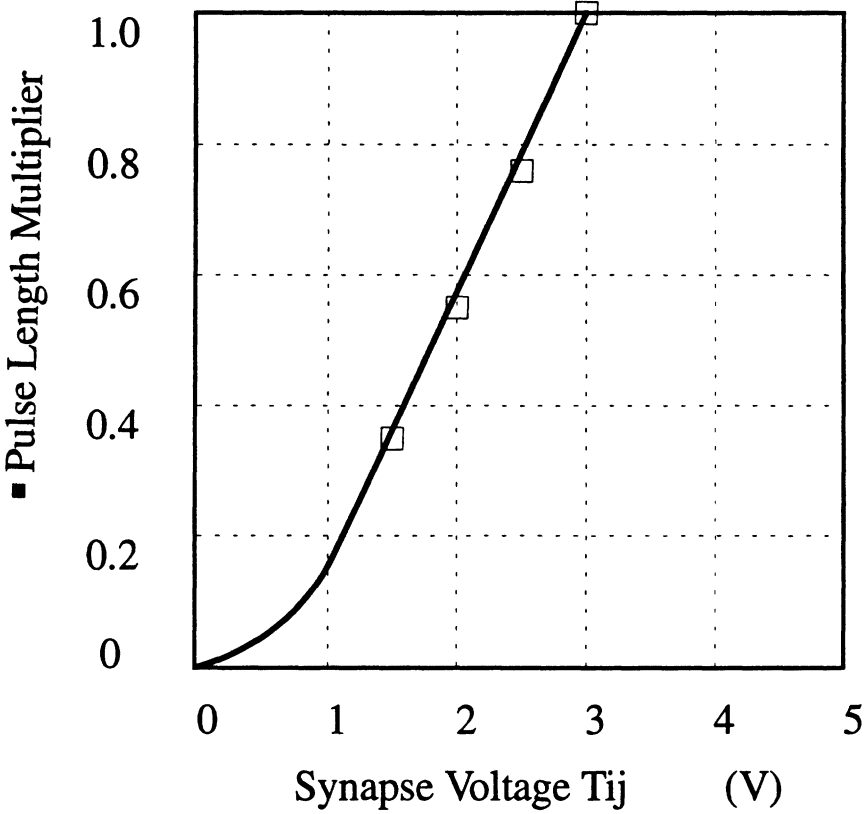


Figure 5. Range of linear operation of time-modulation synapse.

lengths W_6 , W_7 , L_6 , and L_7 .

Therefore, a pulse on the gate of T6 dumps a packet of charge of value

$$Q_{dumped}(T_{ij}) = \int I_6 dt = I_6 Dt \times T_{ij} \text{ coulombs}, \quad (1)$$

while a pulse on the gate of T7 removes

$$Q_{removed} = \int I_7 dt = I_7 Dt \text{ coulombs}. \quad (2)$$

The net charge added to the activity x_i is therefore

$$Q_{total}(T_{ij}) = Q_{dumped}(T_{ij}) - Q_{removed}. \quad (3)$$

If we choose values of W_6 , L_6 , W_7 and L_7 such that $Q_{dumped}(2V) = Q_{removed}$, then $Q_{total}(2V) = 0$. This effectively splits the range of T_{ij} such that a value $T_{ij} > 2V$ will result in an increase in x_i proportional to $(T_{ij} - 2V)$ and a value $T_{ij} < 2V$ will result in a decrease in x_i proportional to $(2V - T_{ij})$.

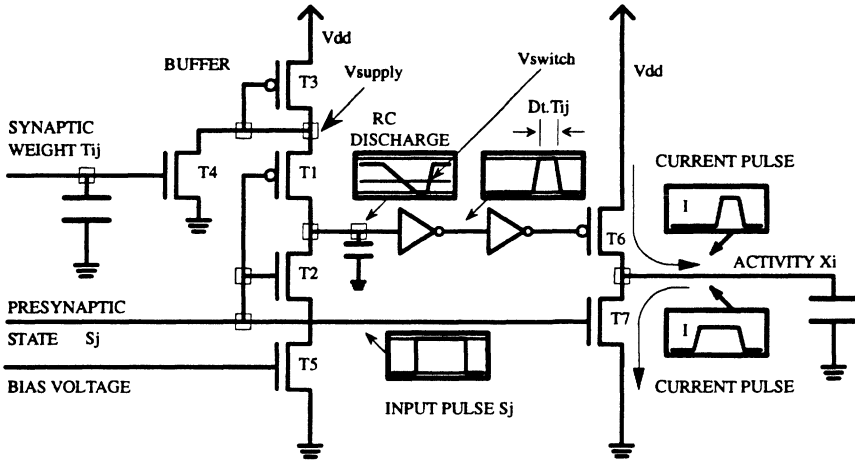


Figure 6. Time-modulation (pulse-width) synapse - details.

This is exactly what we set out to do. A column of these synapses, with the associated distributed capacitors on the drain connections of T6 and T7 will aggregate the total activity from all neurons connecting to neuron i to represent x_i by a slowly varying voltage. The rise in the voltage representing x_i caused by a single pulse passing through synapse T_{ij} is:-

$$\Delta V(x_i) = \frac{Q_{total}(T_{ij})}{C_{total}(x_i)} \quad (4)$$

As the number of synapses in the column is increased, the capacitance $C_{total}(x_i)$ in the denominator of (6) increases proportionately, and therefore the individual contributions to $\Delta V(x_i)$ become less significant, as we would wish. Naturally, as more synapses are added, more terms of the form $Q_{total}(T_{ij})$ are added, and the synapse is therefore 100% cascable, both topologically and electrically.

To ensure that transistors T6 and T7 remain in saturation, two additional devices, T8 and T9, are incorporated, as shown in Fig. 7. This incurs little penalty in silicon area, as they have the additional effect of reducing the need for the lengths of T6 and T7 to be large to restrict their source-drain currents. These devices act as voltage attenuators, as for instance, the gate voltage on T7 cannot be driven above $V_{drain}(9) - V_{in}$. The effectiveness of the attenuation process is increased by the *body effect*, whereby a MOS device has its threshold raised as its source rises above the substrate potential. The attenuated pulses on the gates of T6 and T7 ensure that these transistors operate in the subthreshold region, and are therefore essentially always saturated, and always operate as current source and sink respectively.

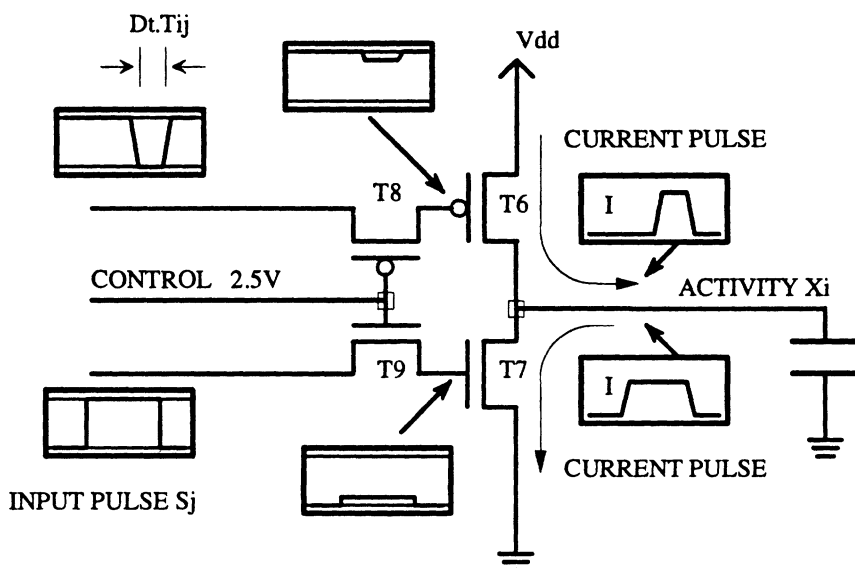


Figure 7. Modified Time-modulation (pulse-width) synapse - improved output characteristic.

The test chip based on this technique proved functional, and was able to implement small perceptron networks. However, the effects of process variation across the silicon surface - variations in the threshold in different manifestations of transistor T2, for instance, created wide variations in synapse characteristics. This problem, coupled with the still-high transistor count (13 MOSFETs, for the full synapse) drove a move towards a more analog solution to the multiplication problem.

Fully Analog. The initial motivation behind the unusual pulse stream form was the analogy with biological neurons, coupled with a desire to use an essentially digital CMOS process for asynchronous analog circuits. These reasons remain, but have been augmented by the discovery that some arithmetic operations such as multiplication can be implemented very efficiently using pulse streams. Furthermore, when states are represented by $0 \rightarrow 5V$ pulses, the neural state can be used to *switch* analog circuits in and out of a system. When analog voltages are used to represent neural states, it is much more difficult to keep all the MOSFETs used to perform arithmetic operations in known operating regimes.

The technique described in this section uses the MOSFET transistor characteristic equations to produce a **current** that is proportional to the product of two **voltages**. Current is subsequently summed by an integrator circuit. The equation is that for the drain-source current, I_{DS} , for a MOSFET in

the *linear* or *triode* region:-

$$I_{DS} = \frac{\mu C_{ox} W}{L} \left[(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right] \quad (5)$$

Here, C_{ox} is the oxide capacitance/area, μ the carrier mobility, W the transistor gate width, L the transistor gate length, and V_{GS} , V_T , V_{DS} the transistor gate-source, threshold and drain-source voltages respectively.

This expression for I_{DS} contains a useful product term:-

$$\frac{\mu C_{ox} W}{L} \times V_{GS} \times V_{DS}$$

However, it also contains two other terms in $V_{DS} \times V_T$ and V_{DS}^2 .

One approach might be to ignore this imperfection in the multiplication, in the hope that the natural robustness of neural systems renders it irrelevant. We have chosen, rather, to **remove** the unwanted terms via a second MOS-FET, as shown in Fig. 8.

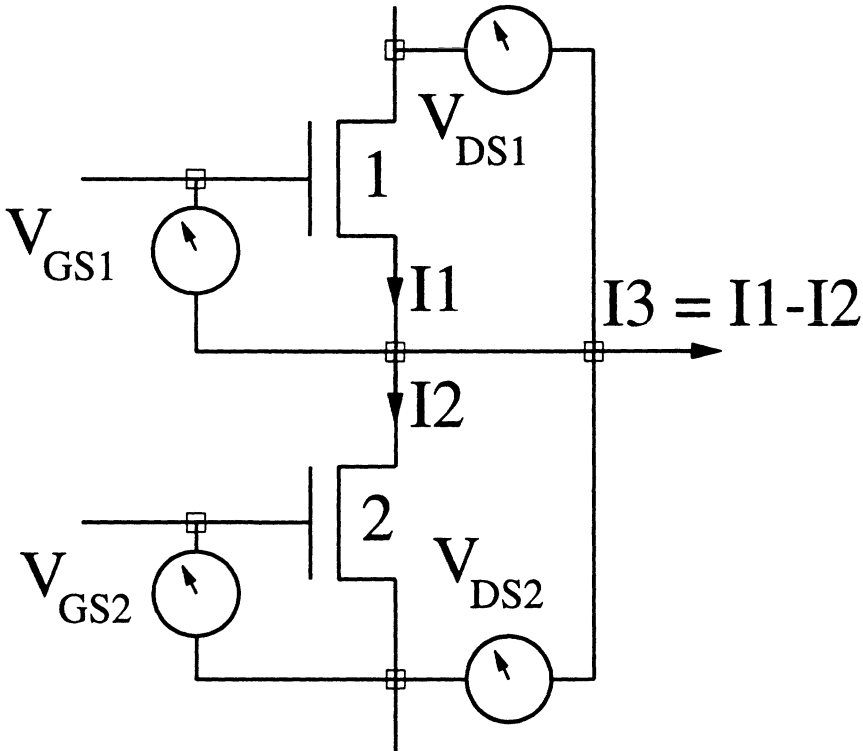


Figure 8. Two-transistor (transconductance) multiplier.

The output current I_3 is now given by:-

$$I_3 = \mu C_{ox} \left[\frac{W_1}{L_1} (V_{GS1} - V_T) V_{DS1} - \frac{W_1}{L_1} \frac{V_{DS1}^2}{2} - \frac{W_2}{L_2} (V_{GS2} - V_T) V_{DS2} + \frac{W_2}{L_2} \frac{V_{DS2}^2}{2} \right] \quad (6)$$

The secret now is to select $W_1, L_1, W_2, L_2, V_{GS1}, V_{GS2}, V_{DS1}$ and V_{DS2} to cancel all terms except

$$\mu C_{ox} \frac{W_1}{L_1} V_{GS1} \times V_{DS1} \quad (7)$$

This is a fairly well-known circuit, called a **Transconductance Multiplier**. It was reported initially for use in signal processing chips such as filters [Denyer 81] and later in [Han 84] It can be used directly in a continuous time network, with analog voltages representing the $\{S_i\}$. We choose to use it within a pulse-stream environment, to minimise the uncertainty in determining the operating regime, and terminal voltages, of the MOSFETs, as described above.

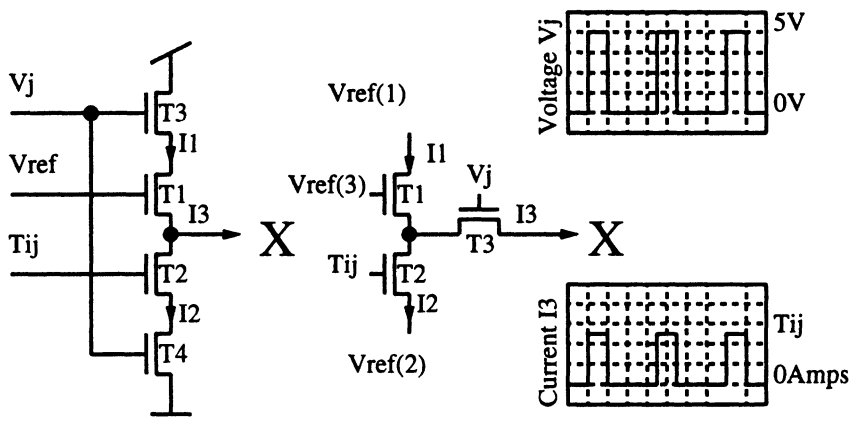


Figure 9. Two versions of the pulsed transconductance multiplier.

Fig. 9 shows two related pulse stream synapses based on this technique. The presynaptic neural state S_j is represented by a stream of 0-5V digital, asynchronous voltage pulses V_j . These are used to switch a current sink and source in and out of the synapse, either pouring current to a fixed voltage node X (excitation of the postsynaptic neuron), or removing it (inhibition). The magnitude and direction of the resultant current pulses are determined by the synapse weight, currently stored as a dynamic, analog voltage T_{ij} .

The fixed voltage at point X and the summation of the current pulses to give an activity $x_i = \sum T_{ij} S_j$ are both provided by an operational amplifier integrator circuit, whose saturation characteristics incidentally apply a sigmoid nonlinearity. The transistors T3 and T4 act as power supply "on/off" switches in Fig. 9a, and in Fig. 9b are replaced by a single transistor, in the output "leg" of the synapse. Transistors T1 and T2 form the transconductance multiplier. One of the transistors has the synapse voltage T_{ij} on its gate, the other a reference voltage, whose value determines the crossover point between excitation and inhibition. The gate-source voltages on T1 and T2 need to be substantially greater than the drain-source voltages, to maintain linear operation. This is not a difficult constraint to satisfy.

The attractions of these cells are that all the transistors are n-type, removing the need for area-hungry isolation well structures, and in Fig. 9a, the vertical line of drain-source connections is topologically attractive, producing very compact layout, while Fig. 9b has fewer devices. When weight address circuitry is added, the two synapse circuits comprise 6 and 5 transistors respectively.

Implications for the Neuron

While the synapse form described above is attractively compact and topologically simple, it places a heavy requirement on the current integrator. As more and more synapses are cascaded, the integrator must sink or source more and more current to maintain the virtual ground at its input, in order that point X in the circuits of Fig. 9 remain at a fixed reference voltage, and the transconductance multiplier property is maintained. To cope with this requirement, either:-

- a) The operational amplifier that underpins the integrator circuit must have a high enough output current drive capability to cope with the maximum number of synapse circuits it has to serve.
- b) The drive capability of the operational amplifier must be distributed amongst the synapses themselves.

We have chosen the latter approach, effectively distributing the output drive transistors of the operational amplifier between the synapses. As a result, each synapse comprises two extra transistors, at a small cost in area, giving a fully cascadable form. The EPSILON chip, which uses this technique, underpins the remainder of this chapter.

THE EDINBURGH PULSE-STREAM IMPLEMENTATION OF A LEARNING-ORIENTED NETWORK (EPSILON) CHIP

We have designed a large pulse stream demonstrator chip using the transconductance multiplier circuit outlined above.

Circuits

The neuron circuits we use have all been based on voltage- or current-controlled oscillators. The details of the circuitry used [Murray 92ab] are complex, and discussion in the context of this chapter is unwarranted. The synapse circuit, however, illustrates an unusual use of the transconductance multiplier form, and is worthy of special mention here.

The synapse circuit is based on the transconductance circuit described above, but embedded in the system in a manner that is not obvious (Fig. 10). Here, the output drive transistors of the Operational Amplifier that underlies the neural integration process have been distributed through the synaptic column. This effectively devolves the current drive capability associated with the integrator output **to the synaptic sites themselves**. This brings about a number of desirable features:-

- 1) The system is now much more cascadable, as each synapse adds its own current drive ability, and the Op-Amp at the foot of the column only drives transistor gates (purely capacitive).
- 2) Power is now dissipated more evenly across the chip, alleviating problems associated with "hot-spots".
- 3) Perhaps most unexpectedly, and most beneficially, the synapse is now well-matched to its own "drive" stage, as they are physically close together. If, for instance, transistor thresholds are lowered locally, both the synapse and its drive transistors will become "faster". As a result, the effect of the process variation that caused the threshold shift is cancelled (to first order).

The combination here may be described as "analog computation under digital control". The transconductance multiplier is an analog circuit, while the presynaptic neural input signal is a digital pulsed waveform, which effectively switches the analog multiplier in and out of the neural column. Long-range communication (of neural states) is thus performed by robust, pulsed signals, while the advantages of small area and high speed of an analog multiplier are enjoyed at the synapse level. While this combination is not "better" than all other pulsed approaches, as the remaining chapters of this book make abundantly clear, it is a powerful one, which we intend to exploit via EPSILON and extend into other chips with increased functionality [Murray 92c]. and enhanced technology [Reeder 91]

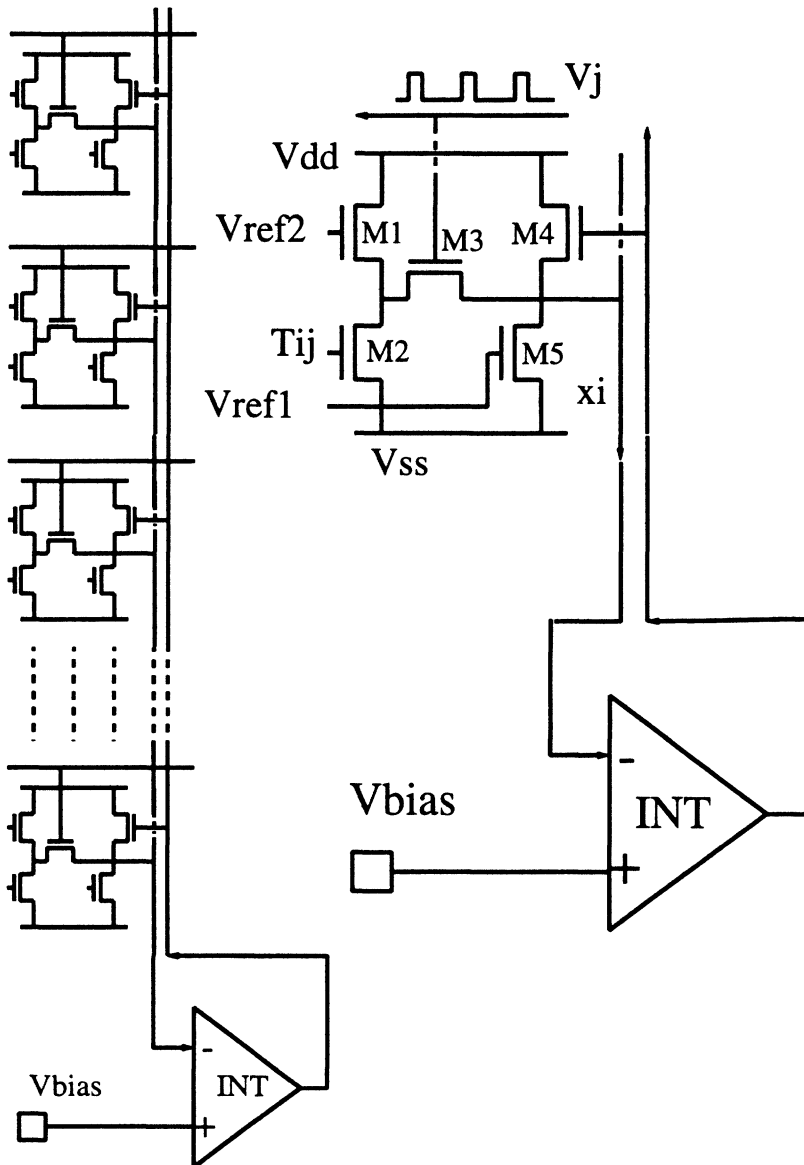


Figure 10. Transconductance multiplier "system". The drive stage of the integrator has now been distributed throughout the synaptic column as MOSFETs M4 and M5.

Specification

EPSILON consists of an array of 120x30 synapses with 30 on chip neurons (see Fig. 11)

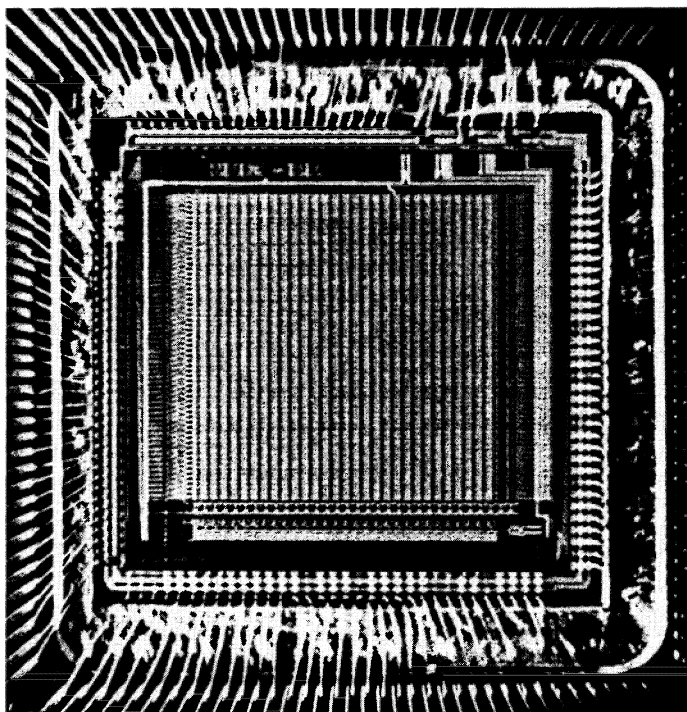


Figure 11. The EPSILON chip - $\approx 1\text{cm} \times 1\text{cm}$.

The chip has been implemented using European Silicon Structures' (ES2) $1.5\mu\text{m}$ CMOS Digital process and occupies an area of 10mm by 9.5mm. The estimated worst case power consumption for this device is 350mW which is well within the safe limits for a chip of this size. All of the subcircuits on the chip have been tested and found to be functionally correct. A printed circuit board allows the chip to operate as a hardware accelerator to both Sun and PC host computers.

As a rough, and hopefully not invidious comparison, Table 1 shows the capabilities of the EPSILON device, alongside those of INTEL's ETANN product [Holler 89].

EPSILON is still a new device, and we only have limited experience of its capabilities as a network. Early experience suggests that it will prove a powerful device where it can be part of the learning cycle - a situation common to all analog implementations, where inclusion in the learning loop is

ETANN vs. EPSILON - COMPARISON	
80170NX (ETANN)	30120PI (EPSILON)
Floating Gate Technology	Standard CMOS Process
64 Variable Gain Neurons	30 Variable Gain Neurons
128 Inputs	120 Inputs
10,240 Synapses	3,600 Synapses
2x(80x64) Arrays	120x30 Array
Inputs: Analog Outputs: Analog	Inputs: Analog, PFM and PWM Outputs: PFM and PWM
3 μ s per layer	10 μ s per layer (PWM)
2B Connections/s/chip	0.36B Connections/s/chip
Non-Volatile Weight Storage > 100 μ s per Weight Update	Capacitive Weight Storage 1 μ s per Weight Update
Multiplication Non-Linear at <i>extrema</i>	Linear Multiplication over Weight Range

essential. Where weights are pre-calculated and "down-loaded" to the device, EPSILON, ETANN and all such devices will cause problems. Some immunity can be gained by noisy off-line training, but it is likely that success will be limited where the chip cannot exhibit its own idiosyncracies during learning. As an example of EPSILON's functionality and performance, Fig. 12 shows the linearity of the synaptic multiplication. Here, the sending neuron state is swept through its full range for a range of different weight voltages - inhibitory and excitatory. It can be seen that the output state varies with impressive linearity, although the characteristic is not perfect.

This points towards some problems that remain with the EPSILON device, and it would be dishonest to conceal them. While the techniques described above have produced synaptic columns that are extremely process-

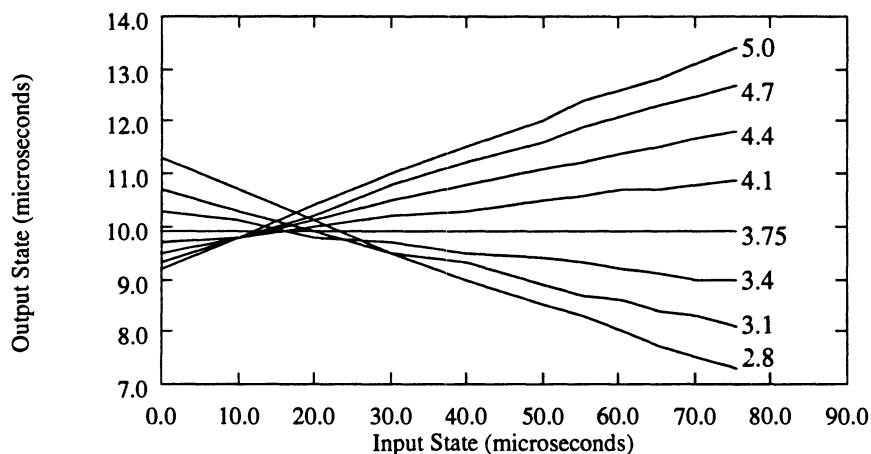


Figure 12. (Transconductance) multiplier characteristic from the EPSILON chip. The sending neuron state is swept through its full range for a range of different weight voltages - inhibitory and excitatory.

tolerant, there are still significant variations between neurons, arising from variations in the integrator characteristics. In all but the "tightest" neural architectures, we can remove these by using a row of synapses (one for each neuron) to effectively cancel out these variations. However, this is a "hack", and we would wish to render it unnecessary, by including auto-bias circuitry, in a revised EPSILON design. The other major flaws in EPSILON are coupling between the pulsed neural outputs and the activity voltages, and a small voltage drop in the power supplies across the device. Both of these can be rendered irrelevant by careful programming, but both should be removed by better layout in EPSILON Mark II.

In summary of EPSILON, therefore - it is an impressive and startling device. It demonstrates beyond any doubt that pulsed techniques can be made to work over large silicon substrates without oscillator coupling and high power consumption - the two concerns often expressed with respect to scaling up pulsed circuits - causing any problems. The device is flawed, but not fatally so. We are looking forward to demonstrating its capabilities in a range of applications, and correcting its minor faults in our subsequent designs.

CONCLUSIONS AND ISSUES

Many issues remain to be addressed. Dynamic weight storage, the technique used for EPSILON, is successful and conceptually simple. It is, however, fraught with the practical problems of refresh and corruption. Existing non-volatile technologies such as MNOS [Sage 89] and Floating-Gate [Holler

89] are expensive, intrusive (they place extra layers within the CMOS process) and inconvenient. We are developing a novel amorphous-silicon (α -Si) device [Reeder 91] that attempts to address these problems by adding a programmable α -Si resistor structure **on top of** the existing CMOS layers. The device is fast and compact, but as this chapter is being written, its yield is atrocious. We have high hopes that these problems will be overcome, but we are realistic about the likely timescales for getting such a novel device "under control".

Even if such a device can be made to yield satisfactorily, it is likely to require on-chip training to optimise the use of what are likely to be idiosyncratic weight memories. We are therefore developing on-chip learning strategies and chips, [Murray 92c] which do not warrant inclusion in detail in this chapter. We have, however, "stumbled upon" some fascinating and, we believe, very important results indicating how we might best make use of inaccurate (in the analog sense) memories. The discovery came about as we were probing the capabilities of the on-chip algorithm, injecting analog noise at the the synapses to simulate a "real" analog environment. The results are startling enough to deserve inclusion, offering as they do a partial solution to the accuracy problem inherent in all analog systems. That they endow a network with an ability to make good use of "imperfect" components, by training with "imperfect" arithmetic is hardly a real cause for surprise - the nervous system has been doing the same for a long time!

Noisy Learning

Accepted dogma hold that high (\approx 16-bit) accuracy is needed during neural learning. This has discouraged attempts to develop analog learning circuitry, although some recent work on hybrid analog/digital systems [Hollis 90] suggests that learning is possible, if perhaps not optimal, in low (digital) precision networks. We found that analog noise, far from impeding neural learning, actually enhances it. While the results relate to a MultiLayer Perceptron (MLP) network, using an algorithm with its roots in back-propagation, they have implications for all learning schemes using gradual weight increment/decrement techniques.

Noise in Learning

The experiments described in this section relate to a series of experiments using a training algorithm devised by the author. [Murray 91, 92c] The details of the algorithm are not important here, except that it is susceptible to imprecision in the same way as is the ubiquitous back-propagation algorithm. It is important to realise here that the noise in question is **not** in the training data. Training with noisy data is a ubiquitous technique, which effectively

expands the training set artificially, and consequently penalises overfitting of decision boundaries. Here, we are injecting noise **at the synapses** - a less invasive procedure with more complex implications. As an example of a "real" classification task, with both training- and test- data sets, the Oxford/Alvey vowel database formed the vehicle for a 54 : 27 : 11 MLP to learn to classify vowel sounds from 18 female and 15 male speakers. The data appear as the analog outputs of 54 band-pass filters, for 11 different vowel sounds, and 33 speakers, with a 1-out-of-11 coding scheme on the output neurons for each of the 11 vowels.

Experiments were conducted in pairs, which were in every respect identical, using the same set of randomised initial weights, *except for the presence of noise*. Learning is enhanced by the presence of noise at a high level (around 20%) on both synaptic weights and activities. This result is surprising, in the light of the normal assertion alluded to above that back-propagation requires up to 16-bit precision during learning. The distinction is that digital inaccuracy, determined by the significance of the Least Significant Bit (LSB), implies that the smallest possible weight change during learning is 1 LSB. Analogue inaccuracy is, however, fundamentally different, in being noise-limited. In principle, infinitesimally small weight changes *can* be made, and the inaccuracy takes the form of a spread of "actual" values of that weight as noise enters the forward pass. The underlying "accurate" weight does, however, maintain its accuracy as a time-average, and the learning process is sufficiently slow to effectively "see through" the noise in an analog system.

The implication is that while analog noise may introduce temporarily inappropriate changes in the $\{ T_{ab} \}$ and $\{ \tilde{o}_{jp} \}$, the underlying trend reflects the accurate synaptic weight values, and makes the appropriate *averaged* adjustments. The further implication is that drawing parallels in this context between digital inaccuracy and analog noise is extremely misleading. The former imposes constraints (quantisation) on allowable weight values, while the latter merely smears a continuum of allowable weight values. The incidental, and highly interesting finding - that higher levels of noise actually assist learning - is not so easily explained, although injection of noise into adaptive filter training algorithms is not unusual. Corrupting the training data with noise is held to have the same effect as penalising high curvature in decision boundaries - in other words it causes the network to draw sweeping curves through the decision space, rather than fitting convoluted curves to the individual training data points [Bishop 90]. In this way, underlying trends are modelled, while fine "irrelevant" detail is ignored. These two findings concerning noise would seem to be perfectly general in the neural context, and have ramifications for all learning processes where weights evolve

incrementally, and slowly. We can explain what is happening in the presence of *arithmetic* noise by examining the mean-squared error on the outputs $\{o_k\}$. The total mean squared error, over all P patterns $\{o_{ip}\}$ is given by:-

$$E(\text{tot}) = \frac{1}{P} \times \sum_{p=1}^{p=P} E_p$$

Where:-

$$E_p = \frac{1}{2} \sum_{k=0}^{k=K-1} E_{kp} = \frac{1}{2} \sum_{k=0}^{k=K-1} (o_{kp}(\{T_{ab}\}) - \tilde{o}_{kp})^2 \quad (8)$$

If we Taylor-expand the output value o_{kp} to second order, around the noise-free weight set, we eventually achieve an error function of the form:-

$$\begin{aligned} \langle E \rangle &= \frac{1}{P} \sum_{p=1}^{p=P} \langle E_p \rangle = \langle E(\{T_{\text{nominal}}\}) \rangle + \\ &\frac{1}{2P} \sum_{p=1}^{p=P} \sum_{k=0}^{k=K-1} \langle \Delta \rangle^2 \sum_{ab} \left[\left(\frac{\partial o_k}{\partial T_{ab}} \right)_p^2 + E_{kp} \left(\frac{\partial^2 o_k}{\partial T_{ab}^2} \right)_p \right] \end{aligned} \quad (9)$$

This has added two terms to the normal error expression of the form:-

$$\langle \Delta^2 \rangle \sum_{ab} \left[\frac{1}{2P} \sum_{p=1}^{p=P} \sum_{k=0}^{k=K-1} \left(\frac{\partial o_k}{\partial T_{ab}} \right)_p^2 \right] \quad (10a)$$

and

$$\langle \Delta^2 \rangle \sum_{ab} \left[\frac{1}{2P} \sum_{p=1}^{p=P} \sum_{k=0}^{k=K-1} E_{kp} \left(\frac{\partial^2 o_k}{\partial T_{ab}^2} \right)_p \right] \quad (10b)$$

In other words, the **error**, which is being notionally minimised by the learning process, is made **greater** by these two terms. The effect of the second term is complex, as it involves second derivatives, and also depends on the errors $\{E_{kp}\}$. The first term, however, can be re-written as

$$K \times \langle \Delta^2 \rangle \left[\left(\frac{\partial o_k}{\partial T_{ab}} \right)_p^2 \right] \quad (11)$$

averaged over all patterns, output nodes, and weights

This is more illuminating. It implies that solutions will be favoured where the dependence of outputs on weights is smeared out as much as possible - where the average of the derivative is minimised. We have performed some simple but exhaustive experiments to verify this finding.

Fig. 13 shows the effect of different levels of noise on a learning cycle with the same initial conditions. The effect on the network's ability to sustain synaptic damage is clear. The resistance to synaptic corruption can be seen to be even greater [Murray 92d].

These results indicate that, as expected, the network's tolerance to weight damage is enhanced by the inclusion of noise in the arithmetic during the learning phase. We have also predicted and demonstrated, by examining both of the above terms, [Murray 92d, 93] that generalisation ability and learning times should be, and are, improved by noisy training. It is not appropriate to deal with these issues here, although they have far-reaching implications for all MLP learning, whether implemented as hardware or software.

In conclusion, the results show that noisy training enhances the quality of a learnt weight set in terms of generalisation, reduces the training time, and perhaps most importantly for VLSI, **distributes the information optimally across the weight set**. We are able to show that learning with the EPSILON chip, where arithmetic precision is limited, is enhanced by noisy training, and we intend to extend the technique to train networks using α -Si weights, where the inaccuracy issue will rear its head once again.

Closing Remarks

This seems a good note on which to end. Pulsed techniques remove some of the concerns (susceptibility to corruption of information, for example) that surround conventional analog VLSI. However, the technique is fundamentally analog, and thus has limited accuracy - limited by the fundamentals of the device physics. We should strive to work **with** the physics rather than against it - in much the way that nature has done.

The remainder of this book will underline the diversity and elegance of design that has emerged in the pulsed field. We have converged upon a range of techniques and issues which are outlined in this chapter and its associated references. Others, with a different perspective and background, have gone a different route. What unites us as a "community" is the simple realisation that pulses offer a combination of robustness, simplicity, circuit surprises and biological parallels that is both satisfying scientifically, and attractive in engineering terms.

ACKNOWLEDGEMENTS

The work reported here has benefitted from the direct input of many co-workers, and the advice of many others. They are, in alphabetical order :- Donald Baxter, Chris Bishop, Mike Brownlow, Zoe Butler, Steve Churcher, Peter Edwards, Alister Hamilton, Peter LeComber, Martin Reekie, Tony

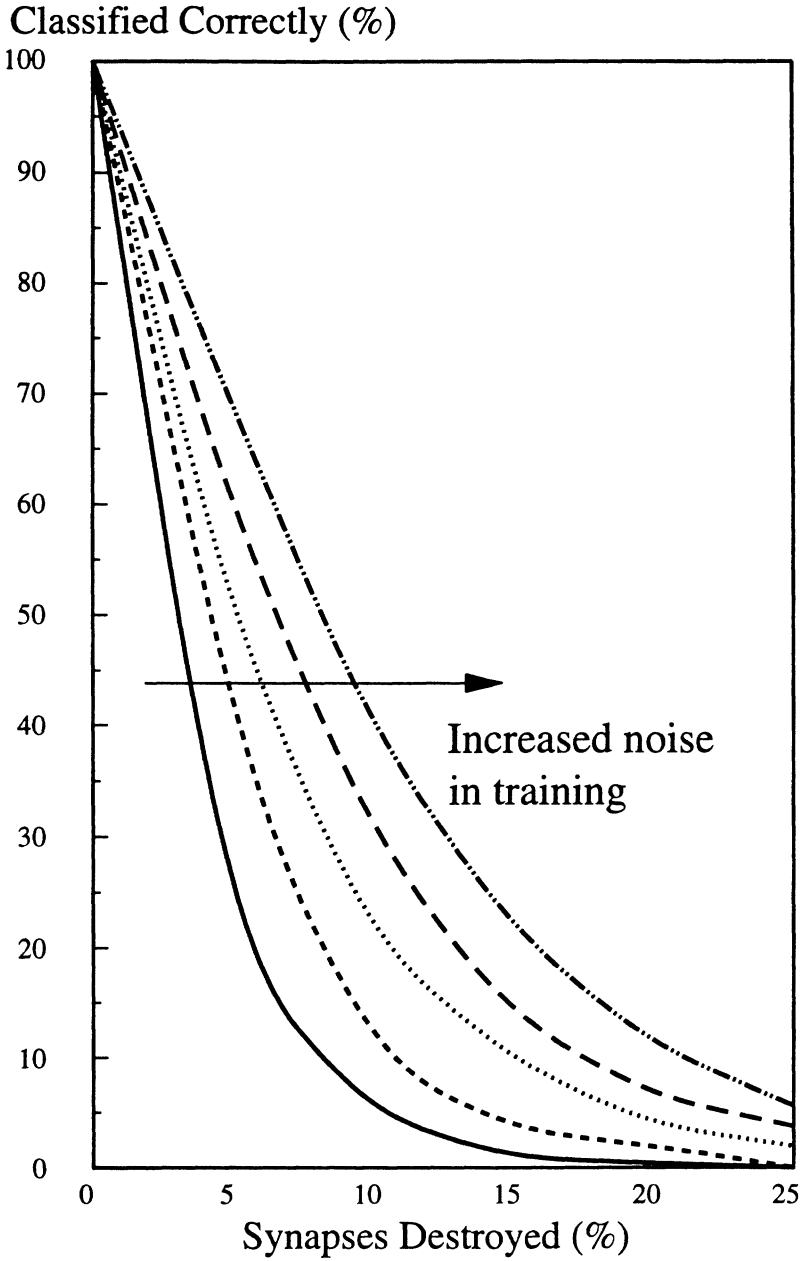


Figure 13. Comparison of Noise-Injected and Noise-Free learning

Smith, Tony Snell, Lionel Tarassenko and Jon Tombs. Financial support has been provided by the Science and Engineering Research Council, the CEC (ESPRIT), British Telecom, British Aerospace, and Thorn-EMI. For all of this, and many other sources of encouragement and wisdom, the author is extremely grateful.

REFERENCES

- [Bishop 90] C. Bishop, "Curvature-Driven Smoothing in Backpropagation Neural Networks," *Proc. Int. Neural Networks Conference*, vol. II, pp. 749-752, 1990.
- [Denyer 81] P.B. Denyer and J. Mavor, "MOST Transconductance Multipliers for Array Applications," *IEE Proc. Pt. 1*, vol. 128, no. 3, pp. 81-86, June 1981.
- [Han 84] Il S. Han and Song B. Park, "Voltage-Controlled Linear Resistors by MOS Transistors and their Application to Active RC Filter MOS Integration," *Proc. IEEE*, pp. 1655-1657, Nov., 1984.
- [Holler 89] M. Holler, S. Tam, H. Castro, and R. Benson, "An Electrically Trainable Artificial Neural Network (ETANN) with 10240 "Floating Gate" Synapses," *International Joint Conference on Neural Networks - IJCNN89*, pp. 191-196, June, 1989.
- [Hollis 90] P.W. Hollis, J.S. Harper, and J.J. Paulos, "The Effects of Precision Constraints in a Back-Propagation Learning Network," *Neural Computation*, vol. 2, pp. 363-373, 1990.
- [Horowitz 89] P. Horowitz and W. Hill, in *The Art of Electronics*, Cambridge University Press, 1989.
- [Mars 81] P. Mars and W.J. Poppelbaum, in *Stochastic and Deterministic Averaging Processors*, Peter Peregrinus Ltd., 1981.
- [Murray 87a] A.F. Murray and A.V.W. Smith, "Asynchronous Arithmetic for VLSI Neural Systems," *Electronics Letters*, vol. 23, no. 12, pp. 642-643, June, 1987.
- [Murray 87b] A.F. Murray and A.V.W. Smith, "A Novel Computational and Signalling Method for VLSI Neural Networks," *European Solid State Circuits Conference*, pp. 19-22, VDE-Verlag, Berlin, 1987.
- [Murray 87c] A.F. Murray, A.V.W. Smith, and Z.F. Butler, "Bit-Serial Neural Networks," *Neural Information Processing Systems (NIPS) Conference*, pp. 573-583, 1987.
- [Murray 88a] A.F. Murray and A.V.W. Smith, "Asynchronous VLSI Neural Networks using Pulse Stream Arithmetic," *IEEE Journal of*

- Solid-State Circuits and Systems*, vol. 23, no. 3, pp. 688-697, 1988.
- [Murray 88b] A.F. Murray, L. Tarassenko, and A. Hamilton, "Programmable Analogue Pulse-Firing Neural Networks," *Neural Information Processing Systems (NIPS) Conference*, pp. 671-677, Morgan Kaufmann, 1988.
- [Murray 88c] A.F. Murray, L. Tarassenko, and A.V.W. Smith, "Fully-Programmable Analogue VLSI Devices for the Implementation of Neural Networks," in *VLSI for Artificial Intelligence*, ed. J.G. Delgado-Frias, W.R. Moore, pp. 236-244, Kluwer, 1988.
- [Murray 89] A.F. Murray, A. Hamilton, H.M. Reekie and L. Tarassenko, "Pulse-Stream Arithmetic in Programmable Neural Networks," *Int. Symposium on Circuits and Systems, Portland OR.*, pp. 1210-1212, 1989.
- [Murray 91] A.F. Murray, "Analog VLSI and Multi-Layer Perceptrons - Accuracy, Noise and On-Chip Learning," *International Conference on Neural Networks (Munich)*, pp. 27-34, 1991.
- [Murray 92a] A.F. Murray, D.J. Baxter, S. Churcher, A. Hamilton, H.M. Reekie, and L. Tarassenko, "The Edinburgh Pulse Stream Implementation of a Learning-Oriented Network (EPSILON) Chip," *Neural Information Processing Systems (NIPS) Conference*, 1992.
- [Murray 92b] A.F. Murray, A. Hamilton, D.J. Baxter, S. Churcher, H.M. Reekie, and L. Tarassenko, "Integrated Pulse-Stream Neural Networks - Results, Issues and Pointers," *IEEE Trans. Neural Networks*, pp. 385-393, 1992.
- [Murray 92c] A.F. Murray, "Multi-Layer Perceptron Learning Optimised for On-chip Implementation - a Noise Robust System," *Neural Computation* vol. 4, no. 3, pp. 366-381, 1992.
- [Murray 92d] A.F. Murray and P.J. Edwards, "Synaptic Weight Noise During MLP Training Enhances Fault Tolerance," *Neural Information Processing Systems (NIPS) Conference*, 1992.
- [Murray 93] A.F. Murray and P.J. Edwards, "Synaptic Weight Noise During MLP Training: Enhanced MLP Performance and Fault Tolerance Resulting from Synaptic Weight Noise During Training," *IEEE Trans. Neural Networks*, 1993. To be published.
- [Reeder 91] A.A. Reeder, I.P. Thomas, C. Smith, J. Wittgreffe, D. Godfrey, J. Hajto, A. Owen, A.J. Snell, A.F. Murray, M. Rose, and P.G. LeComber, "Application of Analogue Amorphous

- Silicon Memory Devices to Resistive Synapses for Neural Networks," *International Conference On Neural Networks (Munich)*, pp. 253-260, 1991.
- [Sage 89] J.P. Sage, R.S. Withers, and K. Thompson, "MNOS/CCD Circuits for Neural Network Implementations," *Proc. Int. Symposium on Circuits and Systems*, pp. 1207-1209, IEEE, May 1989.
- [Simmon 89] J.A. Simmons, "Acoustic-Imaging Computations by Echolocating Bats: Unification of Diversely-Represented Stimulus Features into Whole Images," *Neural Information Processing Systems (NIPS) Conference*, pp. 2-9, Morgan Kaufmann, 1989.
- [Smith 88] A.V.W. Smith, "The Implementation of Neural Networks as CMOS Integrated Circuits," *PhD.Thesis (University of Edinburgh)*, 1988.

SILICON DENDRITIC TREES

John G. Elias

*Department of Electrical Engineering
University of Delaware, Newark DE, 19716*

INTRODUCTION

In the vertebrate nervous system, communication between distant neurons is done using encoded pulse streams. Closely packed neurons may not produce impulses at all, relying instead on electrotonic spread of membrane potential differences to communicate (e.g. McCormick, 1990). Impulses are also used within the dendritic trees of some, or perhaps all, spatially extensive neurons (e.g. Llinas and Sugimori, 1980). However, their role is not well understood, partly because experimental measurements are extremely difficult to obtain within thin dendritic branches.

Although long distance communication needs may have been the primary evolutionary force behind encoded pulse streams, their impact goes well beyond simple communications. Neurons are continuously bombarded by hundreds or thousands of afferent pulse streams whose impulse responses are integrated by the cell's active and passive membrane elements. A cell's response depends on, among other things, its morphology, membrane electrical and chemical properties, and the location of afferent synapses. The resulting membrane voltage trajectory due to the integration of individual impulse responses plays a critically important role in the dynamic behavior of the cell (e.g. Rapp et. al., 1992). Even a relatively small number of afferent pulse streams leads to complex changes in membrane voltage that may be useful in neural computation (e.g. Northmore and Elias, 1993).

As pointed out by Murray et. al. (1987, 1991), the use of encoded pulse streams for computation and for communication represents an attractive approach towards implementing highly robust large-scale artificial neuronal systems. In their approach, binary signals in the form of encoded pulse streams are used to control analog circuitry and carry information between computational units. The pulse coded approach has since been adopted by a number of research groups, some of which are represented by chapter contributions in this book.

In this chapter, we describe the silicon implementation of an ADT (artificial dendritic tree) that is intended to emulate the behavior of spatially extensive biological neurons which have passive dendrites. We begin with a presentation of background material that should illustrate the linkage between our artificial structures and their biological paragons. This is followed by a description of the artificial neuronal elements and their behavior under impulse excitation. The next section covers aspects of silicon implementation, and the final section is a brief discussion of a tracking controller that makes use of ADTs.

BIOLOGICAL COMPUTATIONAL ELEMENTS

Neuron anatomy can be generalized as having three major parts with the following highly simplified functional descriptions: 1) a spatially extensive dendritic tree, which receives and integrates afferent signals at specific synaptic sites distributed over the entire tree structure; 2) a soma, which forms a response based on the collective dendritic tree electrical signal; and 3) an axon, which propagates efferent impulsive signals from the soma to distant dendritic trees of other neurons. All parts of the neuron depend with varying degree on electrical and chemical changes elsewhere in the cell. For many neurons, the dendritic tree occupies 80-90% of the total surface area of the neuronal membrane, and it is believed that this extensive morphology gives the neuron powerful dynamic signal processing capabilities (e.g. Rall and Segev, 1987) which are not represented by highly abstract models such as the perceptron (Rosenblatt, 1962).

Dendritic Trees and Chemical Synapses

Although the electrical properties of dendrites and axons have been studied for over one hundred years, much of what is known about electrical signal spread in dendrites has been developed during the last thirty years by Wilfrid Rall and his co-workers (e.g. Rall, 1957; 1989). In this section, we limit the brief discussion to passive dendrites and refer the interested reader to (Llinas and Sugimori, 1980; Koch et al., 1983; Shepherd et al., 1989;

Hounsgaard and Midtgaard, 1988) for more information on the active properties of dendrites.

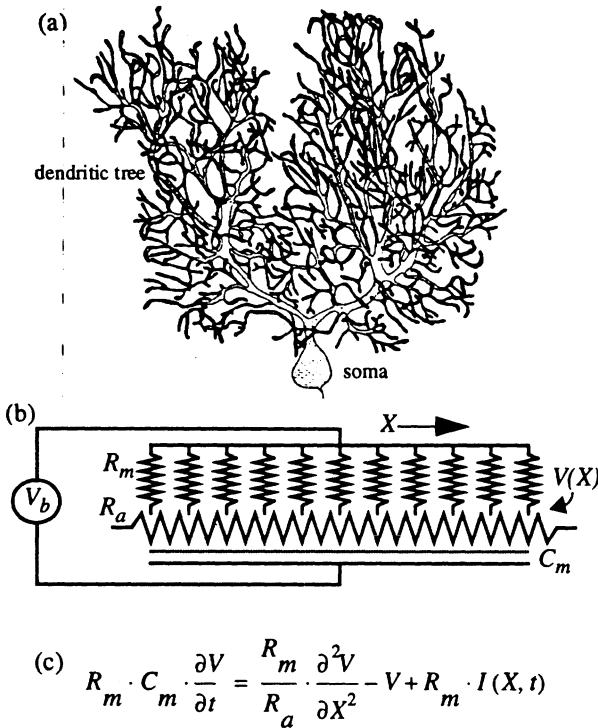


Figure 1. a) Drawing of typical Purkinje cell (after Berry and Bradley, 1976) from cerebellum showing extensive dendritic tree which supports $\sim 10^5$ synapses in humans. b) Electrical model for a section of passive dendrite with a membrane capacitance, C_m , in parallel with a membrane resistance, R_m , and with a series axial cytoplasmic resistance, R_a . The resting membrane voltage is determined by voltage source, V_b . c) One-dimensional cable equation which gives the membrane voltage, V , at location X at time t as a result of current density, $I(X,t)$.

It is common practice in neural network research to model the neuron as a point entity that receives and processes inputs at the soma (cell body). However, most neurons have a spatially extensive dendritic tree structure, which not only forms most of the cell's surface area but provides a spatiotemporal signal processing capability not present in traditional neural network models. Figure 1a depicts a drawing of a Purkinje neuron (Berry and Bradley, 1976) from the cerebellum which attempts to show the extensive

dendritic tree structure that is common with these types of cells. The dendritic tree receives most of the afferent impulses, which in human Purkinje cells is at approximately 10^5 different synaptic locations.

Figure 1b is an electrical model for a section of passive dendrite with a membrane capacitance, C_m , in parallel with a membrane resistance, R_m , and a series axial cytoplasmic resistance, R_a . A linear second-order differential equation, the cable equation shown in Figure 1c, describes the one-dimensional voltage profile for a given current density, $I(x,t)$. Transmembrane dendrite current (outward or inward) results in a soma voltage that depends in a nonlinear way on the location of the transmembrane current on the dendrite (Rall, 1964). This characteristic provides a means to scale or weight an input signal, over a wide dynamic range, by simply selecting the position for inward or outward current on the dendritic branches. The means to produce inward or outward current at particular sites is provided by the chemical synapse.

The synapse is the basic computational element of the nervous system. It is the site of signal transduction, where afferent signals, which originate from either distant or adjacent neurons, are received and combined to effect a new neuronal state. The extent of the state change is dependent on a large number of factors, not least of which is the past state trajectory. There is much experimental evidence that suggests that certain levels of afferent activity density leads to potentiation or depression of the cell's normal resting voltage that persists over widely varying times (e.g. Andersen, 1987; Hebb, 1949). These short and long term potentiation effects are thought to play an important role in neurocomputation and memory. There are at least three types of synapses in nature: chemical, electrical, and ephaptic (e.g. McCormick, 1990). In this chapter, we will focus only on emulating the computationally important behavior of the chemical synapse, which is believed to be, by far, the most common type.

We shall assume that the chemical synapse can be modeled as an ensemble of opened or closed charge-passing channels that open transiently due to the arrival of neurotransmitter at receptor sites on the postsynaptic terminal (e.g. Nicoll, 1988). If the released transmitter molecules are short lived and arrive and bind to receptor sites on the postsynaptic terminal in a time that is short compared to the postsynaptic membrane response then this process can be modeled as an impulsive event. The impulse response then depends only on the dynamics of the postsynaptic cell. If the postsynaptic cell has an Nth-order low pass filter characteristic behavior, where N is greater than unity, then its impulse response closely resembles the alpha function (Rall, 1967).

The nature of the interaction of synapses with respect to the electrical distance between simultaneously active sites is a critically important property of passive dendritic trees. An active synapse is defined as the transient opening of ion-specific channels that permits current to flow if a potential difference exists across the membrane for those specific ions. For active synapses that are temporally coincident and that are located at electrically nearby sites (i.e. the resistance between them is relatively small), the resultant signal is a sublinear function (e.g. Koch et al., 1983; Shepherd and Koch, 1990b). Conversely, for temporally coincident signals in which the active synaptic sites are electrically distant from each other, the resultant signal is nearly linear. This phenomenon is extremely important in providing a rich environment for computation.

An excitatory synapse results in a positive or depolarizing voltage trajectory if its reversal potential is greater than the membrane resting voltage. An inhibitory synapse results in a negative or hyperpolarizing voltage trajectory if its reversal potential is less than the membrane resting voltage. An important class of inhibitory synapses have their reversal potential near the normal membrane resting potential and therefore produce no voltage transitions whenever the membrane voltage is near rest. These silent synapses are believed to play a critical role in visual processing (e.g. Torre and Poggio, 1978).

The transient response of dendrites to depolarizing or hyperpolarizing synaptic activation shows two important features. First, the peak voltage amplitude as measured at the soma is largest for active sites nearest the soma and gets progressively smaller for sites further away. Second, the time at which the peak occurs shows a similar dependency on the distance from the active site so that distal sites are spread out in time and reach their peak values later than more proximal sites. This transient behavior suggests the possibility of powerful dynamic signal processing capabilities using simple circuit elements modeled after dendritic trees and chemical synapses.

Somata and Axons

The electrical response due to synaptic stimulation diffuses or propagates through the dendritic tree to all other portions of the cell interior. As the electrical signal spreads to other parts of the cell, an action potential or impulse might be generated at specialized sites along the dendrites, axons, or soma. An action potential is a highly nonlinear voltage transition that is usually triggered by the cell membrane voltage exceeding some threshold value. As mentioned previously, information is often conveyed from neuron to neuron by modulating the interpulse interval and their phase relationships. The interpulse interval can vary over a large range. Some neurons produce high pulse density

bursts with long intervals in between bursts. Other neurons are continuously producing pulses with a particular interpulse interval. Synaptic activity in the dendritic tree along with cell morphology and membrane electrical properties have a dominant role in determining the resulting pulse behavior. In general, depolarizing excitatory responses tend to reduce the interpulse interval, and hyperpolarizing inhibitory responses tend to increase the interpulse interval. Silent synapses tend to play a localized inhibitory role in particular regions of the dendritic branches (e.g. Northmore and Elias, 1993).

Axons carry information in the form of encoded impulses to distant neurons. Axons are described by the same electrical model as passive dendrites and therefore are fairly lossy. In order to transmit pulses over long distances, axons often have an additional coating (myelin) that helps to reduce losses, and they have repeaters every few millimeters that regenerate the pulses.

ARTIFICIAL DENDRITE AND CHEMICAL SYNAPSE

In this section, we describe electronic circuits that 1) emulate the electrical behavior of passive dendritic trees and chemical synapses and 2) are simple and robust enough to ensure that networks, which ultimately need to support huge numbers of synapses, can be constructed with standard VLSI processing. Electronic analogs of active dendrite behavior (e.g. Llinas and Sugimori, 1980; Shepherd et al., 1985, 1989; Hounsgaard and Midtgaard, 1988) will not be treated in this chapter.

Artificial Dendrites

Passive artificial dendrites are formed by a series of standard compartments, where each compartment has a capacitor, C_m , that represents the membrane capacitance, a resistor, R_m , that represents the membrane resistance, and an axial resistor, R_a , that represents the cytoplasmic resistance (e.g. Rall, 1989). Figure 2a shows a section of artificial dendrite with five standard compartments that is part of a much longer branch like that shown in Figure 2b.

The transient response of the artificial dendrite is of primary importance. Figure 2c shows the impulse response measured at point S due to inward impulse current at four different locations, A , B , C , and D on a passive artificial dendrite as represented in Figure 2b. The location S represents the position of the soma. Therefore, the voltages measured at S are those that would affect somatic voltage-sensitive circuits and perhaps cause the generation of an efferent impulse.

As with biological passive dendrites, the peak voltage amplitude is largest for transmembrane current nearest the soma and gets rapidly smaller for sites

further away. The time for the voltage to peak shows a similar behavior: time-to-peak-voltage increases with distance from S (e.g. Rall, 1989). The behavior shown in Figure 2 illustrates how the concept of weight is an inherent property of the dendritic physical structure. It is clear that position along the artificial dendrite can be used to produce an effective weighting, in both time and amplitude, of afferent signals that are in the form of transient inward or outward currents.

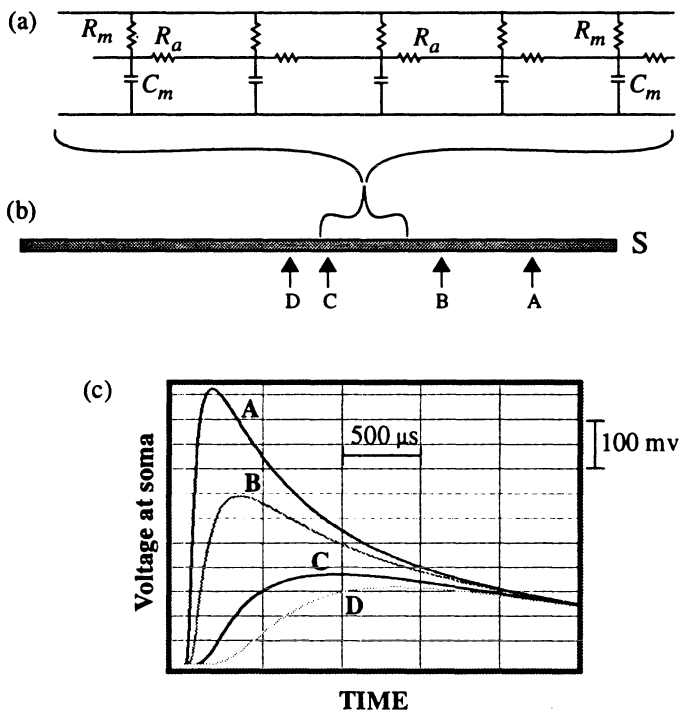


Figure 2. a) Compartmental model of passive dendrite. Each RC section, R_m , R_a , and C_m , is a standard compartment that simplifies VLSI layout. b) Standard compartments are abutted on substrate to form silicon dendritic branches. c) Impulse response of single artificial dendritic branch due to transient transmembrane current at indicated locations on branch.

Artificial Chemical Synapse

The means for enabling inward or outward impulsive current at a specific artificial dendrite location is accomplished by using a single MOS field effect

transistor. A p-channel transistor enables inward current, which produces a depolarizing excitatory type response, and an n-channel transistor enables outward current, which results in an inhibitory type response. Some n-channel transistors have one of their terminals connected to the resting voltage, V_{rest} , and play the role of silent or shunting inhibitory synapses. Two variants of artificial dendrites are shown in Figures 3a and 3b, where p-channel (upper) and n-channel (lower) transistors are placed at uniform positions along the branch. In Figure 3a, only hyperpolarizing inhibitory synapses are present, while in Figure 3b, both hyperpolarizing and shunting are shown. The transistors are turned on by an impulse signal applied to their gate terminals. Both transistor types operate in the triode region. Therefore, the amount of transmembrane current depends on the conductance of the transistor in the on state, the duration of the gate terminal impulse signal, and the potential difference across the transistor, which is dependent on the state of the dendrite at the point of the synapse. All excitatory transistors have identical drawn dimensions (as do, currently, inhibitory transistors), and both excitatory and inhibitory artificial synapses are placed at the same locations in the current chip implementations. In normal operation, both excitatory and inhibitory transistors at the same site may turn on simultaneously.

Electrical Response of Artificial Dendritic Trees and Synapses

In Figure 2, we illustrated the behavior of the impulse response amplitude as a function of the synapse position on the ADT, thus demonstrating the effective weighing of inputs that are mapped onto the tree structure. The impulse response amplitude as a function of the afferent impulse signal width is shown in Figure 4a, which represents measured responses from one of our VLSI circuits for four different impulse widths. A similar postsynaptic behavior is found in biological preparations under presynaptic voltage-clamp: presynaptic depolarization produces a nearly linear increase in postsynaptic voltage (e.g. Angstadt and Calabrese, 1991). This behavior may be due to a lengthening of the time over which transmitter is released, thereby increasing transmembrane current in the postsynaptic terminal. In any event, the efficacy of existing connections can be changed by altering the impulse width. We are investigating how this may be done on a local basis, perhaps consistent with Hebb's postulate (Hebb, 1949), such that both local synaptic strength and the location of the synapse on the branch combine to produce an effective synaptic weight for a given connection.

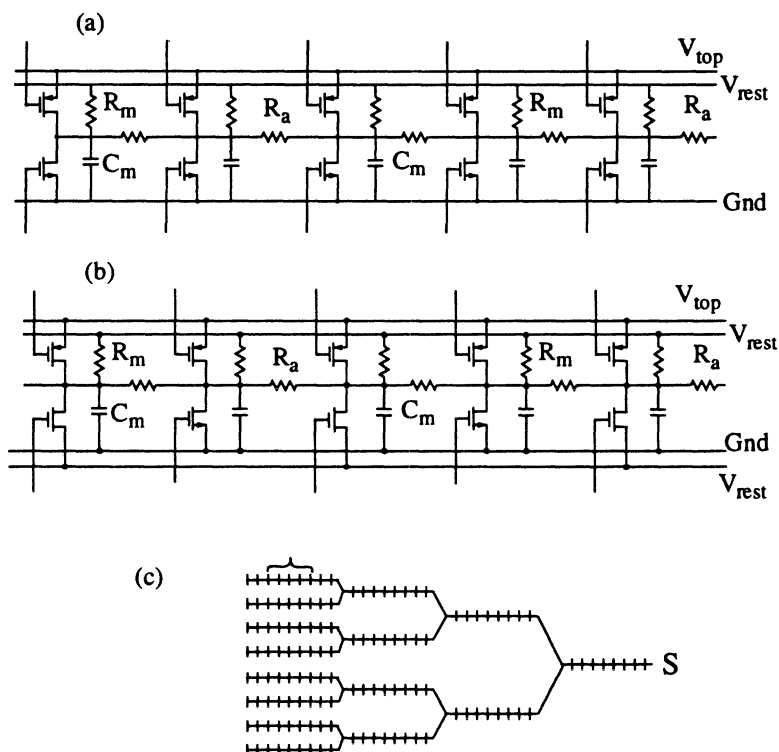


Figure 3. a) A five compartment section of artificial dendrite with five excitatory and five hyperpolarizing inhibitory artificial synapses. (b) Same as (a) but includes three shunting inhibitory synapses. V_{rest} is the resting voltage, V_{top} is the maximum membrane voltage. (c) A multibranching ADT which is constructed by piecing together artificial dendrite sections like that in (a) and (b).

The artificial dendrite's voltage response to closely spaced impulses is shown in Figure 4b. The response due to each synaptic event is added to the resultant branch point voltage from past events until the voltage reaches a maximum value. This behavior is the expected impulse response of an N th-order system and is solely due to the effective postsynaptic membrane. The same behavior would be observed if the phasing of multiple, transiently conducting, artificial synapses was short compared to the effective membrane constant.

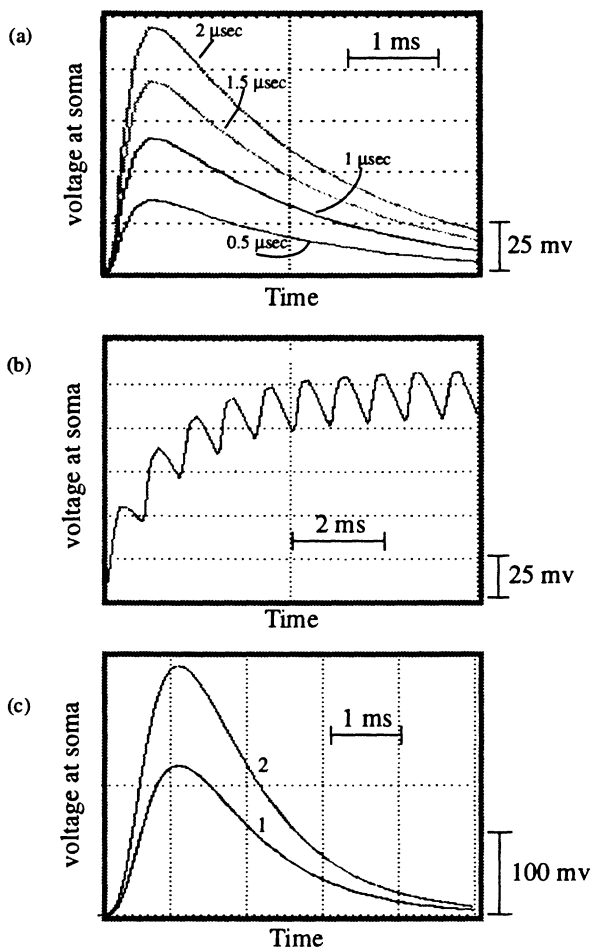


Figure 4. Experimental results from artificial dendrite-synapse circuit to afferent stimulation a) Graded response: amplitude of voltage peak at soma is linearly related to afferent impulse width over wide range. b) Tetanus response: closely spaced impulses cause voltage response to saturate if impulse rate is faster than membrane decay time c) Nonlinear and nearly-linear response: curve 1 is the resultant somatic voltage for simultaneous stimulation of two adjacent synapses on same branch (see fig 3c). Curve 2 is somatic voltage for simultaneous stimulation of two synapses on different branches. Positions of synapses for curves 1 and 2 were equidistant from soma.

Multiple, simultaneously conducting synapses that are electrically close together produce a voltage at the soma that is less than the sum of their

individual responses (e.g. Shepherd and Koch, 1990b). This sublinear effect is due to the shunting load seen by each synaptic site when electrically nearby synapses open their channels. In contrast, multiple, simultaneously conducting synapses that are electrically far apart produce a nearly linear resultant voltage at the soma. Both of these behaviors, as measured at the trunk of a two-branched ADT (e.g. point S in Figure 3c), are shown in figure 4c. The smaller voltage transient (curve 1) was measured when two adjacent artificial synapses were simultaneously active on the distal end of one of the branches. The larger voltage transient (curve 2) shows the resultant voltage when two artificial synapses on separate branches were simultaneously active. In this case, the resultant is nearly twice that of the previous example. In both cases, the artificial synapses were equal distance from the point of measurement. This type of behavior clearly enriches the signal processing capabilities of systems comprised of spatially extensive dendritic trees (Koch and Poggio, 1987).

Artificial Somata and Axons

The analog output voltage of each ADT must be processed by an impulse generating artificial soma. Although the behavior of biological somata is quite complex, and it varies considerably depending on cell type and species (e.g. Koch and Poggio, 1987), we believe a simple circuit will suffice. Soma circuits like those described by Mead (1989), Meador et. al. (1991), and Lazzaro (1992) are simple to implement and exhibit desirable behavior. We are currently investigating the use of these circuits or adaptations of them with our ADTs.

In nature, the axon is the channel over which information is conveyed to distant locations. If this was its only function we would have little need to emulate its behavior. However, axons, like dendrites, impart a delay in the flow of information, which we believe is an important system capability. Fortunately, information carried by axons is in the form of impulses which makes delaying them by arbitrary amounts of time a fairly simple matter. In our system, impulses produced by either sensors or artificial somata are captured by monitoring circuitry (see below) that can hold the impulse for a specified time before sending it to its final destination(s).

SILICON DENDRITIC TREES

If artificial dendrites are to be used in real systems then they must be implemented via a process that can make huge numbers of them in a small area inexpensively. The only feasible path for doing this currently is with standard silicon processing methods. In this section, we discuss briefly the implementation of a dendritic system in silicon. Before discussing the silicon

implementation, we must say a few words about our method of making connections between synapses and impulse sources.

Convergent, Divergent, and Recurrent Connections

Networks that are built with artificial dendrites and synapses process signals that have a spatiotemporal significance by mapping afferent signal pathways to specific locations on the dendritic trees. The connections between synapses and the outputs of sensors and neurons determine the overall system response for a given dendritic dynamic behavior. The number of different connection patterns is quite large and is a factorial function of the number of synapses and sensor elements. If we limit, for the moment, the number of divergent connections to one, then the total number of different connection patterns is given by

$$\frac{N!}{(N - M)!}$$

where N is the number of artificial synapses and M is the number of sensor and neuron outputs. Artificial systems may have thousands of afferents and many times more synapses, resulting in an extremely large number of possible connection patterns. In our system, we allow each sensor element or artificial neuron to make unrestricted divergent connections and each synapse to receive multiple convergent connections from both sensor elements and artificial neurons. This tends to make the number of possible connection patterns much larger than that indicated by the equation.

Virtual Wires

In the implementation of an electronic system, the number of data pathways in or out of modules is limited by the available technology. Integrated circuit packages rarely exceed 500 pins; our current artificial dendrite chips are in 40 pin packages. This limitation in pin count is of special concern with dynamic artificial neuronal systems because of the analog nature of the computation. Each sensor or neuron output must be able to connect to any one of the artificial synapses in the system, and the spiking outputs from sensors and neurons must arrive at their artificial synapse destinations in a parallel fashion.

In order to overcome I/O limitations and to meet the connectivity and timing requirements, we make use of a multiplexing scheme that we refer to as virtual wires. In this scheme, the outputs of active neurons and sensors (i.e. those that are currently producing a spike or impulse) cause the synapses that they connect with to become activated after a delay that is specific for each

effluent connection. The process of reading an active output causes that output to return to the inactive (i.e. nonspiking) state.

Virtual wires are formed using four circuits: digital or analog Stimulus Memory, which is closely associated with each synapse, Address Decoding, which serves all on-chip synapses, State Machine, which determines sensor and neuron output states, and Connection List, which specifies the locations of synapses and the axonal delay associated with each connection. Stimulus Memory and Address Decoding are on-chip circuits; the Connection List and State Machine are off-chip. With digital Stimulus Memory, the activated synapses throughout the system are turned on transiently by a global impulse stimulus signal. With analog Stimulus Memory, the synapses turn on as soon as they are activated and turn off after either a globally set or individually set delay.

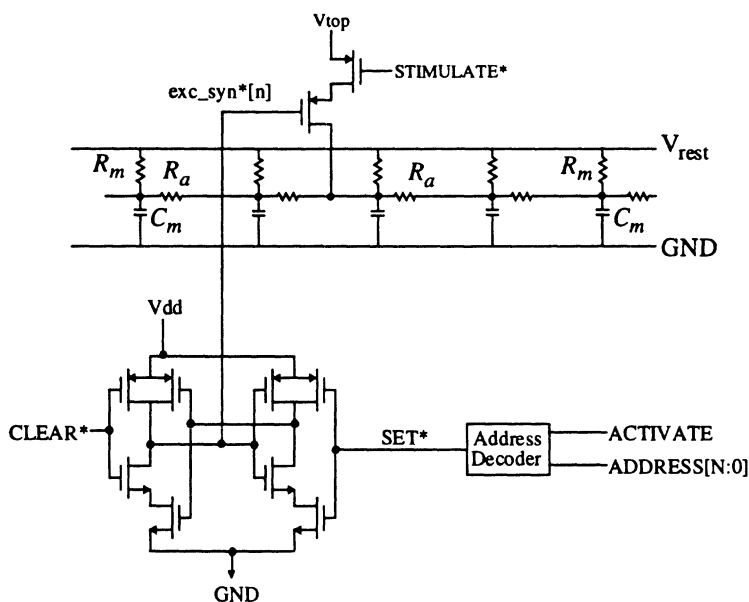


Figure 5. Digital excitatory Stimulus Memory shown with its p-channel synapse transistor. SET* is asserted by applying the proper address and asserting ACTIVATE. When SET* is asserted the synapse is activated (i.e. exc_syn*[n] is set to logic 0) and the artificial synapse will turn on while STIMULATE* is asserted. CLEAR* inactivates all Stimulus Memory locations throughout the system. Both STIMULATE* and CLEAR* are global signals in the system.

The circuit diagram of a digital excitatory Stimulus Memory connected to its p-channel artificial synapse transistor is shown in Figure 5. With the digital

Stimulus Memory, nine transistors are needed for each artificial synapse. A synapse is activated when its $\text{exc_syn}^*[n]$ is set to logic 0 by asserting SET^* while CLEAR^* is unasserted. The SET^* signal is asserted by the on-chip address decoder when the proper combination of external address lines and control signal are asserted. An activated synapse will turn on while the global impulse signal, STIMULATE^* , is asserted. The global signal CLEAR^* is asserted after every STIMULATE^* assertion to inactivate synapses in preparation for the next round of sampling and activation.

Digital Stimulus Memory requires a rather large number of transistors for each synapse and does not lend itself to individually variable synapse on-times as discussed above and illustrated in Figure 4a. Therefore, we have designed an analog Stimulus Memory cell that is considerably simpler and provides some control over the synapse on-time. The circuit diagram of an analog inhibitory Stimulus Memory connected to its n-channel artificial synapse transistor is shown in Figure 6. The same address decoder is used in both the digital and analog designs and the interface is the same except the signals CLEAR^* and STIMULATE^* are not required. The duration of the on-time for a particular synapse type (i.e. excitatory, hyperpolarizing, shunting) throughout a chip is controlled by separate bias voltages, V_b .

The Connection List is a multiple-bit-wide memory that holds the synapse addresses and axonal delay of each efferent connection in its domain. For large systems, we plan to divide the network into domains that will permit a certain level of parallel sampling of neuron and sensor outputs which should

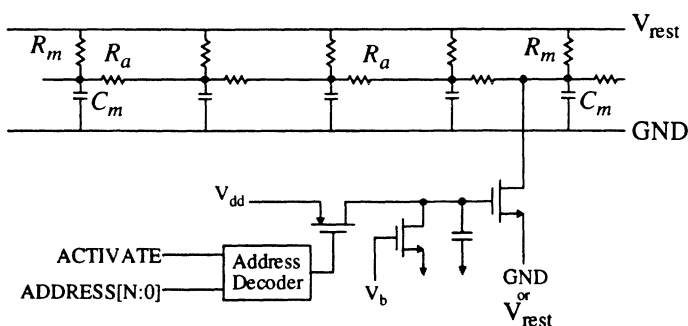


Figure 6. Analog inhibitory Stimulus Memory. Connection of synapse source terminal to GND results in a hyperpolarizing behavior while connection to V_{rest} produces a shunting or silent synapse behavior. The synapse turns on when a proper address is present and ACTIVATE is asserted. The on-time duration is controlled by V_b which is different for each synapse type on a chip.

enhance system scalability. The Connection Lists across all domains hold the pattern of connectivity for the system and thus their contents determine system behavior. A connection pattern can be fixed in ROM, or as in our present system, loaded via computer for experimentation

Figure 7 illustrates a simplified single-domain system comprising Connection List, sensor, State Machine, and four neuromorphic chips, each of which contain a number of artificial neurons. The outputs of the artificial neurons on each chip are sampled via a multiplexer which is selected by the

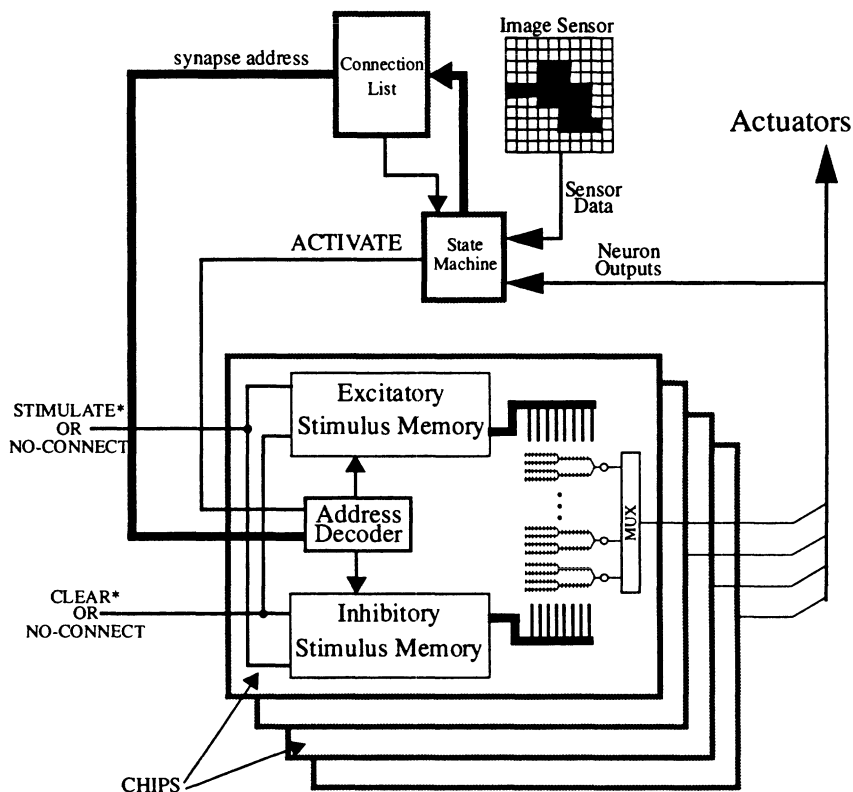


Figure 7. Simplified block diagram for single domain system showing basic operation. All sensory and neuronal outputs activate the artificial synapses that they connect to through the virtual wires. NO-CONNECTs are for analog Stimulus Memory implementations.

on-chip Address Decoder. Each neuron is in one of two states, so only a single output pin is needed to read all of them. In operation, the State Machine reads the state of each sensor element and every neuron in its domain. A spiking neuron or sensor output is detected by the State Machine which then activates all of the synapses that connect to that particular sensor or neuron. For chips with digital Stimulus Memory, STIMULATE* is asserted transiently after reading all outputs, thus turning on activated synapses while it is asserted. This is followed by asserting CLEAR*, which inactivates all synapses. For chips with analog Stimulus Memory, synapses turn as they are activated and stay on for a time that is different for each synapse type. As with Mahowald's method of connecting neuron outputs to synapses (Mahowald, 1992) addresses of synapses and neurons are used rather than direct connections carrying spikes.

Standard Dendrite Compartment

Figure 8 illustrates the basic integrated circuit layout of our standard dendrite compartment *sans* synapse transistors. Each compartment has an effective membrane capacitance, C_m , an effective membrane resistance, R_m , and an effective cytoplasmic resistance, R_a . The size of the artificial dendrite standard compartment varies with each chip design. In the past, it has ranged from 18 μm by 180 μm to 18 μm by 360 μm with most of this area being taken up by the capacitor.

The capacitor is the largest element in the standard dendrite compartment and is implemented using conventional silicon processing methods (e.g. Allen and Holberg, 1987). The capacitor was fabricated with two layers of polysilicon separated by a thin oxide layer. The top plate of the capacitor is polysilicon layer 2 (poly2) and connects to a ground bus that runs perpendicular to the long axis of the capacitor. The bottom plate is polysilicon layer 1 (poly1) which connects directly to the resistors, R_a and R_m , and to the synapse transistors in the stimulus memory (see Figures 5 and 6). The capacitance varies, depending on chip design, between 1 and 2 pf, which is based on an oxide thickness of approximately 700 \AA . There are many techniques to reduce the footprint of the capacitor while keeping the same capacitance: thinner dielectric, use material with a higher dielectric constant, e.g. silicon nitride, employ three-dimensional capacitors, e.g. trench or tower capacitors, but we will not explore these further here.

The compartmental resistors may be implemented by a number of standard silicon fabrication techniques: well, pinched, active, and SC (e.g. Allen and Sanchez-Sinencio, 1984). The resistor footprint for a particular resistance depends not only on the resistance value but also on the implementation technique. Well resistors have a footprint advantage over the other techniques because the well resistor can be put under the capacitor.

Therefore, a well resistor does not take up any silicon real estate but it has the disadvantage of relatively small resistance (measured as $5 \text{ k}\Omega$ per square for our chips). Pinched resistors have a higher resistance but can not be placed under the capacitor. Active and SC resistors require control voltages and clock signals and emulate resistor behavior over a certain range of terminal voltages and resistance values.

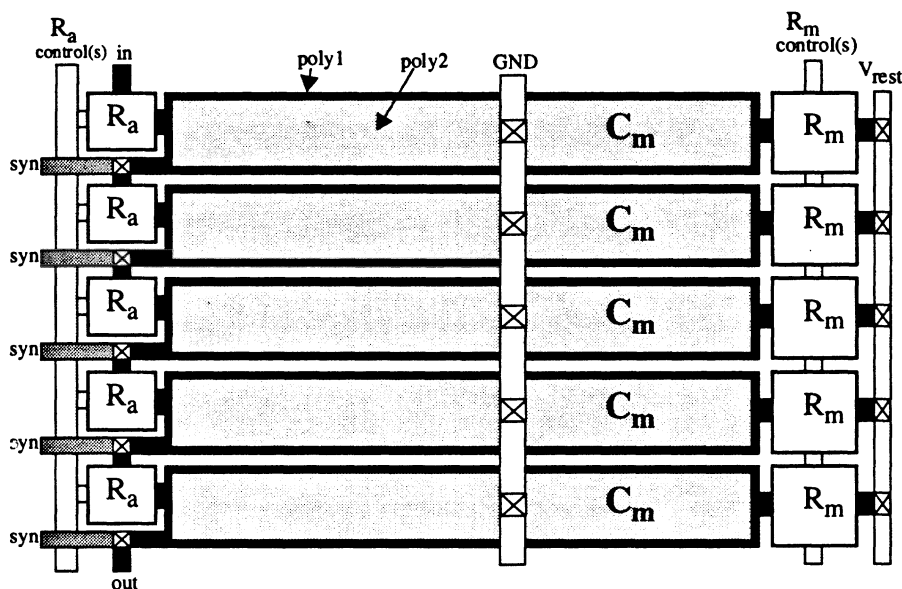


Figure 8. Basic VLSI layout for standard dendrite compartment. Five compartments are shown. Control lines for resistors permit adjustment of resistance over a limited range. V_{rest} establishes the resting voltage (typically 1V). The synapse transistors (not shown) connect to terminals on the left hand side labeled syn.

We have implemented n-well, active, and SC resistors on different chips and will report on the details of their design and relative behavior later (Elias and Meshreki, 1993). In our standard dendrite compartment, n-well resistors go under the capacitor and active or SC resistors are placed at the ends of the capacitor as shown in Figure 8. Independent control signals for changing the resistance of the SC or active R_m and R_a pass along both sides of the compartment. For chips with active resistors, the control signals are DC voltages that permit a certain range of adjustment. With SC resistors, the control signals are AC voltages in which the frequency determines the resistance. Presently, the R_m resistors in all of the compartments share the

same control signals and the R_a resistors share a different set of control signals. Therefore, all compartments have nominally the same R_m and R_a resistances.

The standard dendrite compartment was designed to abut with adjacent compartments and was pitch-matched to the on-chip virtual wire circuitry. This method makes the construction of ADTs a relatively simple task: to make a branch, standard compartments are placed side-by-side until the desired branch length is reached. Branches are then connected via metal or poly wires to form trees. The spacing between compartments is the minimum distance between capacitors (2 μm). The compartments are aligned such that the inputs of one compartment connect to the outputs of the previous compartment.

Silicon Implementation

The ADT circuits and their on-chip virtual wiring are fabricated using a 2 μm CMOS double-poly n-well process on a 2mm by 2mm Mosis Tiny Chip (e.g. Mead, 1989). The artificial somata and output multiplexer have been left off the chips thus far to permit experimentation with different soma circuits. Four artificial dendritic branches each having 15 excitatory and 15 hyperpolarizing inhibitory artificial synapses have been implemented on most of the chips. The number of synapses are kept low in order to leave open silicon areas on the chips for other analog test circuits.

Figure 9 shows an ADT chip layout that uses digital stimulus memory cells. In this implementation, the four branches are in-line, with a gap in between each branch, and centered on the die. The ends of each branch are taken out of the chip through package pins to allow experimentation with different tree structures. Multiple chips can be combined as well to produce tree structures with more branches, longer branches, or higher order branching. The remaining circuitry makes up the virtual wires.

TRACKING CONTROL

We are investigating the use of ADT circuits in various control and sensory processing applications. One of these makes use of ADTs to control the pointing of a video camera such that a maneuvering-target is always kept in the center of the camera's field of view. In this system, the sensor is a standard RS-170 type camera and its pointing actuators are torque motors that are antagonistic to each other. The actuators couple to the camera body via cables and cause the camera to pivot about a single point which is fixed on a multi-axis platform. Both camera and platform are controlled by similar ADT circuitry and actuators. The paired torque motors and cables are simple analogs of antagonistic muscle groups and tendons of the primate oculomotor and head systems, the camera representing an eye, and the platform functioning as a

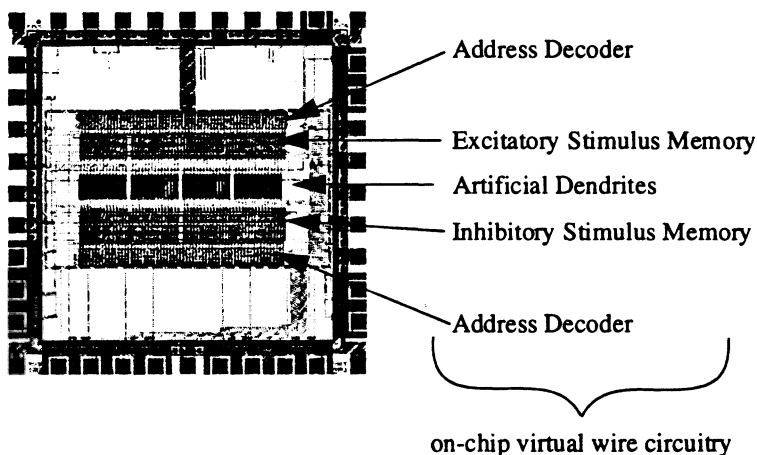


Figure 9. Chip layout of artificial dendrites fabricated using a MOSIS 2 μm double polysilicon standard CMOS process. The four artificial dendritic branches can be seen in the center of the die. The ends of each branch are connected to pads which allows experimentation with different branching structures. Each branch has 30 synapses (15 excitatory and 15 inhibitory) which are uniformly spaced along the branch.

head. In operation, the paired motors, linked by camera and platform attachment points, move against each other by applying unequal tension on the cables, thereby changing the gaze direction. In the complete system, there are two pairs of antagonistic actuators attached directly to the camera and two pairs attached to the camera's supporting platform.

Live video, directly from the camera, is displayed on a monitor, which allows observers to follow the progress of the tracking system. In operation, any target that moves into the camera's field of view will attract the system's interest and cause it to rapidly rotate the camera and platform in a direction that makes the target's image less eccentric with respect to the field of view. Rotational movement continues until the target's image is centered. The greater the target eccentricity, the greater is the peak rotational velocity of the camera.

The one dimensional system shown in Figure 10 can serve to illustrate the basic control operation of the complete system. Only two ADTs are needed for this one-dimensional target tracker. Each artificial neuron uses a single dendritic branch which has $N/2$ excitatory and $N/2$ inhibitory artificial

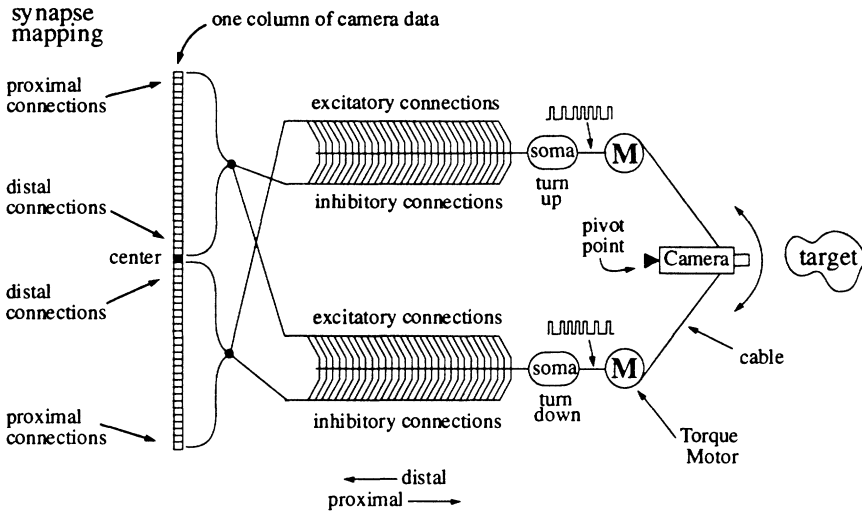


Figure 10. Simplified block diagram of a one-dimensional target tracking system that makes use of ADTs and oscillating artificial somata. Camera sensor elements near center make connections to distal synapses on ADTs while sensor elements near edges make connections to proximal synapses.

synapses, where N is the number of sensor elements. The connections between the sensor and each artificial neuron have a simple triangular structure (Elias, 1992). However, the inhibitory and excitatory connections for each artificial neuron are mirror images of each other: the top half of the sensor makes excitatory connections to the lower ADT and inhibitory connections to the upper ADT; the bottom half of the sensor makes inhibitory connections to the lower ADT and excitatory connections to the upper. This connection pattern gives the system the ability to detect imbalance in the sensor's field of view. With an unbalanced sensor field, the outputs of the neurons are no longer of equal pulse density which forces the actuators to pull the camera in a direction that reduces the eccentricity between target and camera center.

The artificial somata are voltage controlled pulse generators which, in the absence of dendritic transients, oscillate with a center frequency of f_0 . Their output pulse streams are not synchronous and they have approximately the same center frequency, f_0 . When the target is in the center of the camera's field

of view, the ADTs receive a balanced set of stimuli from the camera. Under balanced conditions, the transient output from the dendritic branch is approximately zero. Therefore, the soma center frequency remains at f_0 . When the target is off center, the dendritic branches receive an unbalanced stimulus pattern that results in a non-zero output transient. The response of the artificial somata to an excitatory input is to transiently increase the output frequency above f_0 . Presently, hyperpolarizing membrane voltage does not reduce the pulse frequency below f_0 . The amount by which f_0 changes depends on the magnitude of the integrated impulse responses from the ADT. The dynamics are dependent on the effective membrane properties and the spatial location of active artificial synapses.

In one dimension, the system operates in the following way. Assume a non-maneuvering target suddenly appears near the top of the sensor's field of view. The camera must be moved up in order to put the target's image in the center of the monitor. Due to the excitatory synaptic activity the pulse rate to the top actuator increases which causes it to pull harder on its cable. Although there are inhibitory voltage transients on the lower ADT, the tension on the bottom stays above a minimum value. As the camera rotates towards the target, all the while reducing the eccentricity, tension on the top cable lessens until the target is centered, when once again the tension on the bottom and top cables is equal and the camera stops rotating.

As the target maneuvers away from the camera's gaze in an attempt to escape, the impulse patterns applied to the dendritic branches induce voltage transients which increase nonlinearly as the distance between target image and monitor center increases. The exact response with distance from the center depends on the actual connection pattern (Elias, 1992) between sensor elements and ADTs. A desirable connection pattern would have the controller rotate the camera faster towards the target when the distance between target image and monitor center is large compared to when the distance is small. With the target's image near the monitor center, the response is much reduced which allows a slower approach to the final camera position and a minimum amount of overshoot.

SUMMARY AND DISCUSSION

Our research program attempts to capture useful neurocomputational principles from biology by applying structure and behavior modeled after synaptic and dendritic levels of implementation. The ADT described in this chapter is the basic computational substrate in our system. Although our electronic models of chemical synapse and passive dendritic tree are, in many respects, extreme simplifications of biological structures, their dynamic

electrical behavior appears to satisfactorily follow that of their biological paragons. The ADT structure is based on a current understanding of passive dendritic trees which results in an extremely simple circuit implementation that is highly scalable. Artificial neurons with extensive dendritic trees have the capability to process signals that have both temporal and spatial significance. In our networks, weights are replaced with connections which, when combined with the sublinear behavior of electrically close synapses and the nearly linear behavior of widely separated synapses, provide a rich computational substrate for signal processing.

ACKNOWLEDGMENTS

The author wishes to thank Peter Warter for several useful suggestions on chip architecture, Hsu Hua Chu and Samer Meshreki for assisting with chip layout, design, and testing.

References

- Andersen, P.O., (1987) "Properties of hippocampal synapses of importance for integration and memory," in *Synaptic Function*, Eds. G. M. Edelman, W. E. Gall, W. M. Cowan, John Wiley & Sons, chapter 16
- Allen, P. E. and Holberg, D. R. (1987) *CMOS Analog Circuit Design*, Holt, Rinehart and Winston, New York
- Allen, P. E. and Sanchez-Sinencio, E. (1984) *Switched Capacitor Circuits*, New York, Van Nostrand Reinhold
- Angstadt, J. D. and Calabrese, R. L (1991) "Calcium Currents and Graded Synaptic Transmission between Heart Interneurons of the Leech," *J. Neuroscience*, 11(3), 746-759
- Berry, M. and Bradley, P., (1976) "The Growth of the dendritic trees of purkinje cells in the cerebellum of the rat," *Brain Res.*, 112:1-35
- Elias, J. G., (1992) "Genetic Generation of Connection Patterns for a Dynamic Artificial Neural Network," *IEEE Computer Society Press Proceedings of COGANN-92*, a workshop on combinations of genetic algorithms and neural networks

- Elias, J. G., Chu, H. H., and Meshreki, S. (1992) "A neuromorphic impulsive circuit for processing dynamic signals," *IEEE International Conference on Circuits and Systems*, vol. 5, 2208-2211
- Elias, J. G. and Meshreki, S. (1993) In preparation
- Hebb, D. O., (1949) *The Organization of Behavior*, Wiley, New York
- Hounsgaard, J. and J. Midtgaard. (1988) "Intrinsic determinants of firing pattern in Purkinje cells of the turtle cerebellum in vitro." *J. Physiol.* 402: 731-749
- Koch, C., Poggio, T., and Torre, V. (1983) "Nonlinear interactions in a dendritic tree: Localization, timing and role in information processing," *Proc. Natl. Acad. Sci.*, 80:2799-2802
- Koch, C. and Poggio, T., (1987) "Biophysics of computation: neurons, synapses and membranes," in *Synaptic Function*, Edelman, G. M., Gall, W. E., and Cowan, W. M., (eds) chap 23
- Lazzaro, J. (1992) "Low-power silicon spiking neurons and axons," *IEEE International Conference on Circuits and Systems*, vol. 5, 2220-2223
- Llinas, R. and M. Sugimori. (1980) "Electrophysical properties of in vitro purkinje cell dendrites in mammalian cerebellar slices." *J. Physiol.* 305: 197-213
- McCormick, D. A. (1990) "Membrane Properties and Neurotransmitter Actions", in *The Synaptic Organization of the Brain*, Ed. G. M. Shepherd, Oxford University Press, chapter 2
- Mahowald, M.A. (1991) "Evolving Analog VLSI Neurons," in *Single Neuron Computation*, McKenna, T., Davis, J., and Zornetzer, S. F. (eds) Academic Press, Chap 15.
- Mead, C. (1989) *Analog VLSI and Neural Systems*, Addison Wesley
- Meador, J. L., Wu, A., Cole, C., Nintunze, N., and Chintrakulchai, P. (1991) "Programmable impulse neural circuits," *IEEE Transactions on Neural Networks*, vol. 2, no. 1, 101-109

- Murray, A. F. and Smith, A. V. W., (1987) "Asynchronous arithmetic for VLSI neural systems," *Electronic Letters*, vol. 23, no. 12, 642-643
- Murray, A. F., Del Corso, D., and Tarassenko, L. (1991) "Pulse-stream VLSI neural networks mixing analog and digital techniques," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, 193-204
- Nicoll, R. A. (1988) "The coupling of neurotransmitter receptors to ion channels in the brain," *Science* 241:545
- Northmore, D.P. and Elias, J. G. (1993) "Evolving networks of directionally selective artificial dendritic trees," in preparation
- Rall, W. (1957) "Membrane time constant of motoneurons," *Science* 126:454-455
- Rall, W. (1964) "Theoretical significance of dendritic trees for neuronal input-output relations," in *Neural Theory and Modeling*, R. F. Reiss, Ed., Sanford University Press, Stanford, CA., p. 73
- Rall, W. (1967) "Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic inputs," *J. Neurophys.* 30, 1138-1168
- Rall, W. (1989) "Cable Theory for Dendritic Neurons", in *Methods in Neuronal Modeling: From Synapses to Networks*, eds. C. Koch and I. Segev, MIT Press, chapter 2
- Rall, W. and Segev, I. (1987) "Functional possibilities for synapses on dendrites and dendritic spines", in *Synaptic Function*, Eds. G. M. Edelman, W. E. Gall, W. M. Cowan, John Wiley & Sons, chapter 22
- Rapp, M., Yarom, Y., and Segev, I. (1992) "The impact of parallel fiber background activity on the cable properties of cerebellar Purkinje cells," *Neural Computation* vol. 4, 518-533
- Rosenblatt, F. (1962). *Principles of neurodynamics*. New York, Spartan
- Shepherd, G. M., Brayton, R. K., Miller, J. F., Segev, I., Rinzel, J., and Rall, W. (1985) "Signal enhancement in distal cortical dendrites by means of interactions between active dendritic spines," *Proc. Natl. Acad. Sci.* 82:2192-2195

- Shepherd, G. M., Woolf, T. B., and Carnevale, N. T.,(1989) "Comparisons between active properties of distal dendritic branches and spines: Implications for neuronal computations," *J. Cognit. Neurosci.* 1:273-286
- Shepherd, G. M. and Koch, C. (1990) "Dendritic electrotonus and synaptic integration", in *The Synaptic Organization of the Brain*, ed. G. M. Shepherd, Oxford University Press, appendix
- Shepherd, G. M. and Koch, C. (1990b) "Dendritic electrotonus and synaptic integration", in *The Synaptic Organization of the Brain*, ed. G. M. Shepherd, Oxford University Press, appendix
- Torre, V. and Poggio, T. (1978) "A synaptic mechanism possibly underlying directional selectivity to motion," *Proc. R. Soc. Lond. B.* 202, 409-416

SILICON NEURONS FOR PHASE AND FREQUENCY DETECTION AND PATTERN GENERATION

Mark DeYong and Chris Fields

Intelligent Reasoning Systems Inc.

4200 S. Research Rd., Las Cruces, NM88003

and Mt. Airy, MD 81771

INTRODUCTION

Biological neurons employ rapid pulses - action potentials (APs) - for long- distance transmission of signals. A typical AP has a pulse width of one to a few ms and an amplitude of up to 100 mV. Cells that fire action potentials typically fire either continuously or in bursts of several APs separated by quiescent periods. Increases or decreases in activity correspond to increases or decreases in the firing frequency for continuously-firing cells, and to increases or decreases in the frequency or duration of bursts for bursting cells [Kuffler 84, Kandel 85]. Action potentials provide the advantages of relatively low-noise, lossless signal transmission typical of digital encodings of information. Nervous systems do not, however, appear to employ any standard digital encoding. The key features of communication in biological systems are timing, frequency, and phase relations between parallel streams of signals. These features are consistent with the requirements of asynchronous, data-driven computation in environments in which the temporal relations between events are usually important, and in which the system must process information and respond appropriately in times commensurate with those of relevant environmental changes.

An action potential is fired when the internal potential of a specialized

region of the axon, the axon hillock, exceeds a threshold. The potential at the hillock is a convolution of both excitatory and inhibitory post-synaptic potentials (PSPs) generated at synapses on the dendritic tree, the cell body, and the axon itself. The PSPs are typically tens of mV in amplitude and tens of ms in duration at the receiving synapse; transmission through the dendritic tree generally dampens, delays, and broadens each PSP. Neurons may have hundreds to thousands of distinct synapses from as many distinct input neurons, and PSPs from each input may have different amplitudes and durations, and require different amounts of time to affect the potential at the hillock. Neurons do not just count incoming APs, and fire if the number exceeds a threshold. Each action potential reflects the combined effect of inputs received at different times from different sources by the firing cell. Firing is typically the result of the cell's receiving multiple excitatory inputs, and many different patterns of input may initiate firing. In some cells, the majority of inputs are inhibitory; AP firing by such cells therefore indicates a relaxation of inhibition relative to the default state as opposed to an enhancement of excitation over the default state. In either case, the response of the cell is very sensitive to the temporal relations between signals received at the same or different parts of the dendritic tree. The output of a neuron may drive synapses on many other neurons. The high fan-in and fan-out of most neurons, the wide variation in synaptic strengths, the presence of both excitatory and inhibitory synapses on single neurons, and the complexity of the convolution of PSPs due to the complex geometry of dendritic trees make neurons computationally powerful devices. Streams of action potentials from single cells, or from coordinated groups of cells, often have very complex, irregular temporal spectra. The representation of neuronal activity by a smoothly-varying AP frequency that is a simple function of weighted, summed input frequencies, as is typical in many artificial neural-network models can be seriously inaccurate and misleading [Selverston 88, Miall 89]. More detailed and realistic model neurons are needed to investigate the computational properties of neural systems and to assess the utility of neuron-like computational mechanisms for solving real-time vision, sensory integration, motion control, and other engineering problems.

We have developed a hardware model of a typical spiking neuron in order to investigate the types of behavior such devices can produce and their potential for application in signal processing and control systems [Fields 92, DeYong 92a]. Our hybrid temporal processing element (HTPE) explicitly models excitatory, inhibitory, and shunting synapses and PSPs, PSP convolution at the hillock, action potential generation, and AP transmission. The HTPE is more realistic than pulse-stream counters [Card 89, Foo 89] and more complete than the operational-amplifier based silicon neurons of Mahowald and Douglas

[Mahowald 91]. The basic components of the HTPE are few-transistor modules that generate "slow" signals representing PSPs and "fast" signals representing APs.

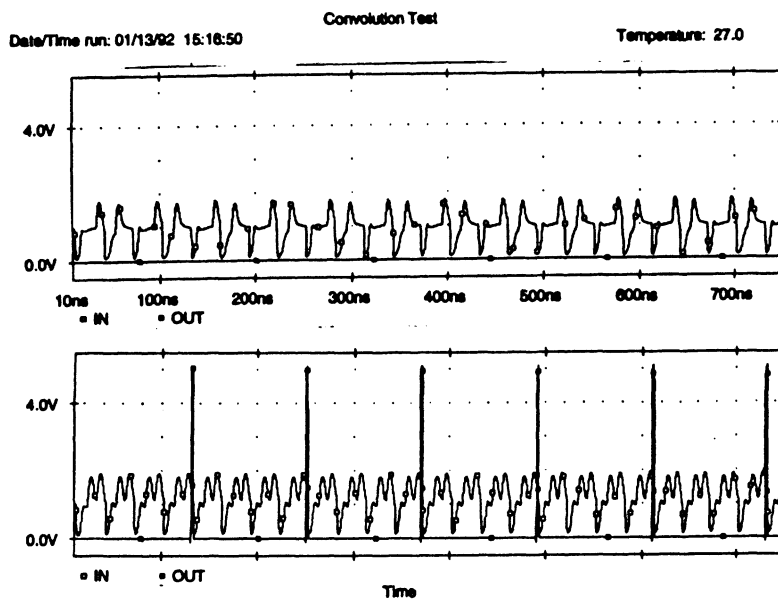


Figure 1. Simulation of HTPE Behavior. SPICE simulation of the convolution of excitatory and inhibitory PSPs to a single HTPE. Upper panel: the excitatory input frequency is too low to overcome the inhibition, so no APs are produced. Lower panel: the excitatory input frequency is sufficient to overcome the inhibition. APs are superimposed on the PSP background.

All signals have been scaled to the V - ns operating range for ease of VLSI fabrication and application to very high-speed signal processing tasks. The widths of these signals may be varied from roughly 10 ns to several ms for slow signals and from roughly 2 - 10 ns for fast signals by varying externally-applied modulating voltages. These PEs can be used as multifunctional building blocks to develop relatively simple circuits for basic logic functions [Fields 92], pulse-stream generation, signal selection via a winner-take-all mechanism [DeYong 92a], and a variety of signal-processing functions (DeYong 92b). Here we show that HTPEs very naturally implement the frequency and phase detection

and shifting, pulse-stream filtering, and complex pattern encoding functions that are ubiquitous in biological neural circuits.

SIMPLE CIRCUITS FOR GATING AND PHASE SHIFTING

The basic HTPE consists of one or more excitatory, inhibitory, or shunting inputs (synapses), a resistive load representing the cell body (soma), and an AP firing circuit (hillock). HTPEs accept pulses as input and produce them as output, and compute internally by analog convolution [DeYong 92a]. A single HTPE with both excitatory and inhibitory inputs acts as a frequency detector that produces output pulses only when the pulse stream on the excitatory input is above a critical frequency set by the strengths of the synapses and the frequency of the inhibitory input.

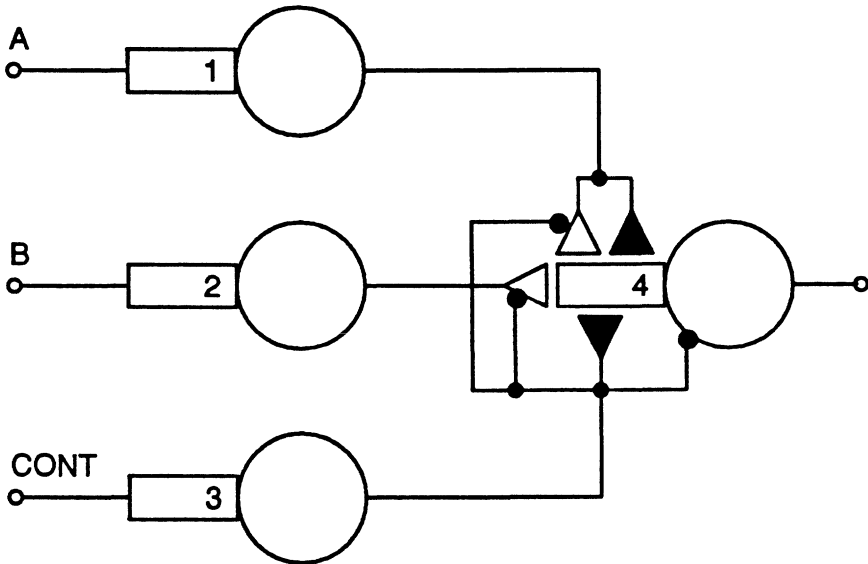
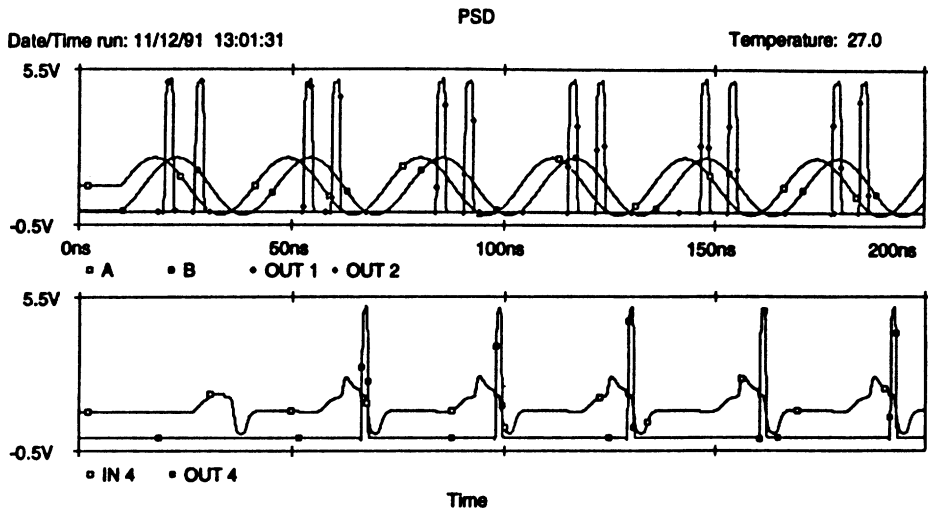
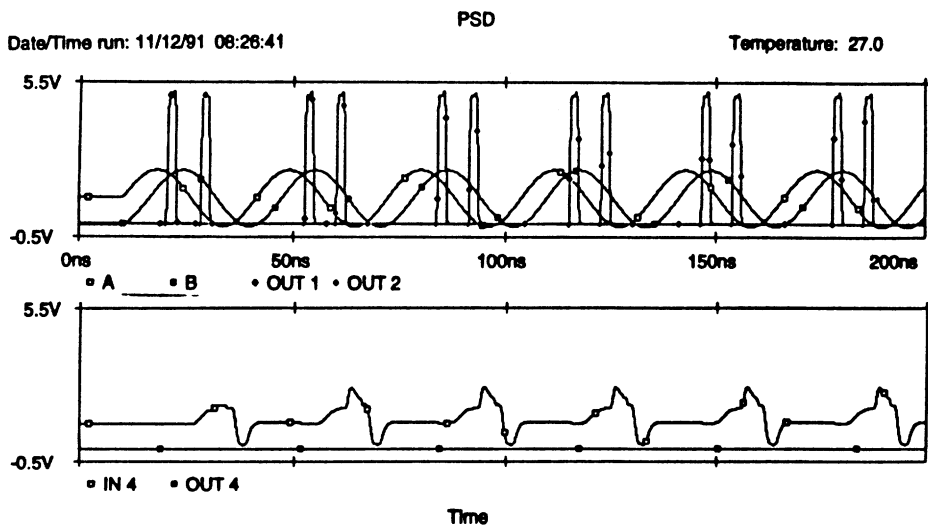


Figure 2. Phase Shift Demodulator (PSD). This single-HTPE system determines if the phase shift between the signals on the A and B inputs is within a specified window. HTPEs 1 - 3 provide input signals. The length of the window is set by the phase difference between the A and CONT signals. If the signals are within the window the output of the PSD (HTPE 4) becomes active. The window duration is a linear function of the A-CONT phase shift.

The frequency of the output increases as the excitatory frequency increases, up to a saturation point set by the refractory period of the hillock. This behavior is the result of the finite width and a-function shape of the PSPs, which convolve to generate periodic waves on a subthreshold background (Figure 1).



a)



b)

Figure 3. Output of the PSD circuit (Fig. 2) for signals with phase differences less than (A) and greater than (B) the phase window set by the inhibitory signal.

The simplest case of phase detection is the detection of coincidence within a fixed time window. The detection of coincidence between two excitatory inputs, neither of which is sufficiently strong to generate an output, is the basis for AND gating by the HTPE [Fields 92]. The width of the coincidence window is set by the widths of the excitatory PSPs; if these are different, the order in which the signals arrive will also influence whether they count as "coincident." Coincidence detection becomes general phase detection if both excitatory signals are periodic; an output pulse is produced at the input frequency if the input signals are of the same frequency and within the coincidence window. An output pulse is produced at the beat frequency if the input signals have different frequencies. The introduction of a third, inhibitory input sufficient in strength to pull the combined excitatory PSP below threshold, as shown in Fig. 2, allows the coincidence window to be varied. Figure 3 shows the response of a single-HTPE phase detection system to acceptable (A) and unacceptable (B) phase shifts between two input signals of the same frequency.

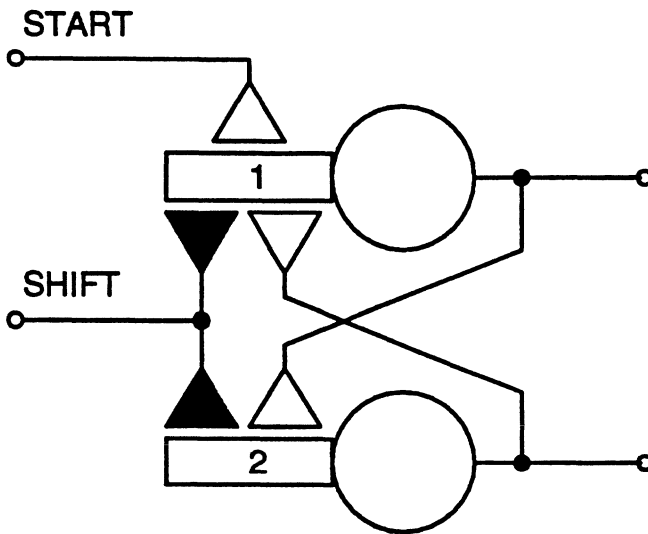


Figure 4. Two-HTPE Oscillator. Oscillation is produced by the cross-coupled excitatory feedback and is initiated by a single AP on the START input. The frequency of oscillation can be varied or stopped by pulsing the inhibitory SHIFT input.

Convolution of excitatory and inhibitory PSPs generated by inputs with different frequencies can be used to shift the output frequency by an amount fixed by an inhibitory input. Figure 4 shows two HTPEs cross-coupled to form

a stable oscillator. The oscillation frequency can be shifted by applying an inhibitory input at the "SHIFT" synapse. The frequency of the inhibitory input determines the amount by which the oscillation frequency is shifted, as shown in Fig. 5. Frequency shifting, with the limiting case of on/off gating, is the basis for pattern generation in nervous systems [Selverston 88]. Very complex periodic and aperiodic patterns can be generated by coupling oscillators of the type shown in Fig. 4 to single-HTPE phase detectors of the type shown in Fig. 2.

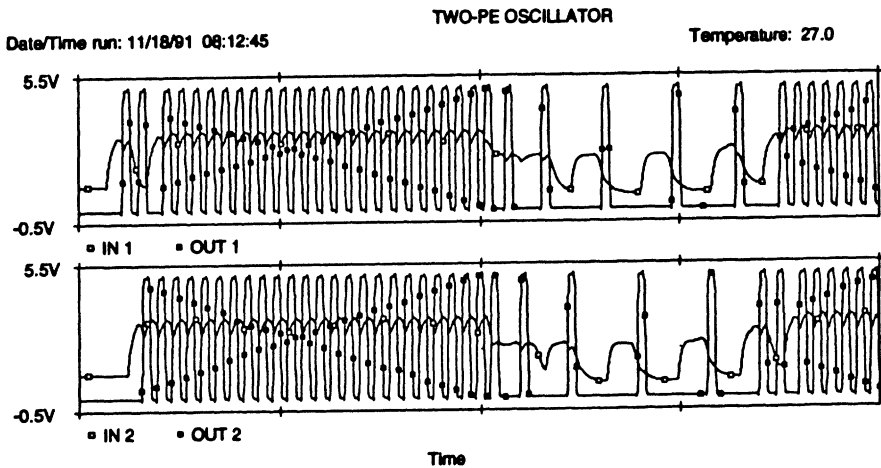


Figure 5. Frequency shifting using the two-HTPE oscillator (Fig. 4). Different shift frequencies correspond to different signal frequencies at the SHIFT input.

Phase shifting requires introduction of a fixed delay without altering frequency. This can be done by replacing the inhibitory synapse in the HTPE shown in Fig. 2 with a shunting synapse, the function of which is to maintain the soma at its resting potential [De Yong 92a]. A shunted HTPE produces an output in response to an excitatory input only if the excitatory PSP is sufficiently broad and strong that it is still above threshold when the shunting

signal decays. Varying the width of the shunting PSP for a fixed excitatory PSP thus effectively varies the delay between the arrival of the excitatory pulse and the generation of the AP, resulting in a pure phase shift [DeYong 92]. Phase shifts allow the simulation with HTPEs of the delaying functions of extended dendritic trees. Coincidence circuits that include phase shifts can be used as delayed coincidence gates, which produce output only when the input signals have particular time separations. Such circuits are basic building blocks for systems capable of detecting and responding selectively to complex temporal patterns.

PULSE-STREAM FILTERING

Filters resolve complex signal streams with many superposed components into simpler patterns. Filtering is used ubiquitously in signal processing applications to remove noise from input signals or to demodulate signals from carrier waves.

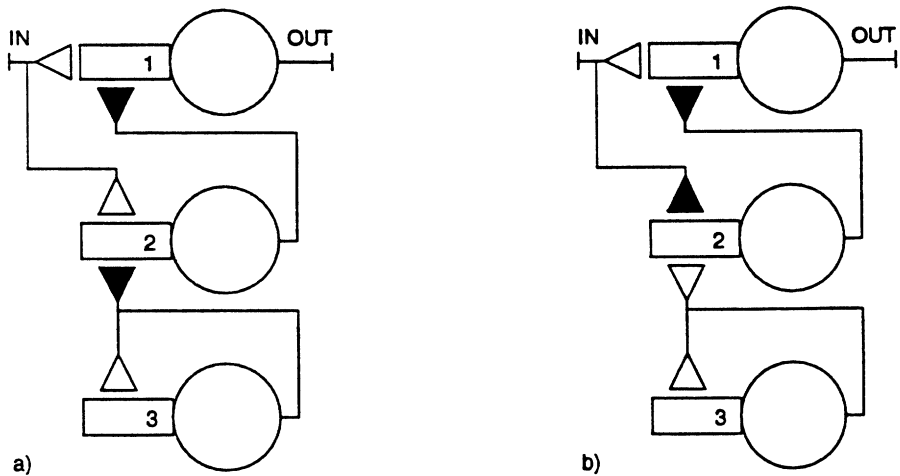


Figure 6. Low (A) and high (B) pass filter systems. Band-pass and band-stop filters are formed by ORing and ANDing the low-pass and high-pass filter outputs, respectively.

In conventional filtering systems the system behavior is characterized by the filter type (e. g. high or low pass) and the critical frequency or frequencies, where the critical frequency signifies the half power (3dB) point of the filter transfer function.

The half-power frequency of the filter and the filter type effectively determine which frequencies are passed and which frequencies are attenuated by the filter, i.e. the pass-band of the filter. If a frequency of interest is within the pass-band of the filter, it should be allowed to pass. If, on the other hand, the frequency of interest is not in the pass-band of the filter, it should be attenuated by at least fifty percent.

HTPE-based systems that perform the four basic filtering operations on input AP streams (low pass, high pass, band pass, and band stop filtering) have already been reported [DeYong 93]. The 3dB point (critical frequency) of these pulse stream filter is taken to be the frequency at which half of the APs are removed from the input stream, thus reducing the total energy contained in the stream by half. This definition of pulse stream attenuation allows pulse stream filtering to be viewed as a form of pulse frequency modulation.

HTPE-level diagrams of low- and high-pass pulse stream filters are shown in Fig. 6. In the low-pass filter (Fig. 6A), an incoming AP stream fires EPSPs on the inputs of both HTPEs 1 and 2. HTPE 1 is biased so as to allow a one-to-one firing ratio between the input and output if the inhibitory synapse on HTPE 1 is inactive. The inhibitory feedback from HTPE 2 to HTPE 1 is inactive if the input frequency is sufficiently low to allow the oscillator formed by HTPE 3 to maintain subthreshold activity on the input of HTPE 2. If the input frequency increases to the point where it begins to overcome the oscillator-driven inhibitory input on HTPE 2, the inhibitory feedback to HTPE 1 will become active, causing a reduction in the input/output firing ratio of HTPE 1. When the 3dB frequency is reached the input/output firing ratio of HTPE 1 will be 2:1. The high-pass filter system of Fig. 6B operates on the same principle, except the roles of the excitatory and inhibitory synapses on the input of HTPE 2 are reversed. These filters can be combined in an OR configuration for a band-pass filter, and in an AND configuration for a band-stop filter [DeYong 93].

PATTERN ENCODING

HTPE filters can be used to transform simple periodic signals with fixed frequency to complex quasiperiodic or aperiodic patterns. Both the quantitative and qualitative characteristics of the output signal from a given filter are sensitively dependent on the difference between the input frequency and the 3db point of the filter. Figure 7 shows the response of an HTPE high-pass filter to an input signal with a frequency equal to the filter's half-power frequency. This filter passes signals at 15.5 MHz at 100%, and attenuates signals at 14.3 MHz to zero. With a 15.152 MHz input, the filter produces a

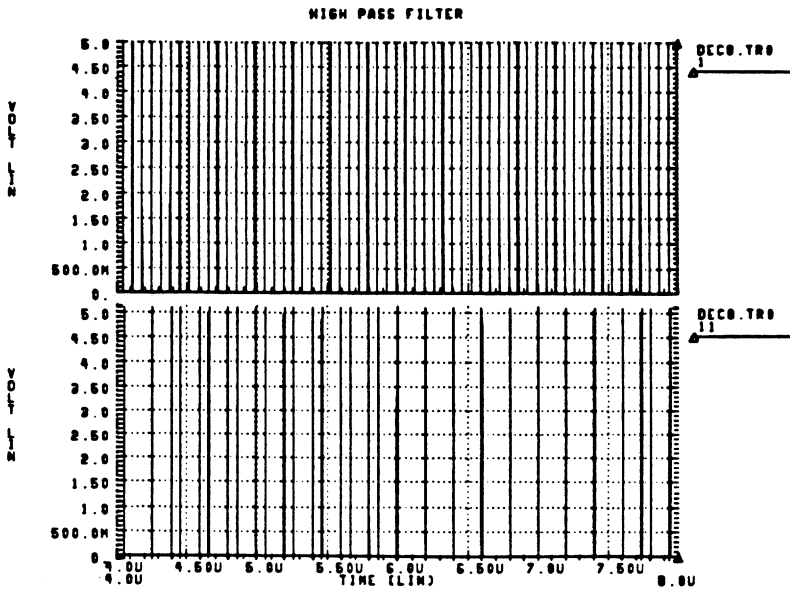


Figure 7. I/O Response of the high-pass filter (Fig. 6B) at an input frequency of 15.152 MHz. This filter passes 15.5 MHz at 100% and 14.3 MHz at 0%. The input pulse stream (top) is allowed to pass to the filter output (bottom) at a ratio of 25 output APs to 50 input APs, 50%.



Figure 8. I/O Response of the high-pass filter at an input frequency of 14.706 MHz. The input pulse stream (top) is allowed to pass to the filter output (bottom) at a ratio of 7 output APs to 25 input APs, 28%.

quasiperiodic signal with 25 output APs for every 50 input APs (50% power). The local frequency of this output signal shifts periodically from the input frequency to $1/3$ the input frequency. As the input frequency is decreased, both the high and low frequency components of the output signal change. At 14.706 MHz (28% power), for example, the filter produces 7 output APs for every 25 input APs, in a stable pattern with components at $1/3$ and $1/5$ the input frequency (Fig. 8).

At power levels for which the average output frequency is not a rational fraction of the input frequency, the HTPE filters produce aperiodic output with multiple local frequency components. While the stability and repeatability of these patterns have yet to be extensively investigated with fabricated hardware, the wide range of behaviors observed in simulations suggests that single HTPE filters may be useful as pattern memories of the sort contemplated by [Selverston 88] that would be addressable by single input frequencies. The possibility of storing complex patterns, combined with robust pattern transformation capabilities, suggests that HTPEs will be applicable to a wide range of signal processing problems.

CONCLUSIONS

The circuits outlined here provide the basic functionality required for a wide variety of temporal signal processing applications, and illustrate the power of hybrid analog-digital computing for solving signal processing problems. None of the circuits shown here require more than 500 μm by 350 μm of chip area (in a CMOS 2 μm process), using custom layout masks that we have already developed [DeYong 92a]. The hybrid processing, custom design, and temporal nature of the HTPE produce efficient high-speed low- hardware solutions to signal processing problems that are unwieldy when addressed by conventional analog and digital methods. A single filter circuit may operate on frequencies from the sub-Hertz range up to the GHz range by simply adjusting the DC biases of the system; no changes in the physical structure of the system are required. These systems exhibit complex behaviors ranging from sensitive pattern detection and filtering operations to addressable complex pattern memory.

References

- [Card 89] Card, H. and W. Moore, VLSI devices and circuits for neural networks, *Int. J. Neural Systems*, 1: 149-165 (1989).

- [DeYong 92] DeYong, M. and C. Fields, Applications of hybrid analog-digital neural networks in signal processing, *Proc. IEEE Int. Symp. on Circuits and Systems*, San Diego, CA. pp. 2212-2215 (1992).
- [DeYong 92a] DeYong, M., R. Findley, and C. Fields, The design, fabrication, and test of a new VLSI hybrid analog-digital neural processing element, *IEEE Transactions on Neural Networks* 3: 363-374 (1992a).
- [DeYong 92b] DeYong, M., T. Eskridge, and C. Fields, Temporal signal processing with high-speed hybrid analog-digital neural networks, *J. Analog Integrated Circuits and Signal Processing* 2: 367-388 (1992b).
- [De Yong 93] DeYong, M., T. Eskridge, and A. Palmer, Complex and emergent behavior from neural-network pulse stream filters, *Proc. IEEE 35th Midwest Symp. on Circuits and Systems*, Washington, DC. In press (1993).
- [Fields 92] Fields, C., M. DeYong, and R. Findley, Computational capabilities of biologically-realistic analog processing elements, *VLSI for Artificial Intelligence and Neural Networks*, J. Delgado-Frias (Ed.), pp. 175-183. New York: Plenum (1992).
- [Foo 89] oo, S., L. Anderson, and Y. Takefuji, Analog components for the VLSI of neural networks, *IEEE Circuits and Devices*, 6 (4): 18-26 (1989).
- [Kandel 85] Kandel, E. and J. Schwartz. Principles of Neural Science, New York: Elsevier (1985).
- [Kuffler 84] Kuffler, S. W., J. G. Nicholls, and R. A. Martin, From Neuron to Brain, 2nd Edition. Cambridge, MA: Sinauer Associates Inc. (1984).
- [Miall 89] Miall, C. The diversity of neuronal properties, *The Computing Neuron*, R. Durbin, C. Miall, and G. Hutchinson (Eds). Wokingham: Addison-Wesley. pp. 11-34 (1989).
- [Mahowald 91] M. Mahowald and R. Douglas, A silicon neuron, *Nature*, 354: 515-518 (1991).

[Selvestom] Selverston, A. A consideration of invertebrate central pattern generators as computational databases, *Neural Networks 1*: 109-117 (1988).

PULSE CODED WINNER-TAKE-ALL NETWORKS

Jack L. Meador and Paul D. Hylander

*School of Electrical Engineering and Computer Science
Washington State University, Pullman WA, 99161-2752*

INTRODUCTION

This chapter introduces a pulse-coded winner-take-all (PWTA) network which employs a unique combination of presynaptic and lateral inhibition that can be efficiently implemented in VLSI. The manner in which the network not only selects the winner but also indicates the weight of the decision made is unique among established winner-take-all networks. A combination of all-or-nothing and graded responses is encoded as a variable rate pulse train appearing only at the output of the winning unit. The mechanism used is closely related to the presynaptic inhibition approach introduced in [Yuille 88] with the exception that it is self-resetting and has properties which make it well suited for electronic realizations using asynchronous pulse-coded circuitry.

The chapter begins with some background on winner-take-all network architecture. The development continues with information coding and processing using asynchronous pulse streams. A functional model of the new PWTA network implementation is then developed. The design of a 2-micron CMOS PWTA circuit based on that model is then presented complete with experimental measurements. The chapter concludes with a discussion of some of the unique features which distinguish this approach from other previously established winner-take-all circuits.

BACKGROUND

The Winner-Take-All Function And Competitive Neural Networks

The winner-take-all (WTA) function plays a central role in competitive neural networks and is related to recurrent on-center off-surround models of natural neural systems [Grossberg 73]. The WTA function can be computed sequentially using numerical comparisons, or in parallel using a Hopfield network [Majani 88] or MOS current conveyors [Lazzaro 88, Andreou 91].

The WTA function is useful because it is an essential component of well established neural networks such as ART, self organizing feature maps, and counterpropagation networks [Zurada 92]. It also finds use in applications such as vector quantization and coding, statistical data clustering, and optimization [Hertz 91].

In simple competitive networks, connection weight vectors (or "prototype vectors") are updated to move toward closely related input vectors. The function of the WTA subnetwork is to determine which of the prototypes is nearest by some distance measure. The inner product measure is used in simple competitive learning while Euclidean distance is employed in the more complex feature map algorithms. The two are equivalent when normalized vectors are used. A simple competitive learning rule based on the inner product distance measure can be expressed concisely in terms of a randomly distributed input vector and a set of prototype vectors associated with unit outputs:

$$Y_i = \begin{cases} 1 & \text{if } W_i^T X > W_j^T X \quad \forall j \neq i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$\Delta W_i = \eta Y_i (X - W_i)$$

where W_i and X correspond to the prototype vector for unit i and the input vector respectively. Here, only unit output Y_i is active, so only row i of W is adapted in proportion to learning rate constant h . In competitive networks the inner product computation and winning unit computation are distributed across two distinct subnetworks as illustrated in the networks of Figure 1. The focus of this paper is upon the implementation of the winning unit computation independent of others associated with distance measures and adaptation.

One characteristic of particular importance to the VLSI implementation of WTA networks is area complexity. WTAs using the Hopfield network organization (Figure 1a) are less practical for VLSI than those using an "inhibitory interneuron" approach (Figure 1b) since winning unit feedback requires a more complex interconnect. The Hopfield net approach has $O(N^2)$

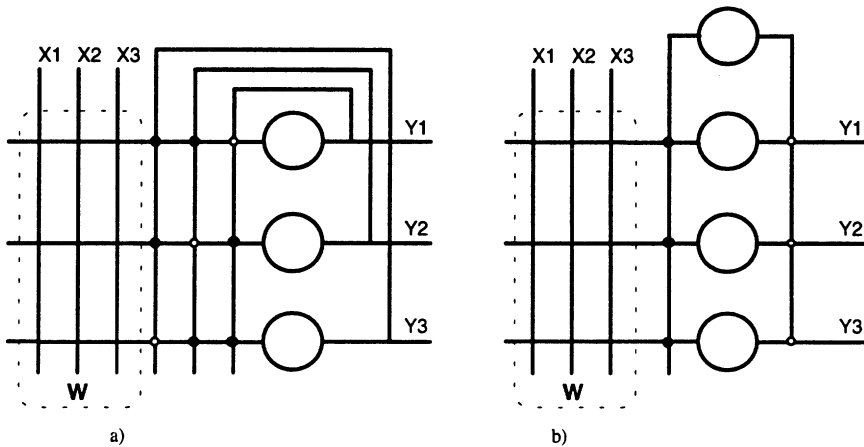


Figure 1. Winner-take-all network organization within a simple competitive network: a) Hopfield network approach and b) the use of a recurrent inhibitory interneuron. Inner product distance computations are computed in the distinct W subnetwork, independent of the WTA implementation. Excitatory connections are indicated by open circles with filled circles indicating inhibition.

area complexity while the use of an inhibitory interneuron requires only $O(N)$ area for N units. The architecture of the WTA network introduced here uses the more space-efficient inhibitory interneuron approach.

Computation In The Pulse Probability Domain

Asynchronous pulse coded processing units similar to the axosomal circuits first described in [Meador 91] are the basic elements of the WTA network to be presented here. The functional organization of a simple axosomal circuit is shown in Figure 2.

The axosomal circuit operates by cyclically charging and discharging the integrating capacitor C at a rate in proportion to the instantaneous magnitude of the synaptic current I_{net} . During the integration phase of circuit operation, S_1 is closed and S_2 is open. In this state weighted synaptic currents are summed via Kirchoff's current law to form the input current I_{net} which is integrated over time on C . With increasing time, the voltage across C increases, eventually reaching the upper threshold of the Schmitt trigger. At this threshold the output of the Schmitt trigger toggles, causing S_1 to open and S_2 to close causing C to discharge through R and S_2 . When the capacitor voltage reaches the lower Schmitt threshold, the circuit reverts to the integration state. The width of the output pulse is dictated by the time-

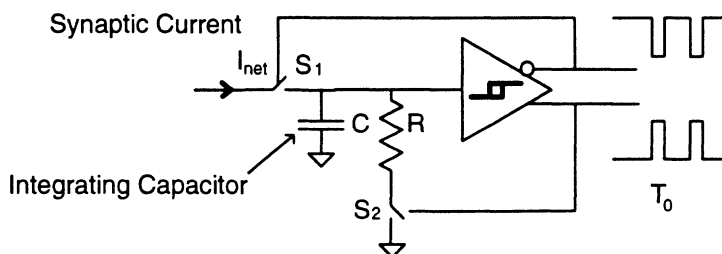


Figure 2. Functional structure of an "axosomal circuit" for asynchronous pulse stream processing. Pulses having a fixed width of T_0 are generated at a rate determined by the magnitude of the synaptic current I_{net} .

constant of the RC circuit, while the rate at which pulses occur depends upon the magnitude of the synaptic current. The axosomal circuit is effectively a current controlled oscillator exhibiting several orders of magnitude of dynamic range and essentially mimics the basic firing cycle of most natural neurons [Guyton 86]. In fact, the term "axosomal" is inspired by nature in the sense that the locus of action potential generation commonly lies in the axon hillock of the neuron soma.

The axosomal circuit of Figure 2 can be used to compute a scaled and weighted summation of pulse probabilities. When used in conjunction with certain pulse stream multipliers or "synapse circuits," the probability that the output pulse stream is generating an action potential or is "firing" is proportional to the weighted summation of the probabilities that input pulse streams are in the action potential state. This is best understood by examining axosomal circuit behavior in conjunction with a simple fixed synapse circuit such as that diagrammed in Figure 3. In the figure, W_{ij} is a voltage which represents the weight of the synapse. This voltage is maintained such that M1 approximates an ideal current source that is gated by transistor M2. Several orders of weight magnitude are available if W_{ij} is maintained in the weak inversion region of M1 [Meador 91]. Since transistor M2 in turn is controlled by the pulse stream input signal X_j , the more frequently (inverted) pulses arrive at X_j , the longer M2 remains on, transferring more charge to the axosomal circuit. Similarly, with increasing weights (decreasing weight voltage in this case), more charge per pulse will flow. If a continuously varying W_{ij} and a fixed X_j having firing probability one (is continuously firing) is presented to the circuit the response will be the integral pulse frequency modulation (IPFM) of W_{ij} [Bayly 69, Gestri 71, Sanderson 80] as illustrated in Figure 4.

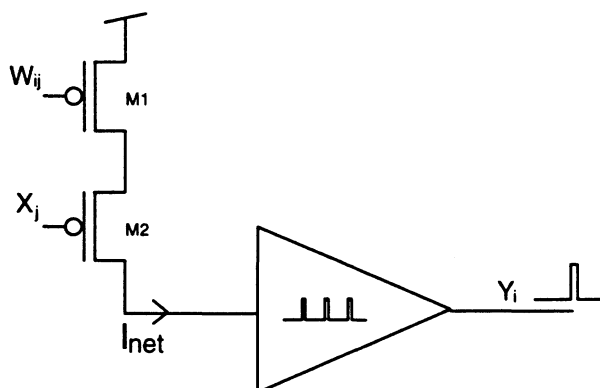


Figure 3. The axosomal circuit of Figure 2 controlled by a simple fixed synapse circuit. The rate at which output pulses are generated varies in direct proportion to with the connection weight (which increases with decreasing voltage W_{ij} in this case) and increasing pulse frequency on X_j .

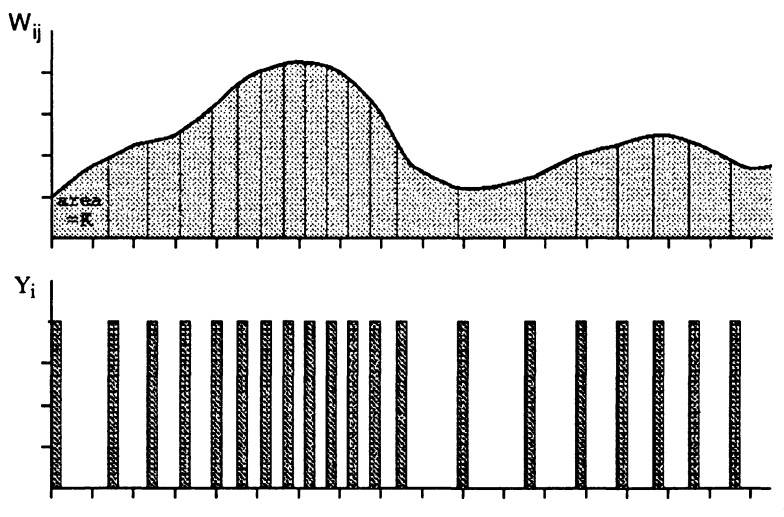


Figure 4. Output pulses generated from a continuously varying weight with a fixed input having firing probability one (continuously firing). The integral of the resulting pulse stream varies in direct proportion to the input signal integral.

IPFM yields a pulse stream having a repetition rate which varies in proportion to the integral of the weight modulating signal. Since each individual output pulse has constant width and magnitude, the integral of the output pulse stream varies in direct proportion to the integral of the input. A new constant area pulse is generated each time the signal integral reaches a multiple of a constant determined by axosomal circuit parameters (equivalent to the area K in Figure 4). Given this and also that the integral of a binary pulse stream per unit time is equivalent to an estimate of pulse probability, it is a simple matter to demonstrate that the axosomal circuit can be used to scale pulse stream firing probabilities, as for example is shown in Figure 5.

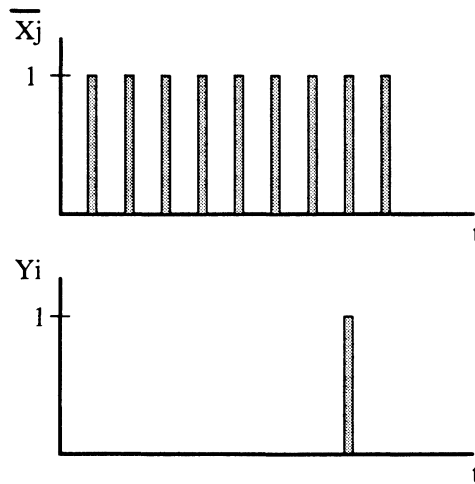


Figure 5. Pulse stream firing probability scaling with binary pulse stream input X_j . W_{ij} is fixed and a variable rate pulse stream applied to X_j . The output firing probability varies in direct proportion with that of the input.

In this figure, the connection weight W_{ij} is assumed fixed and a variable rate pulse train is applied to X_j . Firing probability scaling can be succinctly expressed as:

$$\Pr(Y_i = 1) = K^{-1} w_{ij} \Pr(X_j = 1) \quad (2)$$

where the firing probabilities are sampled over identical time intervals independent of circuit state. K depends upon C and the Schmitt threshold voltages. It is proportional to the quantity of charge required to raise C 's terminal voltage to the firing threshold voltage.

The extension of this result to a weighted summation of several inputs is straightforward. Parallel synapse circuits yield a summation of individual

weighted inputs via Kirchoff's current law. The firing probability of the result is simply a scaled, weighted sum of input pulse probabilities:

$$\Pr(Y_i = 1) = K^{-1} \sum_j w_{ij} \Pr(X_j = 1) \quad (3)$$

This result is presented graphically for two pulse stream inputs in Figure 6. It is important to note that this result proceeds asynchronously independent of signal timing. This means that under certain conditions a relative phase shift between the two signals will yield only a minor variation in the overall result. Any deviation from the ideal sum due to a phase shift is bounded by the accuracy limits established by pulse width and the signal observation interval.

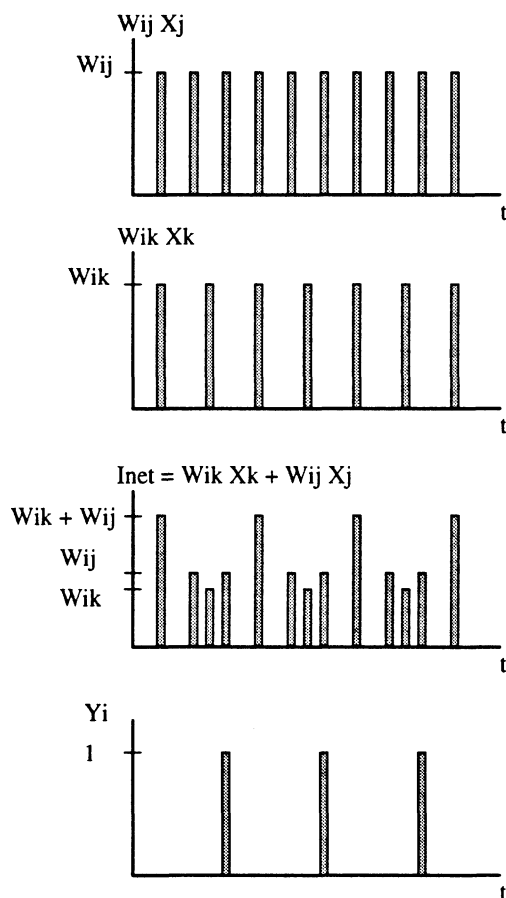


Figure 6. Weighted summation of pulse stream firing probability for two inputs X_j and X_k .

Although the traditional sigmoidal output function is not explicitly represented in Equation (3), it is implied by the probabilistic signal representation. Just as probabilities are bound by 0 and 1, the response of an asynchronous pulse coded neuron circuit consisting of the axosomal and synapse circuits presented here saturates at zero and some maximum firing rate determined by the pulse width T_0 . A small T_0 yields a large dynamic range for pulse repetition rates, increasing the accuracy of the signal representation for constant observation intervals.

Pulse Coding in VLSI Implementation

There are several advantages in the use of pulses as a VLSI communication and computation medium. In general, pulses are very easy to buffer and can communicate continuous analog information off-chip. It is easier to multiplex pulse stream signals for improving the efficiency of both on chip and off chip communication [Murray 91, Mahowald 92]. Furthermore, asynchronous pulse coded neuron circuits have a natural "power down" property where very little power is consumed when no input data is being processed.

Another unique property of pulse coded signals is that both digital and analog information is effectively communicated simultaneously. The "all or nothing" response traditionally considered to be an important feature of natural neural processing is complemented by the ability to express continuous values having a dynamic range limited only by the pulse width and observation interval. One might argue that this property helps explain the flexibility exhibited by neurons in the wide range of specialized functions they perform in natural neural systems. As will be seen later, this property lends a useful feature to pulse-coded WTA networks: that of being able to communicate the "certainty" with which a particular winning decision is made.

AN ASYNCHRONOUS PULSE-CODED WTA

A winner-take-all network that extends the presynaptic inhibition feedback mechanism first described by Yuille and Grzywacz [Yuille 88] is presented in this section. The pulse-coded WTA (PWTA) network introduced here employs a unique combination of presynaptic and lateral inhibition to yield a self-resetting network that preserves winning input strength information and is robust with finite precision distributed computation. The manner in which this network not only selects the winner, but also indicates decision "weight" by the strength of the winning unit response is unique among established WTA algorithms. A combination of all-or-nothing and graded responses is encoded as a variable rate pulse train appearing only at the

output of the winning unit. This section introduces a functional model of the network and analyzes the parametric conditions under which an ideal WTA function is computed.

Model

A PWTA network combines pulse-coded unit dynamics with an inhibitory recurrent interconnect. The activation of a pulse-coded unit (corresponding to the terminal voltage of the capacitor in the axosomal circuit of Figure 3) can be modeled by the nonlinear differential equation:

$$\kappa \dot{v} = -\alpha v + \text{net} - (\beta v - \gamma + \text{net})g(v) \quad (4)$$

where $0 < v(0) \leq V_{tl}$ and $g(\bullet)$ is a binary hysteresis function having a threshold of V_{th} and a threshold of V_{tl} . Provided $\gamma/(\alpha + \beta) < V_{tl}$, v_i will oscillate between V_{tl} and V_{th} . Simultaneously, $g(v_i)$ describes a pulse train having a repetition rate which varies sigmoidally with input net current.

PWTA network dynamics can be expressed in terms of (4) with additional global inhibition from a shared inhibitory interneuron:

$$\kappa \dot{v}_i = -\alpha v_i + \text{net}_i - ((\beta - \lambda)v_i - \gamma + \zeta)g(v_i) - (\lambda v_i - \zeta + \text{net}_i)G(\mathbf{V}) \quad (5)$$

where

$$G(\mathbf{V}) = \bigvee_k g(v_k)$$

indicates a logical OR computed by the interneuron and \mathbf{V} corresponds to the vector of unit activations, $[v_1, \dots, v_n]$. Figure 7 illustrates the network architecture. $G(\mathbf{V})$ is global inhibition which occurs when *any* unit pulses, causing all activations to be inhibited toward equilibria specified by network parameters β , λ , γ and ζ . Ideally, the winning output is indicated by the dominance of the first unit having activation that reaches V_{th} . Since the winning unit establishes the synchronized re-initialization of all others, it is the only one to fire. In general, the winning unit is determined by a combination of network reset state and input magnitude. With appropriate parameter values, the reset state establishes initial conditions at the beginning of each firing cycle that make the winning decision dependent exclusively upon the I_{net} inputs.

Unit activations in a PWTA network have three important equilibria: one associated with an *integration phase* of operation and two others with a *firing phase*. Combinations of these equilibria establish cyclic attractors in the network activation space. The activation trajectory for a unit is determined

exclusively by one target attractor at a time, as established by the values $G(V)$ and $g(v_i)$ (Table 1). During the integration phase all units compete in a race toward the firing threshold V_{th} . The first unit to reach that threshold triggers the firing phase. During this second phase, $(\alpha+\beta)/\kappa$ determines the decay rate and $\gamma/(\alpha+\beta)$ the asymptote of the winning unit activation. $(\alpha+\lambda)/\kappa$ and $\zeta/(\alpha+\lambda)$ determine the respective quantities for losing units.

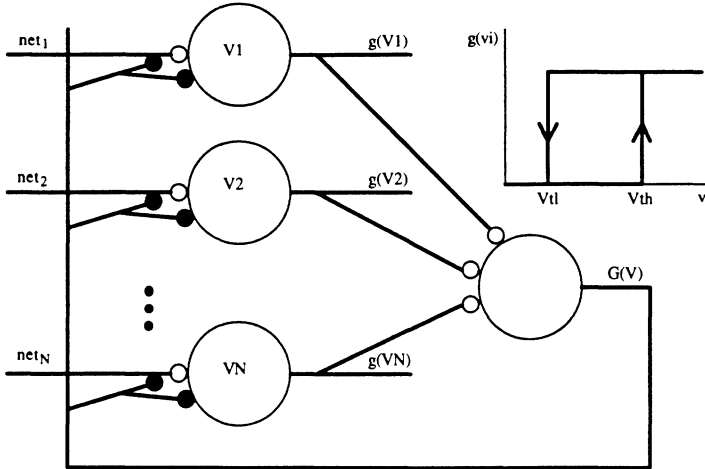


Figure 7. An asynchronous-pulse-coded winner-take-all network architecture. A single-layer with a recurrent inhibitory interneuron yields a linear interconnect organization.

$G(V)$	$g(v)$	$\kappa \dot{v}_i$	Action
0	0	$-\alpha v_i + net_i$	integration phase
1	0	$-(\alpha + \lambda) v_i + \zeta$	firing phase, unit i loses
1	1	$-(\alpha + \beta) v_i + \gamma$	firing phase, unit i wins

Table 1. PWTA processing unit operation is determined by a combination of global inhibition from the recurrent interconnect ($G(V)$) and local unit output ($g(v_i)$).

Winner Determination

The PWTA integration phase corresponds to a specific case of the presynaptic inhibition WTA network [Yuille 88]. Equation (5) can be

rewritten in a form which separates presynaptic inhibition and reset terms:

$$\kappa \dot{v}_i = -\alpha v_i + \text{net}_i (1 - \prod_{j \neq i} g(v_j)) - ((\beta - \lambda)v_i - \gamma + \zeta + \text{net}_i)g(v_i) - (\lambda v_i - \zeta)G(V) \quad (6)$$

where the third and fourth terms realize a reset function that will be discussed in the following section. When considered alone with appropriate choices for V_{th} and V_{tl} , the first two terms realize a presynaptic WTA:

$$\kappa \dot{v}_i = -\alpha v_i + \text{net}_i K_i(V) \quad (7)$$

where

$$K_i(V) = 1 - \prod_{j \neq i} g(v_j)$$

Proof of presynaptic WTA function for this specific K_i requires that it be established that a winner exists, is unique, and directly corresponds to the largest input.

A winner exists provided $\text{net}_w > \alpha V_{th}$ where net_w is the winning input value. Given the premise that a winner *never* exists, then $g(v_i(t)) = 0 \forall (i, t)$. This implies that $K_i(V) = 1, \forall (i, t)$. By Equation (7) then, $v_i \rightarrow \text{net}_i / \alpha$ for all units as $t \rightarrow \infty$. Since $\text{net}_w > \alpha V_{th}$, then $v_w > V_{th}$ meaning that $g(v_w) = 1$, thereby contradicting the negative premise. Therefore, there will always exist a winner at some point in time.

A winner is unique provided $V_{tl} > 0$. Given $g(v_w) = 1$ it follows that $K_i(V) = 0 \forall i \neq w$, which in turn implies that for all units save the winner $v_i \rightarrow 0$; so $g(v_i) \rightarrow 0$ when $v_i < V_{tl}$. Thus only that unit having activation $v_i = V_{th}$ has a non zero output.

The winning unit corresponds to the largest input provided $v_i(0) \leq v_w(0)$. An expression of the difference of unit differential activation can be obtained from Equation (7):

$$\kappa \frac{d}{dt}(v_w - v_i) = -\alpha(v_w - v_i) + (\text{net}_w - \text{net}_i) \quad (8)$$

This expression assumes that $v_i < V_{th} \forall i$, so a winner has yet to be established and $K_i(V) = 1$ for all units. If $v_i(t) < v_w(t)$ for any t , then the only way a unit j associated with a losing input net_j can win is if at some point $v_i(t) = v_j(t)$. Under such circumstances, activation order is determined by:

$$\kappa \frac{d}{dt}(v_w - v_i) = (\text{net}_w - \text{net}_i) \quad (9)$$

The only way order will be exchanged is if this expression is negative. But that contradicts what is implied by input order, so only the largest input can

lead to a winning activation when the two activations are equal. Thus the winning unit corresponds to the largest input provided $v_i(0) < v_w(0)$. This also obviously holds for the trivial case where initial states are identical, $v_i(0) = v_w(0)$.

Clearly, initial conditions are possible where $v_i(0) > v_w(0)$ even though $I_i < I_w$ in which a losing input will yield a winning output. If, for example, $v_i(0)$ is large enough such that $v_i(t)$ can reach V_{th} before winner $v_w(t)$ can "catch up" then input order will not directly determine the winner. $v_i(0) > v_w(0)$ does not necessarily imply erroneous winner determination though, since there also exists the possibility that I_w is large enough to compensate for unequal initial bias against v_w .

Network Initialization

The PWTA firing phase controlled by the latter terms of (6) provides a cyclic reset mechanism for establishing initial conditions. Although a global signal indicates the end of an integration phase, local parameters determine unit initial conditions for the next integration phase. The network automatically resets at a rate dictated by the largest input signal, in a sense implementing a kind of automatic input gain control. It shall now be shown that this reset mechanism functions independently of initial network state, therefore insuring the correct indication of the winning input.

A PWTA correctly indicates the winning input independent of initial network conditions. It does so by converging to a cyclic attractor in which only that unit associated with the largest input fires. That cycle is entered by the end of the second integration phase provided $\gamma/(\alpha+\beta) < V_{th}$, $\lambda \gg \beta$, and $\zeta/(\alpha+\lambda) = V_{th}$.

If initial network conditions are such that $v_i(0) \leq v_w(0) \forall i$, then the result follows directly from the simple argument of the previous section that the state $v_i(t) = v_w(t)$ never occurs so the winner is correctly indicated at the end of the first integration phase.

If the initial conditions are such that $\exists i \mid v_i(0) > v_w(0)$, then the first unit to fire does not necessarily correspond to the true winning input, so it can only be considered a "hypothesized" winner. Since $\gamma/(\alpha+\beta) < V_{th}$, the hypothesized winning unit activation will follow a *truncated* exponential trajectory during the first firing phase:

$$v_w(t) = \begin{cases} (V_{th} - \gamma/(\alpha+\beta)) e^{-(\alpha+\beta)t/\tau} + \gamma/(\alpha+\beta) & \text{if } v_w > V_{th} \\ V_{th} & \text{otherwise} \end{cases} \quad (10)$$

until $v_w(t) = V_{th}$ whereupon $v_w(0) = V_{th}$ is established for the subsequent integration phase at the beginning of the second cycle. Since $\zeta/(\alpha+\lambda) = V_{th}$ all

losing unit activations *asymptotically* converge to V_{th} following the trajectory:

$$v_i(t) = (v_i^* - V_{th}) e^{-(\alpha+\lambda)t/\kappa} + V_{th} \quad (11)$$

where v_i^* is unit i activation when the network firing phase is entered. This establishes $v_i(0) = V_{th} + \epsilon$ at the end of the first firing phase. Therefore, $v_i(0) = v_w(0) + \epsilon \forall i$ at the beginning of the second integration phase. By the properties of exponential decay, ϵ can be made arbitrarily small by increasing λ and causing v_i to more rapidly converge. $\lambda \gg \beta$ implies that ϵ is very small, so the state $v_i(t) = v_w(t)$ occurs very early in the second integration phase. It is theoretically possible to choose λ arbitrarily large relative to β without ϵ ever reaching zero. In practice, λ can be made large enough for ϵ to fall below computational precision limits, as determined by the digital word length or the analog noise floor. Thus in practice, a ratio of λ/β can be found which virtually establishes the condition $v_i(0) = v_w(0)$. Thus unit activations become properly ordered according to (9) before entering the second firing phase. Therefore, the second unit to fire will be that associated with the largest input. This winning cycle will repeat until actual input order changes.

EXPERIMENTAL PWTA IMPLEMENTATION

This section presents a CMOS implementation of a PWTA circuit along with experimental measurements verifying its functionality. A simple variation of the axosomal circuit of Figure 2 is used in combination with 3-bit digitally programmable synapse circuits in a simple competitive neural network. Test results show that the network correctly matches 4-element input vectors with their stored prototypes.

CMOS Circuits

Figure 8 shows a block diagram of the experimental system. The Pulse coded inputs X_0 through X_3 are weighted by the programmable synapse circuit array and distributed to the axosomal circuits. The outputs (Y_0 through Y_3) control a global inhibition generator, I . The global inhibition signal feeds back to all the neurons in the network using an $O(n)$ architecture like that of Figure 1 a).

The programmable synapse circuits are simple 3-bit MDACS controlled by serial-in-parallel-out registers (Figure 9). These registers are chained together in a bit-serial fashion for simplified interfacing to host hardware. Although this simple weight I/O organization would not likely be used in more complex systems requiring synapse adaptation, it does provide a straightforward method for examining the functionality of experimental PWTA circuits. Input signal X_i gates supply current to $M_2 - M_4$ via M_1 . V_{ref} is

selected for subthreshold operation with transistor aspect ratios increasing in powers of two for binary weighting. Dynamic SIPO register **R** controls which

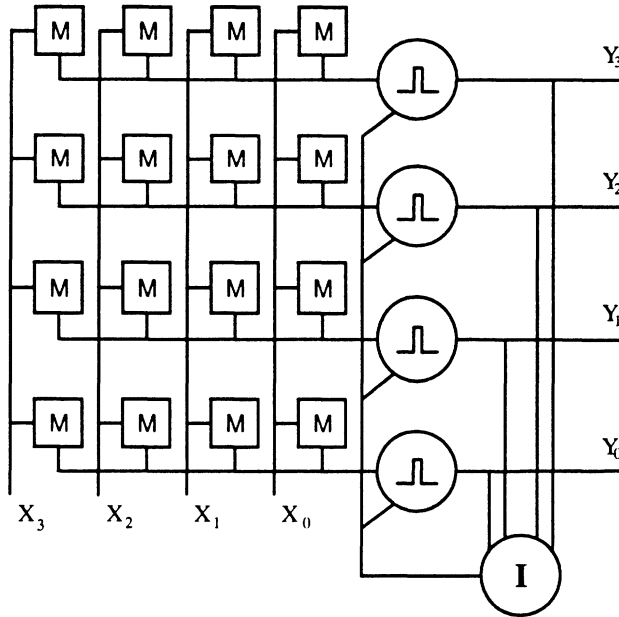


Figure 8. Pulse coded winner-take-all system organization.

branches are active to establish eight synaptic weight levels. A layout organization consisting of mirrored pairs of the circuit has dimensions 318 x 198 microns.

In the axosomal circuit shown in Figure 10, M1 and M2 correspond to S1 and S2 and M3-M8 form the Schmitt trigger of Figure 2. In this implementation, the integration capacitance primarily consists of the gate-bulk capacitance of transistors M4-M7. With the aspect ratios given, the threshold voltages of the Schmitt trigger are 1.5 V and 3.0 V for V_{t1} and V_{t2} respectively. M11 forms the local branch of the global inhibition generator. Each axosomal circuit layout has dimensions of 186 x 48 microns.

The global inhibition signal **I** controls the switching between integration and firing phases of the axosomal circuits. Figure 11 shows how the transistors corresponding to M11 from network axosomal circuits connect together to compute the OR global inhibition function. When the output of one neuron fires, the **I** inputs of all axosomal circuits will be forced high. This will in turn cause all of them to enter the network firing phase, simultaneously discharging all the integrating capacitors and effecting their simultaneous inhibition.

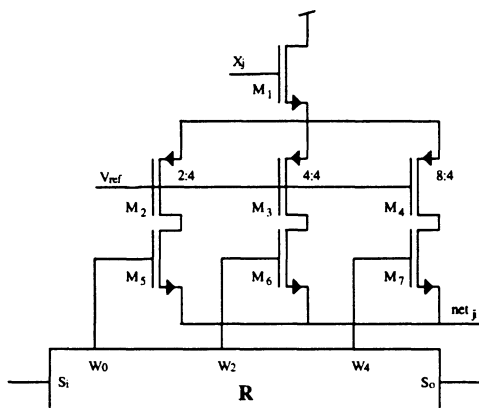


Figure 9. The programmable synapse circuit consists of a 3-bit programmable MDAC. Eight synaptic weight levels are available for testing PWTA response. Minimum size devices are used unless noted otherwise in the diagram.

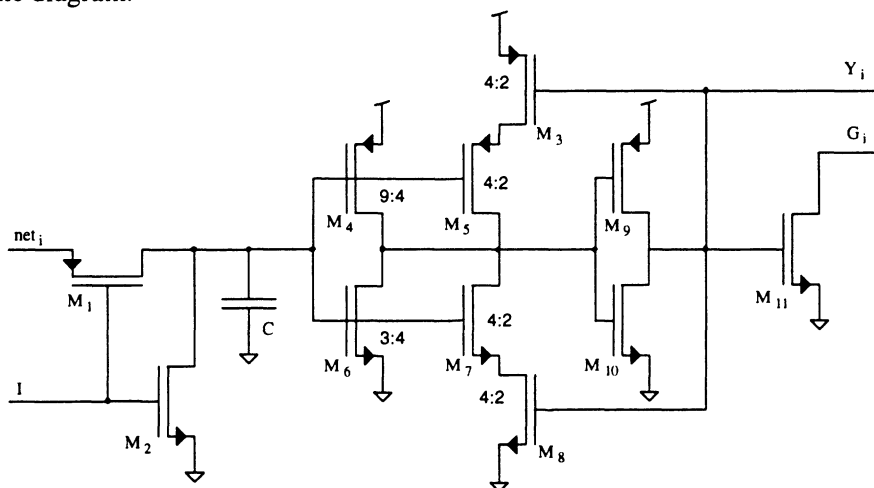


Figure 10. CMOS implementation of the axosomal circuit in Figure 2.

Network Initialization

As detailed earlier, certain initialization conditions must be adhered to for a PWTA network to correctly indicate a winning input. Specifically, for the ideal model previously discussed (equation (5)), system parameters should be selected such that $\gamma/(\alpha+\beta) < V_{th}$, $\lambda \gg \beta$, and $\zeta/(\alpha+\lambda) = V_{th}$. In practical implementation terms, these constraints mean that the activation of the winning unit (voltage across the integration capacitor) should decay toward

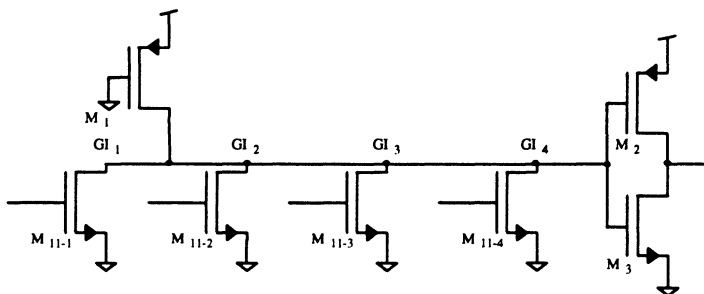


Figure 11. Global inhibition circuitry. One weak pullup transistor and an inverter are needed for the network with transistor M11 is repeated for each axosomal circuit.

some value less than V_u while the activation of all losing units should decay toward V_u precisely. Furthermore, the losing unit decay rate should exceed that of the winning unit so that it has converged upon V_u to a reasonable degree of accuracy when the network exits the firing phase. Meeting these conditions in the ideal case guarantees that transition boundaries between winning units can be made arbitrarily precise.

In actual circuits however, a number of error sources conspire against this ideal goal. Variation in threshold voltages, time delays, and discharge rates between fabricated axosomal circuits means that erroneous decisions can be made in the region of ideal transition boundaries. Functional simulations have previously shown that two nonideal effects, namely vascillation and hysteresis can be observed in transition regions where two inputs are too close to resolve [Meador 92]. Absolute precision is an unreasonable design goal for obvious reasons, so an alternate design criterion must be developed.

In the case of the network presented here, consistent behavior within transition regions is considered to be more important than absolute precision. Specifically, the network reported here has been designed to consistently exhibit hysteresis within indeterminate transition regions. This behavior occurs rather naturally with a relatively simple circuit since both winning and losing units approach ground simultaneously, with the losers by definition starting at a lower activation level than the winner. When the winning activation reaches V_u the losing activations all are significantly smaller since the decay rates are large. Thus the current winner will begin the following integration phase with a higher activation, and will continue to win until some other input reaches a significantly larger value.

IC Test Results

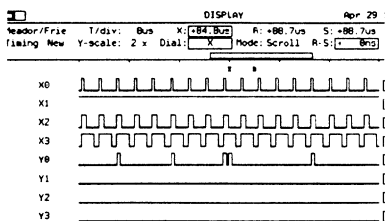
A MOSIS 2-micron CMOS Tiny Chip was used to test an experimental PWTA implementation. The resulting measurements were obtained using a Tektronix DAS9200 / LV500 digital tester in combination with a Phillips PM3580 logic analyzer. The tester was programmed to deliver variable duty cycle pulse trains to each of the four system inputs while the logic analyzer recorded system responses. The network was first programmed bit-serially with the four prototype vectors [1 7 3 5], [7 1 3 5], [1 7 5 3], and [1 3 5 7] respectively. The vectors were chosen to have equal magnitude so that true Euclidean distance would form the basis for all comparisons. Pulse trains were applied to the network inputs having duty cycles corresponding to these four vectors and the system responses recorded (Figure 12). In the figure, it can be seen that the unit programmed with the matching prototype vector is the only one to be activated. Another observation made is that although each winning unit fires with about the same probability for the same winning input magnitude (about 3-4 pulses per 90 uS), the firing is irregular - a consistent, fixed firing period is not observed. This result most likely arises from parasitic capacitance in the synaptic summing junction on the axosomal input. Charge continues to accumulate in the summing junction during the network firing phase, independent of the integrating capacitor within the axosomal circuit. At the beginning of the next network integration phase, the accumulated charge is redistributed onto the integration capacitor, resulting in an apparent instantaneous increase in net input current. Since all units re-enter the integration phase simultaneously and the accumulated charge will be greater for the winner than all other units, this will not affect the winning outcome unless the parasitic capacitance is somewhat larger than the axosomal integrating capacitance. One reasonable way to avoid such error is to simply use a larger axosomal integrating capacitance. A second, more scalable approach would be to distribute the switches S1 and S2 (Figure 2) throughout the synaptic array, thereby maintaining a constant ratio between parasitic and integrating capacitors.

DISCUSSION AND CONCLUSION

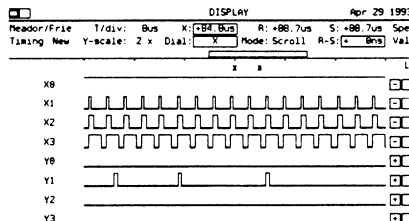
The PWTA system presented in this chapter has several interesting features which distinguish it from other previous WTA implementations. One of those is the ability to indicate the strength of input data upon which a decision is based. Saturating WTA algorithms such as those described in [Majani 88] effectively filter out such information. In the presynaptic inhibition algorithm [Yuille 88] the rate at which system state changes is directly related to input signal strengths. The time required to arrive at a

decision can be considered an indicator of the importance or "weight" carried by the inputs. A larger input signal is interpreted as "significant" so a decision is made rapidly. If all inputs are small, more time is taken to gather evidence before a final decision is made. The PWTA algorithm inherits this property from its presynaptic predecessor, but instead indicates input signal strength by winning unit firing rate.

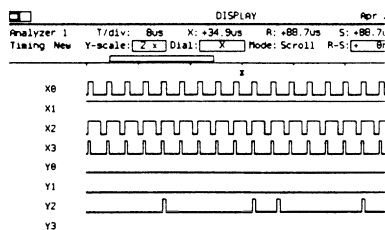
$X=[1\ 7\ 3\ 5]$, Y_0 activated



$X=[7\ 1\ 3\ 5]$, Y_1 activated



$X=[1\ 7\ 5\ 3]$, Y_2 activated



$X=[1\ 3\ 5\ 7]$, Y_3 activated

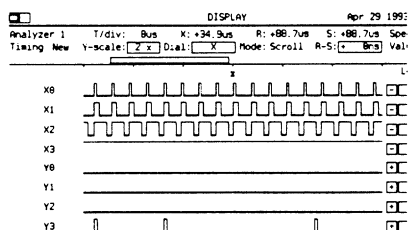


Figure 12. Fabricated IC response to matching inputs. In each example the input vector X matches one of the four stored prototype vectors. In each case, the unit associated with the stored prototype is the only one to fire.

The PWTA algorithm also inherits a useful input averaging property from the presynaptic inhibition algorithm. Unit dynamics are such that activation increases monotonically with a filtered average of the input signal. In electronic implementation terms, this makes it possible to directly process the kind of pulsed input signals employed in asynchronous-pulse-coded ICs [Meador 91, DeYong 92, Hamilton 92, Moon 92, Watola 92]. Although a combination of the original $O(N)$ WTA circuit [Lazzaro 1988] with Mead's self-resetting neuron circuit [Mead 1989] would yield a system that generates pulses, it would not properly process input pulse trains since the current-mode WTA expects data represented as continuous input currents.

One interesting property of this network is that most power is dissipated

only when pulses are generated. With low input currents (low output firing rates), most of the power is dissipated as V_c approaches V_{th} during the transition into the firing phase. Less power is dissipated during the firing phase since much less time is spent resetting the network than integrating input signals. At higher input currents, the average dissipation of the axosomal circuits increases with the winning unit firing rate. At zero input current, the inherent charge leakage of MOSFET source/drain connections will cause V_c to tend away from V_{th} . As a result, the network dissipates very little power when inputs become quiescent, yet responds instantaneously when nonzero inputs become available, dynamically adjusting power requirements to suit the input processing needs. This differs significantly from the self-resetting neuron circuits originally described by Mead [Mead 1989] where continuous power dissipation results from operating a fixed-threshold inverter in its high gain region (see Chapter 8 of this volume for a novel low power variation of that approach).

In conclusion, the pulse coded winner take all system which has been presented in this chapter not only possesses useful physical properties that are well-suited for VLSI implementation, but also novel functional properties. Low-order system layout complexity, compact cell implementations, and dynamic power dissipation combine with the ability to encode decision strength to yield a flexible and efficient implementation of an important neural network function.

ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of E. Frie and P. Bergmann who assisted with IC design, layout, and test. Thanks also to D. Watola for his many contributions and T. Heldt for CAD software and test system support.

References

- [Andreou 91] Andreou, A.G., Boahen, K.A., Pouliquen, P.O., Pavasovic, A., Jenkins, R.E. and Strohhahn, K., "Current-Mode Subthreshold MOS Circuits for Analog VLSI Neural Systems," *IEEE Trans. on Neural Networks*, V.2, 215-213, 1991.
- [Bayly 69] Bayly, E., "Spectral Analysis of Pulse Frequency Modulation in the Nervous System," *IEEE Transactions on Bio-Medical Engineering*, V.15, pp. 257-265, 1969.

- [DeYong 92] DeYong, M.R., Findley, R.L. and Fields, C., "The Design, Fabrication, and Test of a New VLSI Hybrid Analog-Digital Neural Processing Element," *IEEE Trans. on Neural Networks*, V.3, 363-374, 1992.
- [Gestri 71] Gestri, G. "Pulse Frequency Modulation in Neural Systems, a Random Model," *Biophysics Journal*, V.11, pp. 98-109, 1971.
- [Grossberg 73] Grossberg, S., "Contour enhancement, short term memory, and constancies in reverberating neural networks," *Studies in Applied Mathematics*, V.12, 213-257, 1973.
- [Guyton 86] Guyton, A.C., *Textbook of Medical Physiology*, 7th ed., Saunders, Philadelphia, 1986.
- [Hamilton 92] Hamilton, A., Murray, A.F., Baxter, D.J., Churcher, S., Reekie, H.M., and Tarassenko, L., "Integrated Pulse Stream Neural Networks: Results, Issues, and Pointers," *IEEE Trans. on Neural Networks*, V.3, 385-393, 1992.
- [Lazzaro 88] Lazzaro, J., Ryckebusch, S., Mahowald, M.A., and Mead, C.A., "Winner take all networks of $O(n)$ complexity," *Advances in Neural Information Processing Systems 1*, 703-711. Morgan Kaufmann, 1988.
- [Mahowald 92] Mahowald, M.A. "Evolving Analog VLSI Neurons," *Single Neuron Computation*, 413-435. Academic Press, 1992.
- [Majani 88] Majani, E., Erlanson, R., and Abu-Mostafa, Y., "On the K-winners-take-all network," *Advances in Neural Information Processing Systems 1*, 635-642. Morgan Kaufmann, 1988.
- [Murray 91] Murray, A.F., Del Corso, D. and Tarassenko, L., "Pulse-Stream VLSI Neural Networks Mixing Analog and Digital Techniques," *IEEE Trans. Neural Networks*, 193-204, 1991.
- [Mead 89] Mead, C.A., *Analog VLSI and Neural Systems*, 193-204. Addison-Wesley, 1989.

- [Meador 91] Meador, J.L., Wu, A., Cole, C., Nintunze, N., and Chintrakulchai, P., "Programmable Impulse Neural Circuits," *IEEE Trans. on Neural Networks*, V.2, 101-109, 1991.
- [Meador 92] Meador, J., "Finite precision effects on dynamic behavior in a pulse-coded winner-take-all mechanism," *Proc. Int. J. Conf. on Neural Networks*, V.III, 432-437, 1992.
- [Moon 92] Moon, G., Zaghoul, M.E., and Newcomb, R.W., "VLSI Implementation of Synaptic Weighting and Summing in Pulse-Coded Neural-Type Cells," *IEEE Trans. on Neural Networks*, V.3, 394-403, 1992.
- [Sanderson 80] Sanderson, A., "Input-Output Analysis of an IPFM Neural Model: Effects of Spike Regularity and Record Length," *IEEE Transactions on Biomedical Engineering*, V.27, pp. 120-131, 1980.
- [Watola 92] Watola, D. and J. Meador, J., "Competitive Learning in Asynchronous-Pulse-Density Integrated Circuits," *Analog Integrated Circuits and Signal Processing*, V.2, 61-82, 1992.
- [Yuille 89] Yuille, A.L., and Grzywacz N., "A Winner-Take-All Mechanism Based on Presynaptic Inhibition Feedback," *Neural Computation* 1, 335-347, 1989.
- [Zurada 92] Zurada, J. M., *Artificial Neural Systems*, Webb Publishing, 1992.

REALIZATION OF BOOLEAN FUNCTIONS USING A PULSE CODED NEURON

Marc de Savigny and Robert W. Newcomb

*Electrical Engineering Department
University of Maryland, College Park MD, 20742*

ABSTRACT

It is proven here that all of the sixteen Boolean functions of two variables can be realized by a pulse coded neuron, thus guaranteeing that present day digital computers could be implemented using pulse-coded neural networks. The result uses an adjustable pulse coded neuron that is realized with a neural-type cell in each of the two input branches and one in the output branch. These neural type cells allow for the adjustment of pulse repetition frequencies to obtain the appropriate coding for each function by the adjustment of internal weights. Suitable values for the weights are given in Table 2. The strictly greater than function is used to explain the weight choices along with SPICE simulation results for the exclusive Or.

INTRODUCTION

One of the frequent questions posed of researchers in other areas is "what are the capabilities of pulse-coded neural networks?" In particular "are they as good as present day digital computers?" Since the logic of present day digital computers is primarily based upon Boolean functions, as one step in starting to answer these questions is to answer the question "can pulse-coded neural

networks realize all of the Boolean functions?" Here we show that indeed they can by presenting a pulse-coded neuron that realizes all the Boolean functions of two variables. Since we wish to consider digital logic functions [Habib 89], we concentrate on digital data coded on analog signals analogous to biological neurons processing analog data. And although we only consider two inputs to the neuron, extensions to more inputs are readily accomplished.

However, the basic motivation for a pulse coded neuron is the work of neurobiologists who have found that biological neurons pulse code the information they process [Ganong 85, p. 94]. Since it may be desired to have a neuron that mimics the pulse coded processing of the brain [Hartline 89], it has appeared reasonable to try to match a model neuron as closely as possible to what we know by way of a description of pulse handling in biological neurons [Cole 89, Murray 88]. As for engineering importance, it further appears that pulse coding can achieve a better noise-immunity when used for present day artificial neural network computation [Meador 91, Tomberg 90]. This could be important in some environments, or when the accuracy or reliability of a neural network is capital.

The main contributions of this chapter include: the presentation of a pulse coded neuron; showing how to obtain the sixteen Boolean functions of two variables with the same neuron via weights; and simulation of the neuron. As such this paper extends the results reported in [Savigny 92].

DESCRIPTION OF NEURON

Our neuron, shown in Figure 1, uses pulse coding of the information [Murray 89] to implement Boolean functions. We are interested here in implementing Boolean functions which requires an effective coding of the two logical levels "0" & "1". We will code by a neural-type pulse at repetition rate r_1 when the information is a logic "1" and repetition rate r_0 , chosen to be zero, that is, a DC constant, when the information is a logic "0". In Figure 1 the inputs In_1 & In_2 can be these pulse coded signals, or, if needed, TTL-compatible logic levels. Next, we give a short description of how the neuron of Figure 1 works.

General principle

There are two inputs In_1 & In_2 with the signal on each branch being routed through three components: a pulse inverter, a Neural Type Cell (NTC) [El-Leithy 88], and a lowpass filter. The pulse inverter, which is controlled by the weight w_{is} , $i=1,2$, allows us to invert the logic level of the input. Not all of the 16 Boolean functions could be realized without this inversion. The NTC is our pulse generating element coding the logic level passed from the pulse

inverter. Finally, the lowpass filter (LPF) will let pass through the fundamental and possibly the second harmonic of the pulse repetition frequency of the NTC pulses. The choice is made through a weight w_{1f} .

A key element of the neuron is the multiplier that takes the product of the filtered signals x_i , $i=1,2$, yielding $y=k \cdot x_1 \cdot x_2$. Depending on which harmonics are present in the x_i 's, y has certain peaks in its spectrum. We select some of these peaks by a bandpass filter (BPF) which gives the signal z . The weight w_{3f} sets the center frequency of the passband of this filter.

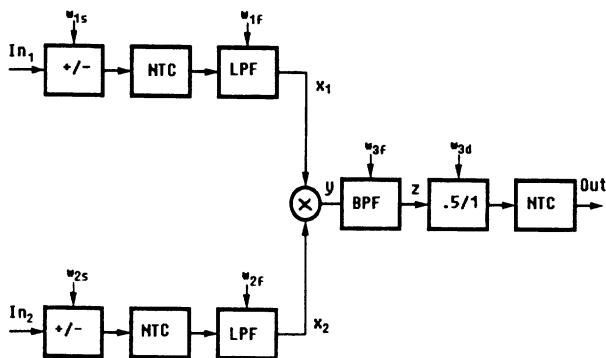


Figure 1. Block diagram of the pulse-coded neuron.

Unfortunately, we may end up with signals that have a repetition rate double that of an input pulse's repetition rate. For Boolean functions, this is not acceptable if we want to cascade our neuron, in which case, the output of a neuron is then the input of a second one, and if a change in the output repetition rate occurs, every stage would need to be optimized for a different rate. Therefore, we design a frequency divider to divide by two ($.5/1$ in Figure 1). The weight w_{3d} on the divider of Figure 1 permits selection of the presence of frequency division or not.

Lastly, we need an output NTC to regenerate the shape of the pulses, and therefore guarantee that the output signal of the neuron will be standardized over the different processing elements in a network regardless of what Boolean function each individual neuron processes.

In the following parts we study how to code information on the NTC pulses and show how we can use the spectrum of the internal signals x_i 's, y , and z to get the Boolean functions.

Choice of Repetition Rates

The repetition rate r_1 representing a "1" is chosen in the higher range of

the frequencies that the NTC can generate in order to further increase the difference between "0" and "1". Because the neural-type pulses are not sinusoidal, the pulse signal for a logic "1" is seen in the frequency domain as made of harmonics denoted $n \cdot f_1$, where n is an integer. We have $r_1 = f_1$. If we had chosen r_0 different from 0, it would have been harder to design efficient filters, since r_1 and r_0 would be given by the same NTC which has a somewhat restricted range. In other words, r_1 would have been closer to r_0 and we would have need for much better filters. Besides, signals at 0 frequency are relatively easy to use.

Role of the frequency in separating cases

The signal generated by the NTC is far from being sinusoidal [Tsay 91, p. 2]. In fact, the fundamental of r_1 is very strong if the NTC's input voltage, V_{in} is close to $V_{in,low}$ or $V_{in,high}$ which are the limits of the range of the values of V_{in} for which the NTC generates pulses. [Savigny 90, p. 38] For r_1 that is chosen in the upper region of the repetition rates generated by the NTC, the second harmonic contains up to 40% of the energy of the fundamental.

Since others have much less energy, we consider only the first two harmonics of the neural-type signals. Therefore, X_i , the Fourier transform of the signal on branch i just before the multiplier, has peaks at $\pm f_1$, $\pm 2 \cdot f_1$, in which case, Y , the signal after the multiplier, has peaks at $\pm f_1$, $\pm 2 \cdot f_1$, $\pm 3 \cdot f_1$, and $\pm 4 \cdot f_1$. This is because, if, with $\delta(\cdot)$, the Dirac function, and $\mathfrak{F}(\cdot)$, the Fourier transform, for $i=1,2$,

$$\begin{aligned} \mathfrak{F}(x_i(t)) = X_i(f) = & A_i + \frac{B_i}{2} \cdot [\delta(f - f_1) + \delta(f + f_1)] \\ & + \frac{C_i}{2} \cdot [\delta(f - 2f_1) + \delta(f + 2f_1)] \end{aligned} \quad (1)$$

where A_i , B_i , C_i are strictly positive real constant unless otherwise specified. Then, with $*$ denoting the convolution operator,

$$\begin{aligned} Y(f) = X_1(f) * X_2(f) = & \mathfrak{F} [x_1(t) \cdot x_2(t)] \\ = & \frac{2A_1A_2 + B_1B_2 + C_1C_2}{2} \cdot \delta(f) \\ & + \frac{A_1B_2 + A_2B_1 + B_1C_2 + B_2C_1}{2} \cdot [\delta(f - f_1) + \delta(f + f_1)] \end{aligned} \quad (2)$$

$$\begin{aligned}
& + \frac{2A_1C_2 + 2A_2C_1 + B_1B_2}{4} \cdot [\delta(f - 2f_1) + \delta(f + 2f_1)] \\
& + \frac{B_1C_2 + B_2C_1}{2} \cdot [\delta(f - 3f_1) + \delta(f + 3f_1)] \\
& + \frac{C_1C_2}{4} \cdot [\delta(f - 4f_1) + \delta(f + 4f_1)]
\end{aligned}$$

We summarize this in Table 1 recognizing that a logic "0" is represented by a DC constant, and a "1" by a signal having harmonics at f_1 and $2 \cdot f_1$. The table gives the peaks in Y for the four logic combinations of the inputs. By choosing between these peaks, we create the different functions.

X1		logic "0"	logic "1"	
X ₂		0	f ₁	2f ₂
logic "0"	f ₂	0	f ₁	2f ₂
logic "1"	f ₁	f ₁	0,2f ₁	f ₁ ,3f ₁
	2f ₁	2f ₁	f ₁ ,3f ₁	0,4f ₁

Table 1. Peaks in the spectrum Y of the product signal for different pulse coded logic inputs. A logic "0" at the i-th input of the neuron is transformed into a constant X_i, on the other hand, a logic "1" at the i-th input is transformed into pulses at repetition rate r₁, the upper harmonics are lowpass filtered out and the resulting signal X_i contains the first, or the first and second harmonics of r₁.

Role of the filters

Changing the cutoff frequency of the lowpass filters will suppress some frequencies in x₁ or x₂ and suppress certain harmonics of r₁ in y. This can be seen in Table 1. Let us study an example: suppose we filter out the harmonic 2·f₁ of the pulses from the first branch (C₁=0 in 2). Practically, this corresponds to disregarding the column of Table 1 labeled 2·f₁. The combination "01" (B₁=C₁=0 in 2) on the inputs can be differentiated from "10" (C₁=B₂=C₂=0 in 2) - where the left bit is the logic state of the first branch, similarly for the second - since peaks at ±2·f₁ are present in y when the inputs are "01", but are not present in Y for "10".

The pulse inverter

It turns out that not all the Boolean functions of two variables can be

realized with filters and a multiplier. To achieve this we add the pulse inverter on the input of the neuron [Savigny 91, p. 46]. Its purpose is to take the incoming signal on its input and invert its logic level. Therefore, if a logic "0" is present on the input of the pulse inverter, the pulse inverter transforms it into a logic "1" and vice versa. That is, if the pulse inverter is selected the pulse repetition rate at the input to the pulse inverter is changed to the "inverse" pulse repetition rate at its output.

We will see in the next section that some functions (out of the sixteen, exactly 6, including and, or, xor, 0) do not need to invert their inputs, while others do (for example, nor, nand). This has the consequence that we need the capability to enable or disable the pulse inverters depending on what function we want to implement.

THE DIFFERENT BOOLEAN FUNCTIONS

We discussed in a previous section that changing the filters and enabling or disabling the pulse inverters and frequency divider in the neuron of Figure 1 results in changing the processing of information in the neuron. We show in this section that all of the sixteen Boolean functions of two variables can be achieved by such changes.

We will call "weight" these quantities that are changed to obtain different functions. This is motivated by the terminology being used for neural networks [Dayhoff 90], where changing the weights modifies the processing. Table 2 lists the weights that are needed to obtain the different Boolean functions.

In the left column of Table 2, we use the notation Bh to label the Boolean function: B refers to "Boolean," and the number h in hexadecimal is a reminder of the output; just translate the hexadecimal number in binary and you have the output of the function. For example, for the four combinations of inputs listed at the top of Table 2, xor has the outputs 0 1 1 0, which is 6 in hexadecimal, so it is labeled B6. Also in the table, the sign weight w_{is} , $i=1,2$ takes on the value "-" if the pulse inverter is enabled, "+" if it is not. Similarly, the divider weight w_{3d} is "Y" when the frequency divider is selected, "N" when it is not required. The weights on the filters are labeled w_{if} , $i=1,2,3$, and are the cutoff frequencies for which we abbreviate the cutoff frequency by n, when an actual cutoff frequency of $n \cdot f_1$ is necessary. In the output columns, we denote zero for a pulsed signal with zero frequency representing a logic "0," and f_1 for pulses at the frequency which denotes a logic "1." In the next section, we show an example to illustrate the choice of weights to be taken from Table 2.

SIMULATION RESULTS

In order to check the validity of the theory, we first present the function

B2 as an example of how the weights of Table 2 apply. Then we implement the pulse coded neuron using SPICE for the xor (B6) function.

Inputs												
In ₁	0 0 1 1											
In ₂	0 1 0 1											
	logic output	Name	w _{1s}	w _{1f}	w _{2s}	w _{2f}	w _{3f}	w _{3d}	Output			
B0	0 0 0 0	0	+	1	+	1	>4	N	0 0 0 0			
B1	0 0 0 1	and	+	1	+	1	2	Y	0 0 0 f ₁			
B2	0 0 1 0	sgt	+	1	-	1	2	Y	0 0 f ₁ 0			
B3	0 0 1 1	In ₁	+	1	+	0	1	N	0 0 f ₁ f ₁			
B4	0 1 0 0	sst	-	1	+	1	2	Y	0 f ₁ 0 0			
B5	0 1 0 1	In ₂	+	0	+	1	1	N	0 f ₁ 0 f ₁			
B6	0 1 1 0	xor	+	1	+	1	1	N	0 f ₁ f ₁ 0			
B7	0 1 1 1	or	+	1	+	2	1	N	0 f ₁ f ₁ f ₁			
B8	1 0 0 0	nor	-	1	-	1	2	Y	f ₁ 0 0 0			
B9	1 0 0 1	nxor	+	1	-	1	1	N	f ₁ 0 0 f ₁			
BA	1 0 1 0	nIn ₂	+	0	-	1	1	N	f ₁ 0 f ₁ 0			
BB	1 0 1 1	goe	+	1	-	2	1	N	f ₁ 0 f ₁ f ₁			
BC	1 1 0 0	nIn ₁	-	1	+	0	1	N	f ₁ f ₁ 0 0			
BD	1 1 0 1	soe	-	1	+	2	1	N	f ₁ f ₁ 0 f ₁			
BE	1 1 1 0	nand	-	1	-	2	1	N	f ₁ f ₁ f ₁ 0			
BF	1 1 1 1	1	In ₁	2	In ₂	2	1	N	f ₁ f ₁ f ₁ f ₁			

Table 2. Weights of the pulse-coded neuron in order to obtain the 16 Boolean functions of 2 inputs variables In₁ & In₂. w_{is}, i=1,2 is "-" if the pulse inverter is selected, "+" otherwise. f_c=w_{if}f_m i=1,2 is the cutoff frequency of the lowpass filters with fm between f₁ and 2·f₁, f_c=w_{3f}f₁ is the cutoff frequency of the bandpass filter. Finally, w_{3d}="Y" when frequency division is necessary, "N" else.

Verification of the weights for "sgt"

The function B2 is also called "sgt," for "strictly greater than." From Table 2, we get the values of the weights for the neuron of Figure 1. We have four cases for the input vector (In_1, In_2) , namely (0,0), (0,1), (1,0), and (1,1). According to the w_{1s} & w_{2s} entries of Table 2, we invert the second branch only, which swaps the bits of that branch. Then, the NTCs transform a logic "0" to 0V, and a logic "1" to pulses at repetition rate r_1 . Both input branch NTCs have pulses with harmonics at $n \cdot f_1$, where n is a positive integer. According to $w_{1f}=1$ & $w_{2f}=1$ a lowpass operation is performed on each input branch cutting off all high harmonics. After the filters, we are left with signals having most of their energy at the frequency f_1 for the logic "1," and we still have 0 for the logic "0". The next step is to perform a modulation by taking the product of the two input branch signals. The resulting signal, y , has energy around frequencies which are the sum and difference of the frequencies of the signals on each branch. From the w_{3f} column, in which $w_{3f}=2$, we take the bandpass around $2 \cdot f_1$ which creates a signal with energy at $2 \cdot f_1$ only when the input to the network is (1,0). A division of the frequency by two ($w_{3d}="Y"$) reduces the frequencies present in the output signal to f_1 or 0 (recall that the logic "0" which is represented by a DC constant is not affected by the frequency division). The logic equivalent of the obtained signal is "0" when the input is (0,0), (0,1), or (1,1) and "1" results from the input (1,0). We therefore have the desired output for the strictly greater than Boolean function B2.

We have chosen to illustrate Table 2 by this function because B2 requires all the elements of the neuron, so if the proof is understood for this function, it should be straightforward to write the proofs for the 15 other functions. Indeed, given an understanding of how to prove the result for B2, it is easy to prove all the results of Table 2; consequently, except for the comments on "xor" which follow, the details of proving the validity of the other entries is omitted, though they can be found in detail in [Savigny 91].

Xor

The point of this section is to realize an exclusive or (xor) using the neuron of Figure 1 with the weights of Table 2 for B6. An MOS circuit of the neuron using MOSIS parameters was simulated via PSPICE.

Figure 2 gives the signals at the outputs of the NTCs on each branch of the neuron and the output. We check that the output (lower curve) is a logic "1" (i.e. pulses, possibly after a delay) if and only if exactly one input is "1"; note that a "0" or "1" in In_i is accurately represented by the pulses displayed in the two upper curves.

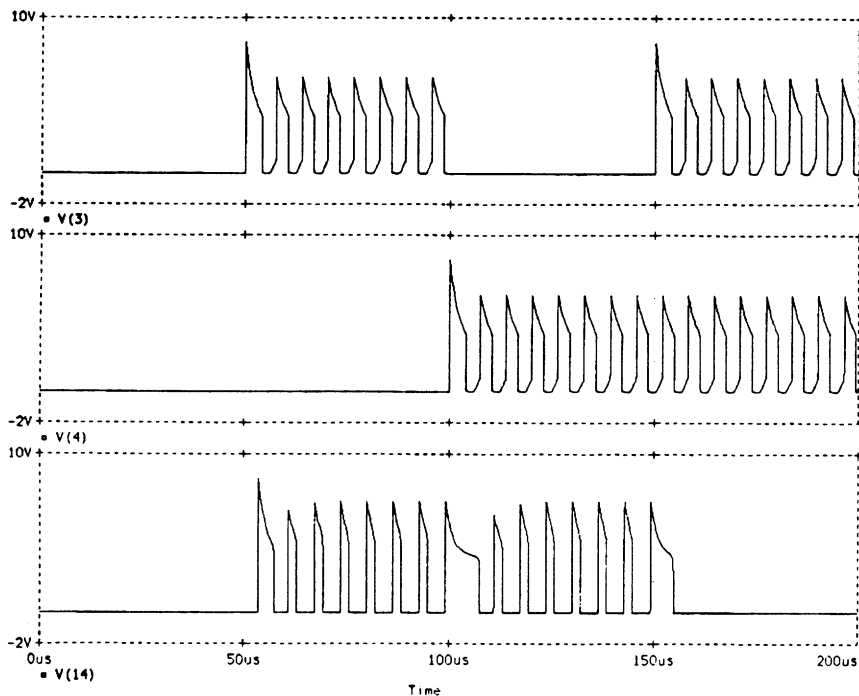


Figure 2. SPICE analysis results for the xor (B6) function.

CONCLUSIONS

We have presented a neuron realizing the sixteen Boolean functions of two variables, noting that the same structure can implement any of these functions, if well-chosen weights are used. Simulation results are given to verify that the neuron works as claimed.

Using de Morgan's work, we know that any binary valued logic function, no matter how complicated, can be realized using the basic two-input functions discussed in this paper. This leads us to the end result that any digital computer can be realized by a pulse coded neural network. Since our pulse coded neuron might be interfaced with TTL-compatible computers, the neuron is designed to allow for TTL input signals, as well as for input pulse coded signals. If it is desired that the output should also be TTL-compatible, a simple device exists [Savigny 91] to convert the pulses to TTL voltage levels. In other words, we can replace logic gates by this neuron, as well as use it throughout a pulse coded computer.

At this point, we are researching a way to reduce the size of the electronic circuit realizing the neuron: The one presented in this paper gives an existence

proof that neurons can implement logic functions, but to get a reasonable size network, its neurons must have the smallest size possible. Thus, it would be best if the filters, frequency dividers, and pulse inverters could be dispensed with.

Finally we wish to acknowledge discussions with many of our colleagues and students as well as the research support of the AFOSR portion of ONR Grant No. N00014-90-J-1114.

REFERENCES

- [Cole 89] Clint Cole, Angus Wu, & Jack Meador, "A CMOS impulse neural network," Proceedings of the Colorado Microelectronics Conference, Colorado Springs, Colorado, March 1989, pp. 678-686. [Dayhoff 90] Judith E. Dayhoff, "Neural networks architectures: an introduction," Van Nostrand Reinhold, New York, 1990.
- [El-Leithy 88] Nevine El-Leithy & Robert W. Newcomb, "Hysteresis in Neural-Type Circuits," Proceedings of the 1988 IEEE International Symposium on Circuits and Systems, Vol 2, June 1988, Helsinki, Finland, pp. 993-996.
- [Ganong 85] William F. Ganong, "Review of medical physiology," Lange Medical Publications, 12th edition, Los Altos, CA, 1985.
- [Geiger 90] Randall L. Geiger, Phillip E. Allen, & Noel R. Strader, "VLSI design techniques for analog and digital circuits," McGraw Hill Publishing Co, New York, 1990.
- [Gray 84] Paul R. Gray & Robert G. Meyer, "Analysis and design of analog integrated circuits," John Wiley & Sons, New York, 2nd edition, 1984.
- [Habib 89] Mahmoud K. Habib & H. Akel, "A Digital Neuron-Type Processor and Its VLSI Design," IEEE Transactions on Circuits and Systems, Vol 36, No 5, May 1989, pp. 739-746.
- [Hartline 89] Daniel K. Hartline, "Simulation of restricted neural networks with reprogrammable neurons," IEEE Transactions in Circuits and Systems, Vol 36, No 5, May 1989, pp. 653-660.
- [Hopfield 87] John J. Hopfield, "Artificial Neural Networks," IEEE Circuits and Devices Magazine, Vol 4, No 5, September 1988, pp. 3-10.

- [Lippmann 91] Richard P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, Vol 4, No 2, April 1987, pp. 4-22.
- [Meador 91] Jack L. Meador, Angus Wu, Clint Cole, Novat Nintunze, & Pichet Chintrakulchai, "Programmable Impulse Neural Circuits," IEEE Transactions on Neural Networks, Vol 2, No 1, January 1991, pp. 101-109.
- [Murray 88] Alan F. Murray & Antony V. W. Smith, "Asynchronous VLSI Neural Networks Using Pulse-Stream Arithmetic," IEEE Journal of Solid-State Circuits, Vol 23, No 3, June 1988, pp. 688-697.
- [Murray 89] Alan F. Murray, "Pulse Arithmetic in VLSI Neural Networks," IEEE Micro, December 1989, pp. 64-p74.
- [Savigny 90] Marc de Savigny, Guy Moon, Nevine El-Leithy, M. Zaghoul, Robert W. Newcomb, "Hysteresis Turn-On-Off Voltages for a Neural-Type Cell," Proceedings of the 33rd Midwest Symposium on Circuits and Systems, Calgary, Canada, August 1990, pp. 37-40.
- [Savigny 91] Marc de Savigny, "Pulse coded neuron realizing Digital functions," Master of Science Thesis, University of Maryland, 1991.
- [Savigny 92] Marc de Savigny and R. W. Newcomb, "Realization of Boolean Functions Using a Pulse Coded Neuron," Proceedings of the 1992 IEEE International Symposium on Circuits and Systems, San Diego, CA, May 1992, pp. 2228-2231.
- [Tomberg 90] Jouni E. Tomberg & Kimmo K. K. Kaski, "Pulse-Density Modulation Technique in VLSI Implementations of Neural Network Algorithms," IEEE Journal of Solid-State Circuits, Vol 25, No 5, October 1990, pp. 1277-1286.
- [Tsay 91] S. W. Tsay, Marc de Savigny, Nevine El-Leithy, and Robert W. Newcomb, "An All MOS Neural-Type Cell," Proceedings of the 34th Midwest Symposium on Circuits and Systems, Monterey, California, May 1991, to be published.

DESIGN OF PULSE CODED NEURAL PROCESSING ELEMENT USING MODIFIED NEURAL TYPE CELLS

Gyu Moon and Mona E. Zaghloul

*Department of Electrical Engineering and Computer Science
The George Washington University, Washington DC, 20052*

ABSTRACT

This chapter presents the hardware realization of Neural Processing Element (NPE). The NPE executes the well-known nonlinear threshold function of the weighted summation, and behaves as an artificial neuron in an artificial neural network. The synaptic weighting and summing using pulse coded Modified Neural-Type Cells (MNTC) are presented. The basic information processing is in the form pulse duty cycle modulation. A prototype CMOS chip with $2\mu\text{m}$ minimum feature size was designed and measurements on the fabricated chip are included.

INTRODUCTION

Artificial Neural Networks (ANN) has been heavily studied during the last two decades in pursuit of non-linear, highly distributed, soft error immune system performance in both scientific and engineering fields [Lipp87, Widr90]. They offer attractive solutions for many problems in which perception is more important than intensive computation [Graf88, Lipp87]. The first pioneering work was done by McCulloch and Pitts [Mccu43] giving mathematical expressions of nervous activities, and thereafter various types of models of a bio-neuron have been introduced as in [Fitz61, Hodg52]. Based on the single neuron model, several different network model algorithms have

been proposed in recent years having perceptual properties, such as learning, associative memory, image or speech recognition, feature extraction, and pattern classification [Carp87, Koho88, Tank86, Ande77, Rume86, Kosk87].

The research on neural networks covers broad topics of modeling, network architecture, and hardware implementation. Hardware implementations of neural network algorithms support two main characteristics of neuromorphic systems: speed and fault tolerance. The hardware implementation method shows inherent fault tolerance specialities and high speed which is usually more than an order over the software counterpart [Lee92, Graf88]. So, it was often pointed out by many authors [Graf87, Sanc91, Murr92, Mead91, Graf88] that in order to obtain the full benefit of neural network algorithms, special purpose hardware must be built.

In this chapter, a CMOS hardware implementation of Neural Processing Element (NPE) is proposed. The NPE serves as neuron in an Artificial Neural Network. The motivation for this task is, as explained above, to enhance computational speed by several order of magnitudes than software simulation and thus will lead us to have real world applications in interactive mode. The inputs and the outputs of the proposed NPE are represented as analog continuous values, while weighting process is done with digitized pulse streams. The analog pulse coded technique for weighting allows us to design in a simple way, yet holding similarities in signal shape with biological neurons. **Pulse Duty Cycle Modulation** technique is adopted only for the weight multiplication. The input and the weight multiplication outcome is converted into a pulse stream, where the average pulse duty cycle [Bell88] of the pulse stream serves as an information variable. The use of the average pulse duty cycle as information medium have many advantages. For example, it supplies advantages from both digital and analog approaches: such as reasonable noise immunity, small hardware realization size, and asynchronous behavior where system clocks are not needed. Another important advantage that justifies the use of pulses is that, since such systems operate on the basis of averaging principles, they are inherently more tolerant to imperfections and non linearities in the components. There is no need for pre- or post-manipulation for signal modulations as often seen in conventional ones in communication [Carl86] chopping clocks [Murr89], Op-amps [Mead91], or switched capacitor circuits [Tomb90]. The proposed design uses the pulse coded operation only for the weighting process and does not need auxiliary circuits needed for pulse shape manipulation, such as pulse frequency modulation (PFM), pulse amplitude modulation (PAM), and pulse position modulation (PPM) circuits.

The analog circuit used to generate pulses as function of input is the

Neural Type Cells (NTC) originally developed by Newcomb [New83]. In the following sections, we will first discuss modifications introduced to the Neural Type Cell so as to adapt to our proposed technique of pulse coded synaptic weights. The Modified Neural Type Cell (MNTC) is used as a synapse for Neural Processing Element (NPE) which serves as a neuron in a proposed Artificial Neural Network.

MODIFIED NEURAL TYPE CELL AS A SYNAPTIC ELEMENT

The Neural Type Cell (NTC) circuit was introduced by Newcomb in [Newc83]. It emulates the spiking pulse generation feature of biological neuron. It has advantages for VLSI implementation such as the simple structure and its functional similarity with its biological counterpart. However, it also has disadvantages of limited oscillation range and complexity of the process which results from its hysteresis nonlinearity. The original NTC has been modified as compared to the original one [Newc83], and named **Modified Neural Type Cell (MNTC)**. The Modified Neural Type Cell (MNTC) will serve as a synaptic element where the weighted multiplication between the input and the weight is occurring. Functionally it receives an input signal from other neurons or from outside and converts that into a corresponding pulse stream whose pulse shape is controlled by the weight signal. A simple functional representation for the MNTC is shown in Figure 1.

The MNTC consists of two functional sub-blocks: voltage controlled oscillator and threshold crossing detector blocks. Both input and weight are analog values in voltage and the output is a pulse stream with a fixed height. The MNTC can be viewed, thus, as a voltage-controlled oscillator with a control signal W (Weight). Inherent hysteresis characteristics of the original NTC contributes to trigger the oscillatory phenomena, and the details for this was explained in [Moon90c]. The weight signal is tied to a voltage controlled resistor [Moon90b], whose value will decide the conductance of a resistor device within the MNTC. As the weight signal changes, the conductance of a resistive device is changing and this will indirectly control the pulse duty cycle of the output pulse

Figure 2 shows the original Neural Type Cell. As can be seen, it consists of three transistors ($M1 - M3$), three passive resistance ($R1 - R3$) and a capacitor (C). Input is $V1$ and outputs are $V2$, $V3$, and $V4$. As the input ($V1$) increases in time domain, $V2$ and $V4$ decrease, and $V3$ increases. When $V1$ reaches a certain level and above, the outputs start to oscillate and their shapes of waves are changing by the input. More specifically, the frequency is increasing while the amplitude of the output is decreasing. The detailed analysis and description for the NTC's operation were reported in [Moon91a].

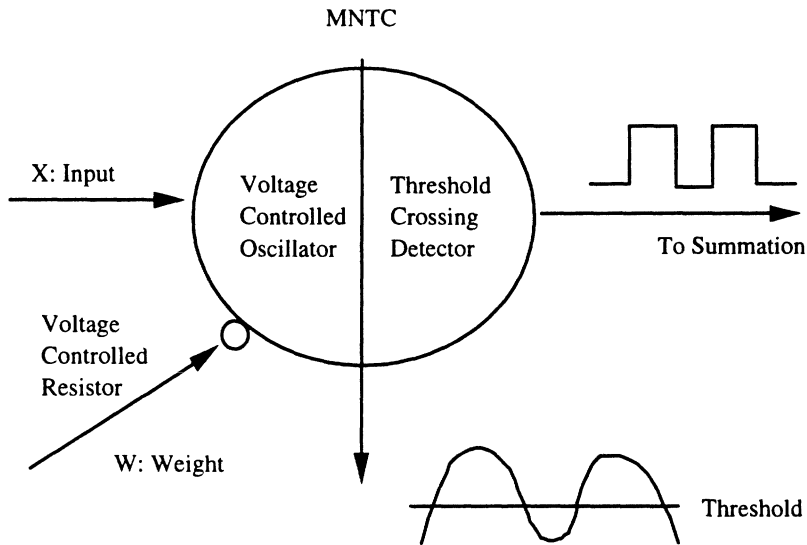


Figure 1. Functional representation of Modified Neural Type Cell.

The original NTC shown in Figure 2 serves as an analog voltage controlled oscillator (VCO) in Figure 1, whose wave is controlled by the input strength. As seen in Figure 2, the output of the original NTC is a totally analog signal whose shape is controlled by the input. These analog continuous properties impose difficulties for signal handling. Thus a threshold crossing detector block for converting the analog wave into pulse streams with a fixed height was introduced.

Figure 3 shows an NTC with a threshold crossing detector. Notice here that V3 is chosen since it is known to swing across the threshold voltage of the feedback transistor (M3) [Moon91a]. The support for this choice was rationalized in an analytical way [Moon91a, Moon92] and will not be covered in here. The threshold crossing detector consists of two inverters. The first inverter, which is composed of M4 and M5, is to convert the analog wave into a pulse stream. It has to have a minimum loading effect on the V3, which might deteriorate the performance of the original NTC. Also, it should have a suitable (near the threshold voltage of the N-type transistor (M3)) level of inverter threshold voltage [Mead80]. Considering above two necessities, the M4 and M5 are designed with sizes of $4/2\mu\text{m}$ and $24/2\mu\text{m}$, respectively. The

inverter threshold voltage of the first inverter was designed to be approximately 1.7V. Using regular device parameters, the loading capacitance on V3 from the first inverter is estimated as 0.0448 PF. This is quite small

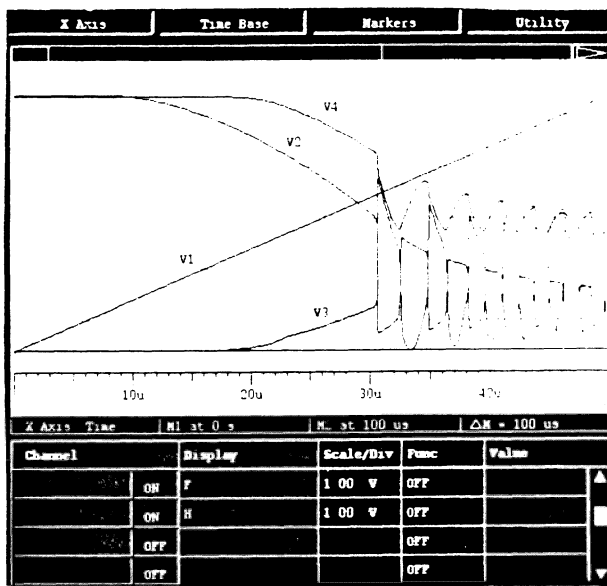
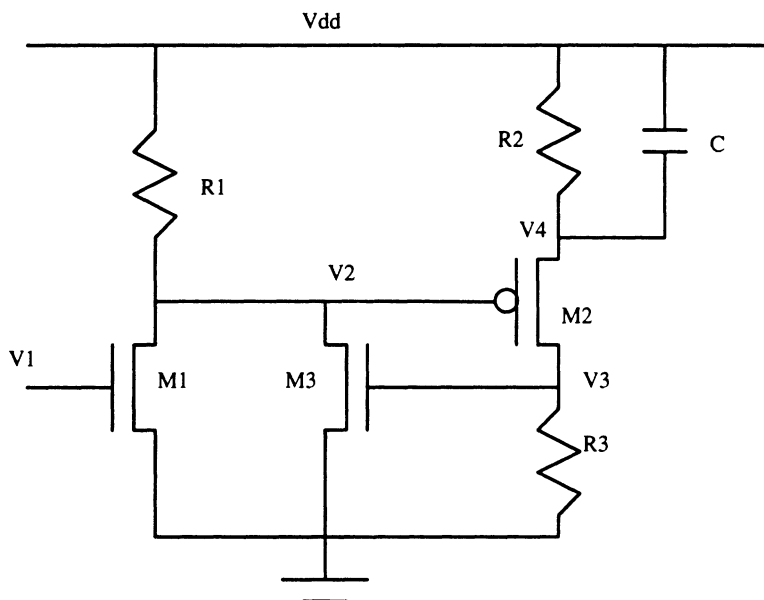


Figure 2. Circuit diagram for the original NTC and the SPICE Simulation.

compared with the load capacitance C (35pF) in Figure 3 and thus we can say that the loading effect is negligible.

The second inverter, which is composed of $M6$ and $M7$, has two purposes. One is to enhance the weak driving-current capability, and the other is to make the output Y_p have the same polarity (phase) as $V3$. $M6$ and $M7$ are designed with sizes of $40/2\mu\text{m}$ and $20/2\mu\text{m}$, respectively. The size of pull-up transistor ($M6$) is designed twice as large as the pull-down transistor ($M7$), considering the mobility difference between the hole and electron carriers.

The simulation result for Figure 3 is shown in Figure 4.

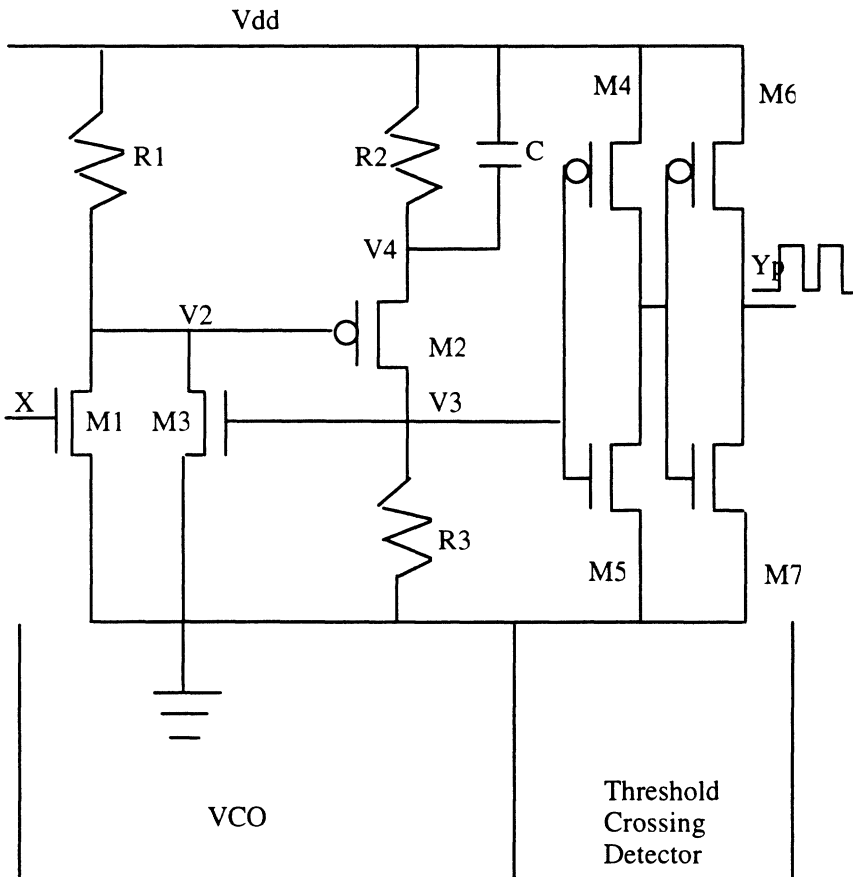


Figure 3. Circuit diagram for a NTC.

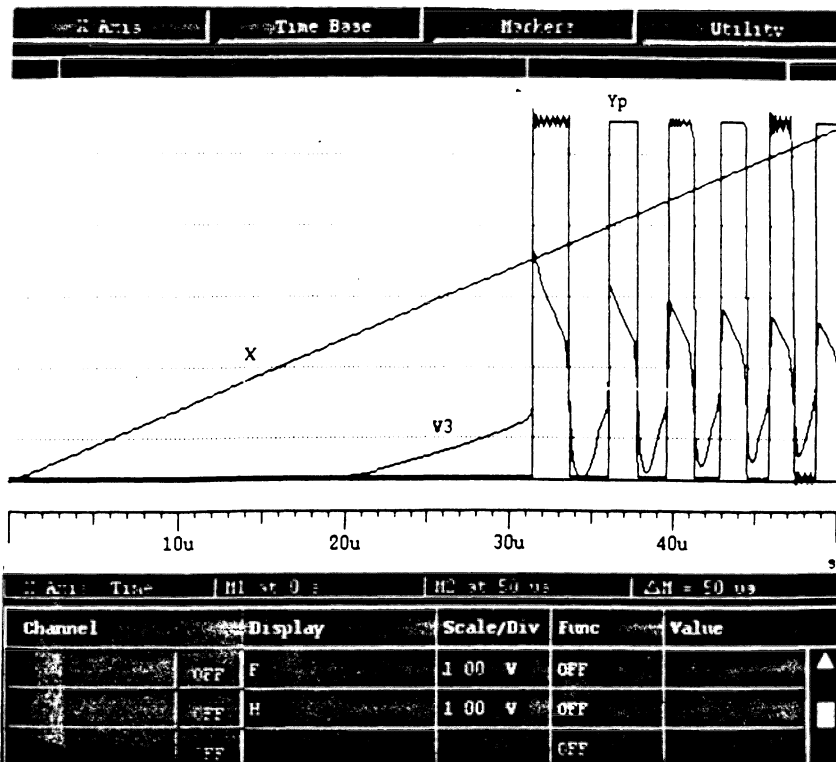


Figure 4. SPICE Simulation result of Figure 3.

As expected the output Y_p is now a pulse stream with same level of swing (0 - 5V), and its pulse shape is still controlled by the input (X). Notice that the frequency of the output Y_p is increasing as the input increases. This can be explained analytically through previous works [Moon90b, Moon92].

Thus far we have only input and output terminals. Now we have to introduce a weighting function into the circuit of Figure 3. To do this, we adopt a voltage controlled MOS resistor [Moon90a] which will replace one of three passive resistors in Figure 3. Instead of R_1 and R_2 , we chose R_3 for this replacement. The control signal of the MOS resistor can be adaptively controlled through a learning block that is limited between V_{dd} and GND.

The voltage controlled MOS resistor is composed of two enhancement-type MOS transistors as shown in Figure 5, and its equivalent resistance was found as [Moon90a]

$$R_{eq} = 1/[K(Vc2 - 2Vt - E1)] \tag{1}$$

where K is transistor gain factor [A/V²], and Vt is the threshold voltage for transistors.

The detail description for this resistor was reported in [Moon90a] and it will not be dealt with here.

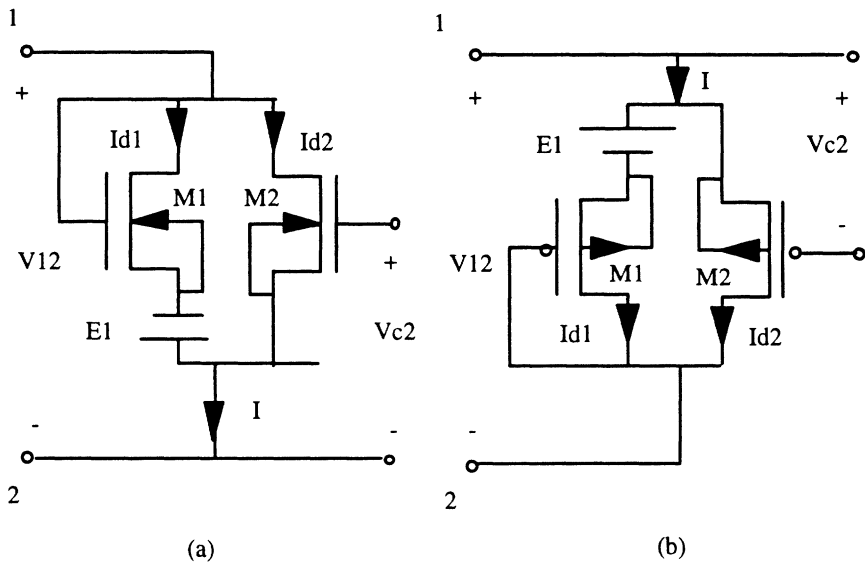


Figure 5. CMOS voltage-controlled linear resistor. (a) NMOS-type and (b) PMOS-type

Replacing R3 with the voltage controlled resistor, we have new circuit diagram as shown in Figure 6. The external voltage source E1 shown in Figure 5 is to accommodate an appropriate dynamic region and is not used in Figure 6 for simplicity. The signal labeled W is the control voltage (Vc2) for the MOS resistor and serves as the weight signal for the MNTC as well. Thus, now we have set up a basic configuration for synaptic element which has an input, a weight, and an output, whose pulse shape is controlled either by the input or by the weight. Figure 7 shows corresponding simulation results of Figure 6 with a fixed weight and a varying input in (a), and with a fixed input with changing

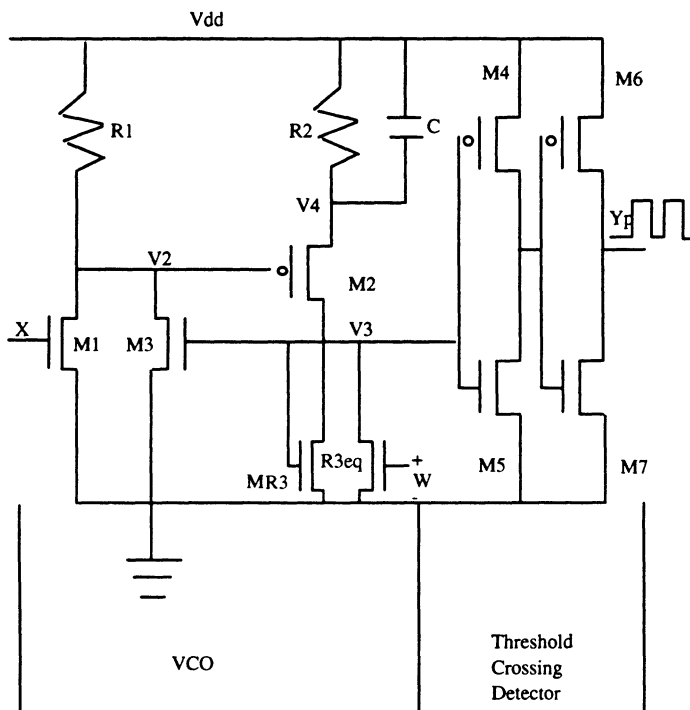
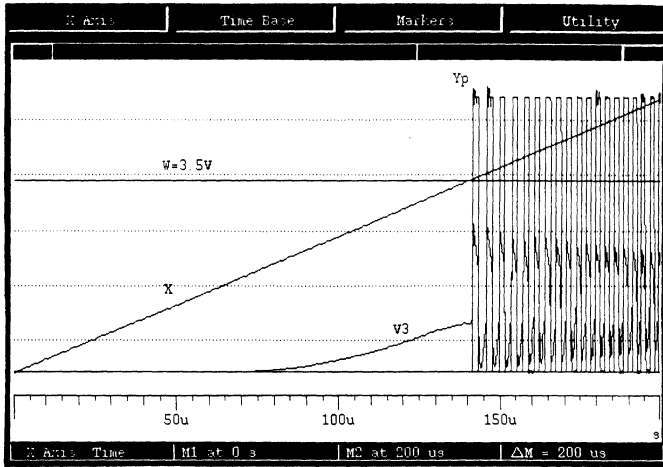


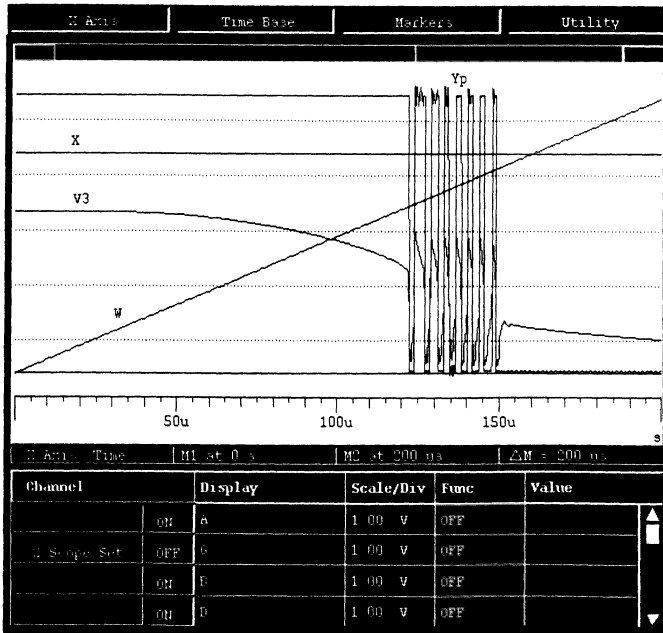
Figure 6. Modified NTC with a voltage controlled resistor replacing R3.

weight in (b).

As can be seen from Figure 7(a), with a fixed weight of 3.5V, as the input increases, V3 also increases. At the moment when V3 reaches the threshold level of the feedback transistor M3, it turns on and hysteresis triggers oscillatory phenomena. After this point, as X input increases, the frequency of the output pulse stream (Yp) increases while the density of the pulse is becoming higher. Figure 7(b) shows a fixed input of 4V, and the increase of the weight voltage will cause frequency increase of the output Yp but with lower density. Instead of picking up the frequency of the output pulse stream as a variable of the system, we chose the pulse density as a control parameter for the system. By doing so, we do not need manipulations or preprocessing on signals, like spectrum analysis or synchronization, as often seen in pulse frequency modulation (PFM) technique [Carl86]. However, the pulse density of the output pulse of Figure 7 does not change in the same direction with respect to both input and weight voltages. It increases as the input voltage increases, while it decreases as the weight voltage increases. Thus, to have the pulse density changing in the same direction with both the input and the



(a)



(b)

Figure 7. SPICE simulations of Figure 6. (a) with a fixed weight of 3.5V, (b) with a fixed input of 4.0V.

weight, we interchanged M1 and R1 as shown in Figure 8.

In addition, the original NTC has a limited range for oscillation. For instance, in Figure 7(a) the oscillation happens only for input range of 3.3 - 5.0V. Out of this range, no oscillation is acquired. This is due to the inherent hysteresis characteristic with the feedback transistor (M3) in the original NTC, which contributes to trigger this oscillatory phenomena [Moon90c]. Conditions for oscillation were already studied in [Savi90] and the focus in the design is to increase the oscillation range as wide as possible.

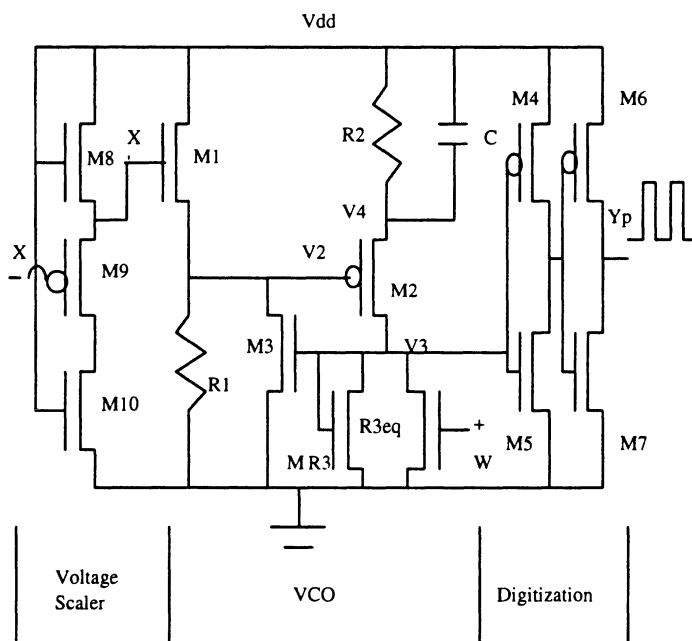
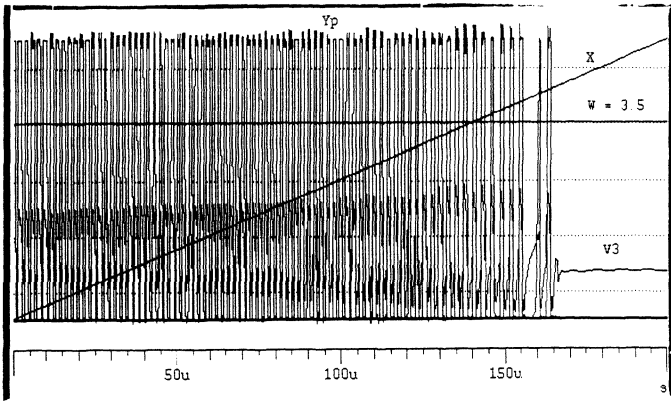


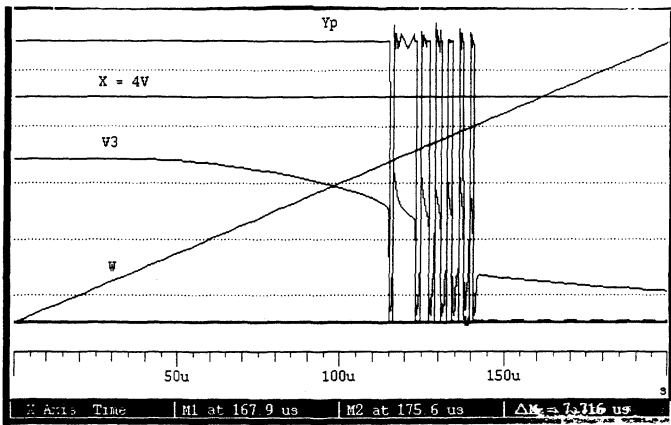
Figure 8. Circuit diagram of Modified Neural Type Cell.

The Modified Neural Type Cell is shown in Figure 8. A voltage scaler is added to scale the range of input. The voltage scaler is a simple circuit composed of three transistors (M8 - M10), which scales a limited input oscillation range into a wide one. The maximum range of the X' signal in Figure 8 for oscillation was found as 3.3 - 4.4V. However, with a scaler circuit, the input value (X) is scaled into this range for X'.

Both M8 and M10 are NMOS transistors with their gates tied to the power line (V_{dd}), and M9 is a PMOS device. With their gates tied to V_{dd} , devices M8 and M10 are always turned on and thus can be seen as two



(a)



(b)

Figure 9. SPICE simulation results of MNTC. a) with a fixed weight of 3.5V, b) with a fixed input of 4.0V

resistive components connected in series constituting a voltage divider. The aspect ratios of both transistors determine the dividing factor which will decide the offset level of the output. M9, on the other hand, determines the slope of the output (X' in Figure 8). Notice here that M9 is a P-type MOS device, and thus as X increases, its gate-to-source voltage (V_{gs}) is decreasing.

Figure 9 shows simulation result of MNTC (Figure 8) with a fixed weight of 3.5V in (a), and with a fixed input of 4.0V in (b). We can see that the oscillation range is widened significantly up to 4V (0 - 4V), and that pulse

density of Y_p is now decreasing as input increases. However, as seen in Figure 9(b), there is a limited range of oscillation for the weight value. Figure 10 graphically shows this oscillation range in the X-W (Input-Weight) plane.

As the weight increases beyond the upper bound on the oscillation range, the output Y_p goes to zero. As the weight decreases below the lower bound of the oscillation range, the output Y_p stays at V_{dd} (5V). This relation can be clearly seen in Figure 9(b). From the above, it appears that the range of oscillation is for $2.9 \leq W \leq 4.1$, thus by using an appropriate scaler circuit for the weight voltage, we can force the MNTC to operate in the oscillatory mode for a larger W range. Notice that this will, however, increase the number of transistors by three for each MNTC.

The above completes a CMOS circuit design of a synaptic element using an MNTC. Both input and output are analog continuous value in voltage, and the output is a pulse stream whose shape is controlled either by the input or by the weight. Notice that in this scheme of MNTC, we established a CMOS design of weighted multiplication in a simple way. The theory of such an operation is discussed in [Moon93b]. The proposed MNTC occupies 73mil². Although the weighing (multiplying) process is not exactly linear and not precisely controllable, we analytically found that it is a monotonic process.

NEURAL PROCESSING ELEMENT

In this section, we will describe the functions of the proposed Neural Processing Element (NPE) as a single node in an ANN. In the NPE, the

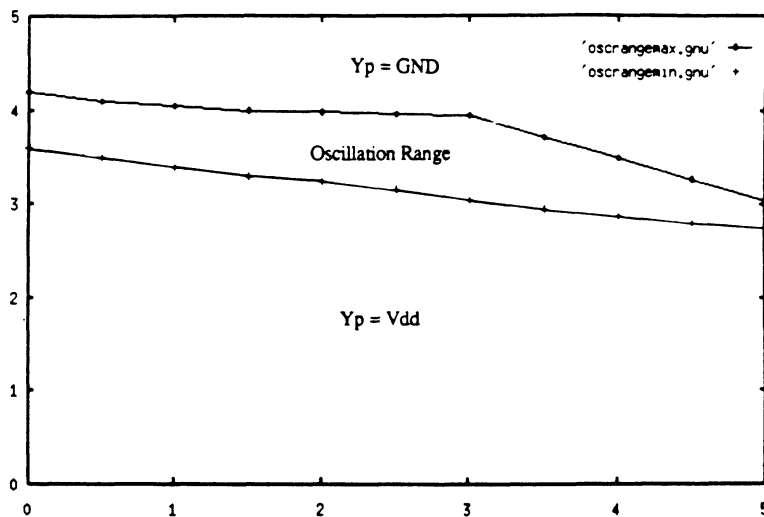


Figure 10. Oscillation range in X-W plane.

information is to be processed and the output be distributed to the other nodes. Thus, it acts as an electronic analogy of a biological neuron in nervous system.

As shown in Fig 11, inputs are applied from other NPE in the structure or by external stimulus. Each input has its corresponding weight. These multiple weighted inputs are to be summed in the NPE, and compared with a threshold value. In case the weighted sum is larger than the threshold, the output is to fire with logic value high (one), otherwise, it remains low (respectively, zero). This speculation gives us a possible function block diagram of the NPE as shown in Figure 11.

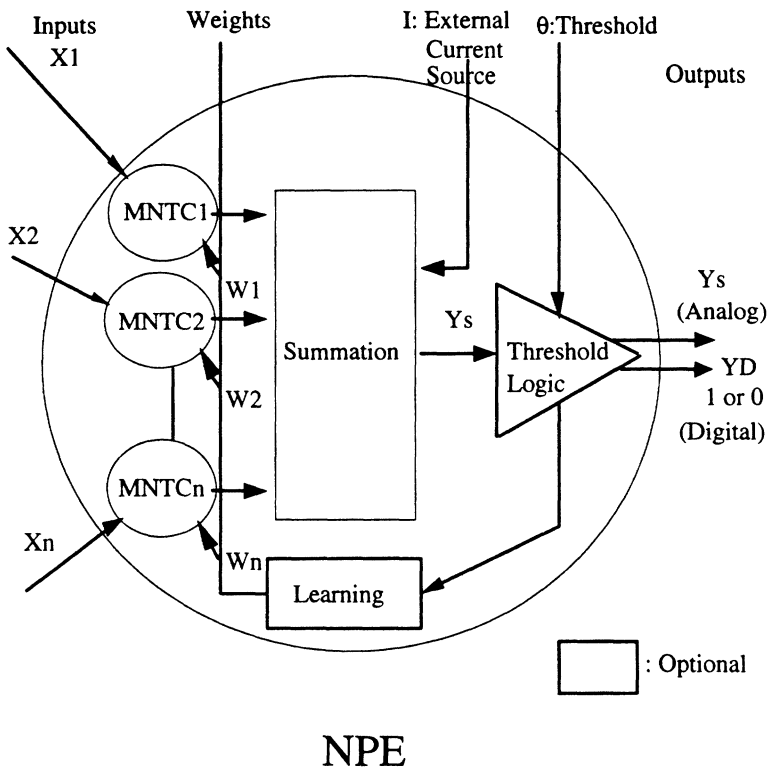


Figure 11. Functional block diagram of Neural Processing Element (NPE).

The NPE consists of four function blocks: Modified Neural Type Cell (MNTC), summation, threshold logic, and adaptive learning blocks. Inputs (X_i) are time-varying analog voltage signals and there are two outputs: one being digital (Y_D) and the other is analog (Y_s). Digital output has two values as a result of the comparison, either high when the Y_s is larger than the threshold (θ), or low otherwise. The analog output (Y_s) is a continuous analog

signal of the output of the summation block. This is not passed through comparison process in the threshold block. The choice between Y_D and Y_S depends on the network model and also upon functionality of the network.

The input signal is multiplied by a corresponding weight as described in the previous section in the MNTC. In the summation block, a summation of all the weighted inputs is executed. A simple capacitive load is adopted for current charge summation (accumulation). An external current source (I) could be applied. In the threshold block the comparison between the weighted summation and a given threshold value is accomplished. The output will be displayed to external, and also will be delivered to other NPE's in the network. Notice that the threshold level θ (θ is normally half of the maximum voltage swing of the Y_s) is applied from outside, and can be changed depending on the function of the network. Finally, in an optional learning block, the weights are updated in an adaptive fashion. The weights are updated according to a **learning rule** and play a major role in determining the overall behavioral characteristics of the networks.

The model of a single processing element in Figure 11 satisfies the basic properties of the artificial neuron. In the following a detailed description of the components of NPE is described.

Summation Block

To achieve an artificial neuron as described, each weighted input must be summed. The summation block is for this task; summing weighted input signals which are generated through the previously described synaptic blocks (MNTCs). A circuit diagram for a proposed summation block is shown in Figure 12.

The n - input pulse stream signals at the left side (Y_{pi}) come from the MNTCs. Each pulse has a certain pulse density which is controlled either by the input or by the weight in the MNTC. The analog output, Y_s , is the voltage across the capacitor C_s . There is also one external current source, I_{ext} , with which we can control the output directly from outside. As can be seen, each pulse works as a single input for pseudo-inverter structure (M_1 , M_2) and as the number of input increases, the number of this transistor pair (M_1 , M_2) is also increasing. Transistors M_3 and M_4 have two purposes: one being to increase resistive value and achieving a large RC constant in transient, and the other is to minimize charge feed-through effect [Alle87, Gray84] over the output when switching, which is normally witnessed in MOS switching device due to stray overlap capacitances. If the inverters and C_s are designed in such a way that the circuit has much larger RC time constant than the pulse period, the output Y_s will converge to a steady-state value in between V_{dd} and ground. One

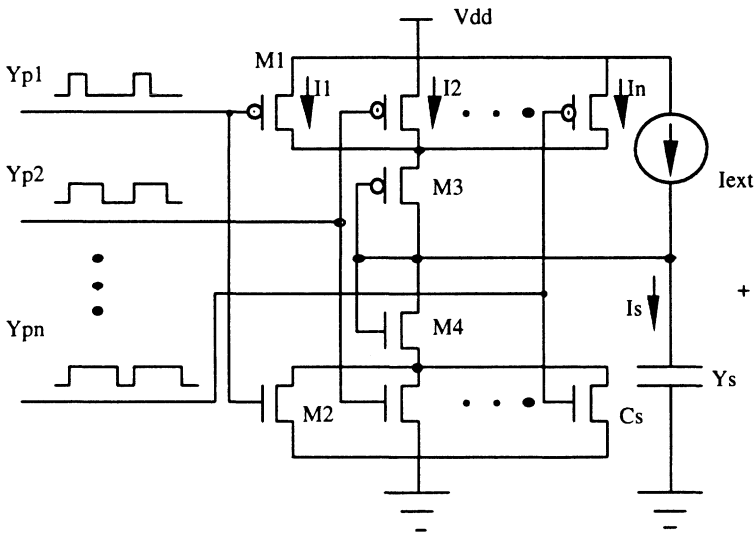


Figure 12. Circuit diagram for summation.

example is shown in Figure 13.

The Pulse Duty Cycle (PDC) of the n -th output of a time-varying pulse stream over a (possibly variable) time interval, t , is defined as:

$$Y_{\text{PDC}}^n(t) = \frac{\sum_{j=1}^m \text{PW}(j)}{t} \equiv \text{PDC} \quad (2)$$

where $\text{PW}(j)$ is the j -th pulse width in the stream, assumed to be of m pulses, in time interval t .

This is for a pulse stream analogous to the duty cycle of a single pulse [Bell88]. As a result of (2), we know that $0 \leq \text{PDC} \leq 1$ and that PDC represents the average value of pulses in time period of t . Thus we can say that the closer PDC is to 1, the denser a pulse stream is. Note also that in order to have a meaningful value of PDC for a given pulse stream, the time t should be chosen larger than any of the $\text{PW}(j)$. In case of dense pulse, the steady-state value of Y_s will be close to the ground level. If the pulse density is small, on the other hand, the output Y_s will be close to the V_{dd} level. In this way the steady-state value of Y_s can be represented as a function of the 'summation' of contributions from each pulse.

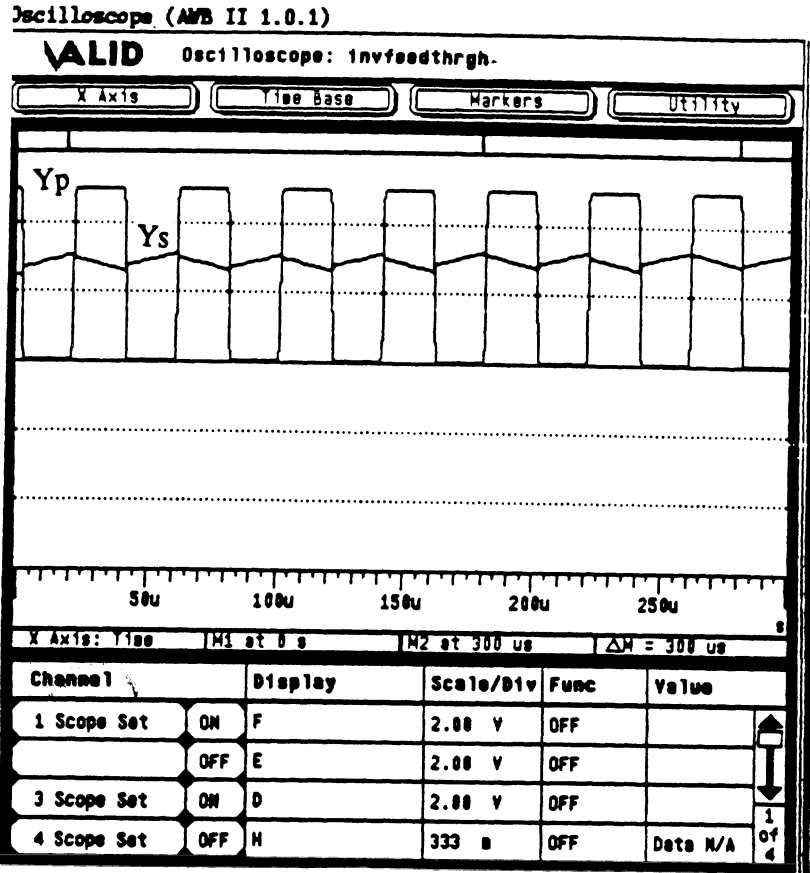


Figure 13. Output of summation block with a large RC time constant.

In this scheme, the circuit in Figure 12 was designed and adjusted so that when average value of PDCs from every pulse streams is equal to 0.5 the steady-state output Y_s is half of V_{dd} . This can be easily done by using suitable sizes of transistors, whose turned-on resistances are same for pull-up and pull-down operations. The sizes of M1, M2, M3 and M4 are chosen as 2/50, 2/100, 4/4 and 4/45, respectively. In this way of design, we differentiate the excitatory operation from inhibitory one, depending on the value of the PDC. If a PDC of single pulse stream is larger than 0.5 generating higher than half of the V_{dd} level, we say that the connected corresponding MNTC is acting as an **inhibitory** junction, otherwise it will act as an **excitatory** one.

This scheme is very unique and has a lot of advantages in its design, for it allows us to have both inhibition and excitation weights to take place on the same synaptic junction of pulse coded MNTC by simply adjusting the PDC. Also we do not need signal manipulation or pre-processing for weighted

summation, like synchronization, pre-filtering, or flip-flops as seen in [Murr89, Find90, Sala91].

The simulations for each discrete PDC value and its corresponding steady-state output in the summation block are shown in Figure 14. Figure 15 shows the PDC versus the steady-state output Y_s . As can be seen and expected, as PDC increases, the corresponding steady-state output decreases and vice versa. Thus, we can say Y_s is a monotonically decreasing function of PDC., i.e.

$$Y_s = f_{(PDC)} \quad (3)$$

The function f is shown in Figure 15. At the point PDC equals 0.5, the expected steady-state value was 2.5V (half of the supply voltage). A small offset (+0.11V), however, was read from the simulation when PDC is 0.5. This offset is due to various secondary effects and does not have significant effect on the overall performance of the weighted summation.

Another aspect to be considered is the fact that a non-refreshed capacitive storage element (C_s) is used for the summation block where charges are summed in a temporal integration form. Here it is necessary to assume that the leakage from a capacitive element is negligible in time-wise, and thus, the circuit operates more rapidly than leakage affects the operation of weighted summation. This assumption is quite acceptable because the leakage from the capacitive element in VLSI package (in dark space) is known in the order of 10^{-15} A [fA] [Botc83] while the charging or discharging current to the capacitor (C_s) is in the order of 10^{-6} - 10^{-3} [μ A-mA]. Therefore, the time constant for leakage is roughly 10^9 less than that of charging or discharging operation, and as a result, the leakage effect in this design is negligible.

Threshold Block

In this work, the well-known differential amplifier stage [Alle87, Greb84] is used for sigmoid type of the non-linear operation. The circuit diagram is shown in Figure 16. It contains five transistors. The input pair M1 and M2 are N-type transistors and the current mirror (M3 and M4) is composed of P-type transistors. A current source is achieved by M5 operating in its saturation region. The two inputs are the output of the summation block (Y_s) and a threshold (θ). The threshold (θ) is an external input. The output of the threshold block is Y_{th} . Simulation result of the transfer curve of Figure 16 is shown in Figure 17.

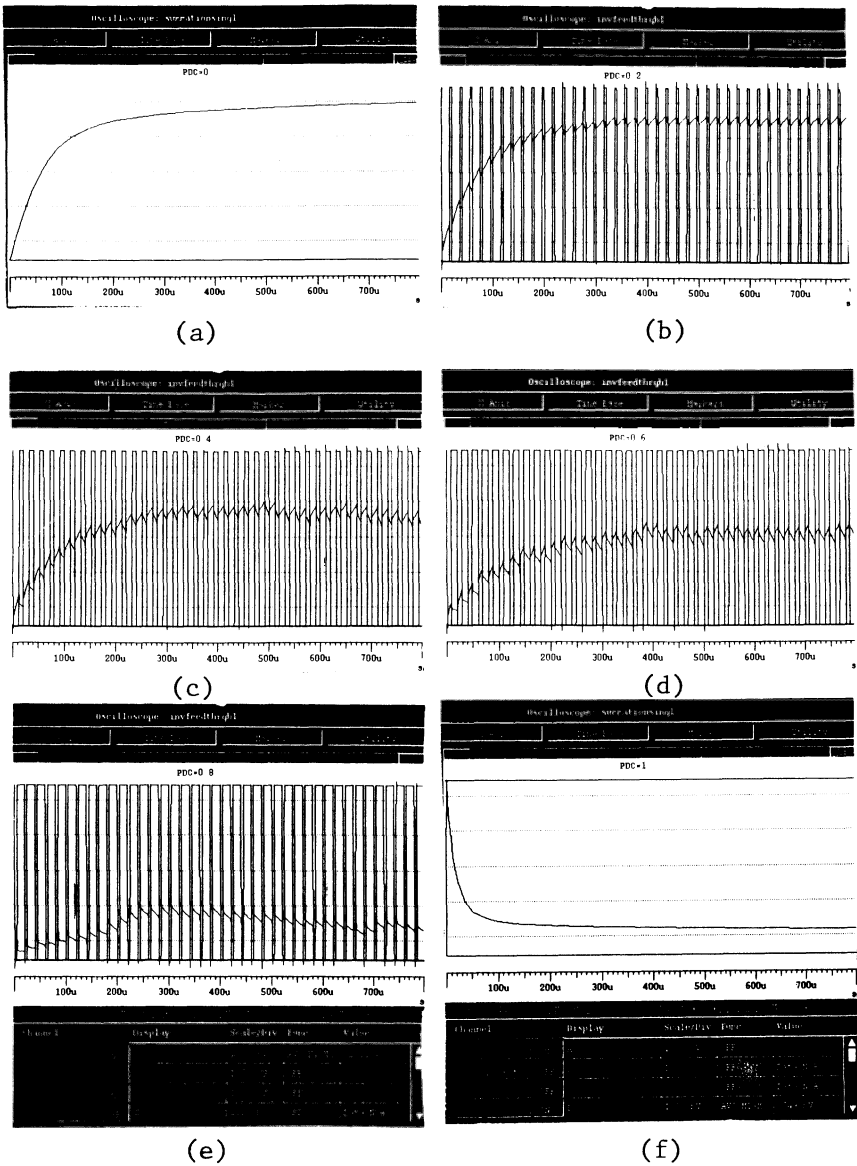


Figure 14. Simulation results of the relation between PDC and steady-state output (Y_s). PDC = a) 0, b) 0.2, c) 0.4, d) 0.6, e) 0.8, and f) 1.

As can be seen, as the difference between Y_s and θ sweeps along the x-axis, the S-shape of Y_{th} is acquired. So, if $Y_s > \theta$ then Y_{th} is high (5V), otherwise Y_{th} is low (0V). As shown in Figure 18 two inverters composed of M6-M9 are added in order to increase the current driving capability of Y_D .

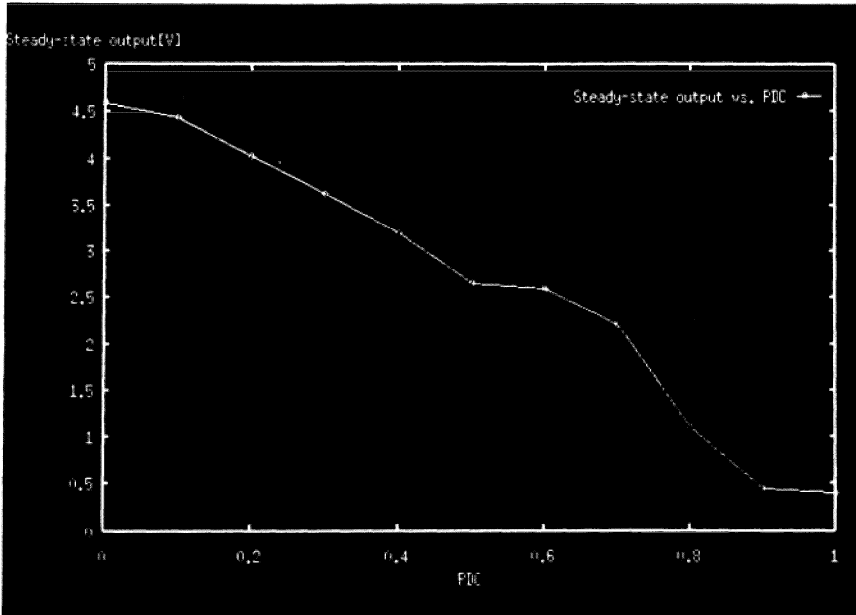


Figure 15. PDC versus steady-state output curve.

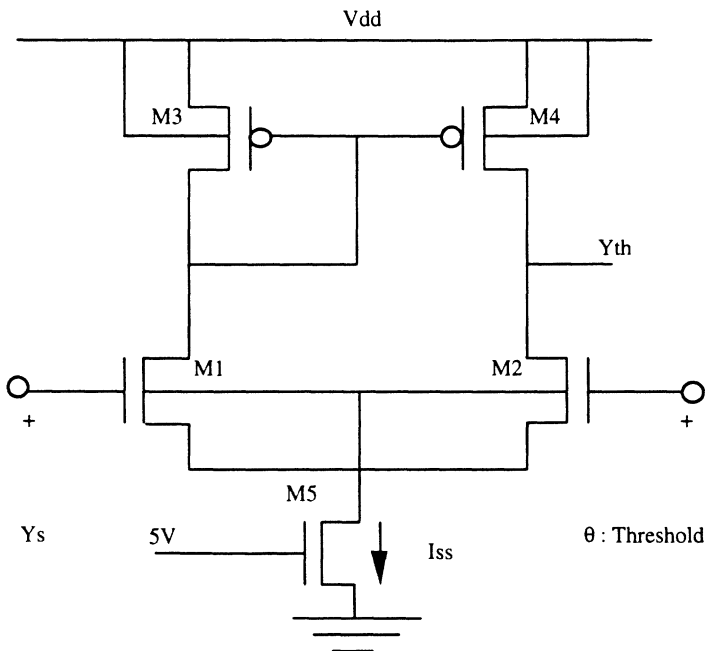


Figure 16. Circuit diagram of CMOS differential amplifier for sigmoid threshold.

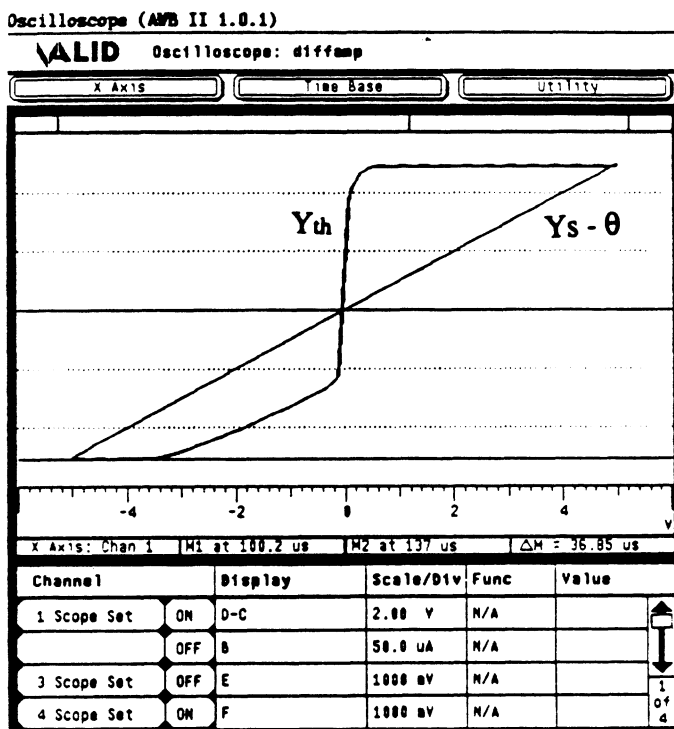


Figure 17. Sigmoid threshold curve of Figure 16.

Specifically, the first inverter (M6 and M7) is designed with minimum feature size of $3/2\mu\text{m}$, which will have minimum interference over the sigmoid operation of the differential stage. The second inverter (M8 and M9) is designed with rather large size (M8 = $40/2$, M9 = $20/2\mu\text{m}$) to accommodate enough current. These two inverters serve to sharpen the slope in the transient region and thus producing a hard limiter-like nonlinearity for YD. The circuit diagram for the threshold block and corresponding simulation result are shown in Figure 18 and 19, respectively

Adaptive Learning Block

An adaptive learning block is not necessarily needed for all networks depending on the application. It is not needed in the case of a predetermined fixed weight vector, with a simple non-linear mapping from the input into the output [Gros85, Feld81]. However, in the case of controlling the network **adaptively** for a special purpose or teaching the network to learn a specific application, an adaptive learning block is needed. This process is called **training or learning**.

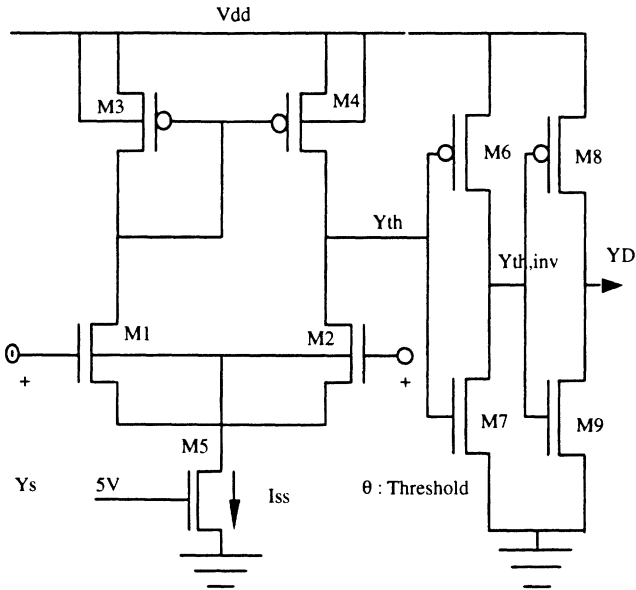


Figure 18. Circuit diagram for threshold block.

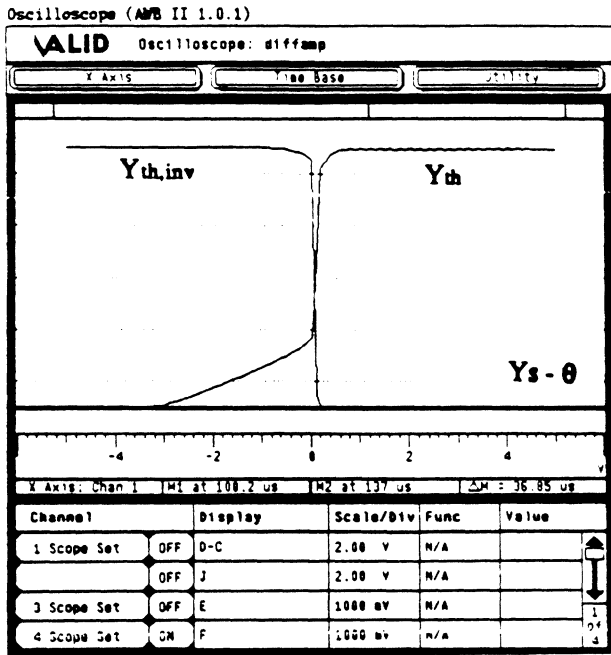


Figure 19. Simulation result of Figure 18.

Learning is the process by which the synaptic weights (or weight vector) are changed. Several different algorithms for learning have been studied by many authors [Hebb49, Widr73, Mins61].

To illustrate how a learning rule can be implemented in CMOS form, consider for example, the Hebbian learning rule [Hebb49] of

$$W_{ij}(\text{new}) = \alpha Y_i(\text{old}) Y_j(\text{old}) \quad (4)$$

where α is a learning constant, and W_{ij} is the weight between i -th and j -th neuron. Y_i and Y_j are outputs (Y_s or Y_D in Figure 11) from i -th and j -th neuron, respectively. Equation (4) is a multiplication between the two neurons' outputs. A single NMOS transistor may be used for this multiplication. Consider the circuit diagram shown in Figure 20.

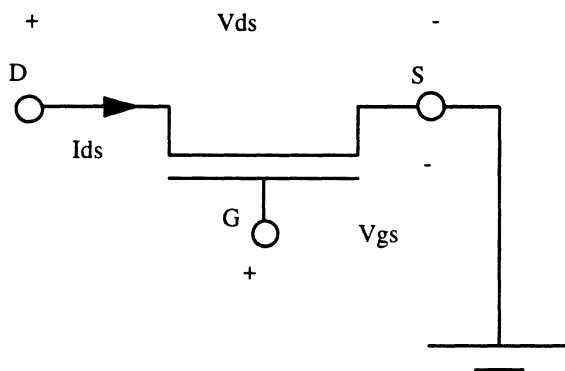


Figure 20. Single transistor multiplier.

When the NMOS transistor is in its linear region of operation, the drain-to-source current is given by [Alle87]

$$I_{ds} = \frac{\mu_n \epsilon_{ox}}{t_{ox}} \frac{W}{L} \left[(V_{gs} - V_t) V_{ds} - \frac{V_{ds}^2}{2} \right] \quad (5)$$

where W and L are the width and the length of MOS transistor, respectively, ϵ_{ox} is permittivity of dielectric material of the transistor gate, μ_n is electron mobility, and V_t is the threshold voltage of the MOS transistor.

For small values of V_{ds} , eq. (5) can be approximated to

$$I_{ds} = \frac{\mu_n \epsilon_{ox}}{t_{ox}} \frac{W}{L} (V_{gs} - V_t) V_{ds} = K (V_{gs} - V_t) V_{ds} \quad (6)$$

where K is the transistor gain factor and is a constant.

Thus, we can consider V_{gs} and V_{ds} as two variables for multiplication. Although it has an offset due to the threshold voltage, V_t , multiplication properties hold.

CMOS CHIP DESIGN AND MEASUREMENTS

A CMOS chip was designed and fabricated to test the design introduced above. The chip was designed so as to achieve an appropriate network configuration for a desired application. A group of basic module corresponding to each of the above functional blocks are stacked in horizontal rows in the layout. The Modified Neural Type Cells (MNTC) are stacked on top of the chip. Each basic cell has the same height with the V_{dd} metal line running horizontally along the top edge and the GND line running horizontally along the bottom edge. By abutting the cells horizontally a stack of synapse cells is constructed. In between metal lines are placed for interconnections among cells. The chip layout is shown in Figure 21.

Each basic functional block is shown in the layout of Figure 21. The chip was fabricated through MOSIS using 2μ double-metal P-well technology. Measurements were done in the VLSI Design/Test Laboratory at The George Washington University. In the following sections, results of separate tests for each of the functional blocks to verify the performance are illustrated. The design of the basic functional blocks above can be configured in many of the known neural models such as the Hopfield model, Perceptron, and Winner-Take-All models.

Modified Neural Type Cell Measurement

The single MNTC occupies $150\mu\text{m} \times 320\mu\text{m}$ (73 mil^2) silicon area. To illustrate the effect of the weight voltage on the output pulse duty cycle of the MNTC, a fixed input (x) was applied to the cell, and a varying weight voltage (W) was applied. Figure 22 shows the relation between the weight voltage (W) and the Pulse Duty Cycle (PDC) of the pulse stream output of the cell. As shown in Figure 22, as the weight decreases in Figs 22a to 22f the PDC increases.

The results of Figure 22 can be summarized as in Table 1. Comparing the results of Table 1 with the simulation results, we see a shift of the measured oscillation range of about +29% from the simulation. These shifts may be due to deviations for the fabrication process. However, the fabricated MNTC shows the intended function behavior of the design and this can be used as synapse junction in the Neural Processing Element (NPE).

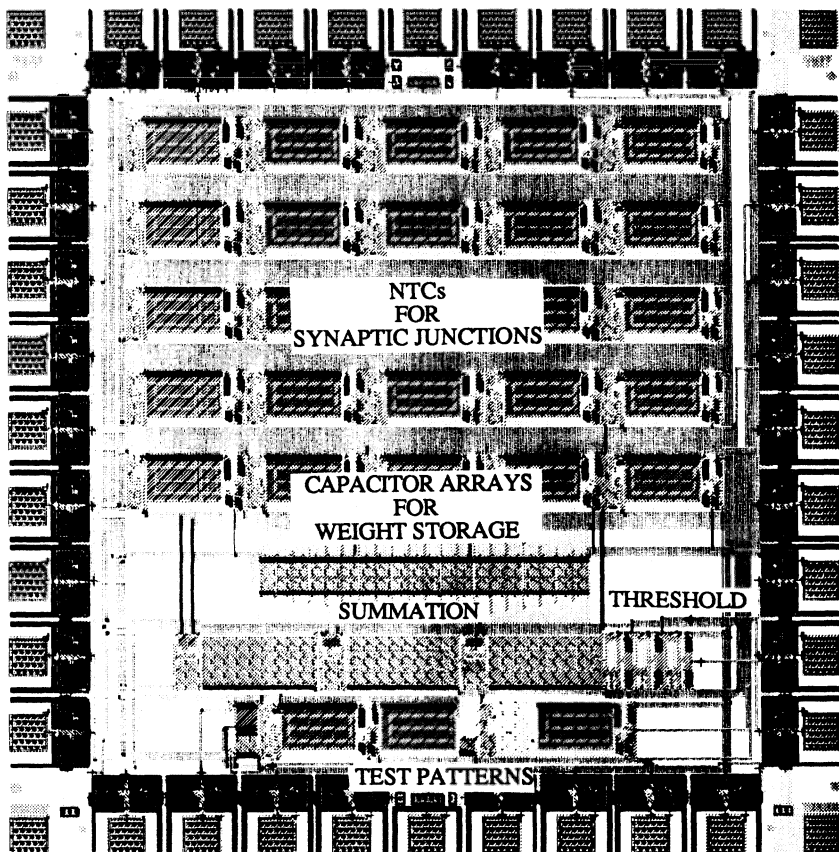


Figure 21. Chip Layout

Neural Processing Element Measurements

To verify the behavior of the NPE, measurements of the fabricated NPE was done. Two sets of measurements were done: one to measure the effect of the weight over the analog output Y_s , and the other to measure the effect of the input over the output Y_s . Figure 23 shows the measurements. In this case W_1 was held at $0v$, and thus Y_{p1} remains high, while W_2 was varied. As W_2 increases the output Y_s is also increasing. We note that although the frequency of Y_{p2} is increasing as the weight W_2 is increasing, the PDC of Y_{p2} is decreasing as the weight W_2 increased. Figure 24 shows the second set of measurements for the NPE. Two weights W_1 and W_2 are fixed, and the input is increasing. The results are summarized in Table 2, where it is clear that the output increased as the input increases.

W(volts)	PDC
4.66	0
4.37	0.49
4.31	0.61
4.25	0.85
4.20	0.97
4.10	1

Table 1. PDC vs weight with input $x=3.88v$

Cases	Ys(V)
(a) X=1.7	2.7
(b) X=2.1	2.9
(c) X=2.5	3.1
(d) X=3.0	3.5
(e) X=3.5	3.6
(f) X=4.0	3.7

Table 2. Ys changes with a varying input and fixed weights in two input NPE.

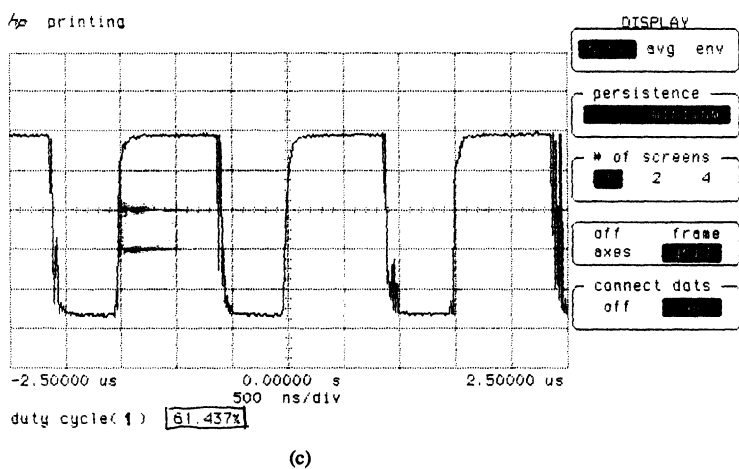
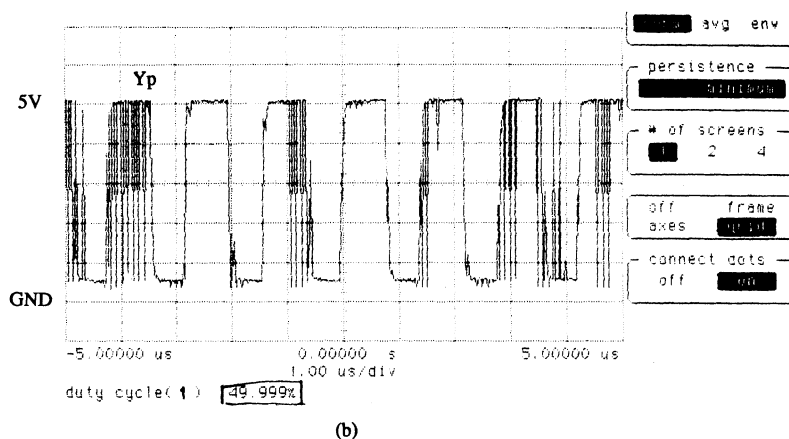
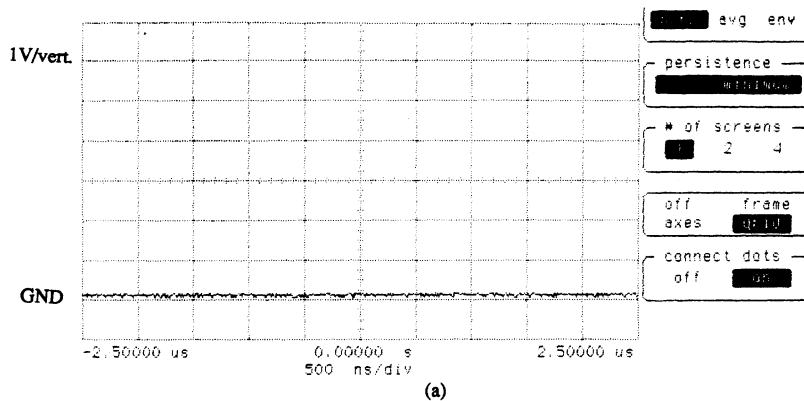
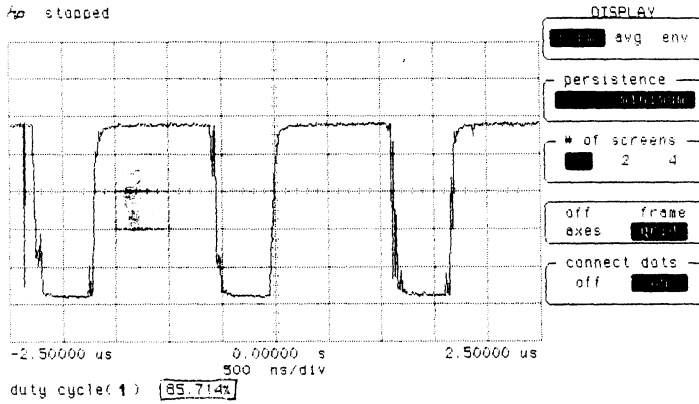
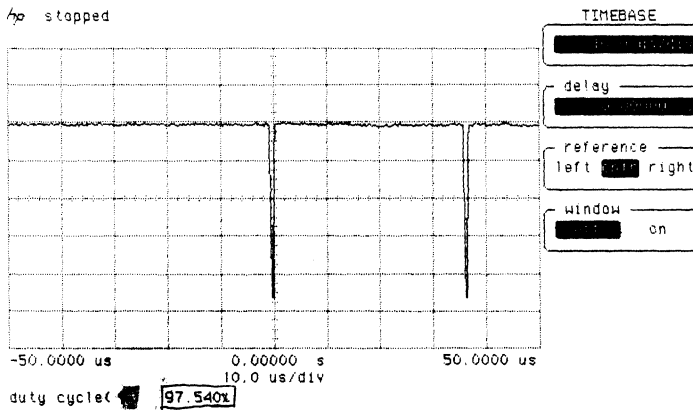


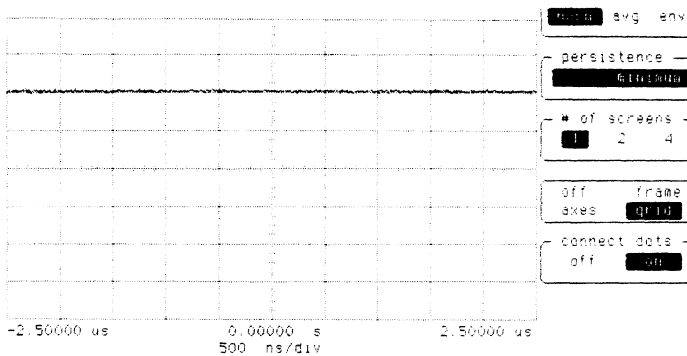
Figure 22. Functional test for MNTC with varying weight (W) and fixed input $x=3.88v$. a) $W=4.66v$, b) $W=4.37v$, c) $W=4.31v$



(d)



(e)



(f)

Figure 22. (cont.) Functional test for MNTC with varying weight (W) and fixed input $x=3.88v$. d) $W=4.25v$, e) $W=4.20v$, f) $W=4.1v$

CONCLUSIONS

In this chapter the Neural-Type Cell was used in the design of the Neural Processing Element. The NTC was modified to achieve design requirements for the NPE. The NPE may be used as a basic cell in different configurations of Artificial Neural Networks. For example, suitable models are Hopfield model, Perceptron, Winner-Take-All, and Autotracking [Moon93c]. This is possible because the introduced NPE is equipped with the basic functional properties of an artificial neuron. Analytical expressions for PDC have been derived in [Moon93b,c]. Measurements on the fabricated chip showed very much encouraging electrical performance which corresponds with the objective functions as well as the simulation results.

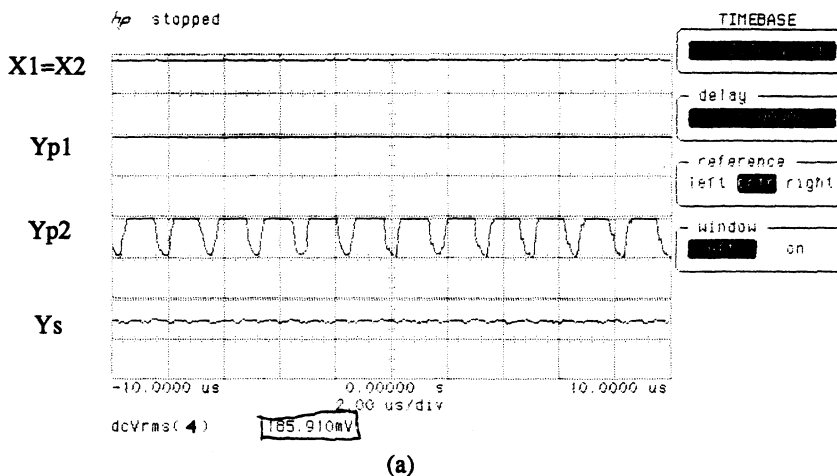


Figure 23. Function test of NPE with two fixed inputs and varying W_2

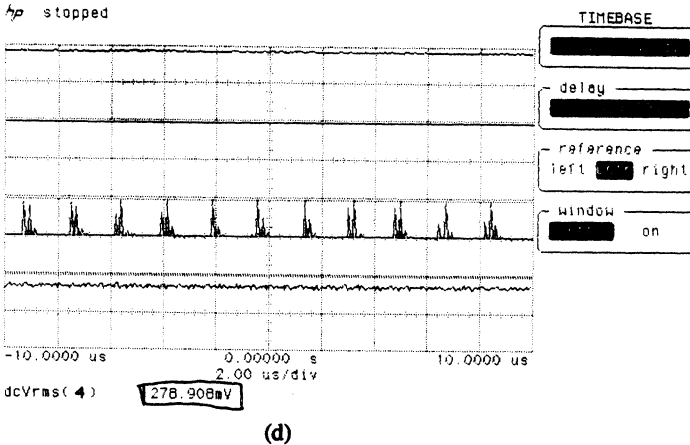
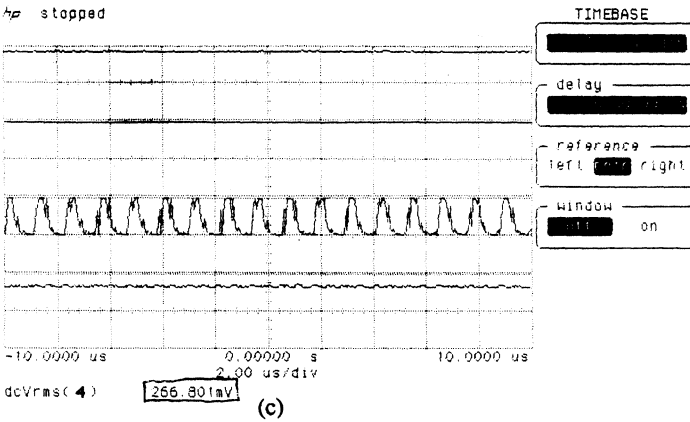
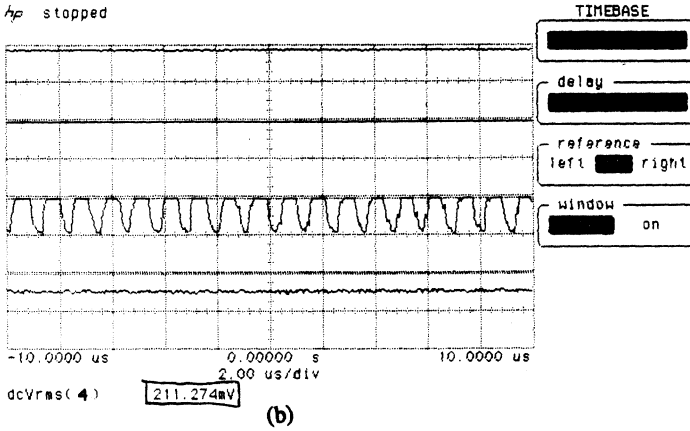


Figure 23. (cont.) Function test of NPE with two fixed inputs and varying W_2

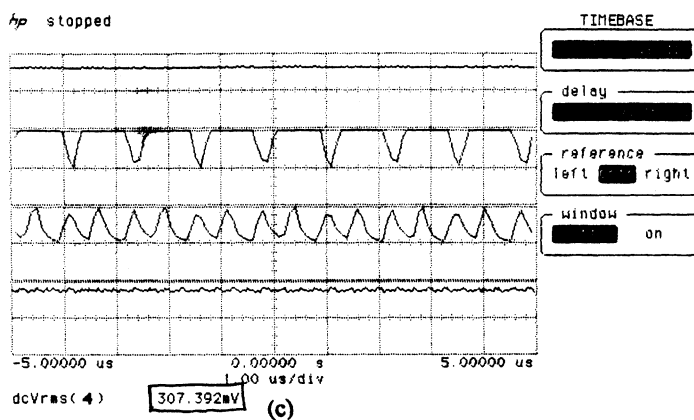
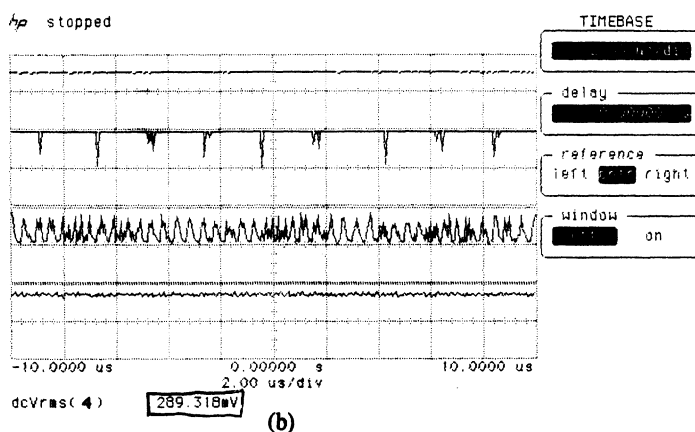
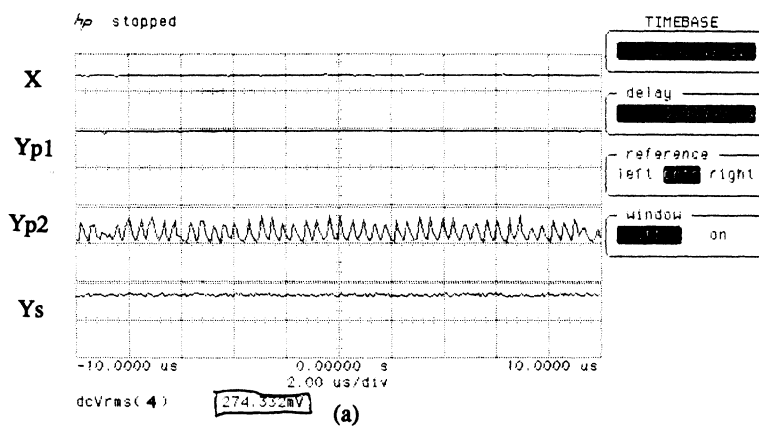


Figure 24. Functional test of NPE with two fixed weights and varying input ($x=x_1=x_2$, $W_1=3.39$, $W_2=3.82$)

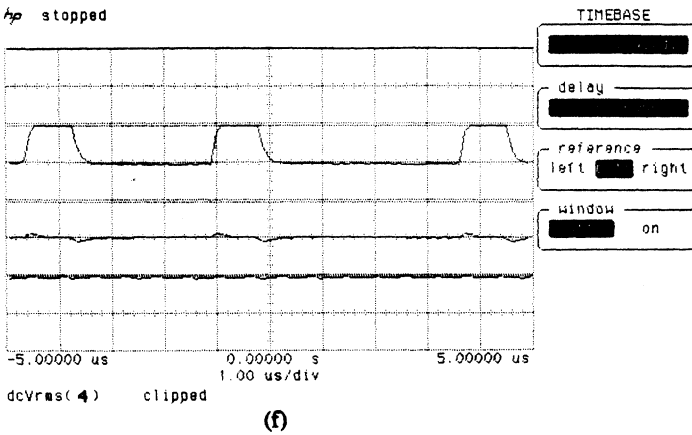
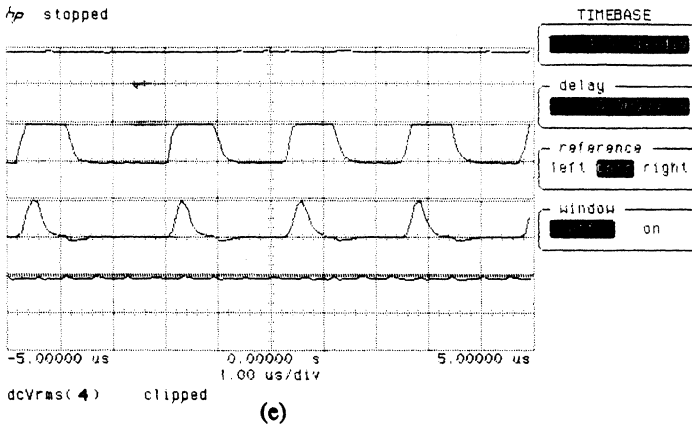
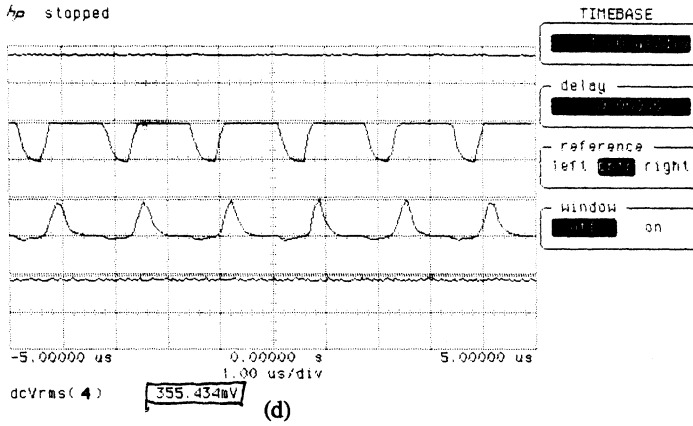


Figure 24. (cont.) Functional test of NPE with two fixed weights and varying input ($x_1=x_2$, $W_1=3.39$, $W_2=3.82$)

References

- [Alle87] P. Allen and D. Holberg, *CMOS Analog Circuit Design*, pp. 198-211, 273-287, Holt, Rinehart and Winston Inc., New York, NY, 1987.
- [Apos74] Tom M. Apostol, *Mathematical Analysis*, Addison-Wesley Co., 2nd Ed., pp. 94, 1974.
- [Awbm88] *Analog Workbench User Manual, Sun Version, Analog Design Tools*, Sunnyvale, CA, 1988.
- [Baba84] J. Babanezhad, G. Temes, "A linear NMOS Depletion Resistor and Its Application in an Integrated Amplifier," *IEEE J. Solid-State Circuits*, vol. SC-19, no.6, pp.932-938, 1984.
- [Bell88] D. Bell, *Solid-state Pulse Circuits*, Reston Publishing Co., pp. 8, 1988.
- [Botc83] C. Botcheck, *Basic MOS Engineering*, vol. 1, pp. 252-261, Pacific Technical Group, Inc., Saratoga, CA, 1983.
- [Carl86] A. Carlson, *Communication Systems*, New York: McGraw-Hill, 1986, pp. 308-314.
- [Cors90] D. Corso, L. Reyneri, "Mixing Analog and Digital Techniques for Silicon Neural Networks", *Proceedings of the 1990 IEEE International Symposium Circuits and Systems*, New Orleans, Louisiana, May 1990, pp.2446-2449.
- [Elle88] N. El-leithy, and R.W. Newcomb, "Hysteresis in Neural-Type circuits", *Proceedings of the 1988 IEEE International Symposium Circuits and Systems*, Espoo, Finland, June 1988, pp.993-996.
- [Find90] R. Findley, M. DeYong, C. Fields, "High Speed Analog Computation via VLSI Implementable Neural Networks," *Proc. Third Microelectronic Education Conference and Exposition*, San Jose, CA, pp. 113-123.
- [Feld81] J. Feldman, *Memory and change in connectionist networks*, Univ. of Rochester, Dept. of Computer Science Technical Report, TR-189.

- [Fitz61] R. FitzHugh, "Impulse and physiological state in theoretical models of nerve membrane," *Biophys. J.*, vol. 1, 1961.
- [Graf87] H. Graf and P. deVegvar, "A CMOS Implementation of a Neural Network Model," *Proc. 1987 Stanford Conf. Advanced Res. VLSI*, Losleben (ed.) MIT Press, 1987, pp.351-367.
- [Graf88] H. Graf, L. Jackel and W. Hubbard, "VLSI Implementation of a Neural Network Model," *IEEE COMPUTER magazine*, pp. 41-49, March, 1988.
- [Gray84] P. Gray, R. Meyer, *Analysis and Design of Analog Integrated Circuits*, John Wiley and Sons, Inc., 1984.
- [Gray80] P. Gray, D. Hodges, and R. Broderen, *Analog MOS Integrated Circuits*, IEEE Press, New York, NY, 1980.
- [Greb84] A. Grebene, *Bipolar and MOS Analog Integrated Circuit Design*, John Wiley and Sons, Inc., 1984.
- [Gros85] S. Grossberg, and E. Mingolla, "Neural dynamics of perceptual grouping: Textures, boundaries, and emergent segmentations", *Perception and Psychophysics*, vol. 38, pp. 141-171, 1985
- [Han84] I. Han, S.B. Park, "Voltage-Controlled Linear Resistor by Two MOS Transistors and its application to Active RC Filter MOS Integration," *Proc. of IEEE J. Solid-State Circuits*, vol. 23, no.1, pp.183-194, Feb. 1988.
- [Hebb49] D. Hebb, *The Organization of Behavior*, John Wiley & Sons, New York, 1949.
- [Hodg52] A. Hodgkin and A. Huxley, " A qualitative description for membrane current and its application to conduction and excitation in nerves," *J. Phys.*, vol.177, pp. 500-544, 1952.
- [Hopf82] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. U.S.*, vol. 79, pp. 2554-2558, April 1982.
- [Hopf85] J. J. Hopfield and D. W. Tank, " Neural Computation of Decisions in Optimization Problems," *Bio. Cybern.* 52, 141-152, 1985.

- [Kand85] E. Kandel and J. Schwartz, *Principles of Neuroscience*, Elsevier Publishing, New York, 1985.
- [Katz66] B. Katz, *Nerve, Muscle, and Synapse*, McGraw-Hill, New York, 1966.
- [Koho87] T. Kohonen, "Self-Organization and Associative Memory," Springer-Verlag, Berlin, 1987.
- [Lipp87] R.P. Lippman, "An Introduction to Computing with Neural Nets," IEEE Acoustic Speech and Signal Processing, pp. 4-21, April, 1987.
- [Magi91] J. Ousterhout, *Magic Tutorial*, Dept. of Electrical Engineering and Computer Science, University of Berkeley, CA, 1991.
- [Mavo83] J. Mavor, M. Jack and P. Denyer, *Introduction to MOS LSI Design*, Addison-Wesley Publishing Co. 1983.
- [Mccu43] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin Mathematical Biophysics*, 5, pp. 115-133, 1943.
- [Mead80] C. Mead and L. Conway, *Introduction to VLSI systems*, Addison-Wesley Publishing Company, 1980.
- [Mead89] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley Publishing Company, 1989.
- [Mead91] J. Meador, A. Wu, C. Cole, N. Nintunze, and P. Chintrakulchai, "Programmable impulse neural circuits," *IEEE trans. Neural Networks*, vol. 2, no. 1, pp. 101-109. January 1991.
- [Mins61] M. Minsky, *Neural-analog networks and the brain model problem*, Princeton University, Ph. D. Thesis.
- [Moon89] G. Moon, M. Zaghoul, R. Newcomb, "IC Layout for an MOS Neural Type Cell," *Proc. of IEEE CAS Mid-western conference*, Urbana, Illinois, pp.482-484, August, 1989.
- [Moon90a] G. Moon, M. Zaghoul, R. Newcomb, "An Enhancement-Mode MOS Voltage-Controlled Linear Resistor with Large Dynamic

- Range", *IEEE Trans. on Circuits and Systems*, vol. CAS-37, no. 10, pp.642-643, June, 1987.
- [Moon90b] G. Moon, "VLSI implementation of a Neural-Type Cell". *Master's Thesis*, Dept. of Electrical Engineering and Computer Science, The George Washington University, September 1990.
- [Moon91a] G. Moon, M. Zaghoul, M. Savigny and R. Newcomb, "Analysis and operation of a Neural-Type Cell," Proc. IEEE ISCAS, Singapore, pp. 2332-2334, June, 1991.
- [Moon91b] G. Moon, M. Zaghoul, and R. Newcomb, "VLSI Implementation of Neural-Type Cell with MOS Linear Resistors," Proc. of IEEE midwest conference, Monterey, California, pp. 784-787, May, 1991.
- [Moon92a] G. Moon, M. Zaghoul, and R. Newcomb, "VLSI Implementation of Synaptic Weighting and Summing in Pulse Coded Neural-Type Cells," *IEEE Trans. on Neural Networks*, vol. 3, no. 3, pp. 394-403, May, 1992.
- [Moon92b] G. Moon and M. Zaghoul, "CMOS Design of Pulse Coded Adaptive Neural Processing Element Using Neural-Type Cells," Proc. IEEE ISCAS, San Diego, pp. 2224-2227, 1992.
- [Moon93a] G. Moon, M. Zaghoul, and R. Newcomb, "An Improved Neural Processing Element using Pulse-Coded Weights," Proc. IEEE ISCAS, Chicago, pp. 2760-2764, May, 1993.
- [Moon93b] G. Moon, M.E. Zaghoul, and R.W. Newcomb, Pulse Coded Weights Neural Processing Element," submitted to IEEE Transactions on Neural Networks.
- [Moon93c] G. Moon, VLSI Design of Neural Networks Using Pulse Coded Weights with On-Chip Learning Capability, Doctoral Thesis, Department of Electrical Engineering and Computer Science, George Washington University, May 1993.
- [Mosi88] Mosis User Manual, The Information Science Institute of the University of Southern California USC/ISI in Marina del Rey, California, 1988.

- [Murr89] A. Murray, "Pulse Arithmetic in VLSI Neural Network", *IEEE MICRO*, vol. 9, no. 6, pp.64-74, December 1989.
- [Murr87] A. Murray, A. Smith, "Asynchronous arithmetic for VLSI neural systems", *Electronics Letter*, vol 23, no. 12, pp. 642-643, June, 1987.
- [Murr91] A. Murray, D. Corso, and L. Tarassenko, "Pulse-stream VLSI neural networks mixing analog and digital techniques," *IEEE trans. Neural Networks*, vol. 2, no. 2, pp. 193-204 March 1991.
- [Murr92] A. Murray, "Pulse Techniques in Neural VLSI: A Review", Proc. of 1992 International Conference of Circuits and Systems, San Diego, CA, 1992, pp. 2204-2207.
- [Nay83] K. Nay, A. Budak, "A Voltage-Controlled Resistance with Wide Dynamic range and low distortion," *IEEE Trans. on Circuits and Systems*, vol. CAS-30, no.10, pp.770-772, Oct. 1983.
- [Newc83] R.W. Newcomb, "Neural-type Microsystems Circuit Status", Proc. of the IEEE Internatioanl Symposium Circuits and Systems, Newport Beach, CA 1983, pp 97-100.
- [Newc79] R.W.Newcomb, "MOS Neuristor Lines", *Constructive Approaches to Mathmatical Models*, edited by Cliffman and G. Fix, Academic Press, 1979.
- [Nils65] N. J. Nilson, "Learning Machines: Foundation of Trainable Pattern Classifying System," McGrow-Hill, New York, 1965.
- [Rose57] F. Rosenblatt, "The perceptron: A Perceiving and recognizing automation," Cornell Aeronautical Laboratory Report, 85-460-1, 1957.
- [Rose62] F. Rosenblatt, "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanics," New York:Spartan Books, 1962.
- [Sala89] F. Salam, N. Khachab, M. Ismail and Y. Wang, "An Analog MOS Implementation of the Synaptic Weights for Feedback Neural Nets," Proc. of the IEEE Internatioanl Symposium Circuits and Systems, 1989. pp. 1223-1226.

- [Sala91] F. Salam, Y. Wang and R. Y. Choi, "On the Analysis and Design of Neural Nets," *IEEE Trans. on Circuits and Systems*, vol. CAS-38, no. 2. pp. 196-201, February 1991.
- [Sanc91] E. Sanchez-Sinencio, Guest Editorial, "Neural network circuit implementations," *IEEE Trans. on Neural Networks*, vol. 2, pp. 192. March 1991.
- [Savi90] M. Savigny, G. Moon, M. Zaghoul, N. El-Leithy, and R. Newcomb, "Hysteresis turn-on-off voltages for a Neural-Type Cell", *Proc. 33rd IEEE Midwest Symp. Circ. Syst.*, Calgary, Canada, August 1990. pp. 37-40.
- [Schi89] H. Schiff, G. Castelletti, G. Stefano, L. Iacino, "A Model for the Dynamic Properties of Integrating Fibers: Target Localization in a Three-Dimensional Space - I. Mathematical Description," *Comp. Biochem. Physiol.*, vol. 92A, no. 3, pp.331-341, 1989.
- [Simp90] P. K. Simpson, *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*, Pergamon Press, San Diego, 1990.
- [Szu92] H. Szu, K. Lee, G. Moon, B. Telfer, G. Rogers, M. Zaghoul, M. Loew and S. Schiff, "Collective Chaos in Neural Networks," *Proc. of International Joint Conference on Neural Networks*, Beijing, China, November 1992 (to be published).
- [Tre189] P. Treleaven et al., "VLSI Architectures for Neural Networks," *IEEE MICRO*, vol. 9, no. 6, pp. 8-27, December 1989.
- [Tomb90] J. E. Tomberg and K. Kaski, "Pulse-density modulation technique in VLSI implementations of neural network algorithms," *IEEE JSSC*, vol. 25, no. 5, pp. 1277-1286, October 1990.
- [Tsiv86] Y. Tsividis, "Continuous-Time MOSFET-C Filters in VLSI," *IEEE J. Solid-State Circuits*, vol. SC-21, no.1, pp.15-30, Feb. 1986.
- [West85] N. Weste, K. Eshraghian, *Principles of CMOS VLSI Design, A System Prospective*, Addison-Wesley, Massachusetts, 1985.

- [Widr73] B. Widrow, N. Gupta, and S. Maitra, "Punish / reward: Learning with a critic in adaptive threshold systems, *IEEE trans. on Systems, Man, and Cybernetics*, vol. SMC-5, pp. 455-465, 1973.
- [Widr90] B. Widrow and M. A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proc. the IEEE*, vol. 78, no. 9, pp. 1415-1442, September 1990.
- [Yous89] H. Youssef, R. Newcomb, M. Zaghoul, "A CMOS Voltage-Controlled Linear Resistor with Wide Dynamic Range," *Proc. of the 21st Southeastern Symposium on System Theory*, Talahance, Florida, pp.680-683, March 1989.

LOW-POWER SILICON NEURONS, AXONS AND SYNAPSES

John Lazzaro and John Wawrzynek

Computer Science Division

UC Berkeley, Berkeley, CA 94720

Power consumption is the dominant design issue for battery-powered electronic devices. Biologically-inspired sensory preprocessors may be an important component of portable computing devices that require real-world visual or auditory input. The silicon neural design style presented in [Mead 89, Andreou 91] naturally supports low-power VLSI design; in this design style, MOS transistors typically operate in the weak-inversion regime. The low-power performance of this design style is outstanding; for example, [Watts 91] reports on a 51-stage silicon cochlea, that computes all outputs in real time and consumes $11 \mu\text{W}$.

However, several popular circuits in this design style operate transistors outside the weak-inversion regime. These circuits, that model the spiking behavior of the axon hillock and the pulse propagation of axons, dominate power consumption in many neural chips [Lazzaro 89ab, Lazzaro 91, Horiuchi 91].

This chapter describes modified versions of these axon circuits. The modified circuits have been designed and fabricated and are fully functional; these circuits show a measured improvement in power consumption over the original circuits. Power consumption decreases of a factor of 10 to 1000 have been measured, depending on pulse width and spiking frequency. The density of the modified circuits is comparable to the original circuits. This chapter also describes describes several low-power synaptic circuits.

AXONAL CIRCUITS

Figure 1 shows the spiking neuron circuit from [Mead 89], that uses a high-gain voltage amplifier with a sigmoidal nonlinearity as a gain element. The circuit converts the unidirectional current I_i into a sequence of fixed-width, fixed-height voltage pulses of V_o . During a pulse $V_o = V_{dd}$, and between pulses V_o is at ground potential.

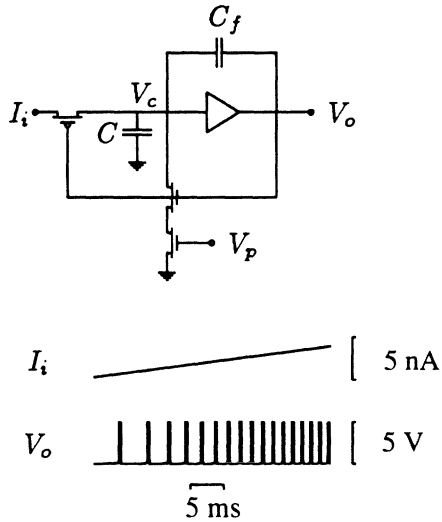


Figure 1. Spiking neuron circuit and function, with unidirectional current input I_i , voltage pulse output V_o , and pulse width control voltage V_p .

To understand circuit operation, consider the circuit condition after an output pulse has completed. In this state, V_o is at ground potential, and V_c is lower than the switching threshold of the amplifier. The discharge path of the state capacitor C is closed, and the charging path of C is open.

The circuit remains in this state until the input current I_i increases V_c to the switching threshold of the amplifier. At this point, V_o switches to V_{dd} . The feedback capacitor C_f ensures the secure switching of the circuit. The new value of V_c is above the switching threshold of the amplifier, and depends on the relative values of C_f and C .

Once a pulse begins, the discharge path of the state capacitor C is open, and the charging path of C is closed. The control voltage V_p sets the discharge rate of C , and thus the width of voltage pulse of V_o . The circuit remains in this state until V_c decreases to the switching threshold of the amplifier. At

this point, V_o switches to ground potential, and the pulse is complete. The voltage V_c is reset to a value below the switching threshold of the amplifier, that depends on the relative values of C_f and C .

This circuit, as described in [Mead 89], uses two digital inverters in series as the high-gain amplifier; Figure 2(a) shows this amplifier implementation. When used in the spiking neuron circuit, the transistors in first inverter of this amplifier are biased outside the weak-inversion regime. In many designs, the static current consumption of these transistors dominates the current consumption of the chip.

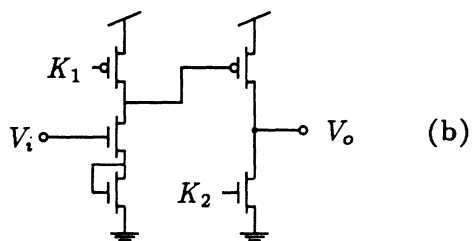
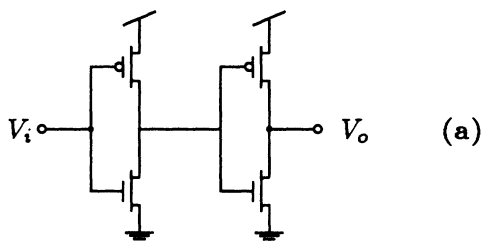


Figure 2. Original gain stage (a) and low-power gain stage (b). K_1 and K_2 are control voltages, set to ensure all transistors operate in the weak-inversion regime.

Figure 2(b) also shows a low-power implementation of a high-gain amplifier suitable for use in the spiking neuron circuit. The control voltages K_1 and K_2 limit the static current consumption of the amplifier. The response time of the amplifier is not symmetric; the speed of crossing the amplifier threshold in a positive direction is not limited by K_1 and K_2 , but the speed of crossing the threshold in a negative direction is directly dependent on K_1 and K_2 . In many applications, this asymmetry allows the bias currents of the amplifier to be set in the weak-inversion regime. The diode-connected transistor acts to raise the switching threshold of the amplifier.

AXONAL EXPERIMENTAL DATA

Figure 3 shows the static current consumption of the two amplifiers shown in Figure 2, as a function of the input voltage V_i . This figure shows data from a test chip fabricated in the $2\mu\text{m}$ double polysilicon n -well Orbit process as supplied by MOSIS. The current meter used in this measurement is not able to measure currents below 5 nA. As expected, the low-power amplifier consumes negligible current below its switching threshold, and a constant current above its switching threshold.

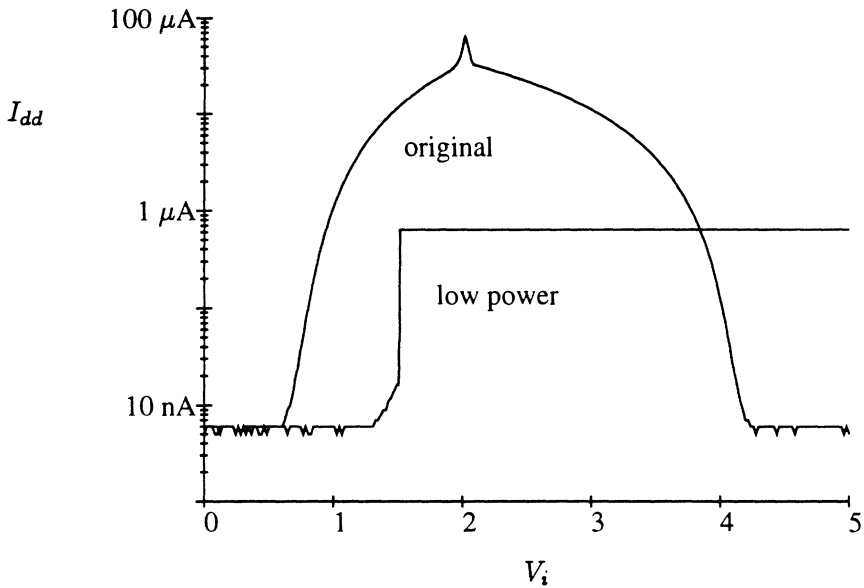


Figure 3. Power supply current I_{dd} for original gain stage and low-power gain stage, as a function of V_i . Note log scale for current.

Figure 4 shows the current consumption of the spiking neuron circuit of Figure 1, implemented with the original amplifier and the low-power amplifier. As expected, the current consumption of the low-power circuit is a linear function of spiking frequency, and increases with the size of the pulse width. All data was taken with the values of K_1 and K_2 necessary for correct circuit function with $10\mu\text{s}$ pulses; the current consumption for longer pulse-width operation can be reduced by adjusting K_1 and K_2 .

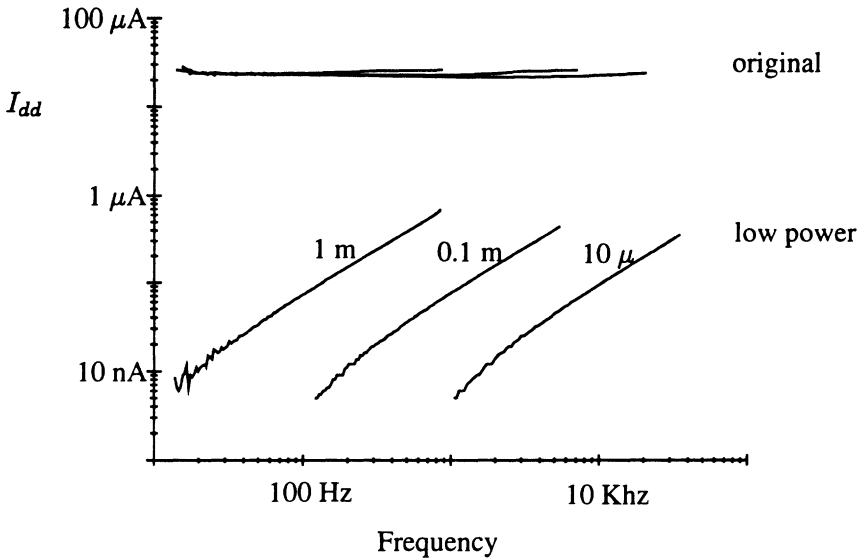


Figure 4. Power supply current I_{dd} for original spiking neuron circuit and low-power spiking neuron circuit, as a function of spiking frequency. Labels next to graphs indicate pulse width of spikes, in units of seconds. Note log scale for frequency and current.

THE AXONAL DELAY CIRCUIT

Figure 5 shows one section of the axonal delay line circuit described in (Mead, 1989). In engineering terms, the circuit is a cascade of non-retriggerable monostables, each with a voltage output V_k , that is either at V_{dd} or at ground potential. To understand circuit operation, consider the condition $V_k = V_{dd}$, $V_{k+1} = 0$, $F_{k+1} = 0$. In this case, the voltage V_c as shown in Figure 5 is below the switching threshold of the amplifier. The discharge path of the capacitor C is closed, and C is charged by a current set by the voltage $V_k - V_p \equiv V_{dd} - V_p$. When the voltage V_c increases to the switching threshold of the amplifier, V_{k+1} changes to V_{dd} , and the axon stage associated with V_{k+2} begins a similar charging cycle. Once this cycle has completed, the voltage F_{k+1} switches to V_{dd} and the capacitor C associated with V_{k+1} quickly discharges, causing the voltage V_{k+1} to switch back to ground. In this way, a constant width voltage pulse propagates down the structure.

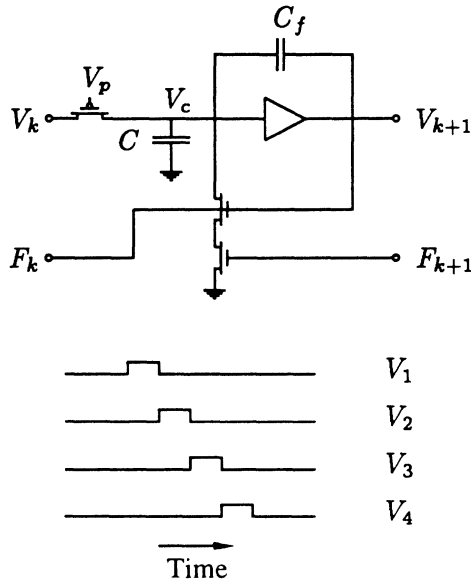


Figure 5. Axon circuit and function. Labels include ports V_k and F_k to previous stage, ports V_{k+1} and F_{k+1} to next stage, pulse width control voltage V_p , state voltage V_c , state capacitor C and feedback capacitor C_f .

As with the spiking neuron circuit, the amplifier in each neuron circuit, if implemented with two digital inverters in series, consumes appreciable static current. Replacing the original amplifier with the low-power amplifier reduces the current consumption of an axonal delay circuit. Figure 6 shows the current consumption of an 18-stage axonal delay circuit, implemented with the original amplifier and the low-power amplifier.

For pulse widths of $100 \mu\text{s}$ and 1 ms , current consumption of the low-power axon circuit is a linear function of spiking frequency. For a pulse width of $10 \mu\text{s}$, the state capacitors C are not fully discharged with a single pulse, and some additional static current consumption occurs. As with the spiking neuron circuit, the current consumption of the low-power axon circuit also depends on pulse width length. All data was taken with the values of K_1 and K_2 necessary for correct circuit function with $10 \mu\text{s}$ pulses; the current consumption for longer pulse width operation can be reduced by adjusting K_1 and K_2 .

The original axon circuit has a single free parameter, V_p , that sets both the speed of pulse propagation and the width of each pulse. The low-power axon circuit provides two additional parameters, K_1 and K_2 . These parameters

permit the pulse width and propagation speed to be set independently, allowing overlapping pulses in successive taps.

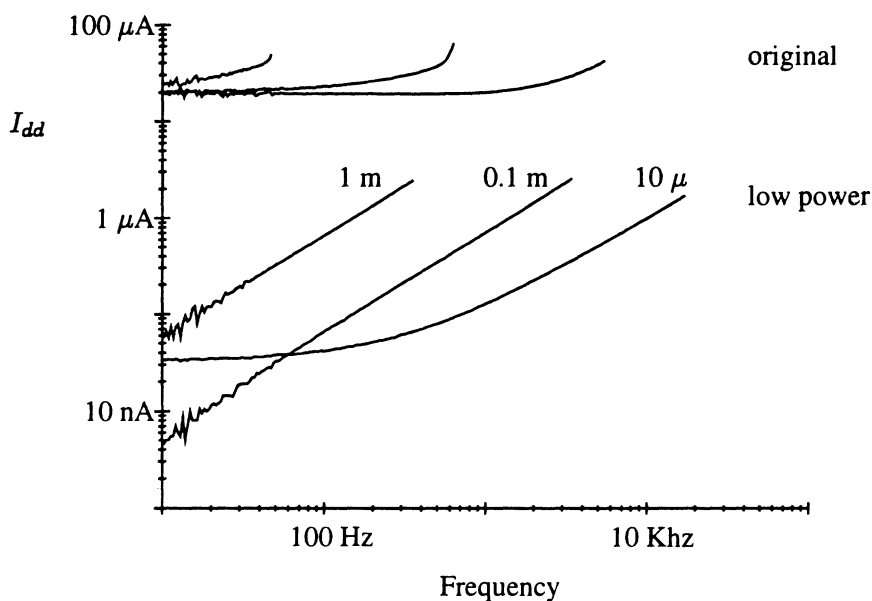


Figure 6. Power supply current I_{dd} for original axon circuit and low-power axon circuit, as a function of spiking frequency (18 sections). Labels next to graphs indicate pulse width of spikes, in seconds. Note log scales for current and frequency.

SYNAPTIC CIRCUITS

This section describes several different types of low-power synaptic circuits. Referring to Figure 1, synaptic circuits convert the pulse output representation of the signal V_o into a unidirectional weak-inversion current signal suitable for connection in I_i . Different synaptic circuits perform different types of signal processing during the conversion. All of the circuits in this section have been fabricated as components in functional systems, except where indicated.

Figure 7 shows simple synaptic circuits. The circuit in Figure 7(a) [Mead 89] is a very simple synapse, converting a downward voltage pulse into the unidirectional current output I_o . The magnitude of the current pulse is set by the control voltage V_w , and the width of the current pulse reflects the width of the input voltage pulse.

In some situations, the width of the input voltage pulse is very small and not under voltage control; in particular, pulse outputs from spiking neuron circuits used in off-chip communication have this property. [Lazzaro 93]. The synapse circuit shown in Figure 7(b) is designed to produce an output current pulse I_o that is wider than the input voltage pulse. The control voltage V_τ sets the width of I_o , and the control voltage V_w sets the magnitude.

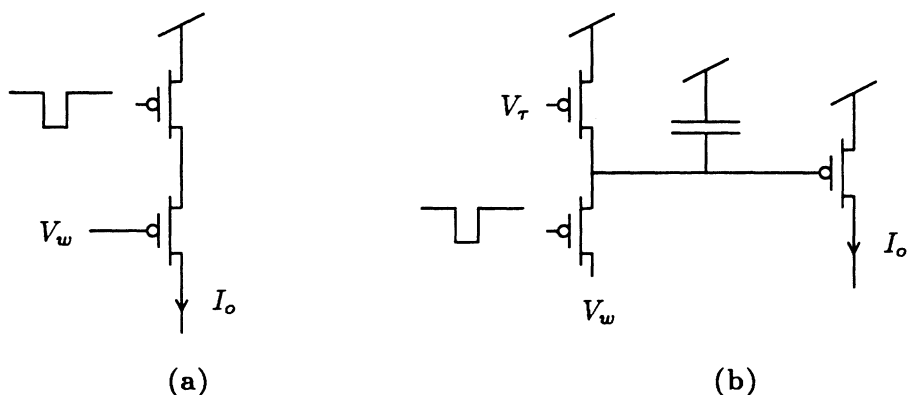


Figure 7. Synapses circuits, producing output currents I_o .

The circuits in Figure 7 allows voltage pulses to be multiplied by a weight set by V_w , and summed together using Kirchoff's current law. Circuits shown in Figure 8 are useful for performing signal processing on this aggregate current signal. These circuits transform a unidirectional input current I_i into a unidirectional output current I_o .

The circuit in Figure 8(a) includes two control voltages, V_1 and V_2 , which can be used together to scale I_i by factors greater than or less than unity. If all transistors are operating in the weak-inversion regime, the equation

$$I_o = I_i e^{(\kappa/2)(V_1 - V_2)/V_o}$$

describes the operation of the circuit, where $V_o = kT/q$ and κ is a fabrication constant as described in [Mead 89].

The circuit in Figure 8(b) is similar to the circuit in Figure 8(a), but uses the currents I_1 and I_2 to scale the input current. If all transistors are operating in the weak-inversion regime, the equation

$$I_o = I_i \sqrt{I_1/I_2}$$

describes the operation of the circuit.

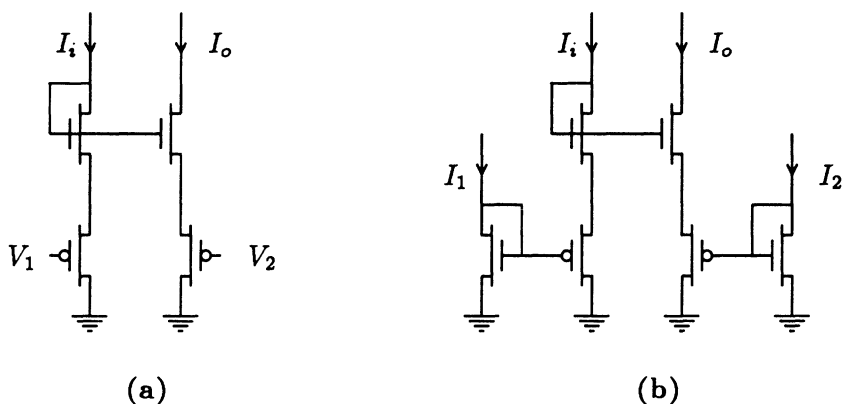


Figure 8. Scaling circuits for currents: $I_o = I_i \alpha(I_1, I_2, V_1, V_2)$.

The scaling circuits of Figure 8 can be used as components for circuits that implement adaptation or facilitation. Figure 9(a) shows an adaptation circuit, that has been used in an auditory nerve model [Lazzaro 92]. The primary input of this circuit is I_i , a current usually obtained by summing the response of many simple synapses. The output of the circuit, I_o , is a scaled version of I_i . If the auxiliary input of this circuit (marked with a pulse) is inactive for an extended period of time, the circuit performs unity scaling. If the auxiliary input is active, however, the output I_o is a reduced version of I_i . The temporal characteristics of the adaptation are set by the control voltages V_u and V_d ; depending on the values of these parameters, the circuit can produce time constants as long as several seconds.

Figure 9(b) shows a facilitation circuit that operates on a similar principal to the adaptation circuit of Figure 9(a). In this circuit, auxiliary input activity scales I_o to be larger than I_i ; a lack of auxiliary input activity performs unity scaling. An addition control voltage, V_l , limits the maximum scaling of the circuit. This circuit simulates correctly but has not be used in a fabricated system.

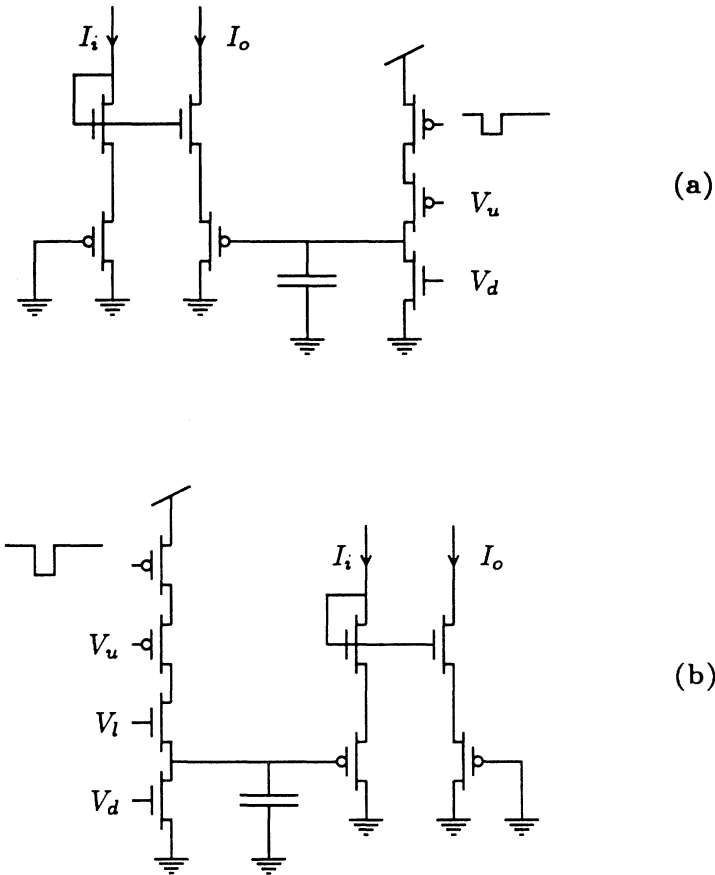


Figure 9. Short-term adaptation and facilitation circuits.

The circuits in Figure 9 are both controlled by a single auxiliary input. This input can be replaced by a logical combination of several inputs, to produce a circuit suitable for learning algorithms.

CONCLUSIONS

This chapter reviews low-power circuit technology for spiking neurons, axons, and synaptic circuits. Using the circuits from this chapter, the reader should be able to recast many of the circuits from other chapters of this book into low-power designs.

Acknowledgements

Spiking neuron and axon research was performed at CU Boulder, and funded by the National Science Foundation; all other research performed at UC Berkeley, and was funded by the NSF (PYI award MIPS-895-8568), AT&T, and the ONR (URI-N00014-92-J-1672). Thanks to Carver Mead for his encouragement of this research.

References

- [Andreou 91] Andreou, A.G., Boahen K. A. , and Pouliquen, P. O., Current-mode subthreshold MOS circuits for analog VLSI neural systems, *IEEE Transactions on Neural Networks*, 2:2, p. 205.
- [Horiuchi 91] Horiuchi, T., Lazzaro, J. P., Moore, A., and Koch, C. , A correlation-based motion detection chip, Tourestzky, D. (ed), *Advances in Neural Information Processing Systems 3*, San Mateo, CA: Morgan Kaufmann Publishers.
- [Lazzaro 89a] Lazzaro, J. P. and Mead, C.A., Silicon models of auditory localization, *Neural Computation*, 1: 41--70.
- [Lazzaro 89b] Lazzaro, J. P. and Mead, C., Silicon models of pitch perception, *Proc. Natl. Acad. Sci. USA*, 86, pp. 9597--96 01.
- [Lazzaro 91] Lazzaro, J. P., A silicon model of an auditory neural representation of spectral shape, *IEEE Journal Solid State Circuits*, Circuits, 26: 772--777.
- [Lazzaro 92] Lazzaro, J. P. (1992), Temporal adaptation in a silicon auditory nerve, Moody, J., Hanson, S., and Tourestzky, D. (eds) *Advances in Neural Information Processing Systems 4*, San Mateo, CA: Morgan Kaufmann Publishers.
- [Lazzaro 93] Lazzaro, J. P., Wawrzynek, J., Mahowald, M., Sivilotti , M., Gillespie, Silicon auditory processors as computer peripherals, *IEEE Journal Neural Networks*, (in press).
- [Mead 89] Mead, C. A., *Analog VLSI and Neural Systems*, Reading, MA: Addison-Wesley.

- [Watts 92] Watts, L., Kerns, D. A., Lyon, R. F., and Mead, C. A., Improved implementation of the silicon cochlea, *IEEE Journal Solid State Circuits*, **27**: 5, 692-700.

SYNCHRONOUS PULSE DENSITY MODULATION IN NEURAL NETWORK IMPLEMENTATION

Jouni Tomberg

Department of Electrical Engineering

Tampere University of Technology, Tampere Finland , SF-333101

INTRODUCTION

Artificial neural networks (ANN) are massively parallel, distributed information processing structures [Wasserman 89]. They consist of huge amount of processing elements interconnected via weighted connections. The idea for these networks is based on the biological world, but their models are considerable simpler. According to the biological models the processing elements are called "neurons" and the weighted connections "synapses". The signal lines from neurons to synapses and from synapses to neurons are called "axons" and "dendrites", respectively. Although the structure of neurons and synaptic connections is relatively simple, the large amount of them needed for practical applications makes the implementation of ANNs quite complicated. One effective way to implement ANNs is VLSI circuits. In the simplest case a neuron can be modeled by nonlinear summing amplifier. The weight values of the synaptic connections, which are responsible for information storage, can be implemented by resistors of different strengths. The learning process changes these weight values according to some specified rule. From the information processing point of view a synapse can be considered as a multiplier which does the product of the incoming neuron value with the stored weight value. A neuron then adds together the output values of the synapses and performs a nonlinear function for the resulting sum. Because of wide range of different ANN algorithms we often need slightly more complicated neuron structure

and the strength of the synaptic connections must be easily programmable. The major goal is to find an effective method to implement the ANN structures on VLSI circuits. Both analog and digital structures can be used [Graf 89]. The main advantages of analog implementations are simple basic blocks and communication which leads to denser chips. The restrictions of these implementations are the difficulties in analog weight storage and noise immunity. On the other hand the digital structures are straightforward to design and off-chip communication and expandability is better. The area required by the digital structures is larger leading often to smaller networks on a single chip. However, the digital structures can better use the advantage of new sub-micron fabrication processes. Thus, the best implementation technique depends strongly on the application. In many cases the mixed ana/digi techniques seem to give the best results. One interesting approach for this kind of implementation is the pulse-density modulation (PDM) technique. In a restricted sense it mimics the biological idea of neuron action using both analog and digital structures.

SYNCHRONOUS PDM TECHNIQUE

In the pulse-density modulation technique the signal value is represented by statistically distributed events occurring asynchronously in time. We have expanded this definition also for synchronous pulse streams. Thus, in the pulse-density arithmetic numbers are represented as streams of digital bits, 0 and 1 [Tomberg 90a]. We interpret the bits in the stream as the sign bits of the two's complement numbers, i.e. zero is + and one is -. The value of a pulse-density number depends on the relation of 0's (+) and 1's (-) in a given window moving along the time axis. Furthermore, we define that all the values are fractional numbers between -1 and +1. Thus the value of a bit stream including N zeros and M ones is $(N-M)/(N+M)$ and by a number including P bits we can represent (P+1) values (Fig. 1).

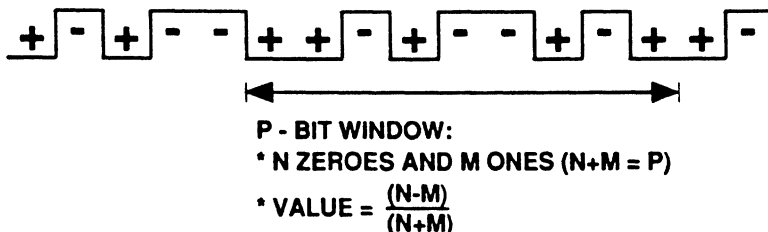


Figure 1. Representation of a PDM-coded value.

The value of a pulse-density stream is continuous, i.e. we can take a sample of P bits any time from the stream and it represents the initial value. In

the normal binary arithmetic we can represent 2^P values with a P-bit word. In that case the overall structure of the arithmetic is much more complicated and requires more area on a chip. In contrast the arithmetic of pulse-density numbers is quite straightforward to realize. The multiplication of two numbers can be done by using a simple bitwise exclusive-or (XOR) function. The addition is done within a defined window by summing the bit values together. The resulting pulse-density bit stream is then defined according to the resulting sum. One should bear in mind that in a pulse density number each bit in the stream has the same weight. This makes the number less sensitive to bitwise errors and the representation more robust than the ordinary binary arithmetic.

PDM ARITHMETIC AND STRUCTURES

The implementation of PDM arithmetic can be done by using digital or analog structures. Both of them have some benefits and drawbacks concerning the implementation and accuracy. Although the arithmetic structures are simple in the PDM technique the data storage for synaptic weights is quite large. If we want to use weight values with P levels we need P-bit register in PDM technique but only $\log_2(P)$ in ordinary binary arithmetic. However, the PDM- weight value can be stored by using dynamic structures which takes less area than the static ones. One interesting approach would be to use dynamic CCD (charge coupled device) structures [Soclof 85] for weight memory to minimize the area. Certainly there is a trade-off where the weight data stored in the PDM format takes more area on silicon than the logic of the ordinary binary arithmetic. Thus, the PDM technique is effective only for implementations where the weight data length is not so large that the benefit of the simpler arithmetic structures will be lost. This break even point depends on the implementation technique, i.e. how small the dynamic data storage structure is compared with the logic structures. Another important point when comparing these different methods is the operation speed. If the top level processing has massive parallelism but the data processing in the lower level is done serially we can lose the benefit compared with the situation when the low level processing is done parallel but the top level processing sequentially. We must take this into account when using the serial PDM technique.

Addition

In the case of N P-bit PDM numbers (P+1 levels) we need N*P bits and clock cycles to represent the final PDM sum. This is of course similar with the ordinary binary arithmetic with the distinction that larger amount of bits is needed for the number representation. The simplest way to implement the digital "neural" PDM addition is to multiplex in time the resulting bits from

synaptic multiplications to dendrite line and sum them together in the neuron [Tomberg 91]. This addition in a neuron can be mixed with the non linearity function and be done within defined window by using ordinary digital binary adders or analog structures. The total circuit area is not so sensitive for the increased area in neuron as in synapse. When using the multiplexing scheme in large networks the amount of bits and, thus, clock cycles will be too high for practical applications.

Another method is to use ordinary serial adders in the dendrite line to get a digital bitwise sum which is sent to the neuron. In this case we need only $\log_2(N)$ clock cycles to perform each bitwise sum. However, the overall control structure would be more complicated. If we use mixed analog/digital technique we can do the bitwise adding by using analog amplitude modulation. Adder structure in each synapse adds or subtracts charge from the common dendrite line depending on the result of the synaptic multiplication. In this case the resulting sum on the dendrite line will remain P-bits long, but each "bit" will have several levels. This is very area effective implementation but the main restriction is the accuracy of the analog structures.

Multiplication

Generally multiplication is implemented by adding the multiplicand recursively to itself as many times as the multiplier defines. In the PDM-multiplier (Fig. 2) a simple XOR-function is used instead of a full-adder which is the case in the ordinary binary arithmetic. Thus, we go through the multiplier bit by bit and add the whole multiplicand into the pulse stream XORed with the multipliers bit. If we have two P-bit numbers (P+1 levels) the result will be P^2 -bit long and it takes P^2 clock cycles to perform. In the ordinary serial binary multiplier the data is $\log_2(P)$ bits long (B bits, P levels) and result would be $2 \cdot \log_2(P)$ bits requiring in serial case $(\log_2(P))^2$ clock cycles.

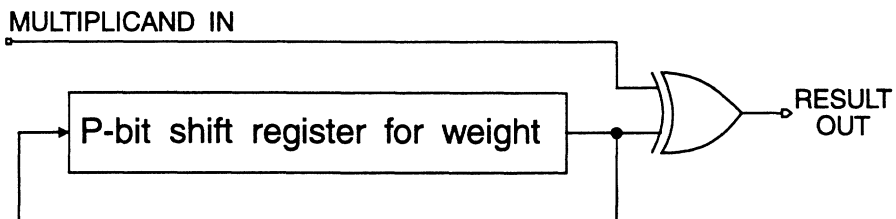


Figure 2. Serial PDM multiplier.

To reduce the amount of clock cycles in the PDM multiplication we can use a serial parallel structure (Fig. 3). Here the multiplicand is coming in serially multiplied by the parallel PDM weight value. The result would be P times parallel P -bit numbers connected further to the adder on the dendrite line. In this case the multiplication takes only P clock cycles. If we use serial-parallel multiplier in the ordinary binary arithmetic we would need only $2 \cdot \log_2(P)$ clock cycles for the multiplications. But in this case the area is much larger since several full-adders are needed.

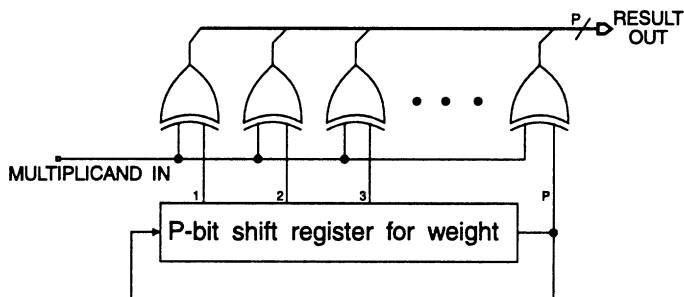


Figure 3. Serial/parallel PDM multiplier.

In Table 1 the ordinary binary multipliers and PDM multipliers are compared with different data lengths. In the comparison two different dynamic shift register structures for weight storage are used. In the 6 transistor case the implementation is an ordinary dynamic CMOS shift register with NMOS transmission gates. In the 3 transistor case a more advanced CCD type shift register technique is used. In the ordinary binary serial/parallel multiplier a static register structure for weight value is used because the weighted bits are fed parallel into the multiplier. This requires more area than the dynamic structures. In the case of PDM value all bits have the same weight and, thus, the value in the serial/parallel PDM multiplier can be moving around the register continuously. This makes the use of minimum area dynamic structures possible.

From Table 1 we see that the smallest area would be achieved by the serial PDM multiplier for 17 level weight values and 3 transistor weight storage. In the 257 level case the ordinary binary multiplier takes much less area. However, these numbers include only the area of the multiplier. The ordinary binary arithmetic requires many control structures and signals to synchronize the data flow. The data stream in the PDM technique is continuous without any other synchronizing signals than the global clock.

METHOD	BITS	LEVELS	SPEED (Clock Cycles)	WEIGHT STORAGE (Trans.)	SIZE (Trans.)
PDM Serial	16	17	256	6	102
				3	54
	256	257	65 536	6	1 542
				3	774
PDM Serial/Parallel	16	17	16	6	160
				3	144
	256	257	256	6	2 560
				3	2 304
Ordinary Binary Serial	4	16	16	6	96
				3	84
	8	256	64	6	152
				3	128
Ordinary Binary Serial/Parallel	4	16	8	8	224
	8	256	16	8	448

Table 1. Efficiency estimations of the different multiplier structures.

It is obvious that when using many levels in the weight values the ideal synaptic multiplication using PDM technique requires more area than the ordinary binary arithmetic and too many clock cycles to be effective. However, when combining the multiplication and addition operations using mixed analog structures the PDM technique offers effective structures for neural arithmetic.

Simplified Arithmetic

Because the PDM numbers have equal weight for each bit and the data stream is continuous some simplifications can be used in implementing the arithmetic. Thus, the required area is decreased and speed increased. If we assume that the density of zeroes and ones in a pulse stream is smoothly and 'randomly' distributed inside a 'window' (length of the number), we can perform the multiplication by taking a simple bitwise XOR-operation between the multiplier and multiplicand. In long run this gives a good approximation of the ideal result in P-bit window. In this case we need only P clock cycles for

multiplication instead of P^2 as was the ideal case. If we assume that the multiplicand has most of the time values near to the maximum or minimum, i.e. the bit stream is dominantly zero or one, this method gives a good approximation for the result even though the bit stream has not ideal random distribution. This non-ideality happens with a strong non linearity (e.g. step function) in the neuron output. This step function simplification has been used with success in our earlier VLSI implementations [Tomberg 89ab, 90].

Nonlinearity Function

In the neuron the non linearity function is performed for the synaptic sum coming from the dendrite line. In the most simplified case a step function non linearity is used. In digital case it can be implemented by adding the bit values from the PDM stream together within a defined window and setting the output to zero if the sum is positive and one if negative. In the analog case a special lossy switched-capacitor integrator which operates as a low pass filter following a highly nonlinear amplifier can be used. The output of the integrator has a value that is related to the last input values. The step function is performed by an amplifier with high gain. In the simplest case the amplifiers can be implemented by using cascaded CMOS inverters [Tomberg 89a].

In some cases more complicated (e.g. sigmoid shaped) non linearities are necessary. For example the backpropagation algorithm [Wasserman 89] needs a non linearity function which is differentiable everywhere. In digital implementation we must define a logic which changes the PDM value in the output register according to the incoming PDM value. This logic defines the used non linearity. In most complicated cases it can be a look-up table. In analog case we have the incoming PDM value in analog format at the output of the integrator. Thus, we can use analog structures to generate the non linearity function. After this the analog value must be converted back to PDM value. This can be done by using a sigma-delta modulator structure [Gray 87] which provides similar pulse stream representation as used in the PDM technique.

Weight modification

The learning process in neural networks is based on the changing the synaptic weight values according to some specified rule. Thus, especially if on-chip learning is desired, the weight values should not be only loadable to the circuit but also on-chip modifications should be possible. We have used a structure where the weight value can be increased or decreased by one unit using two control lines [Tomberg 91].

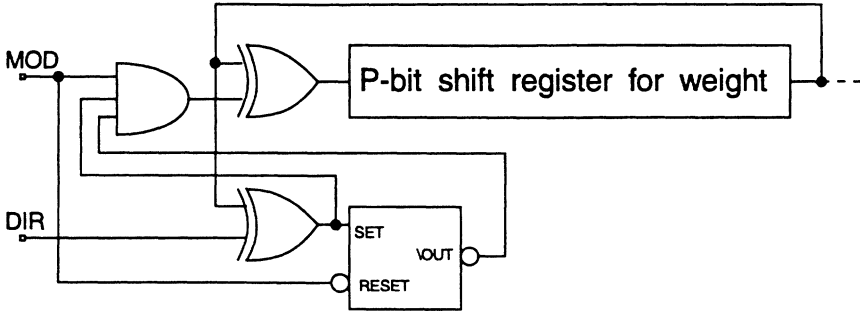


Figure 4. Weight modification logic for the PDM structure.

In the PDM technique the weight modification logic is quite simple (Fig. 4) compared to ordinary binary logic where some extra logic is needed to prevent the overflow situations. In the PDM technique the modification is done by going through all the bits in the weight and changing one bit from one to zero or from zero to one if the value is increased or decreased, respectively.

PDM STRUCTURES FOR IC IMPLEMENTATIONS

Numerous IC implementations of artificial neural networks consisting of tens up to hundreds of neurons on a single chip have been designed during the last years. The switched-capacitor technique has been proposed for some of the neural network implementations [Tsvividis 88]. It offers an effective method for mixed analog/digital implementations. The biologically oriented pulse-stream methods has also been found very promising among many other suitable techniques [Graf 89, Murray 88, Hirai 89]. Limitations of the implementations are usually the network size, the simplifications of the algorithm due to the restrictions of the used technique, and the off-chip interface for application purposes. These limitations could be alleviated by new techniques, such as wafer scale integration and optoelectronic devices.

Next we will discuss two ways of implementing the PDM synapse and neuron structures in silicon using both analog switched-capacitor (SC) and fully digital techniques. These techniques offer effective ways for large and expandable network implementations. The advantage of the switched-capacitor structure is its small area and thus larger networks can be implemented in a single chip. In contrast the digital structure offers better connectivity between individual chips and thus very large networks can be obtained by connecting chips together. Both structures are very modular and thus expandable in a chip.

Analog Switched-Capacitor Structures

In the SC-technique resistors can be substituted by clocked capacitors. Thus instead of currents in resistor networks one deals with charges in switched-capacitor networks [Tsividis 88, Tomberg 89c]. For example a clocked capacitor structure shown in Figure 5 with a capacitor of value C and clocking frequency f , can replace a resistor of value:

$$R = 1/(f \cdot C) \quad (1)$$

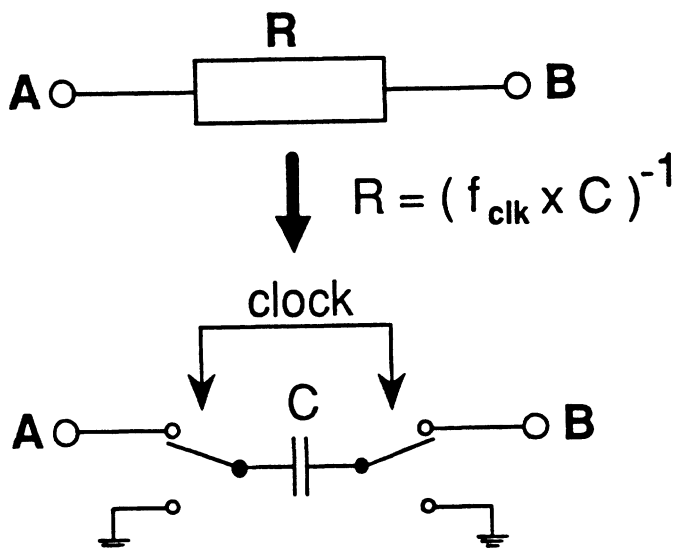


Figure 5. Switched-capacitor (SC) technique.

The result of the synaptic multiplication, performed by the XOR-gate, is connected to control the switched-capacitor structure. The weight value is running around the shift-register and the XOR-operation is performed between the current axon value and the bit of the weight value. The sum of the synapse outputs, i.e. "unit charges", is formed in the dendrite line. If the result of the multiplication is "-" the synapse is adding charge to the dendrite line and in case of "+" it is subtracting it. In Figure 6 we show the structure of the switched-capacitor synapse.

Apart from the nonlinear transfer function the neuron together with the dendrite line acts like an adder. It forms the sum of the synaptic outputs coming along the dendrite line and then performs a nonlinear function to the sum. The first part of the neuron is a lossy switched-capacitor integrator which

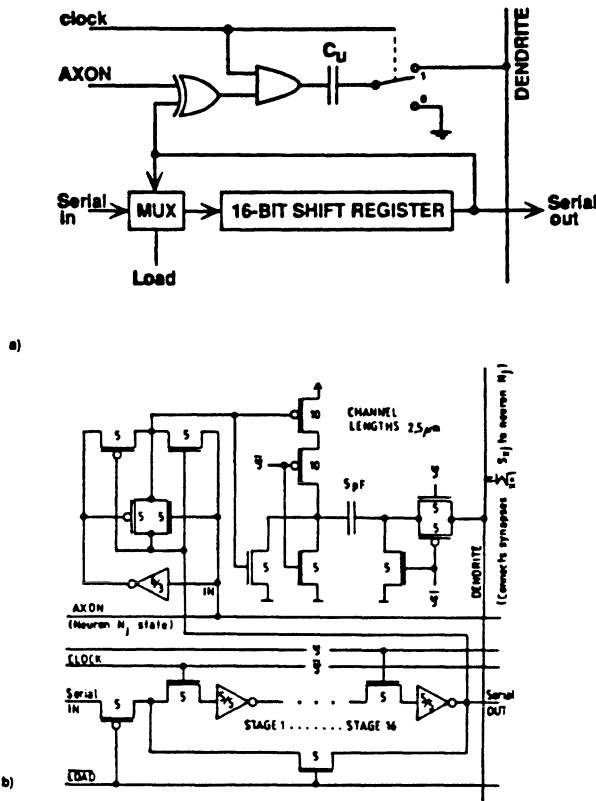


Figure 6. SC implementation of a PDM synapse: (a) block diagram and (b) transistor level diagram.

operates as a low-pass filter. This integrator forms the sum of the incoming pulse values inside the defined window. The sum is then fed to the second part which is a nonlinear amplifier structure implemented by serially connected inverters. The shape of the transfer function of an inverter is ideally sigmoid like. In this case we have maximized the amplification by cascading several inverters. Thus, the output of this stage is either high or low depending on the "sign" of the sum formed by the integrator. Therefore, this structure converts the output signal of the integrator from the voltage domain to pulse-density domain by using the step function shaped non linearity. The last inverter is designed as a tri-state structure to allow high impedance state during the initialization period. The structure of the switched-capacitor neuron is shown in Figure 7. During the initialization period the network is in the write state so

that the outputs of the neurons are controlled to be in high impedance states.

The basic idea of the lossy integrator structure is that the output has always a value which is related to the last input values. For simplicity the amplifiers were chosen to be inverters. If we implement the low-pass part by using resistors and capacitors the bandwidth (BW) and amplification (A) of such a structure are:

$$BW = 1/(2*p*R_1*C_f) \quad (2)$$

$$A = -R_1/R_2. \quad (3)$$

If on the other hand these resistors are replaced by switched-capacitor structures whose switching frequency is f the bandwidth and amplification will be:

$$BW = (f*C_1)/(2*p*C_f) \quad (4)$$

$$A = -C_2/C_1. \quad (5)$$

Thus the bandwidth is directly dependent on the switching frequency. Because we have used 16-bit weight registers and the bit frequency equals the switching frequency, the maximum reasonable bandwidth is $f/16$, though also smaller bandwidths can be used. Such structures give smoother distribution of the pulses and thus better reliability, but increase the convergence time of the network. The advantage of the structure is that the resulting capacitor values are independent on the frequency. This is of course true only for a limited frequency range because of the non-idealities, such as the resistance of the transmission gate switches. Nevertheless, we gain a wider range of proper operating frequencies. The amplification of the structure is chosen to be one.

As described above the data is processed in this circuits both in digital and analog form. Data is moving from neuron to synapse in digital form and from synapse back to neuron in analog form. The synaptic weight values are stored in digital PDM form, synaptic multiplication and adding operations are performed by analog structures and during the non linearity function in neuron data is converted again to digital PDM form. Because of the PDM data representation the conversion between digital and analog forms is very easy.

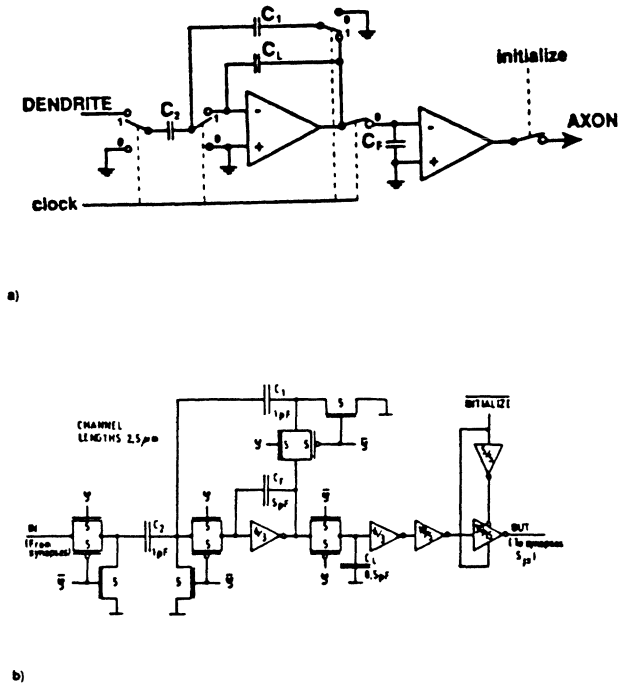


Figure 7. SC implementation of a PDM neuron: (a) block diagram and (b) transistor level diagram.

Digital Structures

The fully digital structures offer straightforward implementation for many different fabrication technologies and also off-chip expandability, which we see very important. However, the realization of the digital pulse-adder in the synaptic dendrite line seemed to be tricky. The sum of the synaptic outputs of one synapse matrix column is formed by chaining the pulse-adders of synapses together by a dendrite line. The one bit pulse stream between synapses makes it possible to connect separate chips together with a minimum amount of extra interface pins.

The synapse structure using simplified pulse-adder block leads to an effective IC implementation (Fig. 8). In this case the result of the bitwise multiplication, the input from the previous synapse and the sum can have values -1 ("-") or +1 ("+"), as represented by one bit. On the other hand the carry values -1, 0 or +1 require one more bit for their representation. We have used the algorithm in which the bitwise result saturates to value -2 or +2, i.e. the sum and carry signals have values -1 or +1 as the sum reaches or exceeds either -2 or +2, respectively. When the sum is zero two successive bits are set

to be complementary. To prevent the clamping of the carry value after saturation the carry input signal does not affect the carry output signal. Thus, the second bit of the carry signal is the same as the sum output and we will save one flip-flop in the implementation. In some cases the synapses near the neuron in this pulse-adder chain can block out the effect of other synapses that are further up in the chain. This is because of the overlapping and saturation conditions, which can occur if pulse-density coded weight values near to +1 or -1 are used. This will especially ruin the operation of large networks, in which two synaptic nodes can block out hundreds of others. The carry signal in the pulse-adder is used to compensate this effect. We could further minimize this effect by modifying the pulse adder, but this would lead to much more complicated structure and the required area of the synapse would increase dramatically. In contrast our aim was to minimize the required structures and obtain larger implementations.

This problem can also be compensated by modifying the learning rule in such a way that the overlapping and saturation conditions are minimized. This can be done by collecting the statistics of the training vectors during the learning and modifying the synaptic PDM weight codes in such a way that the sum of the individual bits in the pulse stream is minimized. This is easiest for simple networks like fully connected content addressable memory algorithms. The performance can be increased more by re-arranging the synaptic connection matrix according to the training set. The synaptic connections between the neurons with highest positive or negative correlation should be placed higher in the matrix, because their possibility to block out the others is larger. This method has been tested by using PC-controlled test board [Palovuori 91]. In spite of the good results which we have obtained in many tasks, the learning performance is also case sensitive for general purpose applications.

To increase the overall performance of the digital synapse structure, we have also tested a version where we replaced the pulse-adder by conventional bit adders even though it makes the operation more complex. There are basically two ways to implement an ideal digital bit adder. We can use an ordinary bit-serial adder in each of the synapses. In this case we need $\log_2 N$ - bits to represent the bitwise sum of the array with N neurons. Therefore, the weight register should be shifted once for $\log_2 N$ clock pulses of the adders. The control signals of this implementation are somewhat more complicated.

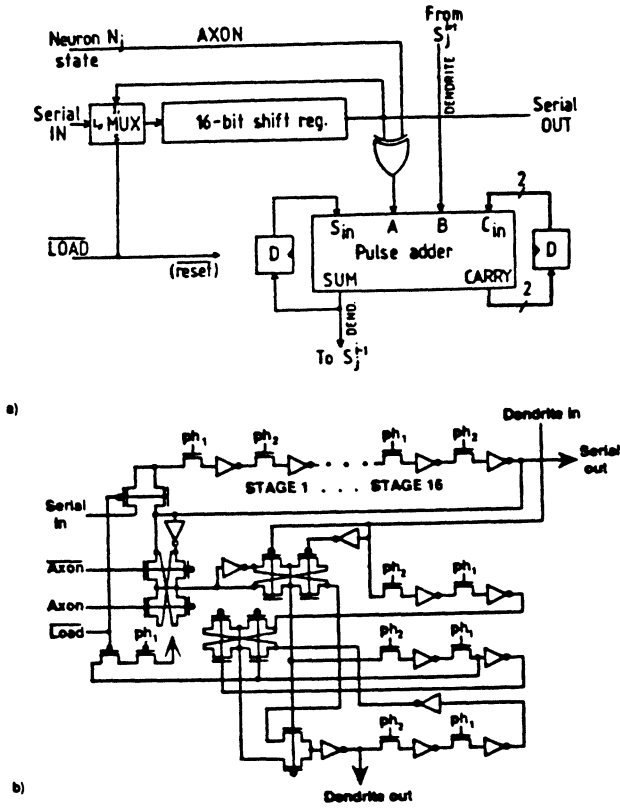


Figure 8. Fully digital implementation of a PDM synapse: (a) block diagram and (b) transistor level diagram.

We must know how many neurons we have in the network, so that we know how many bits and thus clock cycles are needed for each bitwise sum. However, we were looking for even simpler structures. Because the synaptic matrix takes most of the area in the circuit we wanted to minimize the synapse logic. Thus, we end up to an implementation where the bits from the synapses are multiplexed in time and the summing is done in the neuron (Fig. 9).

In this case we need N clock cycles to calculate each bitwise sum, but the clock frequency can be higher because of the simpler structures along the dendrite line. Furthermore, we obtain an automatic configuration of the network, i.e. we do not need to tell for the network how many neurons are connected in it. Each neuron in the end of the dendrite line sends up a pulse

which enables the synapses one at a time. The last synapse sends this pulse back to the neuron, which ends the bitwise summing process.

For realizing the neuron we have used a step function non linearity which in many algorithms seems to give the same results as the ideal sigmoid function. This is implemented by calculating the sum of bits, i.e. -1's ("-") and +1's ("+") inside the chosen 16-bit window (length of the weight register), and setting the output to zero if the result is positive and to one if it is negative.

In the case of simplified pulse-adder synapse structure the 16-bit window is formed by a 16-bit delay line and the sum can be formed by a recursive adder structure in which the input bit of the window is added to the sum and

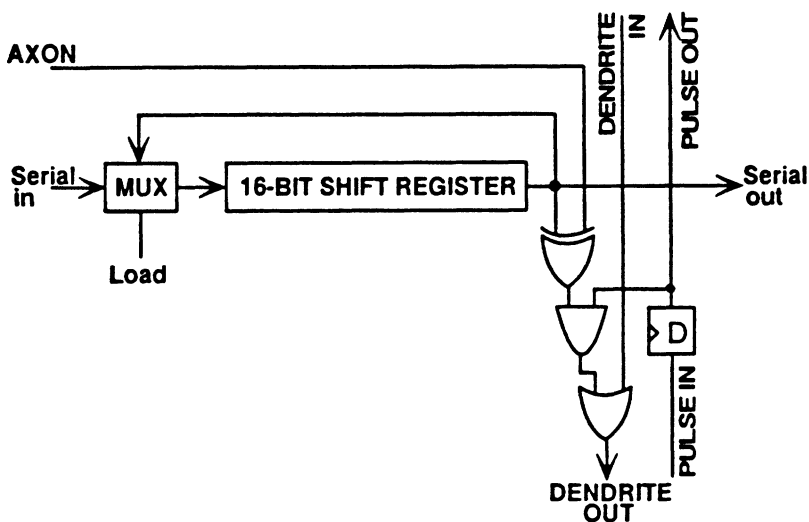


Figure 9. Time multiplexed digital PDM synapse structure.

the output bit is subtracted. This keeps the sum of the last 16-bits always in the adder [Tomberg 89b]. Thus, the number of the bits in the adder of the neuron is independent on the network size. We have optimized the adder structure to be a simple 16-bit dynamic bi-directional shift register which implies up-down counting (Fig. 10).

When the network is initialized, the delay register of the neuron is filled with the initial value. This is done by selecting the input of the shift register from the I/O-pads by WRITE-signal. The neuron can operate in two different modes. In the normal mode the output of the neuron is formed as described above. In the "network" mode the nonlinear function is not performed and the output is the same as the input, i.e. the output of the

synapse. This mode is used when the chip is cascaded together with other chips to form a larger network.

When the ideal bit-adder implementation of synapse structure is used, the amount of the bits in the neuron structures depends on the network size. Therefore we need $\log_2 N$ bits more for the sum and the delay line in the neuron, where N is the number of neurons in the fully connected network. In this case the restriction for the expandability is the size of the adder in the neuron and we should implement an adder large enough for the expansion purposes.

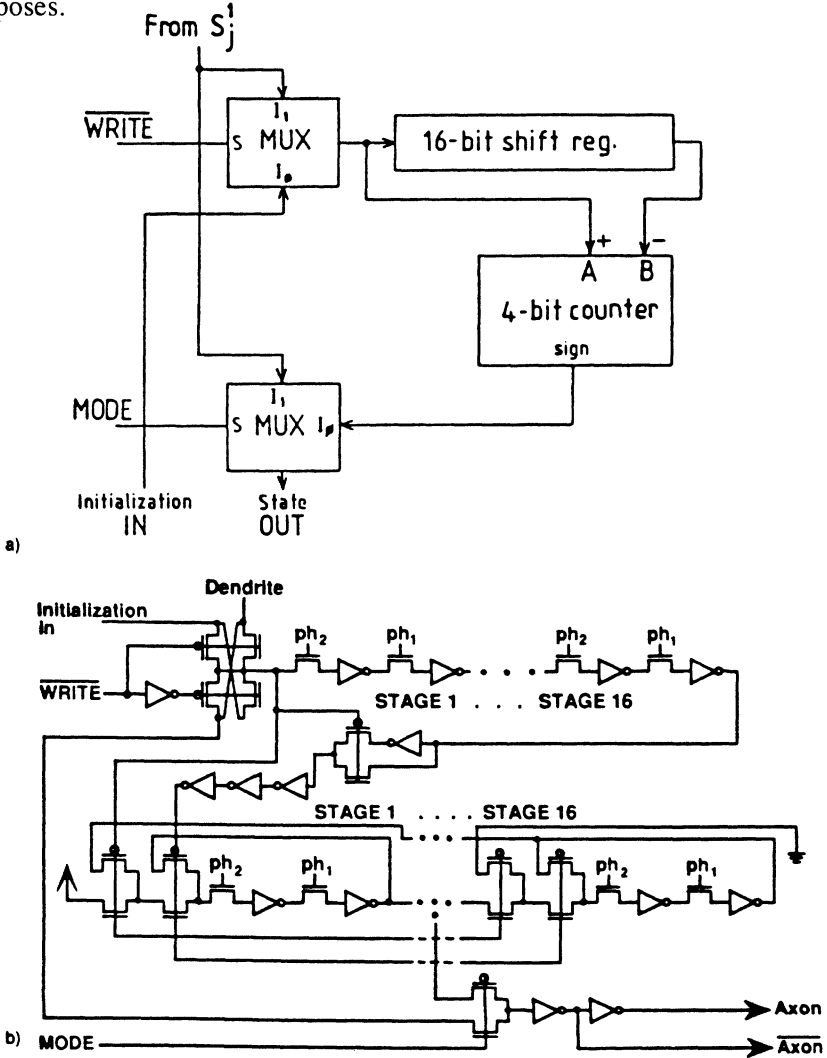


Figure 10. Fully digital implementation of a PDM neuron: (a) block diagram and (b) transistor level diagram.

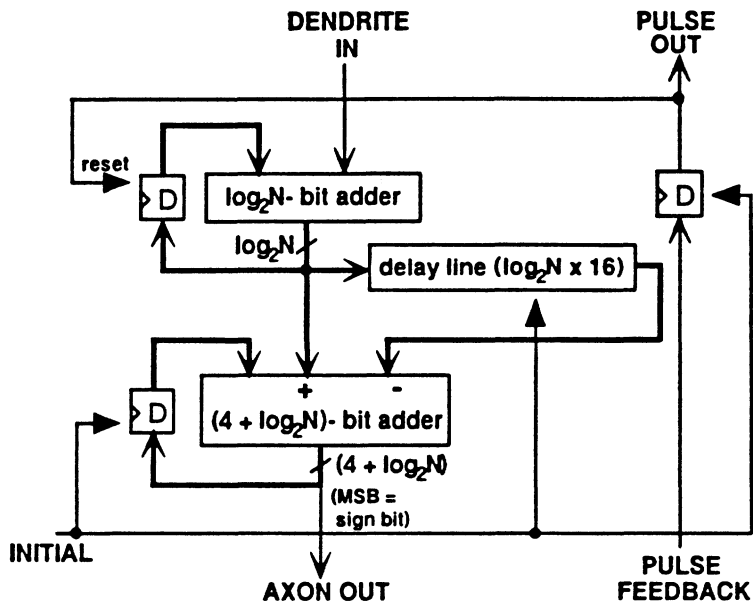


Figure 11. The neuron structure for time multiplexed synapse network.

INTEGRATED CIRCUIT IMPLEMENTATIONS

Several VLSI implementations of artificial neural network algorithms has been designed and tested using the described PDM structures. Fully connected neural network architectures have been used as test benches for the developed structures and later on many other networks were designed.

Implementations of Fully Connected Network

Hopfield type fully connected neural network has been the network most popularly implemented as integrated circuits. This is mainly because of the simple and regular structure and its properties are well known. Thus, it offers a good test bench for the developed structures. We have implemented such a network by using both switched-capacitor and fully digital structures [Tomberg 89ab].

Simple SC-Implementation [Tomberg 89a]

First we will discuss the overall architecture of the SC-network as shown in Figure 12. The neurons are placed on the diagonal line of the synapse matrix. In this way the structure is very regular. The self-connections of neurons are lost, but this is not essential for many applications. The network state is read and written through the bi-directional buffers, controlled by the I/O-logic. There is also a serial output for the weight data, which could be used

for testing purposes and for chaining several network chips together. Circuit level simulations for large networks are very heavy and therefore only small networks have been simulated with SPICE circuit simulator. However, from these simulations we can see that the circuitry operates correctly and larger network can be tested after processing.

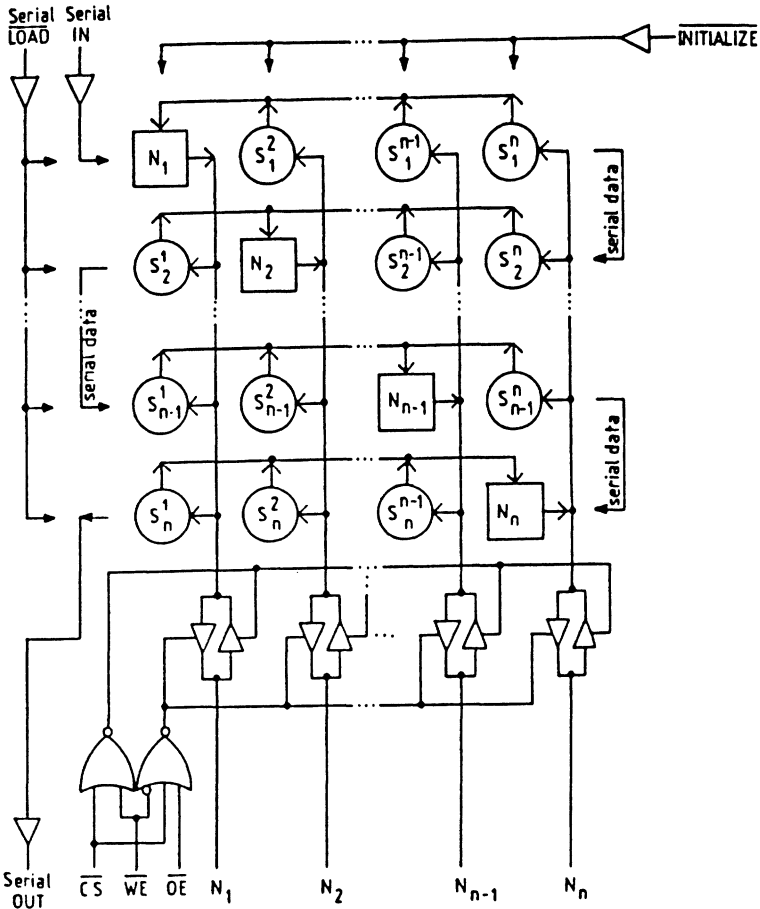


Figure 12. Architecture of the fully connected network with SC implementation.

In Figure 13 simulation results for a network with three fully connected neurons and 10 MHz operating frequency are shown. This kind of small networks converge quite rapidly, i.e. within 32 clock cycles. For larger networks convergence times are of course longer.

Using a 2.5 μm one metal-layer molybdenum gate CMOS-process the area of one synapse with 16-bit weight register, i.e. 17 weight values is less

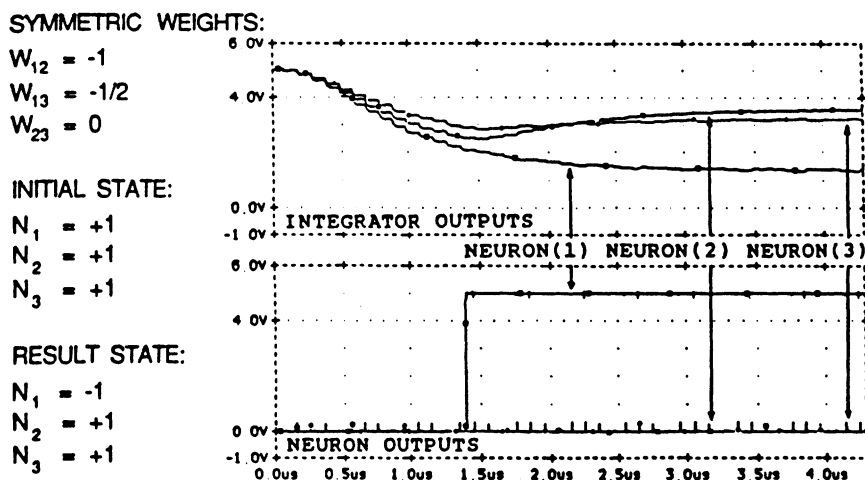


Figure 13. Simulation results of a SC network with three fully connected neurons.

than $50\,000\ \mu\text{m}^2$. Thus for example an array of 16×16 synapses requires less than $12.8\ \text{mm}^2$. The whole active area of the chip including the buffers and simple I/O-control logic is less than $15\ \text{mm}^2$. If this design is scaled down to $1\ \mu\text{m}$ process, we could get in excess of 100 fully connected neurons on a single chip of size $1\ \text{cm}^2$. For ordinary IC-packages the pin count restricts the amount of neurons. One solution to this problem would be to multiplex the outputs so that the neuron states can be read and written.

Simple Digital Implementation [Tomberg 89b]

Now we will turn to the overall architecture of the digital network implemented with simplified pulse-adder synapse structures (Figure 14). Because of the digital interface, these chips can be connected together to form larger networks. For simple connectivity, the neuron outputs must be wired to the "axon" inputs by the user. Each chip has two different operating modes; normal-mode and network-mode. In the normal-mode the chip operates as a 16 neuron and 256 synapse network and can be used alone or to be cascaded together with other chips. In the network-mode the neurons of the chip are bypassed and only the 16×16 synapse matrix is active. In this mode the chip can be connected together with other chips to be a part of large array of

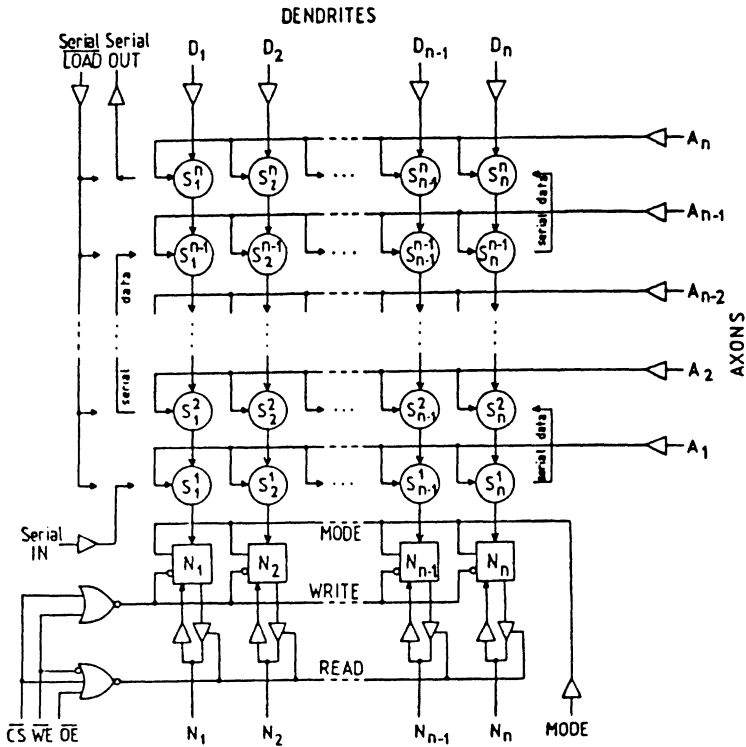


Figure 14. Architecture of the fully connected network with digital implementation.

synapses. Therefore, a large network of $16 \times M$ fully connected neurons contains $M \times M$ chips of which M chips are operating in the normal-mode and $M(M-1)$ chips in the network-mode.

For the digital structures functional level simulations are the most interesting ones. In Figure 15 simulation result for a network with three fully connected neurons is shown. The internal states of the neurons, denoted as "initial" in the figure, are actually output values of the delay line of the neuron, i.e. delayed dendrite values. By creating functional models of the basic blocks, neurons and synapses, we can simulate also larger networks quite easily.

In Figure 16 functional simulations for a network with 25 fully connected neurons are shown. In this case the network is trained with three patterns, i.e. numbers 1,2 and 3, by using the delta rule [Wasserman 89]. We have tested several different learning rules and observed the importance of proper learning. Because the digital implementation of the PDM technique is not

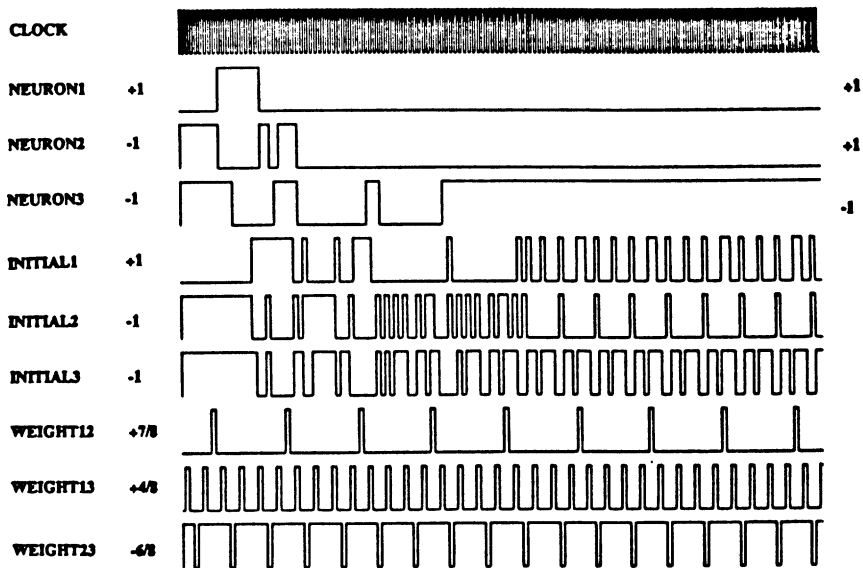


Figure 15. Simulation results of a digital network with three fully connected neurons.

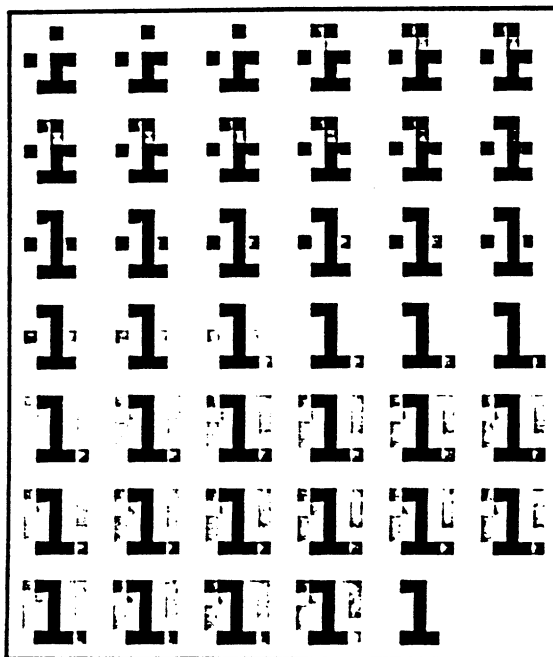


Figure 16. Simulation results of a digital network with 25 fully connected neurons and three trained patterns.

ideal, as discussed earlier, we should use a learning rule which gives the optimum result for the weight coding [Tomberg 91].

Using a 2.5 μm CMOS-process, the area of one synapse with 16-bit weight register is about 70 000 μm^2 as compared with the area 50 000 μm^2 in [Tomberg 89a]. Thus for example an array of 16x16 synapses is less than 18 mm^2 . Again, if the design is scaled down to 1 μm process about 90 fully connected neurons could be placed on a chip of size 1 cm^2 . For 16x16 network we would need 16 I/O-pins for connecting the synapse rows, i.e. axons together and 32 I/O-pins for connecting the synapse columns, i.e. dendrites together. In this case we would need 48 pins for cascading chips together and additional 8 pins for control lines. We have designed this network by using 2 μm two metal layer CMOS process with a 68 pin LCC (Leadless Chip Carrier) package and it is currently been processed by Austria Micro Systems. The total area of the chip including the I/O-pad structures was less than 28 mm^2 . The pin diagram was designed in such a way that single chips can be easily cascaded and a printed circuit board with 16 such circuits was implemented. This board can be joined to parallel port of a PC and programmed easily by the user.

Digital Implementation with On-Chip Learning Facility [Tomberg 91]

For more general purpose implementation we have designed a network with ideal pulse-stream adders together with optimized on-chip learning algorithm. This ensures autonomous operation with minimum amount of external control circuitry.

The learning algorithm is divided into the initialization and iterative parts. During the weight initialization the synaptic weights are calculated according to the training set based on Hebb's rule [Wasserman 89, Lippman 87] which are then set into the network. The first phase in the learning procedure is to reset all weights to zero (PDM) value. Then each training pattern is shown for the network once and the network is modified according to the rule:

$$\Delta w_{ij} = (p_i * p_j) \quad (6)$$

where p_i and p_j are the pixel values of neurons i and j . If Δw_{ij} is 0 the weight is not modified and if it is +1 or -1 one of the bits in the PDM-coded weight value is changed from one to zero or from zero to one, respectively. The second phase consists of iterative weight modification based on the Delta-rule [Wasserman 89] in which all the training patterns are presented for the network randomly and after the stabilization the weight values are modified according to the rule:

$$\Delta w_{ij} = (p_i^d - p_i) * p_j^d, \quad (7)$$

where p_i^d and p_j^d are the desired values of the pixels and p_i is the actual output value. This phase goes on until all the patterns are learned.

The fully digital interface makes it possible to expand the network easily by connecting several circuits together. Each chip has two possible configurations. In the first configuration the chip operates as a neuron and synapse network with control logic and it can be used alone or to be cascaded together with other chips. In the second configuration the neurons and the control logic of the chip are bypassed and only the synapse matrix is active. This mode is used simply to expand the connection matrix of the chips with active neurons. Thus the circuits with active neurons are responsible also for the control of the expansion circuits. The network can operate either in the learning or recognition mode selected by an external signal. The learning mode is fully automatic. First the control logic reads the patterns from the buffer memory and initializes the weights. Then the iterative modifications are done until all the patterns are learned or this phase is stopped externally. The state of the status signal tells whether the network is still learning or finished. During the recognition phase it tells when the network is stabilized. The stabilization of the network is detected by a special logic which checks the output states of the neurons within the defined, in this case 16 bit, window.

The synaptic weight modifications are controlled by two line sets going through every synapse. These are the axon lines going vertically through each synapse in row and the dendrite lines going vertically through each synapse in column. During the learning phase these lines are fed via the control logic. The actual weight modification logic is placed in each synapse (Fig. 4). If both lines going through a synapse have different values no modifications for that weight value is done. If on the other hand both lines have the value zero or one the weight value is increased or decreased by one, respectively. The increasing is done by changing a bit with value one to zero. The decreasing is a reverse operation. For each modified bit the whole weight register is checked serially. If no place for the modification is found, the weight value is obviously saturated to its maximum value.

The control logic is operating synchronously in each circuit which has active neurons. Each one reads and writes the patterns of its own neurons and sends the weight modification signals according to those neuron states. Their operation is basically independent of each other and the necessary synchronization is done by the external control signals which set the operation mode and start the operation. The only information between the adjacent control circuits is concerning the stabilization state of the network. Thus, all the circuits are chained together by the status signals which tells whether the neurons of the circuit are stabilized or not.

The first version of the circuit is done by using Programmable Gate Arrays (PGA's) and therefore the network size is quite small; only four neurons on each circuit. This prototype is used for testing purposes. In the first full custom version with dynamic structures we are aiming to 32 neurons on a chip and clock speeds up to 20 MHz.

Boltzmann-Machine Implementation [Tomberg 90b]

The Boltzmann-machine algorithm can be seen as a stochastic modification of the Hopfield type network [Sompolins 88]. In this case the state of the neuron is based on a statistical rather than a deterministic approach. The idea is to make the nets to find global minima of the energy function instead of local minima as in case of Hopfield nets. The Boltzmann Machine is suitable for various optimization tasks although the algorithm is computationally very heavy. This is because of the simulated annealing method which requires very efficient implementation on hardware. The fully connected part of output and hidden nodes tries to find a global minimum from the energy space defined by the state of input nodes. The output state of a node or neuron depends statistically on the input values, i.e. a noise parameter is added to the non linearity function of the neuron. This parameter is called temperature of the network and it defines the probability of the network to be in either one of the binary states. If the temperature is zero the node operates just like a neuron in the Hopfield network. If the temperature is infinite the output value can be either one of the binary values with equal probability.

To ensure an efficient VLSI implementation of the algorithm some modifications have been made for the original, ideal Boltzmann-machine algorithm. For the number representation we have used digital pulse-density coded values with 17 discrete levels. We have also used a linear probability function instead of the sigmoid. Thus, we end up with two main simplifications namely linearizing the probability function and quantizing the synaptic weight values and the number representation in general. The effects of these simplifications have been tested by simulations [Tomberg 90b]. We found out that linearizing the probability function has very small effect on the performance. In contrast the quantization of the weight levels decreases the performance significantly. Nevertheless, by increasing the amount of the hidden nodes we can decrease this effect. Thus, we obtain better performance either by increasing the amount of the discrete levels or the hidden nodes. Both of these will also increase the area of the silicon implementation and an optimal solution depends on the network size and application. Also the results of K.Kyuma [Kyuma 91] show that the effect of weight quantization in Boltzmann-machine network is essentially insignificant above certain critical

number of quantization levels. This number depends on the training set size. For training set size of the order of the neuron numbers the critical value was about 10.

The architecture of the Boltzmann machine circuit is shown in Figure 17. The circuit is designed to operate together with a host processor, which controls the learning and optimization operations. Thus, no control structures are implemented on chip in this first version. Our initial aim was to implement an efficient test structure for a larger network and include the control structures in a later designs. The network consists of the input nodes and fully connected block of hidden/output nodes. The operation is controlled via several registers, which are written and read by the host processor. The network configuration is selected via a special register which controls the I/O -nodes. During the learning the nodes which are selected to be output nodes can be clamped to desired values by writing the value in the I/O-register and selecting the clamped nodes via the control register. All the I/O-node values, both hidden and output nodes can be read via the I/O-register. The updating is performed by selecting the desired row via the row decoding register and updating the entire row by the value written in the update register. The simulated "temperature" parameter of the algorithm is controlled with a register. The 16-bit parallel Random Number Generator (RNG) block generates a new random

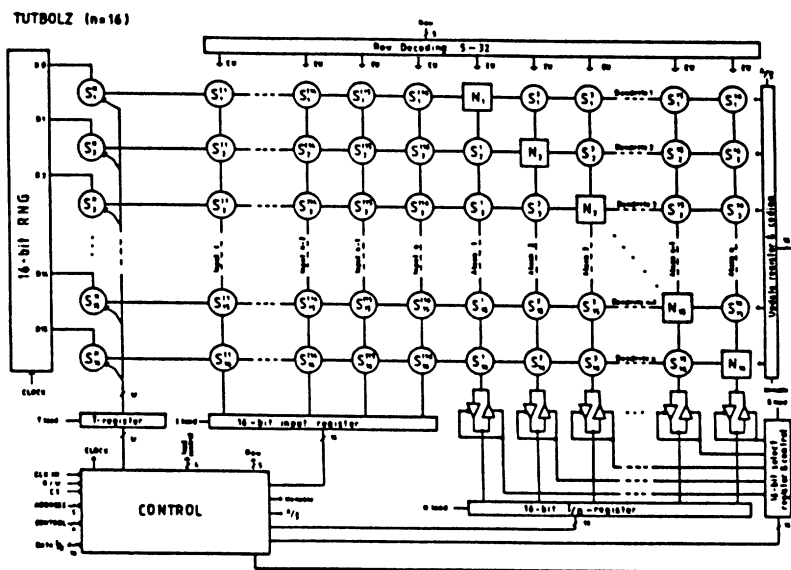


Figure 17. Architecture of the Boltzmann-machine circuit.

value during each clock cycle [Saarinen 91]. These random numbers can be considered as pulse-density coded white noise for 16 different channels. The random number generator is based on the shift-register structure with XOR feedback connections giving a very efficient implementation. The temperature parameter of the network can be changed by modifying the amplitude of the noise added on the dendrite lines. This is performed by feeding the noise via pulse-density synapses as controlled by the temperature register. The noise is multiplied by the temperature value written in the register

Both digital and switched-capacitor implementations of the structures based on the pulse-density modulation technique can be used. The switched-capacitor implementation leads to smaller area, but the digital implementation has the advantage of easier off-chip expandability. The active area of the circuit implemented by the $2.5 \mu\text{m}^2$ CMOS process is approximated to be 40 mm^2 with the switched-capacitor structures and 50 mm^2 with the fully digital structures.

Self-Organizing Network Implementation [Tomberg 92]

Self-organizing neural network models can be seen as biologically plausible because they do not need training pairs during the learning phase, i.e. they fall into the category of unsupervised learning. Kohonen's Self-organizing Feature map is one of the best known models and has many VLSI implementations [Goser 91]. The most critical issues of implementations are the effective calculation of the best matching node and its variable neighborhood. The most common applications for this kind of network can be found in the area of pattern and speech recognition.

In this our design we have combined the knowledge adopted from the earlier PDM implementations for realizing self-organizing network with on-chip learning algorithm. We have used SC-technique together with digital substructures to optimize the overall performance of the circuit. In Figure 18 we show the architecture of the implementation containing 16 nodes for input vectors with 16 components and 16-bit PDM coded weight values. In this first design the neighborhood is one dimensional. In future versions the amount of nodes and bits in weights can be easily scaled up according to the requirements of the application. Also multidimensional neighborhoods are possible.

The operation of the circuit is continuous, i.e. the components of the input vector are fed into the circuit in the PDM format and the result of the network can be read from the outputs continuously. In addition only two control inputs are needed. The first one selects the operation mode of the circuit i.e. it either is in the learning mode or just finding the best matching output node. The other one is used to control the learning phase. In general two parameters are needed

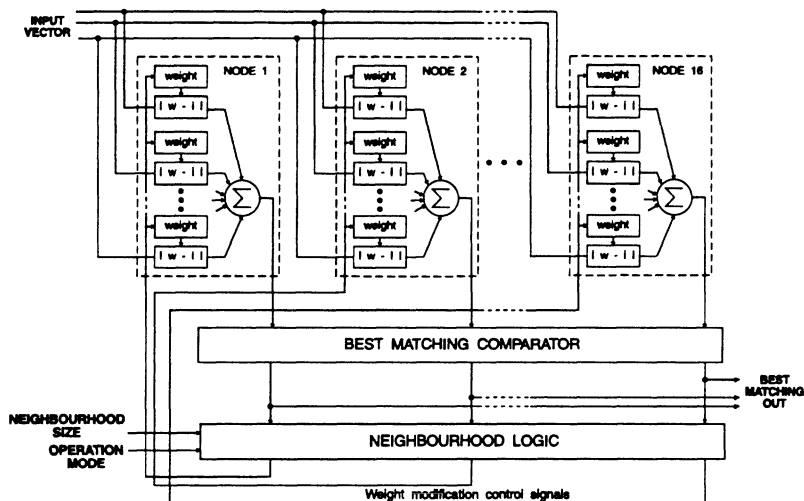


Figure 18. The PDM synapse logic of the self-organizing network.

for controlling the learning operation. One of them controls the neighborhood size and the other defines the shape of the update function of the weights in the neighborhood. Both of these parameters are controlled as a function of time. In our implementation we update all the weights in the defined neighborhood by a unit value. The shape of the update function is implemented by repeating the update operation for the defined input vector with different sizes of neighborhood so that the requirements of the desired update function are satisfied. Thus only one control value for learning purpose is needed when it is used synchronously with the input vector feeding.

The distances between the input vector and the reference vectors of the nodes are calculated by adding together all the absolute values of the differences between the component of the input vector and corresponding weight value (Manhattan distance). These values are then compared in parallel with each node and the neighborhood is defined around the best matching node. The feedback information to the weight updating logic is used to modify the weight value during the learning phase while the next vector is fed in. Due to the fully parallel operation the time of one learning cycle is proportional only to the amount of bits in the weight value. During the recognition phase only the best matching part of the algorithm is performed and the weight updating logic is disabled.

The difference between an incoming PDM-coded value and a PDM weight value is calculated in the synapse logic by using lossy integrator structure implemented by switched capacitor technique (Fig. 19). The sign of

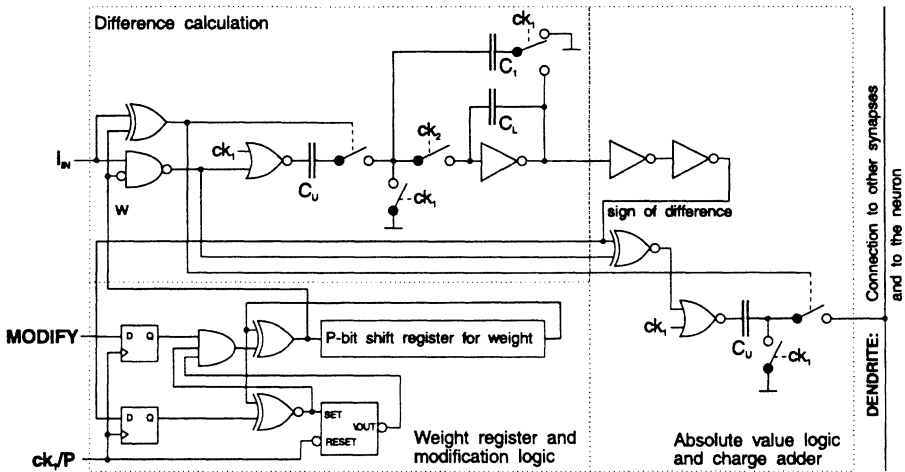


Figure 19. The PDM synapse logic of the self-organizing network.

the difference is then detected by a simplified comparator implemented by CMOS inverters. The absolute value of the difference is generated based on the sign of the result. The results of different components are added together within the dendrite line by using simple charge adding method. The synapse block includes also the weight modification logic. The control signal (MODIFY) comes from the neighborhood logic timed by the window synchronizing pulse. If the MODIFY signal is high the weight value is modified during the next P clock cycles according to the sign of the difference.

The distance of each node from the input vector is calculated in the neuron by integrating the summed charge from the dendrite line within the PDM window (Fig. 20). The charge in the integrator corresponds to the distance between the input vector and the node value. Thus the neuron which has the lowest value in the integrator is closest to the input vector. These values are compared by a special SC-feedback structure. All the integrators are charged equally via the reference capacitors C_0 which keeps the value of the integrator of the best matching node below the threshold value of the fixed comparator. This is done by continuously checking the output values of the comparators via the wire-and structure. If one or more comparator outputs are high all the integrators are charged and if none of them is high all the integrators are discharged by a small unit value. Thus the output of the best matching comparator has positive pulses which are then latched in the output after each comparing sequence. In order to select one of the best matching nodes in the case of equal distance values some extra logic is added to the

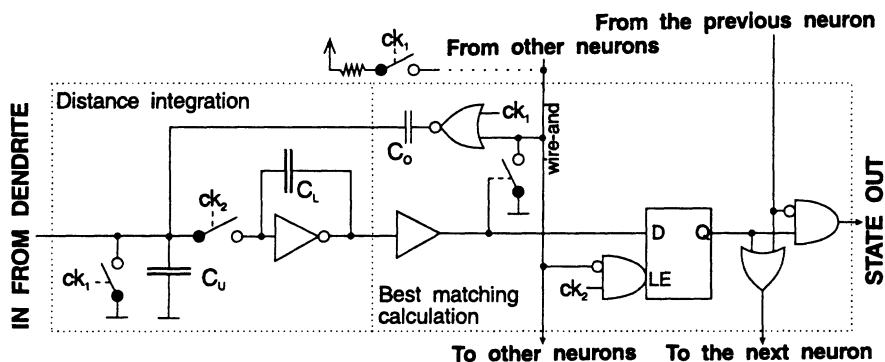


Figure 20. The PDM neuron logic of the self-organizing circuit.

output of the neuron. The final output is then taken to the neighborhood logic which performs filtering of the data and calculates the neighborhood for the weight modification logic. The neighborhood is defined by using a SC-modification of the resistive network [Mann 88, Hochet 91]. Instead of using comparators with selectable threshold value, we decided to use a simple fixed value comparator and modify the strength of the connections in the network by altering the clocking frequency of the SC-connections between the neighboring nodes (Fig. 21). Therefore, a bigger clocking frequency means larger neighborhood and vice versa. The operation mode selection signal is used to disable the weight updating signal when the circuit is not in the learning mode.

The architecture of the circuit is modular offering easy on-chip expansion of the network. However, the off-chip expansion is more difficult due to the global operation based on charge adding principles. The control logic is relatively simple including only the global two phase clock signal generator and sequencers for generating the PDM window synchronizing pulses and the neighborhood size control frequency according to the external control value. In order to achieve an optimum implementation a CMOS process with effective capacitor structures and possibility for small dynamic shift registers is required. Our first design is targeted for 1 μm double poly, double metal CMOS process which offers good possibilities for dense and effective structures.

Other implementations

Other algorithms can also be implemented by using pulse-density modulation technique. However, their requirements concerning the simplifications of the basic algorithm are often stricter. One should also bear in mind that the required accuracy of the algorithm may strongly affect the

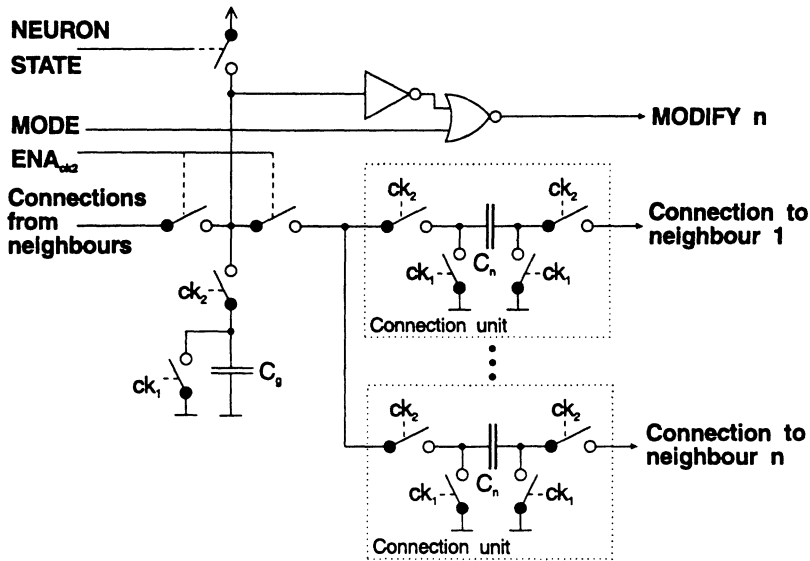


Figure 21. The neighborhood logic of the self-organizing circuit.

suitability of the structure and its implementation. For example, the backpropagation [Wasserman 89] algorithm seems to require more than 256 discrete values for the weights during learning [Graf 89]. This makes analog implementations less suitable due to its smaller inherent accuracy and larger noise sensitivity. Otherwise, for example the Boltzmann-machine is more robust for the weight quantization than the Backpropagation model [Kyuma 91]. The non linearity function of the neuron must also be differentiable everywhere and thus we can not use the step function approximation instead of the sigmoid function. This makes the implementation of the non linearity function in the neuron more complicated. Basically we must use a look-up table in the place of simple up/down counter or recursive adder as was the case in the Hopfield type networks.

CONCLUSIONS

The pulse-density modulation technique offers an interesting method to implement artificial neural networks as VLSI circuits. It is suitable for networks with relatively small amount of weight levels or very effective weight storage techniques. The optimized result will be obtained by using mixed analog/digital structures. One of the benefits of PDM technique is the simple interface to analog world due to the data representation. The analog data can be converted to digital PDM format by using simple sigma-delta modulator structure.

The synaptic weight data is stored in dynamic shift registers and the weights are fully programmable between -1 and +1. The number of discrete levels of the weight value depends directly on the length of the weight registers. This affects also the minimum synapse size and thus the maximum network size on a single chip. The advantage of the switched-capacitor implementation is small area of a synapse, and therefore relatively large networks can be implemented. The architecture of the network is also regular, modular and thus easy to expand. For the same complexity of the network architecture a digital implementation requires 30 per cent more silicon area. This difference can, however, be considered quite insignificant. The advantage of a fully digital implementation is good off-chip expandability to larger networks. The main restriction is that the simplified digital implementation is in many ways not ideal. This can be partly compensated by the learning rule. An ideal digital implementation will be obtained by expanding the structures using more silicon area or time for the processing.

Several successful VLSI implementations have been designed by using the PDM technique [Tomberg 89ab 90 91 92, Palovuori 91]. Both co-processor type circuits and fully autonomous circuits with on chip learning facility has been designed. Currently we are looking for more complicated system level implementations of ANNs using PDM technique. One particular interest is to apply the PDM technique for more biologically oriented neuron model due to the relationships to the biological pulse-stream representation.

ACKNOWLEDGMENTS

This work has been supported in part by Finsoft and Microelectronics programs of the Technology Development Centre in Finland (TEKES) and by the Academy of Finland.

References

- [Goser 91] K.Goser, "Kohonen's Maps - Their Application and Implementation in Microelectronics", in *Proc. of 1991 International Conference on Artificial Neural Networks*, Vol. 1, Helsinki, Finland, 24-28 June 1991, pp. 703-708.
- [Graf 89] H.P.Graf and L.D.Jackel, "Analog Neural Network Circuits", *IEEE Circuits and Devices Magazine*, Vol. 5, no. 4, 44-49, 1989.
- [Gray 87] R.M.Gray, "Oversampled Sigma-Delta Modulation", *IEEE Transactions on Communications*, vol. COM-35, pp. 481-489, May 1987.

- [Hirai 89] Y.Hirai et al., "A Digital Neuro-Chip with Unlimited Connectivity fo Large-Scale Neural Networks", in *Proc. IEEE and INNS IJCNN'89*, Vol. 2, Washington D.C., July 1989, pp. 163-169.
- [Hochet 91] B.Hochet, V.Peiris, S.Abdo and M.J.Declercq, "Implementation of a Learning Kohonen Neuron Based on a New Multilevel Storage Technique", *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 3, March 1991, pp. 262-267.
- [Kyuma 91] K.Kyuma, "Optical Architectures for Neuron Circuits", in *Proc. of 1991 IEEE International Symposium on Circuits and Systems*, Singapore, 11-14 June 1991, pp. 1384-1387.
- [Lippman 87] R.P.Lippman, "An Introduction to Computing with Neural Nets", *IEEE ASP Magazine*, Vol. 4, Apr. 1987, pp. 4-22.
- [Mann 88] J.Mann, R.Lippman, R.Berger and J.Raffel, "A Self-Organizing Neural Network Chip", in *Proc. IEEE CICC'89*, 1988, pp. 10.3.1.-10.3.5.
- [Murray 88] A.F.Murray and A.V.Smith, "Asynchronous VLSI Neural Networks Using Pulse-Stream Arithmetic", *IEEE Journal of Solid State Circuits*, Vol. 23, no. 3, June 1988, pp. 688-697.
- [Palovuori 91] K.Palovuori, J.Tomberg ja K.Kaski, "VLSI Implementation of a Pulse-Density Modulated Neural Network for PC-controlled Computing Enviroment", *International Conference on Artificial Neural Networks*, Helsinki/Finland, 24-28.6.1991.
- [Saarinen 91] J.Saarinen, J.Tomberg, L.Vehmanen and K.Kaski, "VLSI Implementation of Tausworthe Random Number Generator for Parallel Processing Enviroment", *IEE Proceedings-E*, Vol. 138, No. 3, May 1991, pp. 138-146.
- [Soclof 85] S.Soclof, *Applications of Analog Integrated Circuits*, Prentice-Hall, pp. 377-406, 1985.
- [Sompolins 88] H.Sompolinsky, "Statistical Mechanics of Neural Networks", *Physics Today*, December 1988, pp. 70-80.

- [Tomberg 89a] J.Tomberg, T.Ritoniemi, H.Tenhunen and K.Kaski, "VLSI Implementation of Pulse-Density Modulated Neural Network Structure", in *Proc. IEEE Int. Symp. on Circuits and Systems*, Portland/USA, May 9-11, 1989.
- [Tomberg 89b] J.Tomberg, T.Ritoniemi, K.Kaski and H.Tenhunen, "Fully Digital Neural Network Implementation Based on Pulse Density Modulation", in *Proc. IEEE Custom Integrated Circuits Conference*, San Diego/USA, 12.7.1-12.7.4, May 15-17, 1989.
- [Tomberg 89c] J.Tomberg ja K.Kaski, "VLSI Implementation of Neural Network Based on Switched-Capacitor Structures", in *Proceedings of Neuro-Nimes 1989*, Nimes, France, 13-16 November 1989, pp. 203-210.
- [Tomberg 90a] J.Tomberg and K.Kaski, "Pulse-Density Modulation Technique in VLSI Implementations of Neural Network Algorithms", *IEEE Solid-State Circuits Magazine*, No. 5, Vol. 25, October 1990.
- [Tomberg 90b] J.Tomberg, H.Raittinen and K.Kaski, "VLSI Architecture of the Boltzmann-machine algorithm", In *Proc. Int. Neural Network Conference INNC'90*, Paris/France, July 9-13, 1990.
- [Tomberg 91] J.Tomberg and K.Kaski, "An Effective Training Method for Fully Digital Pulse-Density Modulated Neural Network Architecture", *IEEE Int. Symp. on Circuits and Systems*, Singapore, June 11-14, 1991.
- [Tomberg 92] J.Tomberg ja K.Kaski, "VLSI Architecture of the Self-Organizing Neural Network Using Synchronous Pulse-Density Modulation Technique", *1992 International Conference on Artificial Neural Networks*, Brighton, UK, 4.-7 September 1992.
- [Tsividis 88] Y.P.Tsividis and D.Anastassiou, "Switched-Capacitor Neural Networks", *Electronic Letters*, Vol. 23, No. 18, August 1988, pp. 958-959.
- [Wasserman 89] P.D.Wasserman, *Neural Computing Theory and Practice*, Van Nostrand Reinhold, New York, 1989.

CMOS ANALOG NEURAL NETWORK SYSTEMS BASED ON OSCILLATORY NEURONS

Bernabé Linares-Barranco¹, Edgar Sánchez-Sinencio²,
Angel Rodríguez-Vázquez¹, and José L. Huertas¹

¹ *Centro Nacional de Microelectrónica, Dept. of Analog Circuit Design, Ed. CICA, Av. Reina Mercedes s/n, 41012 Sevilla, SPAIN*

² *Dept. of Electrical Engineering, Texas A&M University, College Station, 77843 TX, U.S.A.*

The area of Artificial Neural Networks consists of building machines and algorithms that are based somehow on the structure of natural brains. It has been shown during the past years that such type of machines are able to do human kind of tasks (associative memories, pattern recognition, feature extraction, ...) much more efficiently than conventional algorithms running on conventional computers.

The fact that the area of knowledge called *Artificial Neural Networks* is based on biological brains is not trivial to any outside observer. A biological brain is made of nervous tissue, it contains living cells, it works thanks to an immense collection of biochemical reactions between huge organic molecules, and it is powered by the metabolism of the living being that owns that brain. On the other hand, an artificial neural network is most of the times a *strange* software algorithm on a digital computer. Sometimes, when we talk about hardware implementations of artificial neural networks, we can physically identify the neurons and synapses of the system, but they are just a collection

of transistors and wires built on a rigid silicon substrate that might communicate to a digital computer, and that is powered by a constant voltage source usually plugged to a socket on the wall. Where is then the relation between these two neural systems, the natural and the artificial?

They both consist of a set of processing elements, called *neurons*, interconnected by *synapses*. The relationship is therefore in the structure, not in the implementation. It is the structure of the artificial neural systems that gives them the ability to perform human-kind tasks. And even more, what makes a neural system able to perform a specific task is not the collection of little processors or neurons, but the collection of weights of the synapses and how these change in time.

In this chapter we will put the emphasis on the implementation of the neurons, and we therefore believe it is good to know the '*connection*' between the natural biological neuron and the artificial one. The way natural neurons work and interact will give us a wider vision of the field and might help us in the future if we decide to change the artificial implementation technique or technology.

A first step towards building biology-like neural network hardware is making the artificial neurons to be signal controlled oscillators: the neurons fire a sequence of pulses if the controlling signal reaches a certain threshold value, otherwise no pulses will be fired. This implementation philosophy has already been adopted by many researchers and is growing in interest gradually [1]-[10].

In this chapter we will first describe with some detail the operation of the living neuron and its interaction with the others. Based on this, we will derive a mathematical model (definitely not the most complete available today) that will serve our purposes. Then we will give a hardware implementation of this model and will simplify it gradually until obtaining simple and efficient artificial neurons usable for oscillatory VLSI hardware [10]-[11]. Afterwards, we will use these neural oscillators to build conventional neural networks, namely a Hopfield network and a BAM network. Finally, before concluding, we will give some hints on how to extend these networks to chaotic oscillators based neural systems, which seem to be available in nature as well.

PHYSIOLOGY OF THE BIOLOGICAL NEURON [12]-[13]

In this section we are going to describe briefly the biological mechanisms involved in the interaction between neurons and how, as a consequence of this interaction, a neuron generates an electrical impulse that is called the *action potential*.

A neuron is a living cell immersed in an interstitial fluid. There exists a

voltage difference between the inside and the outside of the cell that is produced by an unequal distribution of electrolytes inside and outside of the cell membrane. This unequal distribution of ions is a consequence of the cell membrane having different permeability factors for each one of the ions. For the time being, and as is illustrated in Figure 1, assume that inside the cell membrane there are K^+ ions and large organic A^- ions, while outside there are mainly Cl^- and Na^+ ions. The cell membrane is always impermeable to the A^- ions so that they always remain inside the cell. These molecules are too large for passing through the cell membrane's pores. During the resting state the cell membrane is permeable only to K^+ and Cl^- ions. Therefore, K^+ will tend to diffuse outwards, while Cl^- tends to diffuse inwards in order to equal their concentration on both sides of the membrane. As a consequence of this diffusion the electrical equilibrium of the state shown in Figure 1 is altered, and an electrical field will arise (negative inside with respect to the outside). This electrical field opposes the diffusion of ions down their concentration gradients. An equilibrium state will be established in which the force of the

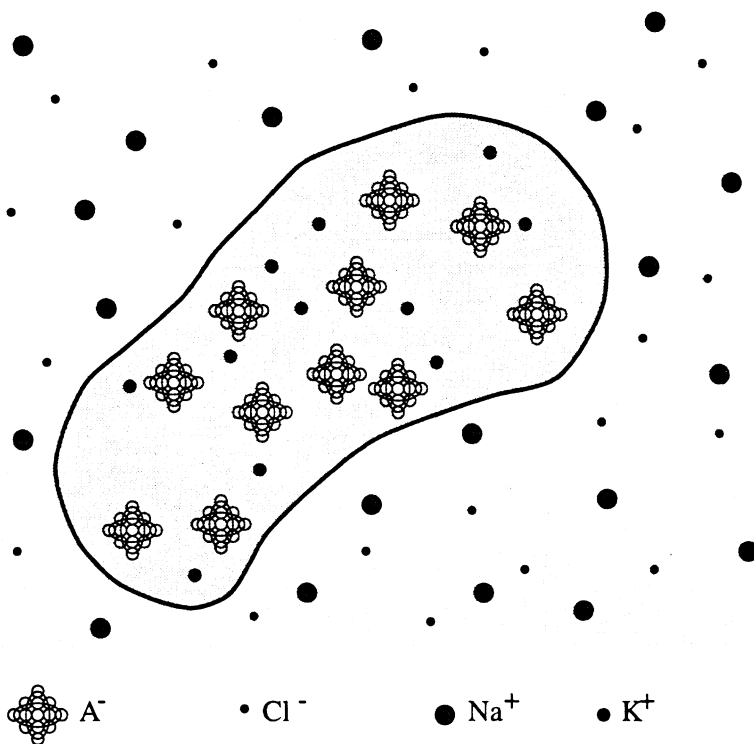


Figure 1. Distribution of Electrolytes inside and outside of a Living Neuron

electrical field against the ions equals the chemical force that makes the ions diffuse. At this equilibrium state a voltage difference of typically **60-80mV** is present between the internal and external walls of the cell membrane, called **membrane resting potential**.

In 1943 David Goldman of Bethesda derived an expression for the membrane potential in terms of the concentration of ions on both sides of the membrane and their relative membrane permeabilities. For the case of our living cell membrane the electric potential V_m (inside with respect to outside) is given by,

$$V_m = 58mV \log \frac{P_K [K^+]_{out} + P_{Na} [Na^+]_{out} + P_{Cl} [Cl^-]_{in}}{P_K [K^+]_{in} + P_{Na} [Na^+]_{in} + P_{Cl} [Cl^-]_{out}} \quad (1)$$

where P_K , P_{Na} and P_{Cl} are the relative membrane permeabilities for K^+ , Na^+ and Cl^- respectively, $[K^+]_{out}$, $[Na^+]_{out}$ and $[Cl^-]_{out}$ are the concentrations of ions K^+ , Na^+ and Cl^- outside the membrane, and $[K^+]_{in}$, $[Na^+]_{in}$ and $[Cl^-]_{in}$ are the concentrations inside.

During the resting state of the neuron the cell membrane is impermeable to ions Na^+ ($P_{Na}=0$) and $V_m \approx -75mV$ (typically between $-60mV$ and $-80mV$). During the generation of the action potential P_{Na} reaches its maximum value and $V_m \approx +50mV$: there is a flow of Na^+ inwards the cell that contributes to the increment of membrane voltage V_m , followed by a flow of K^+ outwards to reestablish the resting potential. Naturally, there is also a mechanism that, after the action potential, is going to pump Na^+ outside, and K^+ inside, so that the resting concentrations of these ions are recovered. This is performed independently by the so-called Na^+-K^+ pumps. These are very complex organic molecules embedded in the membrane that literally pump Na^+ out and K^+ in against their concentration gradients, by means of a sequence of chemical metabolic reactions that consume energy.

An equivalent circuit for the electrical properties of the nerve membrane is shown in Figure 2. The different permeability factors are represented by conductances G_{Na} , G_K and G_{Cl} , and C_m is the capacitance imparted by the lipids of the membrane. E_K , E_{Na} and E_{Cl} are called the Nernst Potentials for K^+ , Na^+ and Cl^- , respectively. They are defined by the expressions

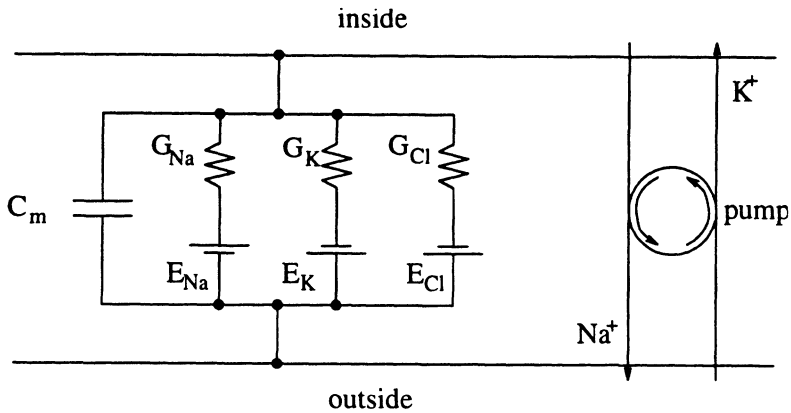


Figure 2. Equivalent Electrical Circuit for Electrical Properties of the Nerve Membrane

$$\begin{aligned}
 E_K &= 58mV \log \frac{[K^+]_{out}}{[K^+]_{in}} \\
 E_{Na} &= 58mV \log \frac{[Na^+]_{out}}{[Na^+]_{in}} \\
 E_{Cl} &= 58mV \log \frac{[Cl^-]_{in}}{[Cl^-]_{out}}
 \end{aligned} \tag{2}$$

According to the Goldman equation (1) when the neuron is resting, since $P_{Na}=0$, it yields

$$V_m = 58mV \log \frac{P_K [K^+]_{out} + P_{Cl} [Cl^-]_{in}}{P_K [K^+]_{in} + P_{Cl} [Cl^-]_{out}} \tag{3}$$

Usually the effect of Cl^- ions diffusion is negligible, so that

$$V_m \approx 58mV \log \frac{[K^+]_{out}}{[K^+]_{in}} = E_K = -75mV \tag{4}$$

and the membrane resting voltage is approximately equal to the K^+ Nernst potential (which is $-75mV$).

During the generation of the action potential P_{Na} increases until it reaches a peak and then returns to zero. At the peak, when P_{Na} is maximum, Goldman's equation can be approximated by

$$V_m \approx 58mV \log \frac{[Na^+]_{out}}{[Na^+]_{in}} = E_{Na} = +50mV \quad (5)$$

which is approximately the peak voltage of the action potential.

The conductances G_{Na} , G_K and G_{Cl} are proportional not only to the relative membrane permeabilities, but also to the concentration of electrolyte at the side of the cell membrane that is the source of the flow. So, for example, for Na^+ the source of the flow is the outside of the cell membrane (the flow is from outside to inside). Therefore,

$$\begin{aligned} G_{Na} &\propto P_{Na} [Na^+]_{out} \\ G_K &\propto P_K [K^+]_{in} \\ G_{Cl} &\propto P_{Cl} [Cl^-]_{out} \end{aligned} \quad (6)$$

Embedded in the cell membrane there are the so-called *ionic channels*. These are physical channels (pores) that can be opened and closed by different stimuli, and when open allow the flow of certain ions through the membrane. The opening and closing of these channels is what changes the permeability of the membrane, and therefore, the membrane potential. Many channels have been identified so far. However, we are going to consider only a few of them that will allow us obtain an appropriate mathematical model for the neuron dynamics.

The Na^+ Channels

We are going to consider only two different types of Na^+ channels: the ones opened chemically by organic molecules called *neurotransmitters* released by the end of a synapse when it receives an electrical impulse, and the ones that are opened when the membrane voltage reaches a certain threshold value (approximately $-50mV$)¹. These two channels receive the name of *Chemically Gated Channels* and *Voltage Gated Channels*, respectively.

1. Recent studies [13] reveal that there are also some intermediate types of channels, i.e., they can be opened by the two mechanisms.

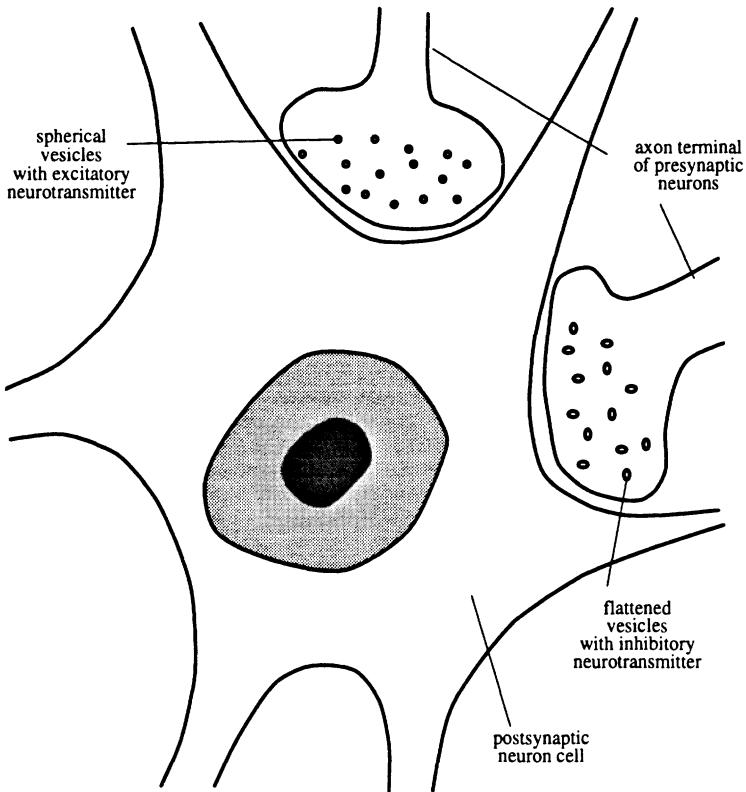


Figure 3. Schematic Illustration of Excitatory and Inhibitory Synaptic Connections to a Neuron.

Chemically Gated Channels

In Figure 3 is depicted a neuron with two synaptic connections, one excitatory and one inhibitory. The end of the synapses do not touch the cell membrane of the neuron. There is a spacing between each synapse and the neuron's membrane of approximately 20-40nm, called *synaptic cleft*. Each synapse is at the end of an axon of another neuron. When an electrical impulse reaches the synapse, vesicles containing large organic molecules called *neurotransmitters* are released. The neurotransmitters that will open the *Chemically Gated Na⁺ Channels* are excitatory neurotransmitters. They are contained in spherical vesicles inside excitatory synapses

The *Chemically Gated Na⁺ Channel* is a large organic molecule (a protein) embedded in the neuron's membrane at the *synaptic cleft* between neuron and an excitatory synapse. When excitatory neurotransmitters are released into the synaptic cleft, they will eventually reach the *Na⁺* channels and bind to them. When this happens the channels are physically deformed so

that they change their structural geometry and open a pore in the membrane that allows the flow of Na^+ ions. This process is schematically represented in Figure 4. The excitatory neurotransmitters only bind temporarily to the channels. They will be hydrolyzed into another substance which is inactive and will be absorbed by the synapse, so that the neurotransmitter can be recycled for future use.

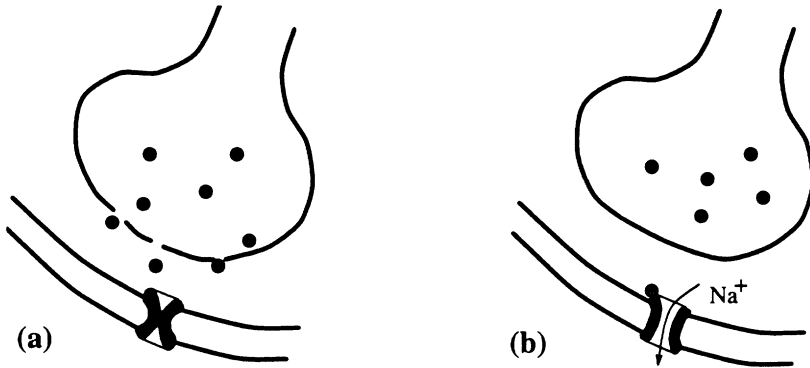


Figure 4. Illustration of Opening Mechanism of a Chemically Gated Na^+ Channel by an Excitatory Neurotransmitter.

Voltage Gated Na^+ Channels

These are channels that are opened when the membrane voltage increases above $-50mV$, approximately. When an electrical impulse is delivered through an excitatory synapse, excitatory neurotransmitter is released, chemically gated Na^+ channels open and Na^+ flows into the cell producing an increase in the membrane voltage. If the electrical impulse was strong enough or if it was a sufficiently long train of pulses, then more channels had been opened and the membrane voltage had reached the $-50mV$ threshold. If this happens the voltage gated channels will open and therefore further increase the membrane voltage. This is like a chain reaction whose result is a very high Na^+ permeability factor (high G_{Na}) which will produce the action potential

The way the voltage gated Na^+ channels work is as follows. It is another protein. It consists of four equal rigid units of 300 amino acids that are joined by other chains of flexible amino acids. These four units are arranged in a cylindrical fashion inside the membrane. For a membrane voltage below $-50mV$ the four units are very close together and the channel is closed. But if the membrane voltage increases above $-50mV$ the four units will separate (no more than 5\AA) allowing flow of Na^+ ions. This is schematically illustrated in Figure 5. Some regions of the protein are charged positively and others

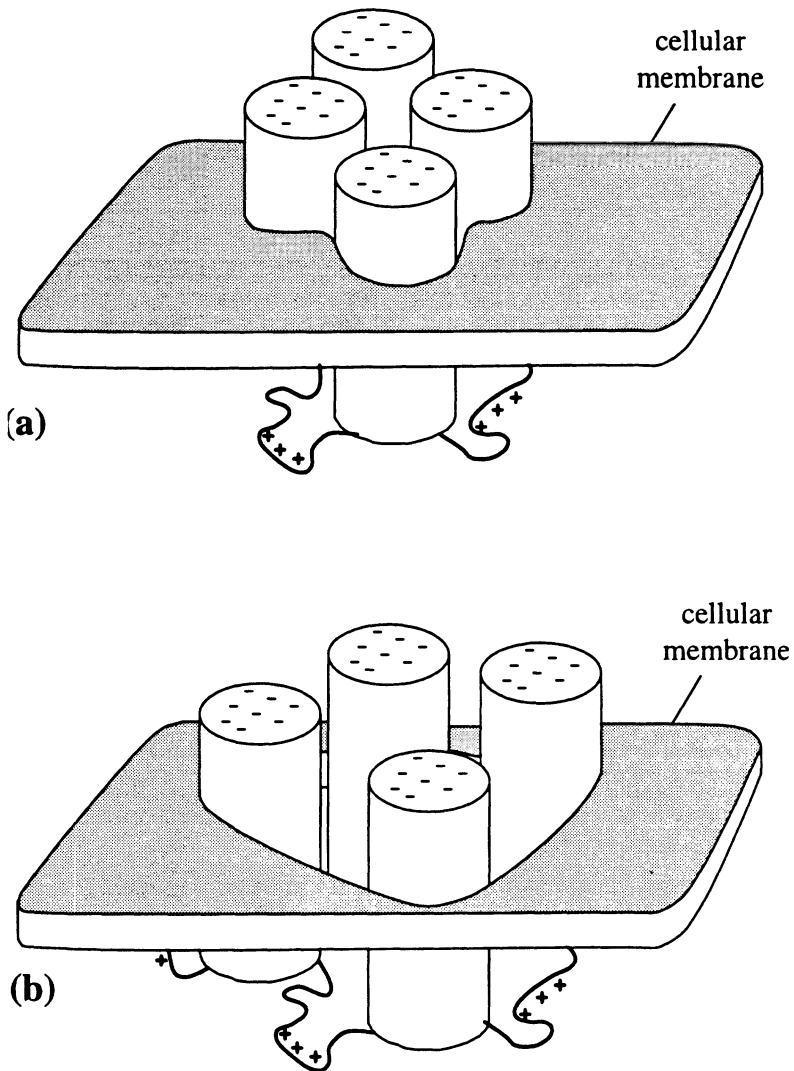


Figure 5. Illustration of Structure of a Voltage Gated Na^+ Channel When Closed (a), and When Open (b).

negatively. It is believed that the interactions between these oppositely charged regions serve as sensors of changes in transmembrane voltage, producing changes in the configuration of the channel protein, which opens the channel slightly allowing flow of Na^+ . Such channels remain open only for a few milliseconds, and the flow of ions can be represented by a square pulse of current (1 to 2 pA) of the same amplitude for all active channels but different

duration. Figure 6 (a) shows the current through three different channels, while Figure 6 (b) depicts the shape of the sum of 200 of them.

Due to the chain reaction produced when the membrane voltage reaches $-50mV$, all the voltage gated channels will open making G_{Na} very high for a few milliseconds. The transient membrane voltage V_m produced under these

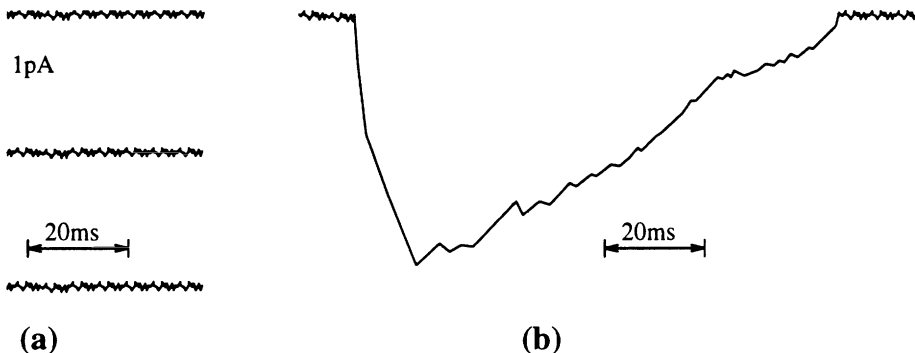


Figure 6. (a) Currents through Three Individual Voltage Gated Channels: (b) Sum of the Currents through 200 Voltage Gated Channels

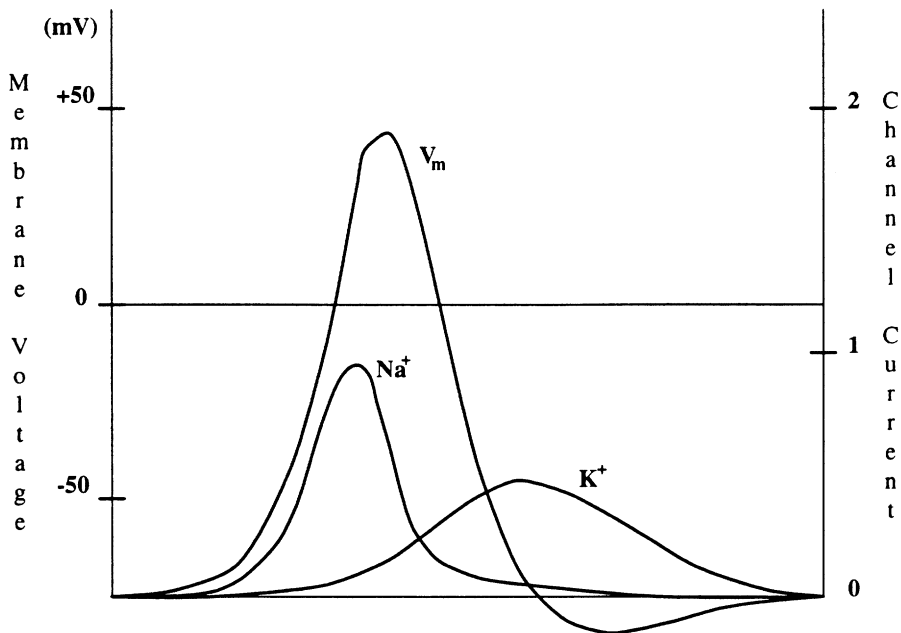


Figure 7. Membrane Voltage and Ionic Currents During an Action Potential in a Cell Membrane

conditions is called the **action potential** (see Figure 7). The amplitude and shape of this action potential are characteristic of the neuron cell and do not depend on the signals that triggered it. If the signal that triggered the action potential is strong enough a train of action potentials might be generated (each one of them of constant amplitude and shape). The number of action potentials and the separation between them does depend on the strength of the triggering signal.

The K^+ Channels

There are several types of K^+ channels, but the action of all of them is to stabilize the membrane potential to the resting voltage. Their effect can be summarized as a current opposite to the Na^+ one that is activated after some delay by an increase in the membrane voltage as shown in Figure 7. Since this current produces a decrease in membrane potential, it will make, after the peak of Na^+ current, the voltage to reach its resting value. Furthermore, if originally not enough Na^+ channels were opened fast enough, this K^+ current will start to make the membrane potential to decrease before the threshold voltage is reached and, therefore, abort the action potential.

The Cl^- Channels

The Cl^- channels are chemically gated channels embedded in the neuron's membrane in the synaptic cleft of inhibitory synaptic connections. When an electrical impulse reaches this synapse, inhibitory neurotransmitter is released into the synaptic cleft and will attach to the Cl^- channels distorting them briefly and opening ionic gates permitting Cl^- ions to move by diffusion into the cell. The result is an ionic current that tends to decrease the membrane voltage, as is shown in Figure 8. The ionic channels for inhibition are surprisingly nonspecific, depending purely on the pore size: all anions smaller than a critical size in the hydrated state ($0.29nm$) pass through.

AN ELECTRICAL CIRCUIT MODEL

So far we have described the physiology of the living neuron and given a partial equivalent electrical circuit (see Figure 2). Now we are going to complete the equivalent circuit so that it will allow us to represent most of the dynamics involved. The circuit is shown in Figure 9. The different ionic channels are represented by the following elements:

- I_e represents the excitatory effect of the Na^+ current passing through the chemically gated Na^+ channels. This current source depends on the signals arriving from other neurons through excitatory synapses.

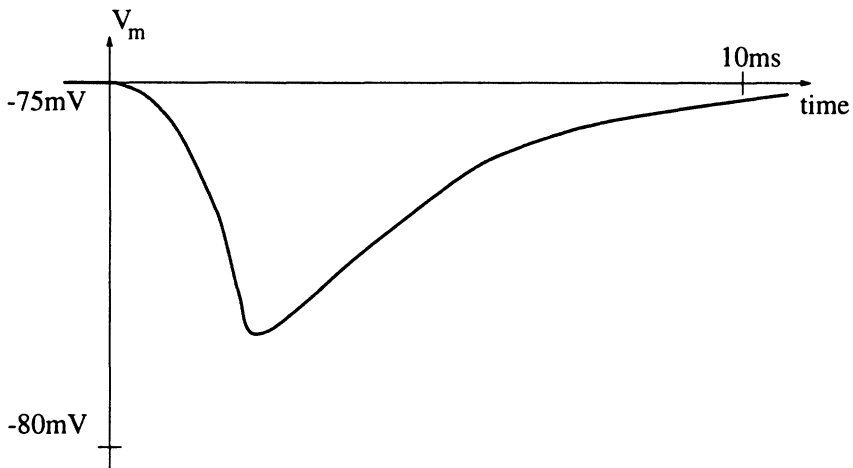


Figure 8. Effect of Inhibitory Transmitter, Released by an Inhibitory Synapse, on the Membrane Voltage.

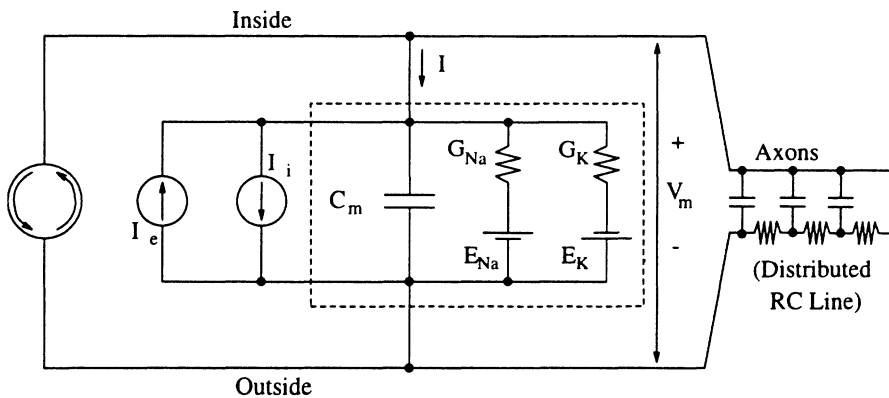


Figure 9. Electrical Circuit Model that Explains the Generation of the Action Potential in a Neural Cell.

- I_i represents the inhibitory effect of the Cl^- current passing through the chemically gated Cl^- channels. This current source depends on the signals arriving from other neurons through inhibitory synapses.
- G_{Na} represents the change in Na^+ permeability of the cell membrane due to the opening of voltage gated Na^+ channels. The value of the conductance G_{Na} will therefore be voltage V_m dependent.

- G_K represents the change in K^+ permeability of the cell membrane due to the opening of voltage gated K^+ channels. The value of the conductance G_K will therefore be voltage V_m dependent, although this dependence is much softer than for G_{Na} .

The loading effect of all the axons of the neuron (that will propagate the electrical impulses to other neurons) is modeled here by a distributed RC line. More precisely, the axons should be modeled by a distributed line of elements like the circuit comprised by broken lines in Figure 9. Such a distributed line would be able to regenerate the action potential along its way to the next synaptic connection without degrading it.

It is worthwhile to mention here that action potentials can be produced in any living cell (not necessarily belonging to the nervous system) if properly excited [13]. Their generation is a property of the cell membrane. What makes the cells of the nervous system unique in this sense is that they are able to propagate action potentials through their axons and synaptic connections to other cells.

The part of the circuit of Figure 9 enclosed by broken lines is very similar to the one that Hodgkin and Huxley proposed in 1952 [14] to relate current and voltage through the nervous cell membrane during an action potential. They provided mathematical expressions for the different conductances,

$$\begin{aligned} G_{Na} &\propto m^3 h \\ G_K &\propto n^4 \end{aligned} \quad (7)$$

that were governed by time and voltage dependent differential equations,

$$\begin{aligned} \dot{m} &= \frac{m_{\infty}(V_m) - m}{\tau_m(V_m)} \\ \dot{h} &= \frac{h_{\infty}(V_m) - h}{\tau_h(V_m)} \\ \dot{n} &= \frac{n_{\infty}(V_m) - n}{\tau_n(V_m)} \end{aligned} \quad (8)$$

The functions $m_{\infty}(V_m)$, $\tau_m(V_m)$, $h_{\infty}(V_m)$, $\tau_h(V_m)$, $n_{\infty}(V_m)$ and $\tau_n(V_m)$ are only voltage dependent and are depicted in Figure 11. The model of Hodgkin

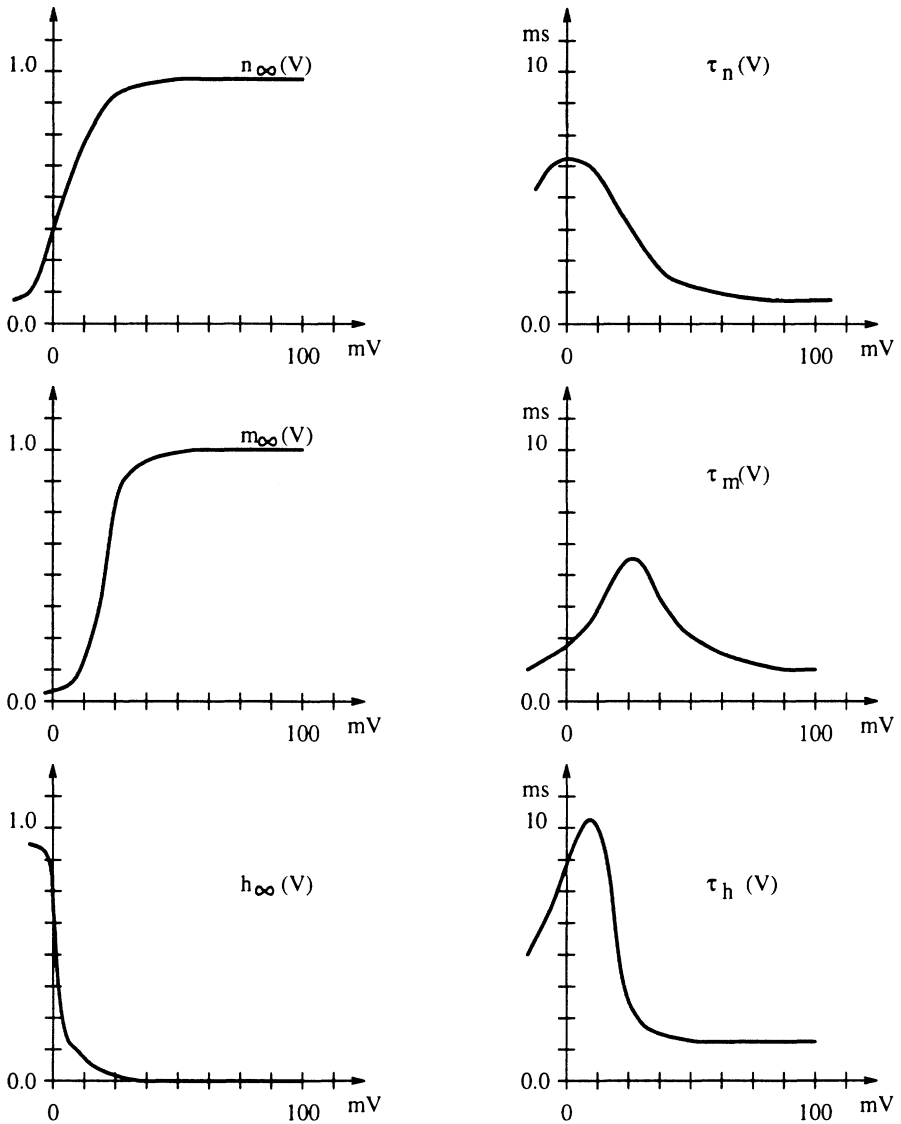


Figure 10. Hodgkin-Huxley Curves for Voltage-Only-Dependent Functions of Equation (8).

and Huxley explains very well the generation of the action potential, but fails to explain the generation of more than a single impulse, such as the complex firing patterns that characterize most neurons [13]. These type of patterns can be explained, however, by the presence of other ionic channels in the membrane. Their global effect is similar to allowing the Na^+ channels to

remain open as long as the membrane voltage is above the threshold that opens them. In the following section we will consider a simplification of Hodgkin and Huxley's model by FitzHugh and Nagumo [15]-[17], where the Na^+ conductance is only voltage dependent (but not time dependent) and, therefore, is able to model the generation of trains of pulses.

In the circuit of Figure 9 we have included a current source that represents the Na^+-K^+ pump. That current source should not be considered as forming part of an electrical circuit that explains the generation of action potentials, because this pump works independently of the action potential and its function is only to avoid accumulation of Na^+ ions inside the cell and of K^+ ions outside. Also, the load of the distributed RC line can be neglected for most practical purposes.

Another aspect we would like to mention before ending this section is how to model the synaptic interconnections between neurons. Remember that when an electrical impulse reaches the end of the axon, i.e. the synapse, a certain amount of neurotransmitter is released into the synaptic cleft between the synapse and the next neuron. These neurotransmitters remain in the synaptic cleft for a few milliseconds and open some of the chemically gated Na^+ or Cl^- channels. The more excitatory neurotransmitters are released, the more chemically gated Na^+ channels will open and the more likely it is that the membrane voltage will reach the $-50mV$ threshold that opens the voltage gated Na^+ channels, generating the action potential. The more inhibitory neurotransmitters are released, the more negative the membrane voltage will become and the less likely the threshold will be reached. For each neuron that is receiving neurotransmitters from all the synapses connected to it, a spatial and temporal summation of all the inputs is performed. Spatial in the sense that each synapse is contributing to increase (if excitatory) or decrease (if inhibitory) the membrane voltage when it receives an electrical impulse, and temporal because the more electrical impulses arrive the more neurotransmitters are present in the synaptic cleft before there is time to inactivate them (for further recycling) and a higher variation in membrane voltage is achieved. This effect can be modeled by the following two differential equations,

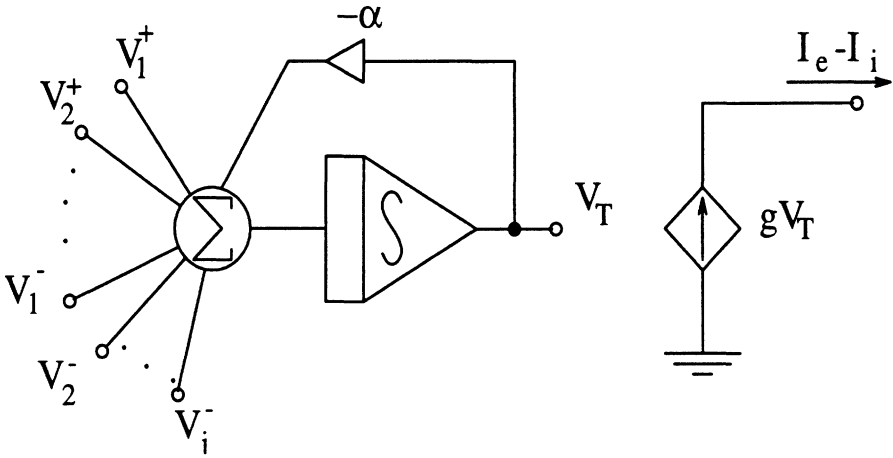


Figure 11. Circuit Diagram for Modeling the Synaptic Connections to One Neuron

$$C_e \frac{dI_e}{dt} = -\alpha_e I_e + \beta_e \sum_i V_i^+ \quad (9)$$

$$C_i \frac{dI_i}{dt} = -\alpha_i I_i + \beta_i \sum_j V_j^-$$

where V_i^+ are the electrical signals at the excitatory synapses, V_i^- are the ones at the inhibitory synapses, and C_e , C_i , α_e , α_i , β_e and β_i are time constants related parameters. If

$$\begin{aligned} C &= C_e = C_i \\ \alpha &= \alpha_e = \alpha_i \\ \beta &= \beta_e = \beta_i \\ I &= I_e - I_i \end{aligned} \quad (10)$$

equations (9) can be reduced to

$$C \frac{dI}{dt} = -\alpha I + \beta \left(\sum_i V_i^+ - \sum_j V_j^- \right) \quad (11)$$

A circuit that implements this equation is shown in Figure 11. The effect

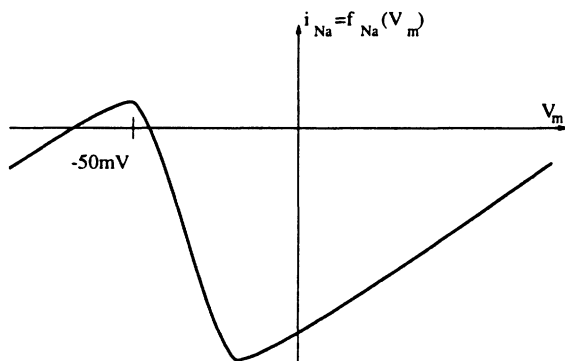


Figure 12. Na^+ Current as a Function of Membrane Potential in the Simplified Model Proposed by FitzHugh and Nagumo

of the integrator in Figure 11 or the time derivatives in equations (9) and (11) is what models the fact that the neurotransmitters remain active for a finite period of time (a few milliseconds) inside the synaptic cleft. Therefore, with eqs. (9) (or the simplified eqs. (11) and the block diagram of Figure 11), we are able to model the interactions between oscillatory neurons.

FITZHUGH-NAGUMO NEURON MODEL AND VLSI CIRCUIT IMPLEMENTATION

Theoretical Model

The simplifications introduced by FitzHugh and Nagumo [15]-[17] in the circuit comprised by broken lines in Figure 9 are a different modeling of the Na^+ and K^+ conductances. Since the Na^+ current characterized by G_{Na} is a fast one that strongly depends on the membrane voltage, it is modeled by a time independent nonlinear conductance, as is shown in Figure 12. On the other hand, the K^+ current is a slow current that does not depend very nonlinearly on the membrane voltage. Therefore, G_K can be modeled by a linear resistor R connected in series with an inductor L and a voltage source E_K that represents the membrane resting potential, as shown in Figure 13. This model is mathematically described by the following set of first-order differential equations,

$$C_m \frac{dV_m}{dt} = I - i_K - f_{Na}(V_m) \quad (12)$$

$$L \frac{di_K}{dt} = V_m + E_K - Ri_K$$

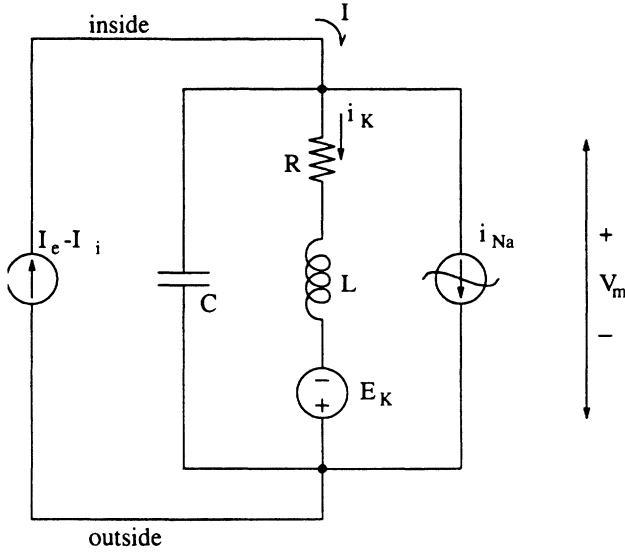


Figure 13. Equivalent Circuit for FitzHugh-Nagumo Neuron Model

Circuit Derivation

We would like to have an equivalent circuit of FitzHugh-Nagumo’s model suitable for a CMOS implementation. The circuit of Figure 13 is not adequate for this purpose because of the presence of inductor L . Therefore, we will use a specific circuit design technique, called **Transconductance-mode** (T-mode), that will allow us to implement the equations directly into a circuit that only has capacitors and transconductance amplifiers, both appropriate for CMOS VLSI.

We will first present this circuit design technique as a general tool for implementing a circuit that solves a general system of N nonlinear first order time differential equations in the variables x_1, x_2, \dots, x_N .

$$y_{oj} + \sum_{i=1}^N g_{ij}x_i + f_j(\dot{x}) + \sum_{i=1}^N B_{ij}\dot{x}_i = 0 \quad , \quad j = 1, \dots, N \quad (13)$$

y_{oj} , g_{ij} and B_{ij} being constant parameters and $f_j(\bullet)$ nonlinear functions of $x=(x_1, x_2, \dots, x_N)$. Consider now the circuit of Figure 14. It consists of N nodes of voltages x_j . Each node j is connected to ground through a capacitor C_{jj} and to each other node i through a capacitor C_{ij} . Two current sources I_{Lj} and I_{Nj} are also connected to each node j . I_{Lj} is linearly dependent on the node voltages of all the other nodes,

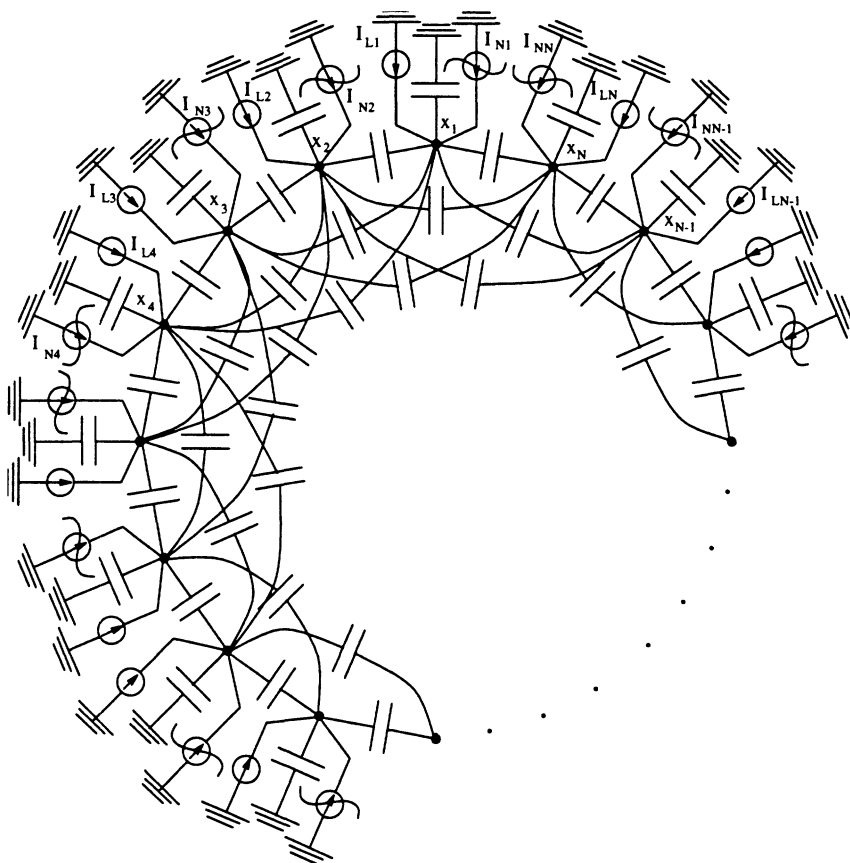


Figure 14. General Topology Representing N Nonlinear Differential Equations

$$I_{Lj} \equiv \sum_{i=1}^N g_{ij} x_i + y_{oj} \quad (14)$$

where g_{ij} (which can be positive or negative) is the transconductance relating interaction between nodes i and j , and y_{oj} is an offset term. I_{Nj} is a nonlinearly dependent current source,

$$I_{Nj} = f_j(x_1, \dots, x_N) = f_j(\vec{x}) \quad (15)$$

For each node j the following KCL equation holds,

$$y_{oj} + \sum_{i=1}^N g_{ij}x_i + f_j(\dot{x}) = \sum_{i=1}^N C_{ij}(\dot{x}_j - \dot{x}_i) + C_{jj}\dot{x}_j \quad (16)$$

If we define now,

$$B_{ij} \equiv \begin{cases} C_{ij} & \text{if } i \neq j \\ -\sum_{l=1}^N C_{jl} & \text{if } i = j \end{cases} \quad (17)$$

we obtain the set of equations (13).

By comparing equations (12) and (13) we can see that equations (12) are a particular case of (13) for $N=2$,

$$\begin{aligned} C_{22}\dot{x}_2 &= y_{o2} - g_{m2}x_1 - f(x_2) \\ C_{11}\dot{x}_1 &= y_{o1} + g_{m1}x_2 - g_{m3}x_1 \end{aligned} \quad (18)$$

if we do the following assignments,

$$\begin{aligned} V_m &= x_2 & i_K &= x_1 & I &= \frac{y_{o2}}{g_{m2}} & E_K &= \frac{y_{o1}}{g_{m1}} \\ C_m &= \frac{C_{22}}{g_{m2}} & L &= \frac{C_{11}}{g_{m1}} & R &= \frac{g_{m3}}{g_{m1}} & f_{Na}(V_m) &= \frac{f(x_2)}{g_{m2}} \end{aligned} \quad (19)$$

By drawing now the circuit of Figure 14 for equations (18) the circuit shown in Figure 15 is obtained. The exact form of the function $f(\bullet)$ seems not to be very critical. Originally, a cubic polynomial [16] for Figure 12 was suggested, but a piece wise linear dependence can give the same basic properties to the system [15]. We will consider $f(\bullet)$ as shown in Figure 16. The nonlinear resistor of Figure 15 with the driving point characteristics of Figure 16 can be implemented in T-mode by the circuit shown in Figure 17 [10], [18].

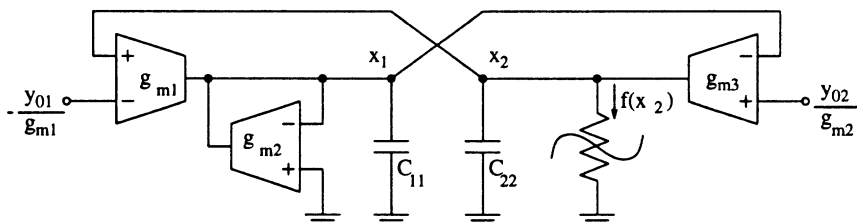


Figure 15. T-mode Implementation of FitzHugh-Nagumo's Equations

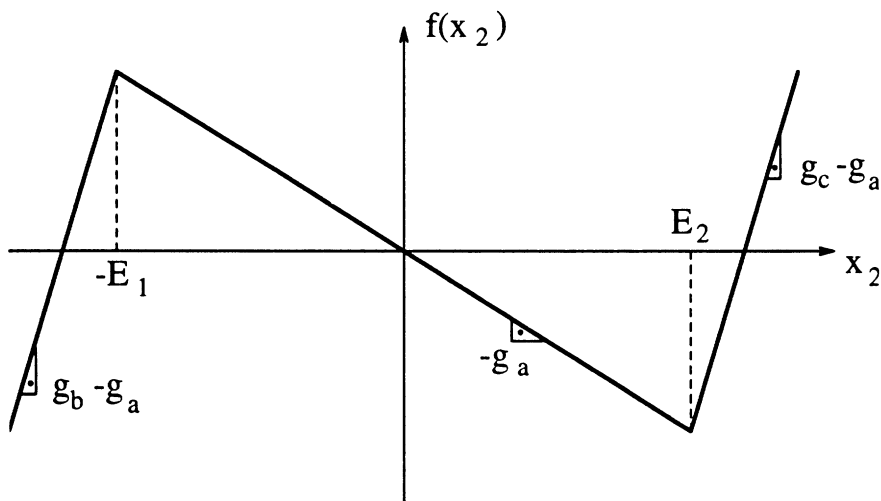


Figure 16. N-Shaped Piece Wise Linear Function for Nonlinear Element of Figure 15

Circuit Dynamics

A phase portrait of the equilibrium points of the system described by equations (18) is shown in Figure 18, where $g_b - g_a = g_c - g_a = g_l$. The equilibrium points are obtained when $\dot{x}_1 = \dot{x}_2 = 0$, i.e., they are obtained by the intersections of the two curves,

$$\begin{aligned} y_{o2} - g_{m2}x_1 - f(x_2) &= 0 \\ y_{o1} - g_{m1}x_2 - g_{m3}x_1 &= 0 \end{aligned} \quad (20)$$

Since there is a nonlinearity with three linear segments, we can divide the phase plane into three linear regions, namely, regions "1", "2", and "3" as shown in Figure 18. Each one of these linear regions will have its own unique equilibrium point. If this equilibrium point lies inside its own region it is called *real equilibrium point*. If it lies outside the region that defines it, it is called

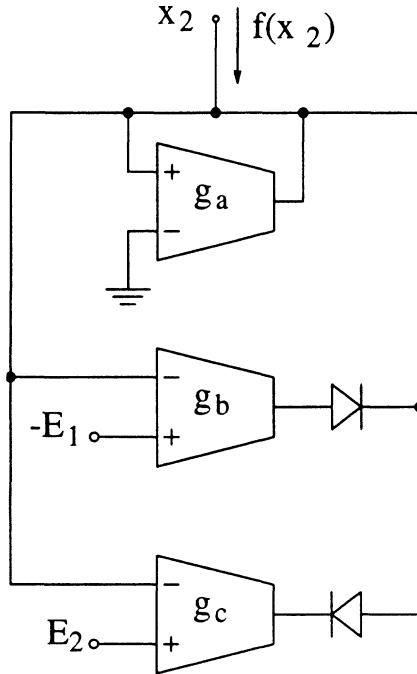


Figure 17. Implementation of the Nonlinear Function Using T-Mode Techniques

virtual equilibrium point. Note that a virtual equilibrium point cannot be reached by the system, because as soon as it goes into another region the equilibrium point of this new region is different. The function $f(\bullet)$ is defined as,

$$f(x_2) = \begin{cases} g_l x_2 - E_2 (g_a + g_l) & \text{for region '3'} \\ -g_a x_2 & \text{for region '1'} \\ g_l x_2 + E_1 (g_a + g_l) & \text{for region '2'} \end{cases} \quad (21)$$

Therefore, according to equations (18), for region "1" the linear differential equations are given by,

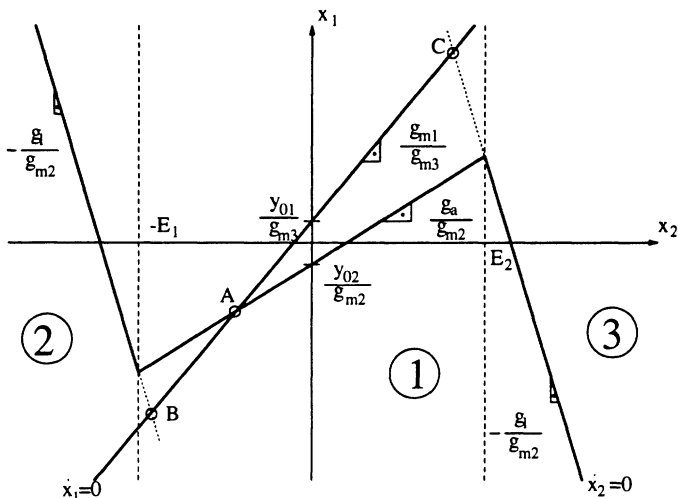


Figure 18. Phase Portrait of the System Characterized by Equations (18)

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{g_{m3}}{C_{11}} & \frac{g_{m1}}{C_{11}} \\ \frac{g_{m2}}{C_{22}} & \frac{g_a}{C_{22}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{y_{o1}}{C_{11}} \\ \frac{y_{o2}}{C_{22}} \end{bmatrix} \quad (22)$$

The equilibrium point **A** for region "1" is defined by,

$$x_{10} = \frac{1}{\Delta} \begin{vmatrix} \frac{y_{o1}}{C_{11}} & \frac{g_{m1}}{C_{11}} \\ \frac{y_{o2}}{C_{22}} & \frac{g_a}{C_{22}} \end{vmatrix} \quad x_{20} = \frac{1}{\Delta} \begin{vmatrix} \frac{g_{m3}}{C_{11}} & \frac{y_{o1}}{C_{11}} \\ \frac{g_{m2}}{C_{22}} & \frac{y_{o2}}{C_{22}} \end{vmatrix} \quad (23)$$

$$\Delta = \begin{vmatrix} \frac{g_{m1}}{C_{11}} & \frac{g_{m1}}{C_{11}} \\ \frac{g_{m2}}{C_{22}} & \frac{g_a}{C_{22}} \end{vmatrix}$$

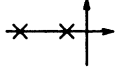
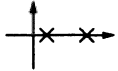

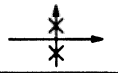
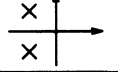
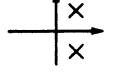
Type of equilibrium state	Real Eigenvalues		Complex Eigenvalues	
	Stable Node	$T < 0$ $\Delta > 0$		
Unstable Node	$T > 0$ $\Delta > 0$			
Saddle Point	$\Delta < 0$			
Center			$T = 0$ $\Delta > 0$	
Stable Focus			$T < 0$ $\Delta > 0$	
Unstable Focus			$T > 0$ $\Delta > 0$	

Figure 19. Classification of Equilibrium Points According to the Values of T_o and Δ_o in their State Equations

and *determinant* Δ_o of the matrix in equation (22). In Figure 19 we give a classification of equilibrium points according to the values of T_o and Δ_o [19]. Equilibrium point *A* of region “1” will therefore be unstable if,

$$T_o = \frac{g_a}{C_{22}} - \frac{g_{m3}}{C_{11}} > 0$$

$$\Delta_o = \frac{g_{m1}g_{m2}}{C_{11}C_{22}} - \frac{g_ag_{m3}}{C_{11}C_{22}} > 0$$
(24)

For regions “2” and “3” we can describe the behavior of the system by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{g_{m3}}{C_{11}} & \frac{g_{m1}}{C_{11}} \\ \frac{g_{m2}}{C_{22}} & \frac{g_l}{C_{22}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{y_{o1}}{C_{11}} \\ \frac{y_{o2} - \alpha(g_l + g_a)}{C_{22}} \end{bmatrix} \quad (25)$$

where $\alpha = -E_1$ for region "2" and $\alpha = -E_2$ for region "3". The equilibrium points **B** (for region "2") and **C** (for region "3") will be stable if

$$T_1 = -\frac{g_l}{C_{22}} - \frac{g_{m3}}{C_{11}} < 0 \quad (26)$$

$$\Delta_1 = \frac{g_{m1}g_{m2}}{C_{11}C_{22}} + \frac{g_l g_{m3}}{C_{11}C_{22}} > 0$$

For proper operation of the system we need to make **A** unstable and **B** and **C** stable equilibrium points. Suppose now that, for a certain value of y_{o1} and y_{o2} , **A** is real while **B** and **C** are virtual. Suppose also that the system is at a certain time in region "1" of Figure 18. Since **A** is unstable the system will move away from it until it eventually reaches region "2" or "3". When this happens, since the equilibrium point (**B** or **C**) is stable, the system will be attracted by it. But before it is reached, the system will find itself again in region "1" and repelled by **A**. As a consequence of all this, the system will oscillate in a limit cycle in which it goes from regions "2" to "3" and vice versa crossing region "1" each time.

By changing, in Figure 18, the relative position of the curves $\dot{x}_1 = 0$ and $\dot{x}_2 = 0$, through y_{o1} and/or y_{o2} , we can make **B** or **C** become real equilibrium points and **A** a virtual one. If either **B** or **C** is real, the system will reach the stable equilibrium point and stay there. Therefore, no oscillations will be produced. This corresponds to the resting state of the neuron where no action potentials are generated. But if y_{o1} and/or y_{o2} is changed beyond the threshold value that makes either **B** or **C** change from real to virtual and vice versa, the system will start to produce oscillations (the neuron is active and firing action potentials). Note that y_{o2} represents the total excitation current $I = I_e - I_i$ of Figure 13. Note also that, as can be seen in Figure 18, for $y_{o1}/g_{m3} - y_{o2}/g_{m2}$ there is an upper and a lower value that will stop the oscillations. This means that FitzHugh-Nagumo's equations represent a double threshold system.

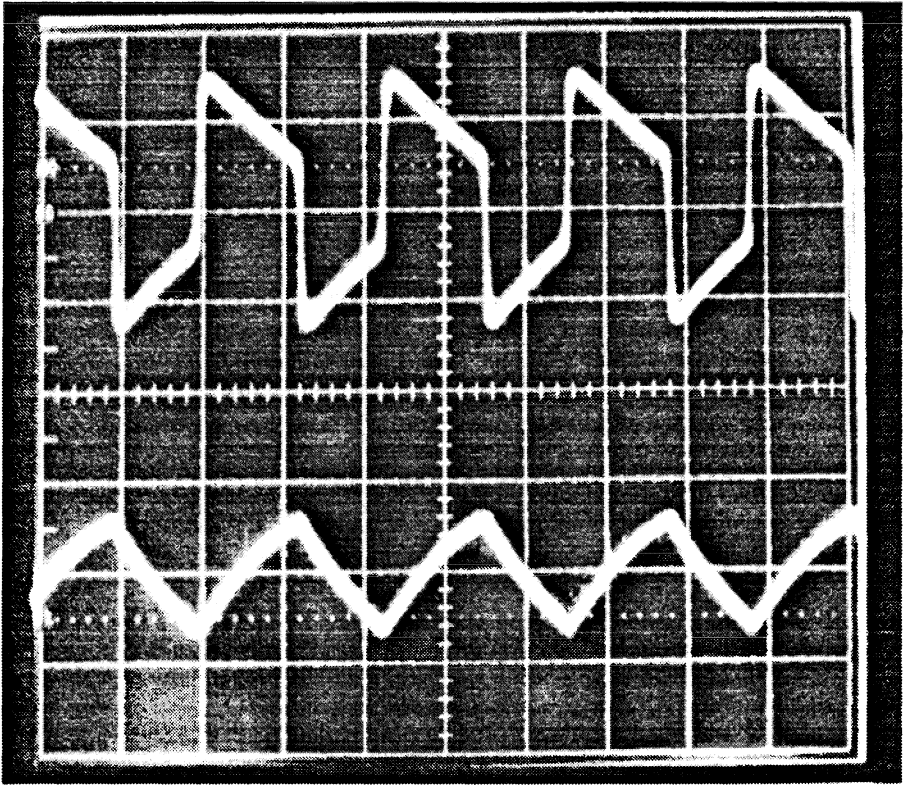


Figure 20. Measured Free-Running Oscillations for the Circuit of Figure 15 When $g_{m1}/C_{11} \ll g_{m2}/C_{22}$

Experimental Results

An IC prototype for the circuit of Figure 15 was fabricated in a standard $2\mu\text{m}$ double-metal, double-poly CMOS process using the MOSIS IC fabrication facility [10], [11]. The OTAs or transconductance amplifiers employed were linearized ones [20]. The diodes were implemented using diode-connected MOS transistors. When the two external inputs y_{o1} and y_{o2} are set to zero, the outputs of the circuit x_1 and x_2 are free running oscillations. If the time constants of the two differential equations (18) are made very different, i.e., $g_{m1}/C_{11} \ll g_{m2}/C_{22}$ then Fitzhugh-Nagumo's equations simulate the behavior of biological cell membranes. The corresponding measured response of the circuit for this case is shown in Figure 20. When signal y_{o2} is considered as the input to the neuron ($y_{o1}=0$) and x_2 as its output, we can see in Figure 21 the, measured input-output relationship of the cell, where y_{o2} is the

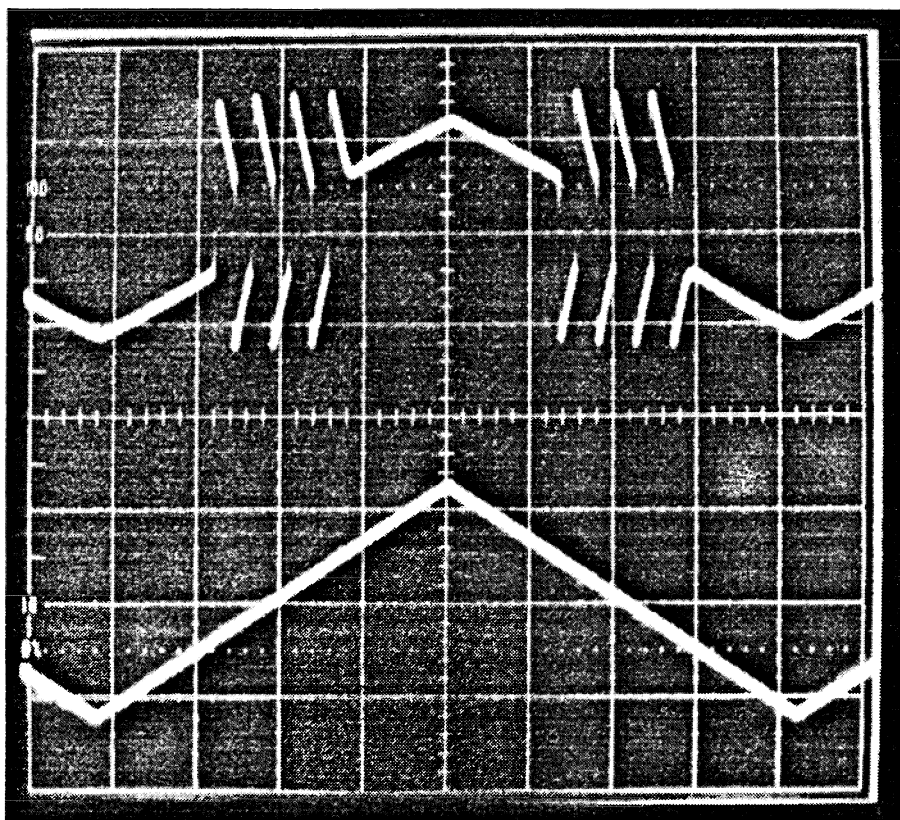


Figure 21. Input-Output Relation of Oscillator of Figure 15; Lower Trace is Input y_{o2} ($y_{o1}=0$), Upper Trace is Output x_2

lower trace and x_2 is the upper trace. Note that the circuit models the behavior of a double-threshold neuron: if the input is either above the upper threshold or below the lower one, no oscillations are produced. But if the input is between the two thresholds, the output is a firing sequence of pulses. Using the interconnection principle of Figure 11 we interconnected two FitzHugh Nagumo cells as shown in Figure 22 using two neurons of Figure 15 and two lossy integrators. The output of the two neurons is shown in Figure 23.

HYSTERESIS NEURON MODEL AND VLSI IMPLEMENTATION

The motivation to develop simpler neuron models (but still keeping the oscillatory nature) is based on their potential use [1]-[11] in implementing hardware for neural network architectures. The free-running oscillator of FitzHugh-Nagumo's system can be further simplified to a hysteresis oscillator

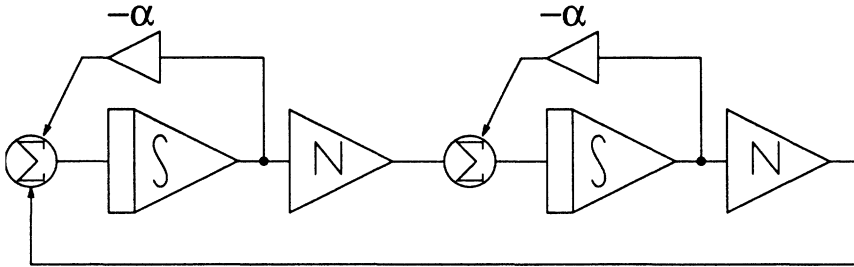


Figure 22. Connections of Two Oscillatory Neurons in a Loop

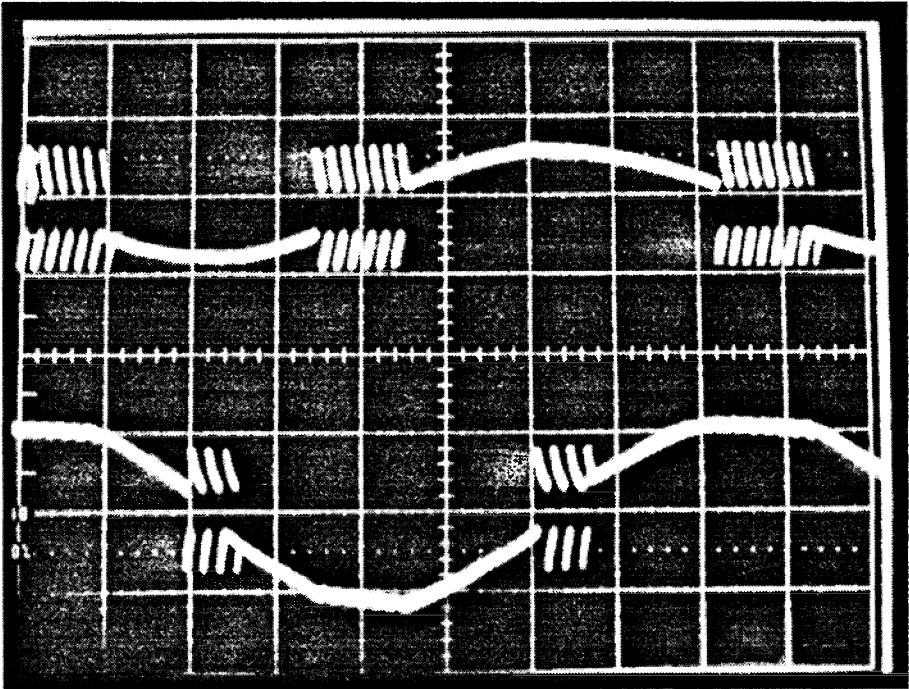


Figure 23. Response of a Two-Neuron-Loop Oscillator

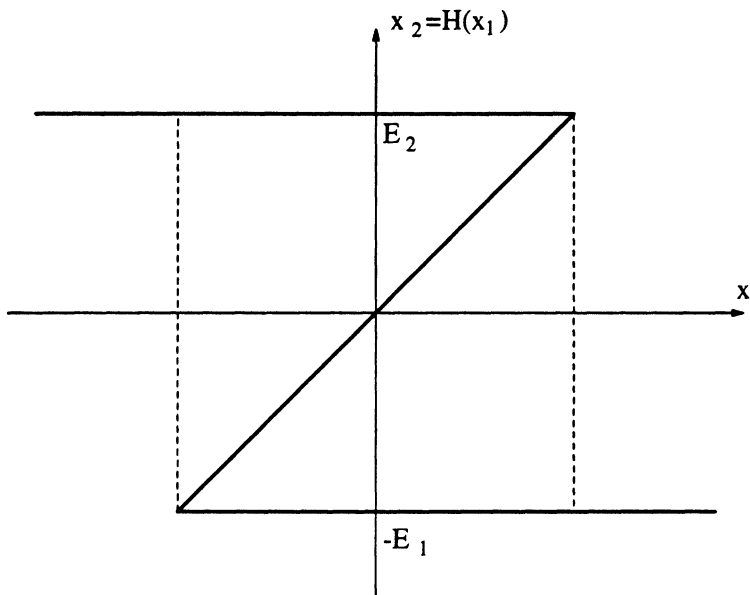


Figure 24. Hysteresis Transfer Function Extracted from FitzHugh-Nagumo's Model

if, in equations (18), we impose the following conditions,

$$\begin{aligned}
 g_c - g_a = g_b - g_a &\rightarrow \infty && \text{in } f(x_2) \\
 y_{o1} = y_{o2} &= 0 \\
 \frac{g_{m1}}{C_{11}} \ll \frac{g_{m2}}{C_{22}} &\rightarrow \infty
 \end{aligned} \tag{27}$$

The consequence of this is that the first equation in (18) will reach its steady state immediately. Therefore, it can be reduced to,

$$x_1 = -\frac{f(x_2)}{g_{m2}} \tag{28}$$

Taking the inverse of equation (28) yields,

$$x_2 = H(x_1) \tag{29}$$

which is a hysteretic transfer function, as depicted in Figure 24. Hence,

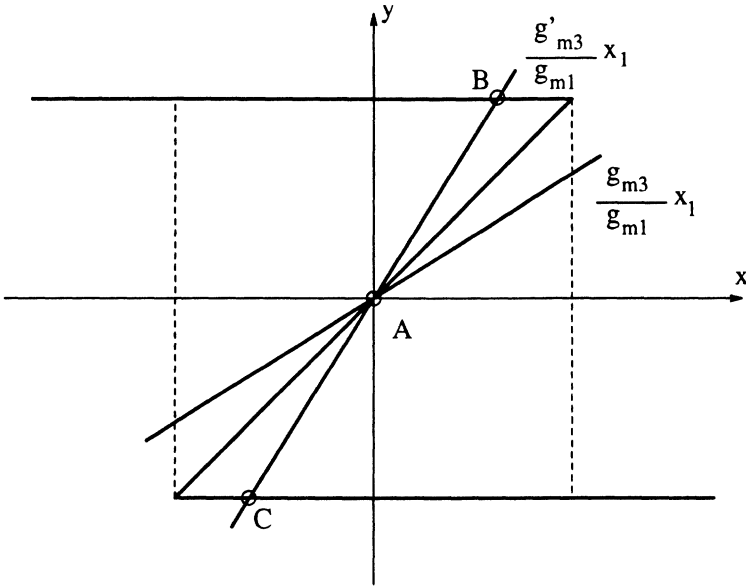


Figure 25. Equilibrium Points of the Hysteretic System

equations (18) simplify into,

$$H(x_1) - \frac{g_{m3}}{g_{m1}}x_1 - \frac{C_{11}}{g_{m1}}\dot{x}_1 = 0 \quad (30)$$

The equilibrium points of this system ($\dot{x}_1 = 0$) are given by the intersection of the two curves,

$$y = H(x_1) \quad (31)$$

$$y = \frac{g_{m3}}{g_{m1}}x_1$$

as is shown in Figure 25. If $(g_{m1}g_{m2})/g_a > g_{m3}$, the only equilibrium point is *A*, which is unstable according to the earlier analysis. In this case, equation (30) represents an oscillator. But if $(g_{m1}g_{m2})/g_a < g_{m3}$ there are two more equilibrium points, *B* and *C*, which are stable, so that the system will stop in either one of them and then no oscillations will be present. Therefore, the oscillator (neuron) can be turned on and off by changing g_{m3} . A circuit implementation of such a system is shown in Figure 26 [21]. For a CMOS implementation it is however easier to substitute the linear resistor g_{m3}^{-1} by a

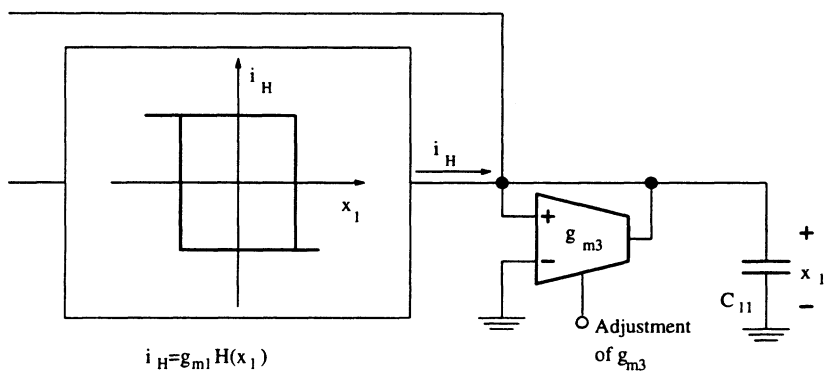


Figure 26. Block Diagram of Hysteretic Neuron Cell

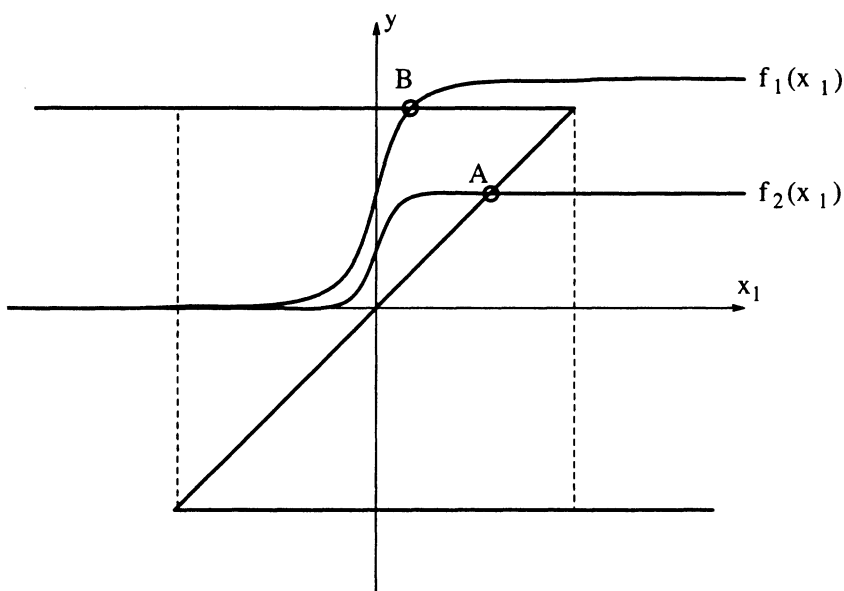


Figure 27. Equilibrium Points of the Modified Hysteretic System

nonlinear one, as shown in Figure 27 [22]. This would change equation (30) into

$$H(x_1) - f(x_1) - \frac{C_{11}}{g_{m1}} \dot{x}_1 = 0 \quad (32)$$

A circuit diagram that realizes equation (32) is shown in Figure 28. Note that

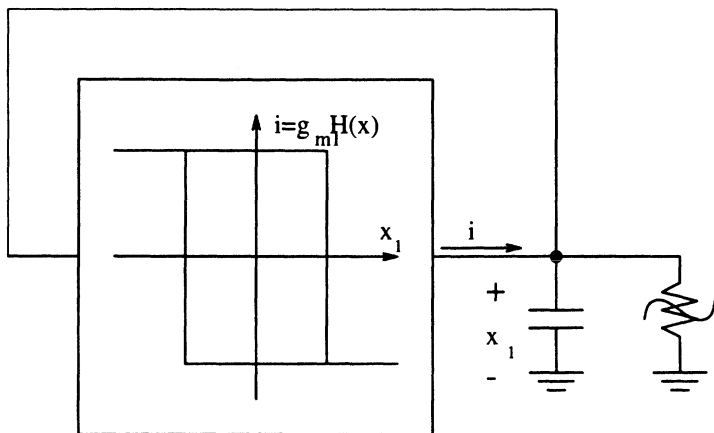


Figure 28. Block Diagram of CMOS Circuit for Modified Hysteretic Neuron Cell

the shape of the nonlinear resistor has to be able to change in the way shown in Figure 27, so that the two equilibrium points *A* and *B* can be obtained. When equilibrium point *A* is obtained, since it is unstable, the system will oscillate. But if *B* is the equilibrium point, since it is stable, the oscillations will disappear. An appropriate CMOS implementation for the nonlinear resistor is shown in Figure 29. If x_1 is positive the current I_c goes through M_2 and is reflected to the input node so that $f(x_1)=I_c$. If x_1 is negative, I_c goes through M_1 and no current is reflected back to the input node, $f(x_1)=0$.

In order to implement the T-mode hysteresis element (note that the output is a current while the input is a voltage) of Figure 28, we can use the circuit diagram given in Figure 30.

The operation of the double output transconductance amplifier in Figure 30 is defined approximately by the following equation,

$$i = \begin{cases} I_{ss} & \text{if } v > 0 \\ -I_{ss} & \text{if } v < 0 \end{cases} \tag{33}$$

Therefore, when $v > 0$ then $i = I_{ss} > 0$ and $v = E^+ - x$, which means that $x < E^+$. On the other hand when $v < 0$, then $i = -I_{ss} < 0$ and $v = -E - x$, which means that $x > -E^-$.

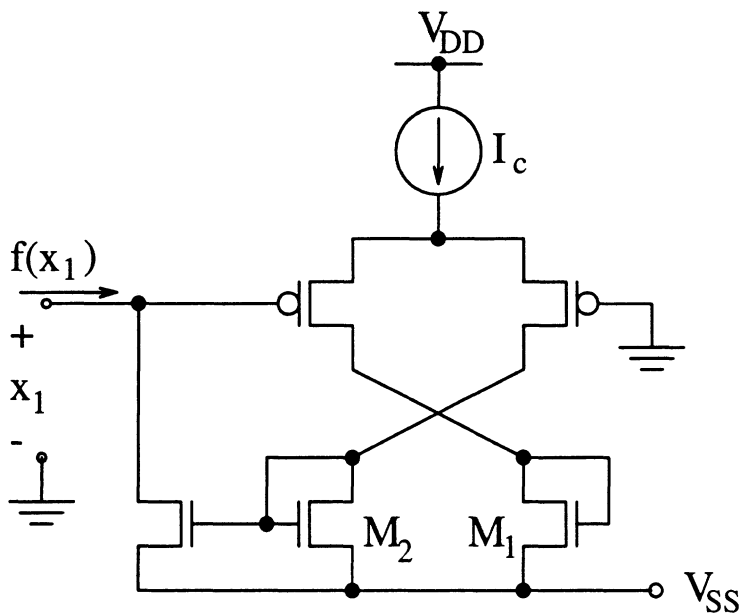


Figure 29. CMOS Circuit Implementation for Nonlinear Resistor

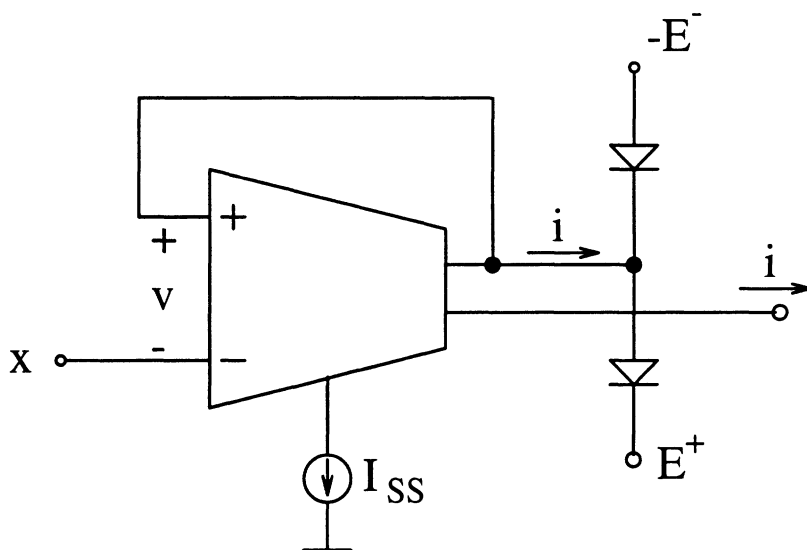


Figure 30. Circuit Diagram for T-Mode Hysteresis Amplifier

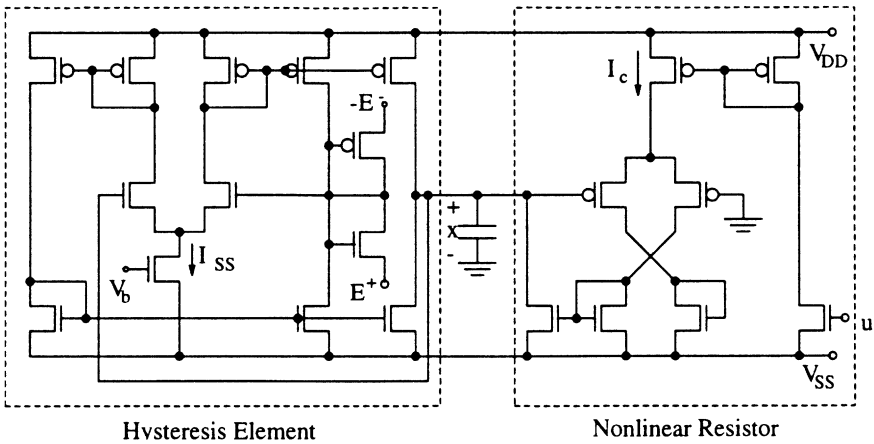


Figure 31. CMOS Circuit for the Modified Hysteretic Neuron Cell

Summarizing,

$$i = \begin{cases} I_{ss} & \text{if } x < E^+ \\ -I_{ss} & \text{if } x > -E^- \end{cases} \quad (34)$$

which is a hysteretic function.

A complete CMOS circuit for the neuron or oscillator of Figure 28 is shown in Figure 31. This simple oscillator was fabricated in a $3\mu\text{m}$ double-metal CMOS process [22] using MOSIS. The input (u_e in Figure 31) output (x in Figure 31) relationship for this neural oscillator is shown in Figure 32. The parameters that can be adjusted in the neural oscillator of Figure 31 are I_{ss} , E^+ and E^- . E^+ and E^- control the amplitude of the oscillations at $x(t)$, and I_{ss} controls the slope of the triangular waveforms. The three of them can be used to change the frequency of the oscillations.

We also connected a two neuron loop, like the previously mentioned in Figure 22, using these hysteretic type oscillators. The result is shown in Figure 33.

OSCILLATORY TYPE T-MODE NEURAL NETWORK SYSTEMS

Here we will show how to use the neural oscillator of Figure 31 to build complete but simple neural network systems. We will illustrate this with two examples: a Hopfield network [23] and a BAM network [24]. In both cases we need to provide some neurons interconnecting circuitry, called *synapses*.

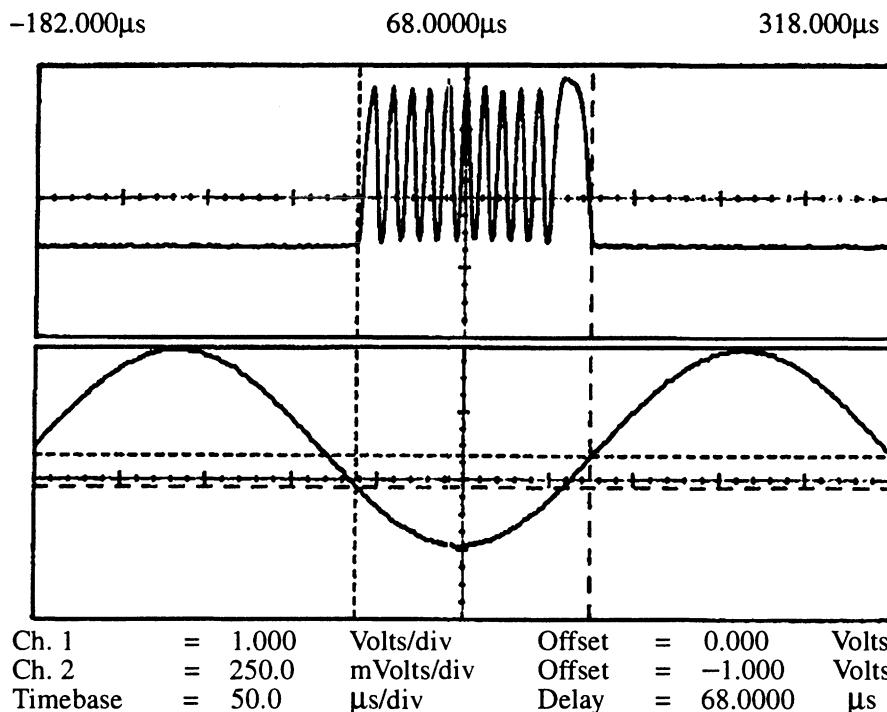


Figure 32. Input-Output Relationship for the CMOS Hysteresis Neural Oscillator

According to what was described previously, each neuron sums the contributions of its interconnecting neurons and performs a lossy integration with this sum. Using T-mode (Transconductance-mode) analog circuit design techniques, the circuit shown in Figure 34 could perform this task, where, x_1, \dots, x_q are the output voltages of previous neurons, x_{ext} is an external input, g_{m1}, \dots, g_{mq} are the transconductance gains of the synaptic transconductance amplifiers, R and C are responsible for the lossy integration¹, and u_e (see Figure 31) is the neural oscillator controlling signal. The values of g_{m1}, \dots, g_{mq} are the *weights* of the synaptic interconnections. For the transconductance amplifiers the circuit shown in Figure 35 was used [11]. The measured DC transfer characteristics for this amplifier, when connecting five of them in parallel and loaded with a 20KΩ resistor as shown in Figure 36, are given in Figure 37. The values of the weights (see w in Figure 37) were varied between $w=-1.2V$ and $w=-2.8V$, and the input signal was swept between $V_{in}=-2.2V$

1. Resistor R will not be implemented physically. The output resistance of the transconductance amplifiers is sufficient.

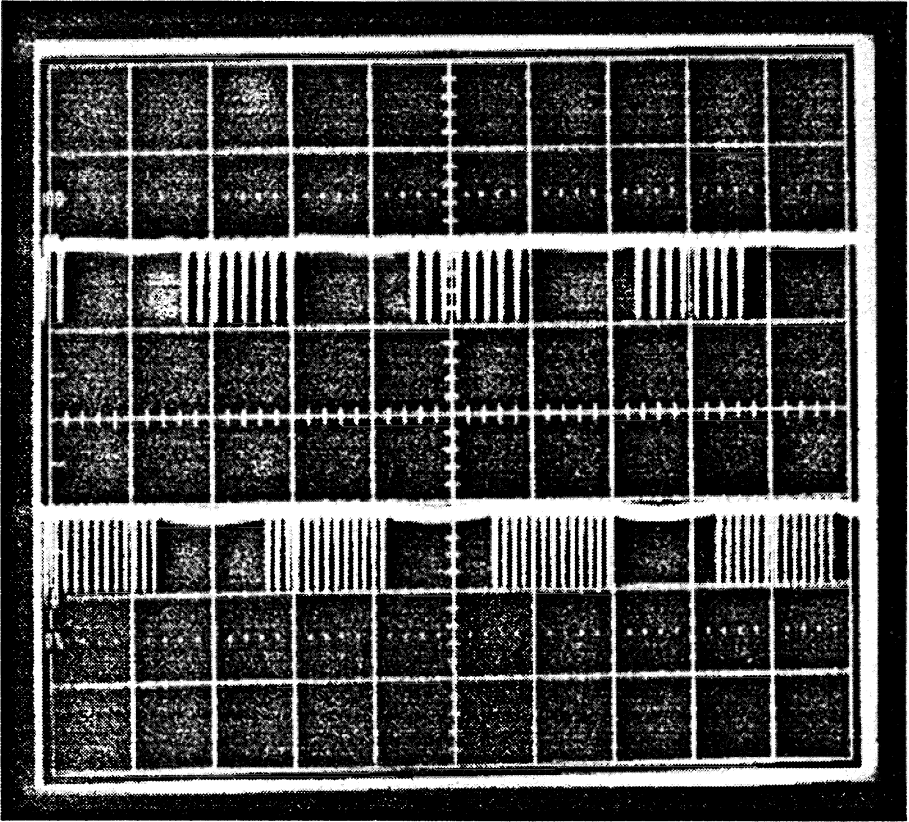


Figure 33. Pattern Generation by a Loop of Two Hysteretic Neural Cells

and $V_{in}=+0.2V$. Note that these multipliers are four-quadrant multipliers. They were designed to be used for non-oscillatory neural networks [11], [25], [26]. But in an oscillatory neural system based on the neurons of Figure 31 only two-quadrant multipliers are needed: if x_i in Figure 34 is not oscillating then it should be $i_i=0$, if x_i is oscillating then i_i should also be an oscillating signal. This can be accomplished by making $x_i(t)|_{non-oscillating}=GND_{top}$, by

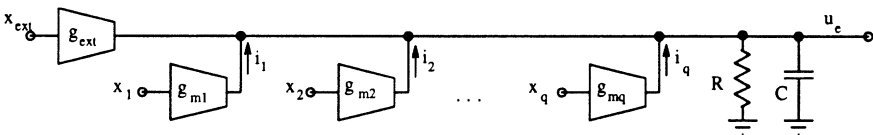


Figure 34. T-mode Circuit for Interconnections of Neurons

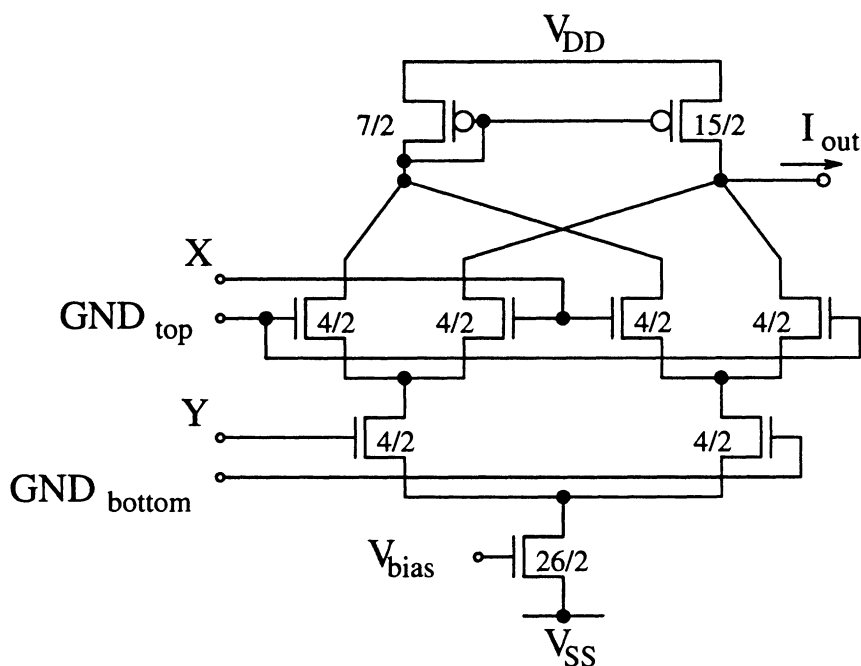


Figure 35. Actual Schematic of Fabricated Multipliers

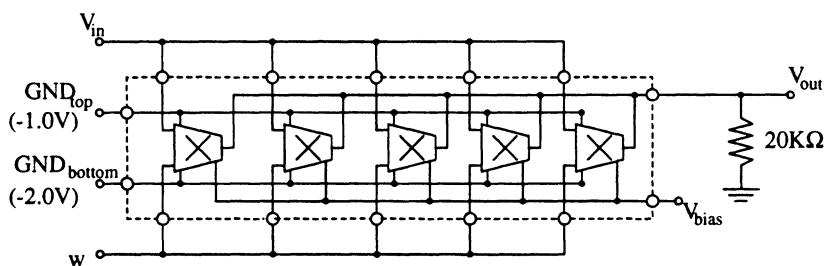
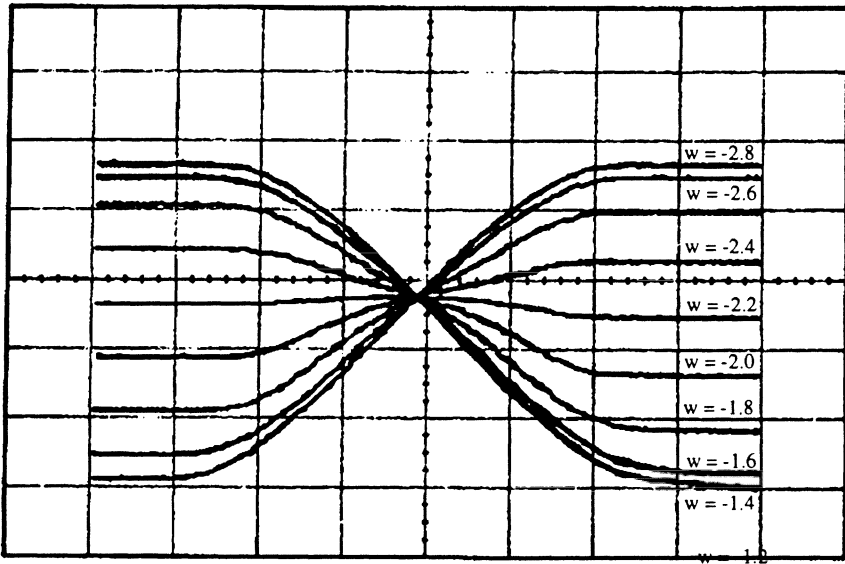


Figure 36. Experimental Set Up for Measurement of DC Characteristics of Transconductance Synaptic Multiplier

properly adjusting the value of E^- in Figure 31. With this in mind we can now assemble the Hopfield and BAM networks using these oscillatory neuron cells.

Oscillatory Hopfield Network

The Hopfield network [23] is a single layer neural network in which every neuron provides input to all others excluding itself. Also the weights



Func1 Vert = 900.0 mVolts/div Offset = 0.000 Volts
 Horizontal = 300.0 mVolts/div Offset = -1.000 Volts

Figure 37. Measurement of DC Transfer Curves of five Synaptic Multipliers in Parallel for $V_{bias}=-3.77V$

are symmetric: the weight of the synapse that connects the output of neuron i to the input of neuron j , w_{ij} , is equal to the one of the synapse connecting the output of neuron j to the input of neuron i , w_{ji} .

We built a 5-neuron Hopfield network, and in Figure 38 is shown the interconnection topology that we had in our experimental set up using the synaptic transconductance multipliers together with the oscillatory neurons. Note in Figure 32 that the neuron behaves equivalently to having a negative gain: if u_e is below the threshold the neuron is firing, otherwise the output of the neuron is steady (and equal to $GND_{top}=-1.0V$). This means that the normalized weights have to be multiplied by -1 . If we want to store the pattern 10101 the corresponding normalized weight matrix with sign change is [23],

$$\begin{bmatrix} 0 & +1 & -1 & +1 & -1 \\ +1 & 0 & +1 & -1 & +1 \\ -1 & +1 & 0 & +1 & -1 \\ +1 & -1 & +1 & 0 & +1 \\ -1 & +1 & -1 & +1 & 0 \end{bmatrix} \tag{35}$$

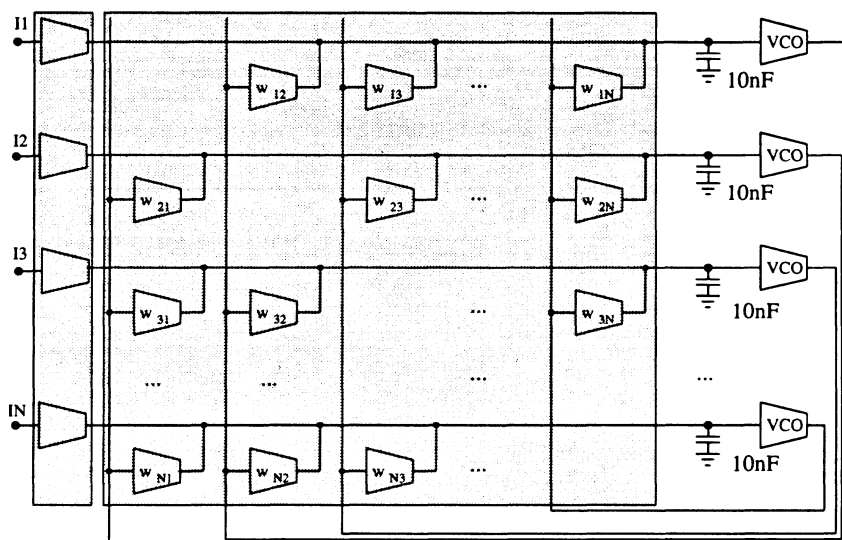


Figure 38. Interconnection Topology for Oscillatory Hopfield Network

For the multipliers we used $V_{bias} = -3.77V$ and

$$\begin{aligned}
 Y &= -2.8V & \text{for} & & w_{ij} &= +1 \\
 Y &= -2.0V & \text{for} & & w_{ij} &= 0 \\
 Y &= -1.2V & \text{for} & & w_{ij} &= -1
 \end{aligned}
 \tag{36}$$

Figure 39 summarizes the stable steady states obtained for all possible input combinations. Note that the system either converges to the stored pattern 10101 or to its complementary 01010. The transient response of the convergence to pattern 10101 is shown in Figure 40. Each one of them shows the input and output voltages of one of the neurons.

Oscillatory BAM Network

A BAM network [24] is a two layer neural network in which each neuron of a layer is only connected to all the neurons in the other layer. The weights are also symmetric: the weight of the synapse connecting the output of neuron i of layer 1 to the input of neuron j in layer 2, w_{ij} , is equal to the weight of the synapse connecting the output of neuron j in layer 2 to the input of neuron i in layer 1, w_{ji} .

We built a 3+3 neurons oscillatory BAM. The interconnection topology is shown in Figure 41. We programmed the pattern shown in Figure 42, which

Input		Stable Pattern	
(0)	00000	01010	(10)
(1)	00001	10101	(21)
(2)	00010	01010	(10)
(3)	00011	01010	(10)
(4)	00100	10101	(21)
(5)	00101	10101	(21)
(6)	00110	01010	(10)
(7)	00111	10101	(21)
(8)	01000	01010	(10)
(9)	01001	01010	(10)
(10)	01010	01010	(10)
(11)	01011	10101	(21)
(12)	01100	01010	(10)
(13)	01101	01010	(10)
(14)	01110	01010	(10)
(15)	01111	10101	(21)
(16)	10000	10101	(21)
(17)	10001	10101	(21)
(18)	10010	01010	(10)
(19)	10011	10101	(21)
(20)	10100	10101	(21)
(21)	10101	10101	(21)
(22)	10110	01010	(10)
(23)	10111	10101	(21)
(24)	11000	01010	(10)
(25)	11001	10101	(21)
(26)	11010	01010	(10)
(27)	11011	01010	(10)
(28)	11100	01010	(10)
(29)	11101	10101	(21)
(30)	11110	01010	(10)
(31)	11111	10101	(21)

Figure 39. Measured Stable States for Oscillatory Hopfield Network Loaded with the Pattern 10101.

has the normalized weight matrix (with its corresponding sign change) [24]

$$\begin{bmatrix} -1 & +1 & -1 \\ +1 & -1 & +1 \\ -1 & +1 & -1 \end{bmatrix} \quad (37)$$

The synaptic multipliers were biased with $V_{bias} = -3.77V$, and their weight inputs Y (see Figure 35) were connected to

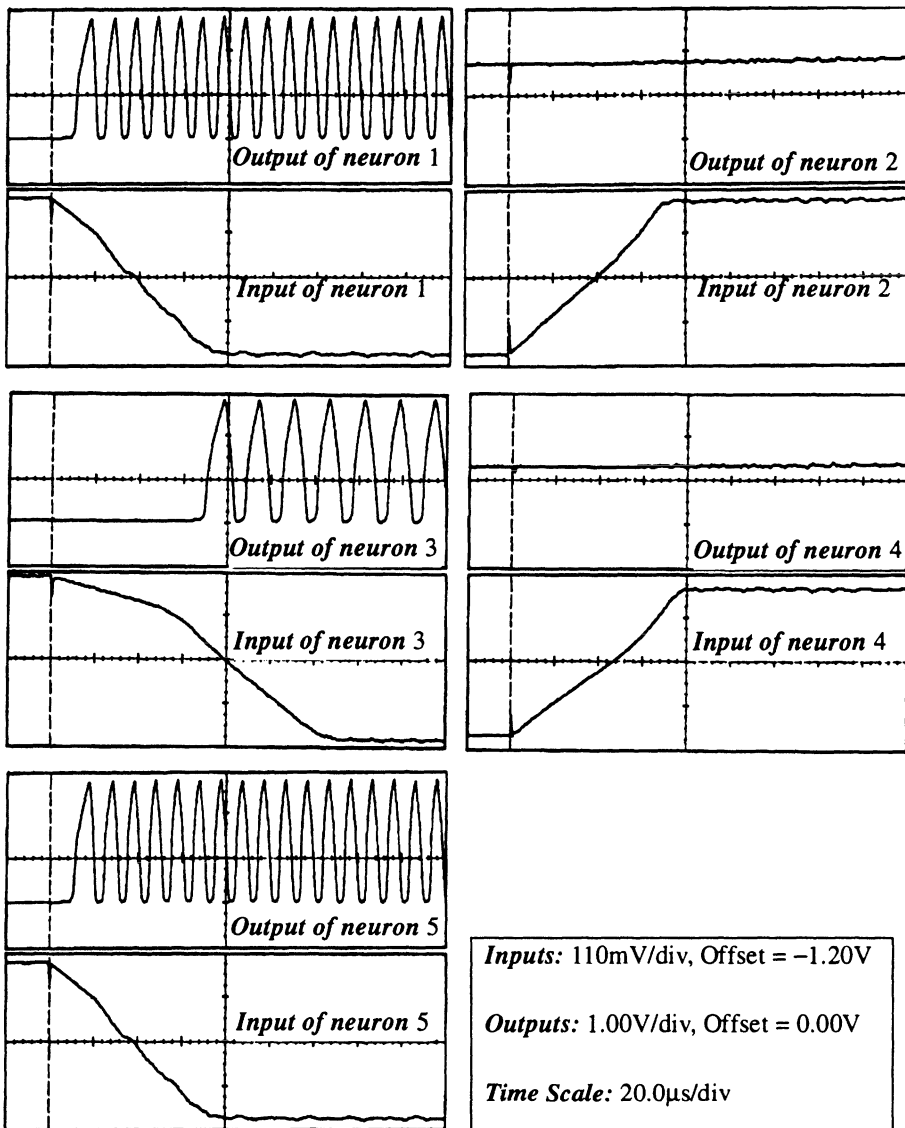


Figure 40. Convergence to Pattern 10101 for Oscillatory Hopfield Network

$$\begin{aligned}
 Y &= -2.8V & \text{for} & & w_{ij} &= +1 \\
 Y &= -1.2V & \text{for} & & w_{ij} &= -1
 \end{aligned}
 \tag{38}$$

shows the transient response of the convergence to pattern A when the input is A. Each figure shows the input and output voltage for one of the neurons.

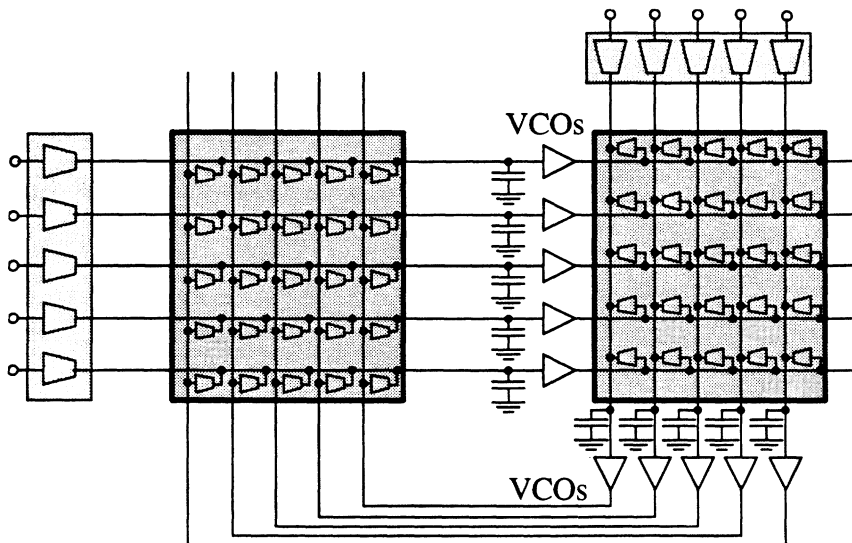


Figure 41. Interconnection Topology for Oscillatory BAM

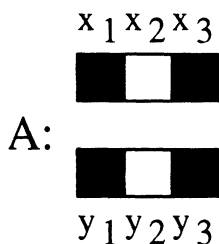
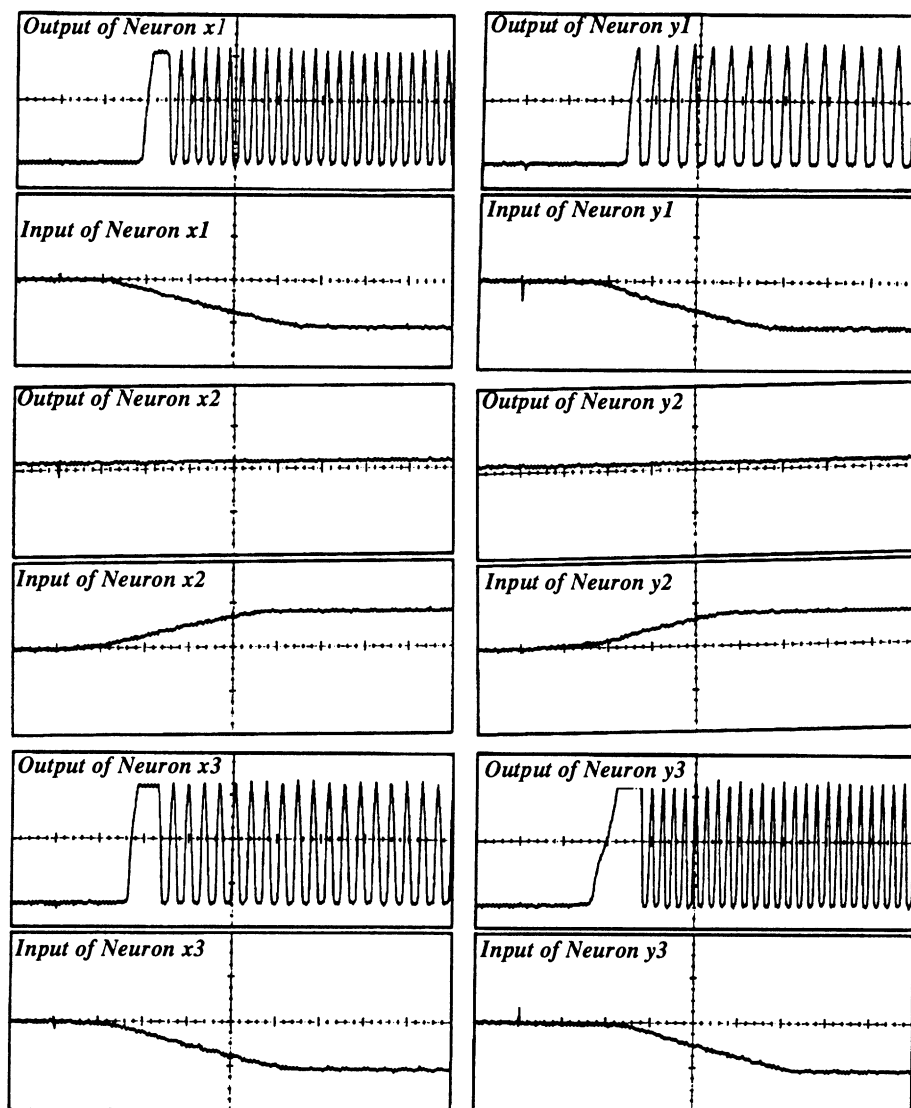


Figure 42. Pattern to be Stored in the Oscillatory BAM

EXTENSION TO CHAOTIC OSCILLATORY NEURONS

An interesting and natural extension of the oscillatory neuron based neural network systems described so far, are neural systems made up of coupled chaotic oscillators. Such systems are not only thought to explain some of the sensory signal processing in living beings [27]-[29], but have been also suggested for engineering applications such as signal detection in very noisy environments [30], robot path planning [31], etc.

There is no generally accepted definition of chaos. However, from a practical point of view it can be defined as a bounded steady state behavior that is not an equilibrium point, not periodic, and not quasi-periodic (i.e., a linear combination of uncorrelated periodic components). A conventional oscillator has a steady state called *limit cycle*, which is a bounded closed path in the state variables space. A chaotic oscillator does not converge to a *limit cycle* but to



Inputs: 500mV/div, Offset = -1.00V

Outputs: 1.00V/div, Offset = 500mV

Time Scale: 50.0 μ s/div

Figure 43. Convergence to Pattern A for Oscillatory BAM

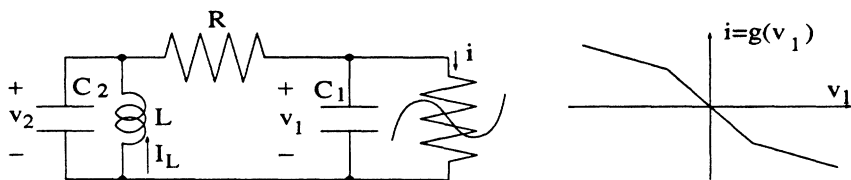


Figure 44. Chua's Circuit

an *attractor*, which is a bounded closed surface in which the steady state trajectories are trapped but are not predictable.

The simplest autonomous continuous time electrical circuit with chaotic behavior proposed so far is Chua's circuit [32], which is shown Figure 44. This circuit is described by the following set of first order nonlinear differential equations

$$\begin{aligned}\dot{v}_1 &= \frac{1}{RC_1} (v_2 - v_1) - \frac{1}{C_1} g(v_1) \\ \dot{v}_2 &= -\frac{1}{RC_2} (v_1 - v_2) + I_L \\ \dot{I}_L &= -\frac{1}{L} v_2\end{aligned}\quad (39)$$

The conditions under which equations (39) exhibit chaotic behavior are available elsewhere [32], [33]. Using the general circuit technique of Figure 14 described earlier to map these equations into a T-mode circuit results in the circuit depicted in Figure 45. The corresponding mapping of variables and parameters is

$$\begin{aligned}v_1 &\equiv x_1 & v_2 &\equiv x_2 & I_L &\equiv x_3 \frac{g_3}{C_2} \\ \frac{1}{RC_1} &\equiv \frac{g_1}{C'_1} & \frac{1}{RC_2} &\equiv \frac{g_2}{C'_2} & \frac{1}{L} &\equiv \frac{g_4}{C_3}\end{aligned}\quad (40)$$

The nonlinear resistor can be implemented using the circuit of Figure 17 and making $0 < -g_l < g_a$. Transistor level hspice simulations of this circuit have been performed [34] and are shown in Figure 46, where steady state trajectories, inside the attractor called *double scroll*, can be seen.

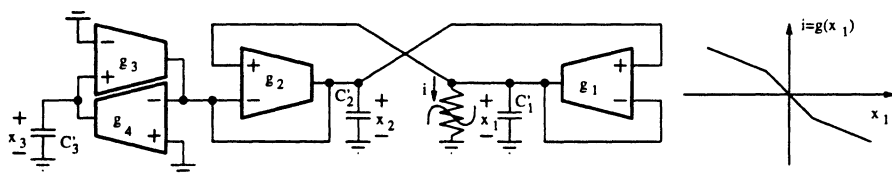


Figure 45. T-mode Implementation of Chua's Circuit

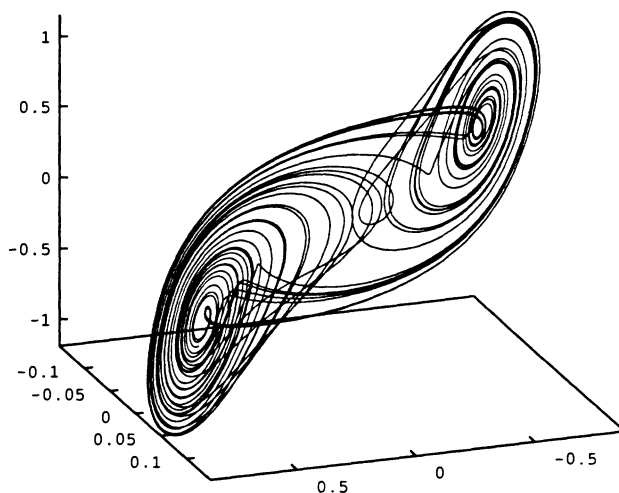


Figure 46. Transistor Level Hspice Simulation of T-mode Chua's Circuit

Such a chaotic oscillator has the potential to be used in a larger VLSI circuit in order to implement chaotic neural network systems for modelling of biological chaotic systems [27]-[29] or for practical engineering applications [30].

CONCLUSIONS

We have seen that using oscillatory type of neurons it is possible to build conventional neural network systems, like for example a Hopfield or a BAM system. Furthermore, we have built an oscillatory neural cell based on successive simplifications derived from the biological principles that define the behavior of living neurons. We started our discussion by describing the biochemical principles responsible for the generation of electrical nerve pulses in biological cells, then an equivalent circuit was derived that modelled the physiological behavior. A simplified circuit model (by FitzHugh and Nagumo)

was obtained and addressed using a circuit theoretical approach in order to derive a VLSI compatible design. This VLSI CMOS design was fabricated, tested and characterized. It was further simplified to a hysteretic based neural oscillator. This one has been used to build a 5-neurons Hopfield network and a 3+3 neurons BAM network.

We have seen that there is a strong connection between the living biological neurons of real nervous systems, and the artificial neural network algorithms available in the literature. This connection is even more evident if oscillatory (or even chaotic) neurons are used, and we have seen that it is feasible to build conventional Hopfield and BAM hardware networks using oscillatory neurons.

ACKNOWLEDGMENTS

The authors would like to thank Mr. Manuel Delgado-Restituto for providing the circuit and results of the section on Chaotic Oscillators.

References

- [1] A. Hamilton, A. F. Murray, D. J. Baxter, S. Churcher, H. M. Reekie, and L. Tarassenko, "Integrated Pulse Stream Neural Networks: Results, Issues, and Pointers", *IEEE Trans. Neural Networks*, vol. 3, May, pp. 385-393, 1992.
- [2] A. F. Murray, D. Del Corso and L. Tarassenko, "Pulse-Stream VLSI Neural Networks Mixing Analog and Digital Techniques", *IEEE Trans. Neural Networks*, vol. 2, March, pp. 193-204, 1991.
- [3] G. Moon, M. E. Zaghoul, and R. W. Newcomb, "VLSI Implementation of Synaptic Weighting and Summing in Pulse Coded Neural-Type Cells", *IEEE Trans. Neural Networks*, vol. 3, May, pp. 394-403, 1992.
- [4] M. R. DeYong, R. L. Findley, and C. Fields, "The Design, Fabrication, and Test of a New VLSI Hybrid Analog-Digital Neural Processing Element", *IEEE Trans. Neural Networks*, vol. 3, May, pp. 363-374, 1992.
- [5] J. E. Tomberg and K. K. K. Kaski, "Pulse-Density Modulation Technique in VLSI Implementations of Neural Network Algorithms", *IEEE J. Solid-State Circuits*, vol. 25, No. 5, pp. 1277-1286, October, 1990.
- [6] N. El-Leithy and R. W. Newcomb, "Hysteresis in Neural-Type Circuits", *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS88)*, Helsinki, pp. 993-996, 1988.

- [7] D. Watola, D. Gembala, and J. Meador, "Competitive Learning in Asynchronous Pulse Density Integrated Circuits", *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS92)*, San Diego, pp. 2216-2219, 1992.
- [8] J. Lazzaro, "Low-Power Silicon Spiking Neurons and Axons", *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS92)*, San Diego, pp. 2220-2223, 1992.
- [9] J. G. Elias, H. H. Chu, and S. M. Meshreki, "A Neuromorphic Impulsive Circuit for Processing Dynamic Signals", *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS92)*, San Diego, pp. 2208-2211, 1992.
- [10] B. Linares-Barranco, E. Sánchez-Sinencio, A. Rodríguez-Vázquez and J. L. Huertas, "A CMOS Implementation of FitzHugh-Nagumo Neuron Model", *IEEE J. Solid-State Circuits*, vol. 26, No. 7, July, pp. 956-965, 1991.
- [11] B. Linares-Barranco, "Analog Neural Network VLSI Implementations", PhD Dissertation, Texas A&M University, College-Station, Texas, December 1991.
- [12] F. L. Straud, *Physiology. A Regulatory System Approach*, New York, MacMillan, 1978.
- [13] G. M. Shepherd, *Neurobiology*, New York, Oxford University Press, 1988.
- [14] A. L. Hodgkin and A. F. Huxley, "A Qualitative Description of Membrane Current and its Application to Conduction and Excitation in Nerves", *Journal of Physiology*, vol. 177, pp. 500-544, 1952.
- [15] J. P. Keener, "Analog Circuitry for the Van der Pol and FitzHugh-Nagumo Equations", *IEEE Trans. Systems, Man and Cybernetics*, vol. 13, No. 5, Sept/Oct, pp. 1010-1014, 1983.
- [16] R. FitzHugh, "Impulses and Physiological States in Theoretical Models of Nerve Membrane", *Biophysical Journal*, vol. 1, pp. 445-466, 1961.
- [17] J. Nagumo, S. Arimoto and S. Yoshizawa, "An Active Pulse Transmission Line Simulating Nerve Axon", *Proc. IRE*, vol. 50, pp. 2061-2070, 1964.
- [18] E. Sánchez-Sinencio, J. Ramírez-Angulo, B. Linares-Barranco and A. Rodríguez-Vázquez, "Operational Transconductance Amplifier Based Nonlinear Function Syntheses", *IEEE J. Solid-State Circuits*, vol. 24, No. 6, December, pp. 1576-1586, 1989.
- [19] Leon O. Chua, Charles A. Desoer and Ernest S. Kuh, *Linear and Nonlinear Circuits*, New York, McGraw-Hill, 1987. .

- [20] A. P. Nedungadi and R. L. Geiger, "High-Frequency Voltage-Controlled Continuous-Time Lowpass Filter Using Linearized CMOS Integrators", *Electronics Letters*, vol. 22, June, pp. 729-731, 1986.
- [21] B. Linares-Barranco, E. Sánchez-Sinencio, A. Rodríguez-Vázquez and J. L. Huertas, "A Programmable Neural Oscillator Cell", *IEEE Trans. Circuits and Systems*, vol. 36, No. 5, May, pp. 756-761, 1989.
- [22] B. Linares-Barranco, E. Sánchez-Sinencio, R. W. Newcomb, A. Rodríguez-Vázquez and J. L. Huertas, "A Novel CMOS Analog Neural Oscillator Cell", *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS89)*, Portland, pp. 794-797, 1989.
- [23] D. W. Tank and J. J. Hopfield, "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", *IEEE Trans. Circuits and Systems*, vol. 33, No. 5, May, 1986.
- [24] B. Kosko, "Adaptive Bidirectional Associative Memories", *Applied Optics*, vol. 26, No. 23, pp. 4947-4960, 1 December, 1987.
- [25] B. Linares-Barranco, E. Sánchez-Sinencio, A. Rodríguez-Vázquez, and J. L. Huertas, "A Modular T-Mode Design Approach for Analog Neural Network Hardware Implementations", *IEEE J. Solid-State Circuits*, vol. 27, No. 5, May, pp. 701-713, 1992.
- [26] B. Linares-Barranco, E. Sánchez-Sinencio, A. Rodríguez-Vázquez, and J. L. Huertas, "A CMOS Analog Adaptive BAM with On-Chip Learning and Weight Refreshing", *IEEE Trans. Neural Networks*, vol. 4, March 1993.
- [27] P. Erdi, T. Grobler, and G. Barna, "Oscillation and Chaos in a Model of the Olfactory Bulb", *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, pp. 917-920, July 1992.
- [28] Z. Li and J. J. Hopfield, "Modeling the Olfactory Bulb and its Neural Oscillatory Processing", *Biological Cybernetics*, 61, pp. 379-392, 1989.
- [29] W. J. Freeman, "Imaging and Interpreting the Geometries of Chaotic Attractors of Brain Systems in Perceptual Information Processing", *Proceedings of the International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, pp. 587-590, 1990.
- [30] D. L. Bix and S. J. Popenberg, "Chaotic Oscillators and Complex Mapping Feed Forward Networks (CMFFNS) for Signal Detection in Noisy Environments", *Proceedings of the 1992 International Joint Conference on Neural Networks*, Baltimore, vol. II, pp. 881-888, June 1992.

- [31] J. Tani, "Diversity and Regularity in Chaotic Wandering of Robot", *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, pp. 955-960, July 1992.
- [32] L. O. Chua, M. Komuro, and T. Matsumoto, "The Double Scroll Family", *IEEE Trans. on Circuits and Systems.*, vol. CAS-33, pp. 1072-1118, Nov. 1986.
- [33] T. Matsumoto, L. O. Chua, and M. Komuro, "The Double Scroll", *IEEE Trans. on Circuits and Systems.*, vol. CAS-32, pp. 798-818, August 1985.
- [34] M. Delgado-Restituto and A. Rodríguez-Vázquez, "A CMOS VLSI Implementation of Chua's Circuit", *Journal of Circuits, Systems and Computers*, (Special Issue on Chua's Circuit), 1993.

A DIGITAL NEURAL NETWORK ARCHITECTURE USING RANDOM PULSE TRAINS

Gamze E. Salam and Rodney M. Goodman

*Department of Electrical Engineering, 116-81
California Institute of Technology, Pasadena, CA 91125*

MOTIVATION AND PREVIOUS WORK

It has been shown that random pulse trains have interesting properties, which make them suitable for many kinds of computations [Tomlinson 91, Tomlinson 90, Murray 91]. Under certain restricted conditions, they can do quite complex computation with simple processing units, using probability and time to advantage. This is quite appropriate for VLSI implementations of neural networks where the high switching speeds of devices can be traded in for small processing units to implement massively parallel architectures. Thus, it is not surprising at all that random pulse trains have already been applied to hardware implementations of neural networks. Their properties have been exploited both in noise generation for Boltzmann learning in Bellcore's stochastic learning microchip [Alspector 91] and in computing the transfer function of neural processing units in Neural Semiconductor's Digital Neural Network Architecture (DNNA)[Tomlinson 91]. Here, we will describe a digital architecture utilizing random bit sequence properties, which combines several features of previous implementations, and adds new ones, such as analog valued inputs and wider dynamic range in digital weight representation.

SIMULATING THE DIGITAL ARCHITECTURE USING RANDOM PULSES

Relevant Properties of Pseudorandom Number (PN) Sequences

Here, we restrict our discussion to binary random number sequences generated by linear feedback shift registers (LFSR), using the properties of irreducible polynomials. Further, we will assume that the two values an element of the sequence can take on are (+1) and (-1). This simplifies the mathematics considerably, but all the discussion that follows applies equally well to sequences of (1)'s and (0)'s.

We will start by stressing that no finite sequence is ever truly random. Hence, we use the term *pseudorandom*. Since sequences generated by LFSR are of finite length, we can not expect true *randomness* itself and will have to contend with certain properties of randomness, instead. We continue to outline these properties [5]:

Periodicity and Maximum Period. The succession of states in a shift register is periodic. The length of the period can not exceed $2^r - 1$, where r is the number of registers. If a sequence has maximum length, its characteristic polynomial is irreducible.

Auto-correlation. If $\{x_n\} = \{x_0, x_1, x_2, \dots\}$ is any finite sequence of binary terms of length N , the auto-correlation function $\Phi(\tau)$ is defined by

$$\Phi(\tau) = \frac{1}{N} \sum_{n=1}^N x_n x_{n+\tau}$$

Here τ can be thought of as a phase shift.

Auto-correlation function $\Phi(\tau)$ of the sequences we describe is two-valued. Namely,

$$\Phi(\tau) = \begin{cases} 1 & \text{if } \tau = 0 \\ -\frac{1}{N} & \text{if } 0 < \tau < N \end{cases}$$

Equal Number of (+1)'s and (-1)'s. In each period of the LFSR sequence the number of (+1)'s are almost equal to the number of (-1)'s. Because in a LFSR of r registers, r consecutive zeros (or as we have been referring to them, (-1)'s) are not generated, the number of (+1)'s exceed the number of (-1)'s by one.

Properties of Consecutive Terms. Here, we will switch back to referring to the elements in the LFSR sequence as (1)'s and (0)'s. In an LFSR of r registers, every possible array of r consecutive terms, except all (0)'s occurs exactly once. The number of consecutive (1)'s or (0)'s of length k (where $0 < k < r - 1$) is 2^{r-k-2} .

Brief Description of Architecture

Consider a typical feed-forward neural network where the activation function of each neuron o_i is derived from the outputs of the previous layer of neurons o_j , through weights w_{ji} . In the digital neural network architecture we describe, the product pairs $(w_{ji}o_j)$ and the synaptic sum $(\sum_j w_{ji}o_j)$ are computed using a different paradigm of arithmetic. An input is represented in pulse-width modulated format. Unlike in some of the previous implementations, this allows for analog valued input to the first layer. The representation of each input is stored in a shift register stack, which uses a random bit stream as clock. As mentioned in the previous section these random bit streams have a rather interesting property that a second bit stream generated by a single shift or more has very negligible correlation to the original stream [Golomb 67]. Since our objective is to perform arithmetic functions between pulses using simple logic gates, this is an essential property [Tomlinson 91, Alspector 91].

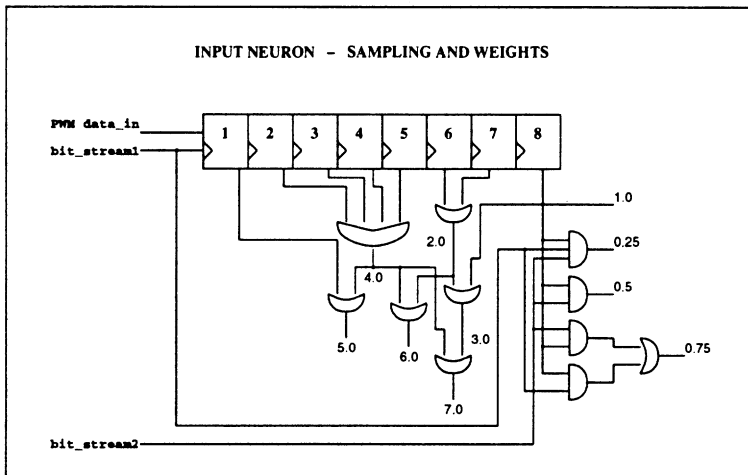


Figure 1. Input neuron with ten weights

An input neuron is shown in Figure 1. The method of computing the $(w_{ji}o_j)$ pairs is also indicated in the figure. Products by weights less than 1 are

achieved by a logical AND of the input with the appropriate number of random bit strings. Products by weights greater than 1 are achieved by a logical OR of the appropriate number of input slices in time.

The fanout from each input neuron to synapses can be kept low by routing only the essential product lines and reconstructing the needed $(w_{ji}; o_j)$ pair at the synapse.

These product pairs are wire-OR'ed at the synapse and fed directly to the output. The saturating nature of the sum compares favorably to a hyperbolic tangent function, thus there is no need for a transfer function for the output neuron.

Pulse Arithmetic using Logic Operations

The 'AND' product has been described previously by Gaines [Gaines 69]. Here, we wish to provide some insight into the 'OR' product case.

OR - product for $(w_{ji} > 1)$. Referring to Figure 1, we define a binary bit stream input to the stack, $x(t)$, which is a deterministic function of time. We further define each bit of the stack as $x(R(I))$, which has some randomness properties. We can see that

$$x(R(I)) = x(t - S_I)$$

where

$$S_I = I + \sum_{i=0}^I \delta(i)$$

and $\delta(i)$ is the number of consecutive zeros of the PN sequence clock in the interval i . (Thus, for the interval between two consecutive ones in the sequence, $\delta(i) = 0$).

For the logic-OR product to give a correct result, the phase shifts introduced by the PN sequence clock should be truly random. Since we know that this is not possible with any finite length sequence, we have to insist, then that the phase shifts introduced in the two or more signals being input to the OR gate go through all possible and equally likely permutations over one period of the sequence. Right away, we can see that this will not hold strictly, even if we limit our analysis only to two adjacent bits:

$$S_{I+1} - S_I \leq 1 + \max_i \{\delta(i)\}$$

Let us assume that the PN sequence clock is generated from an irreducible polynomial of degree P . Then, $\max\{\delta(i)\} = P - 1$, which it occurs only once in every period, and thus only one permutation of it is possible. In practice; however, this does not seem to matter much because of the exponentially decreasing likelihood of larger phase shifts, as was briefly explained in the above section. Thus, we are able to approximate, using the logic-OR function, $(w_{ji}o_j)$ pairs for $(w_{ji} > 1)$.

Eleven Inputs at Ten Different Weights

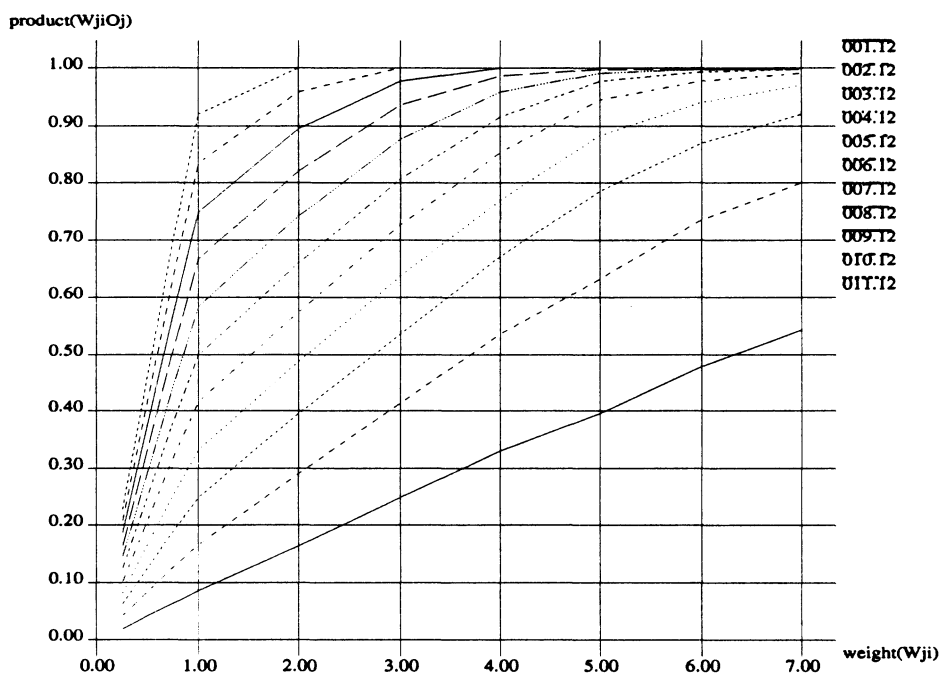


Figure 2. Eleven signal values at ten different weights

To illustrate further using non-adjacent bits in the stack, we need to determine the $E\{S_I - S_J\}$, given $I > J$:

$$S_I - S_J = (I - J) + \sum_{k=J}^I \delta(k)$$

Consider the average value of δ . The probability of δ_i is $p(\delta_i)$, which

corresponds to the probability that $\delta = i$.

$$E\{\delta\} = \sum_i \delta_i p(\delta_i) < \sum_{k=0}^{\infty} 2^{-k} k = 2$$

Thus,

$$E\{S_I - S_J\} < 3(I - J)$$

The above approximation is conservative because we ignored the $\delta = 0$ case, which would further reduce $E\{\delta\}$, and consequently $E\{S_I - S_J\}$.

Sampling PWM Signal (1/12) at 10 Weights for 3000 Cycles

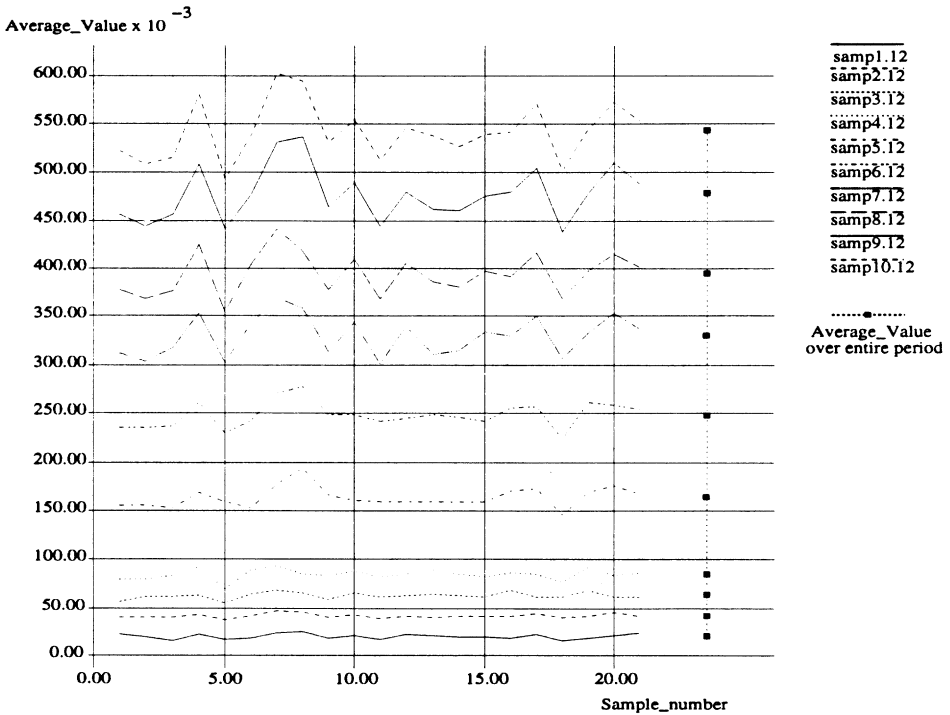


Figure 3. Sampling the weighted signal for 3000 cycles

The $E\{S_I - S_J\}$ being relatively small, implies that logic-OR product could be a reasonable approximation of the actual product. Our simulation results do confirm this. Figure 2 shows $(w_{j_i} o_j)$ product pairs for eleven different inputs and ten different weights. Among those, products corresponding to multiplications with $(w_{j_i} > 1)$ are the result of sampling logic-OR'ed

bits. Inputs range from $\frac{1}{12}$ to $\frac{11}{12}$. Weights range from (0.25) to (7). Note the saturating nature of the product, which resembles a sigmoidal transfer function.

Accuracy and Sampling Intervals. Raw and weighted signal values are quite *accurate* outside regions of saturation when sampled over the whole range of clock cycles. Figure 3 shows the weighted value of a PWM signal with duty cycle of $\frac{1}{12}$ when sampled over 3000 cycles, (approximately $\frac{1}{22}$ of the period with an irreducible polynomial of degree 16) instead of the whole range. The variance of the error is related to the number of sampling cycles. This property suggests an inherent potential for this architecture to be exploited for stochastic learning algorithms.

Sums of equal valued signals

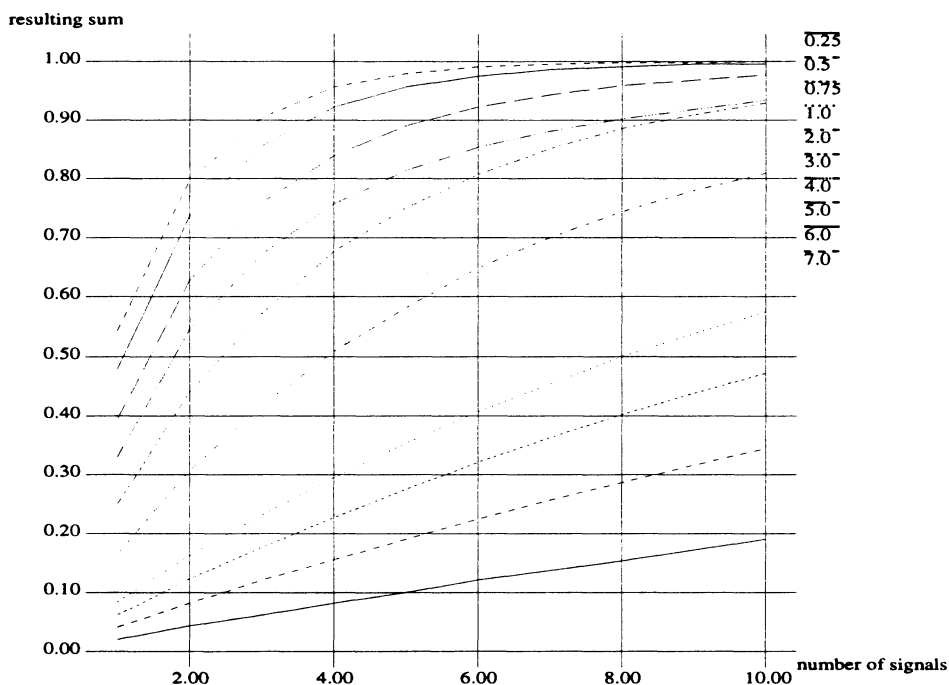


Figure 4. Adding equal valued signals together

Summation by OR'ing. Just as the product pairs corresponding to ($w_{ji} > 1$) are computed using a logic OR gate, the synaptic sums $\sum_j (w_{ji} o_j)$ are computed by wire-OR'ing the ($w_{ji} o_j$) pairs. Figure 4 shows the sum of equal valued signals. The x -axis is the number of signals that are being added on a wire. The signals have duty cycles ranging from $\frac{0.25}{12}$ to $\frac{7}{12}$.

AN APPLICATION TO PATTERN CLASSIFICATION: THE IRIS DATABASE

The computation needed to separate the three types of flowers in the iris database is quite simple. This is a linearly separable case of pattern classification, so no hidden layer is needed. Only two of the four parameters associated with the flowers are relevant to the classification; therefore, two input neurons are sufficient. The weight vector points in a (+,+) direction, which makes it possible to represent it in this implementation, without having to process separate (+) and (-) representations of data.

Simulation of a Classification Problem : The Iris Database

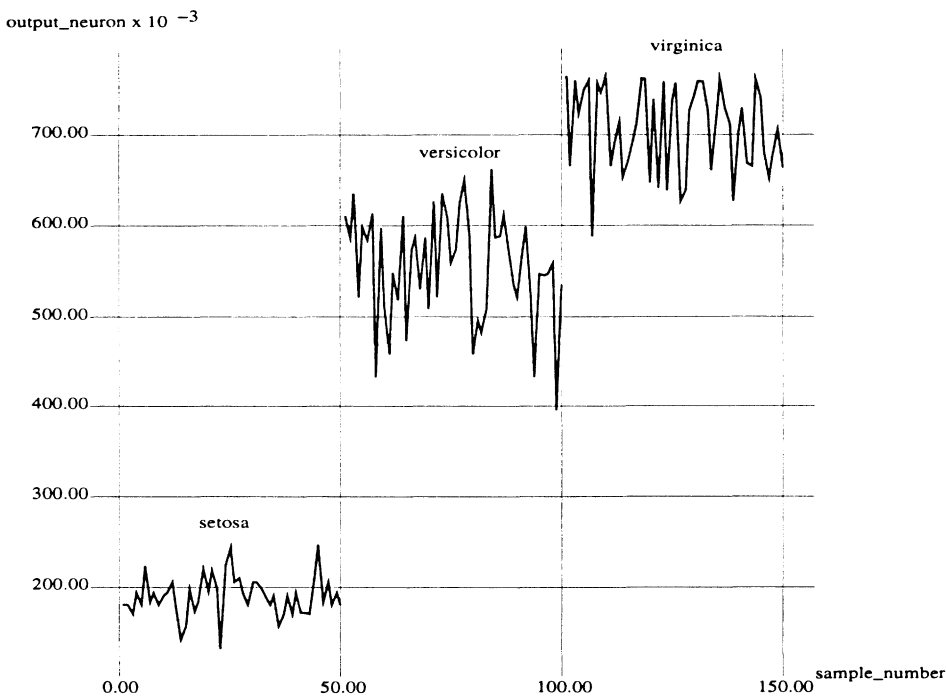


Figure 5. Classification using a ternary unit (with two thresholds).

Solution

For this very simple, single layer architecture simulation we have used the following procedure: First, we selected the two weights for connecting each input neuron to the ternary output neuron by inspecting the distribution

of data in the two dimensional input space. Second, we scaled the weights and the input parameters so they can be represented in this architecture. Here, inevitably quantization errors were introduced. Then, we fed the input-weight pairs to an output neuron with two thresholds to create a ternary function for choosing between the three possible flowers.

Results

Classification output from simulation is shown in Figure 5. There are fifty examples of each flower in the database. The two edges of the clusters is between points 50-51 and 100-101. As expected with this database, iris setosa and iris versicolor are easy to separate. Iris versicolor and iris viginica; on the other hand, are not so straightforward. The Bayesian limit on classification is around 97%. If we were to set the second threshold of the ternary output neuron at 0.637, this architecture would misclassify only five flowers, which puts its accuracy at around 95%. This is a very encouraging result, especially considering the all the sources of noise in this system.

One multiplicand fixed (linear characteristic as expected)

Product at freq ratio 7.5 (V)

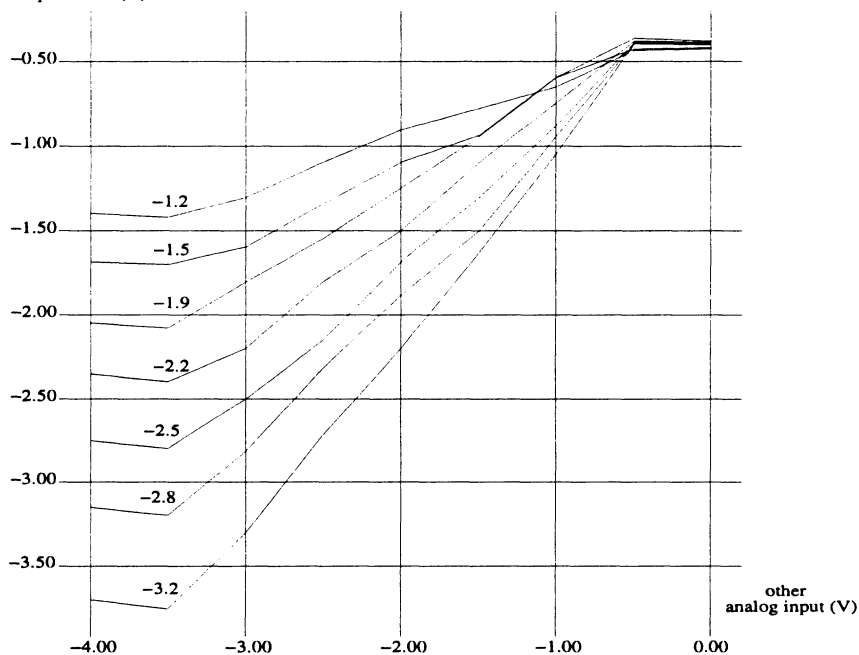


Figure 6. Classification using a ternary unit (with two thresholds).

HARDWARE IMPLEMENTATION

We will report results from the hardware implementation of a multiplication scheme similar to what has been described above for $(w_{ji}; o_j)$ pairs for $(w_{ji} < 1)$ case. Specifically, we have fabricated and tested the random pulse generation and the synaptic multiplication circuits to investigate how accurate and robust multiplication is using our scheme.

One multiplicand fixed (linear characteristic as expected)

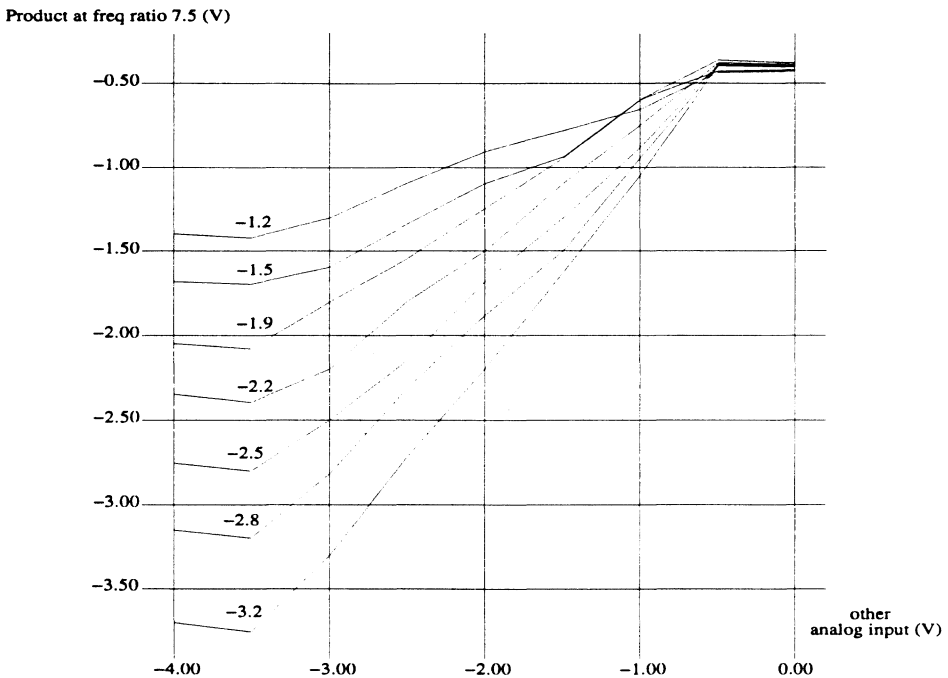


Figure 7. Time averaged product with one multiplicand fixed (as marked).

In the hardware implementation we generate four versions of each analog input, which are exactly 90° out of phase with one another. A random pulse train with the desired average value is obtained by selecting randomly between these four phase shifted versions. The two select inputs to the multiplexer that chooses between the four phase shifted versions are two uncorrelated bit streams, which are PN sequences out of phase. This is similar to introducing random phase shifts to a deterministic pulse, as the PN sequence clock does in

the digital architecture. The product pair $(w_{ji}; o_j)$ is obtained by AND'ing the the selected representation for w_{ji} with o_j .

The specific circuit implemented consists of a digital delay line to simulate the phase shifted weight and input activity level representations. The input to the delay line is provided from a pulse width modulator that is driven by an analog input value. This implementation mixing analog and digital techniques was chosen to test this multiplexing scheme over a continuous range of values.

Both multiplicands equal (parabolic characteristic as expected)

Product at various freq ratios

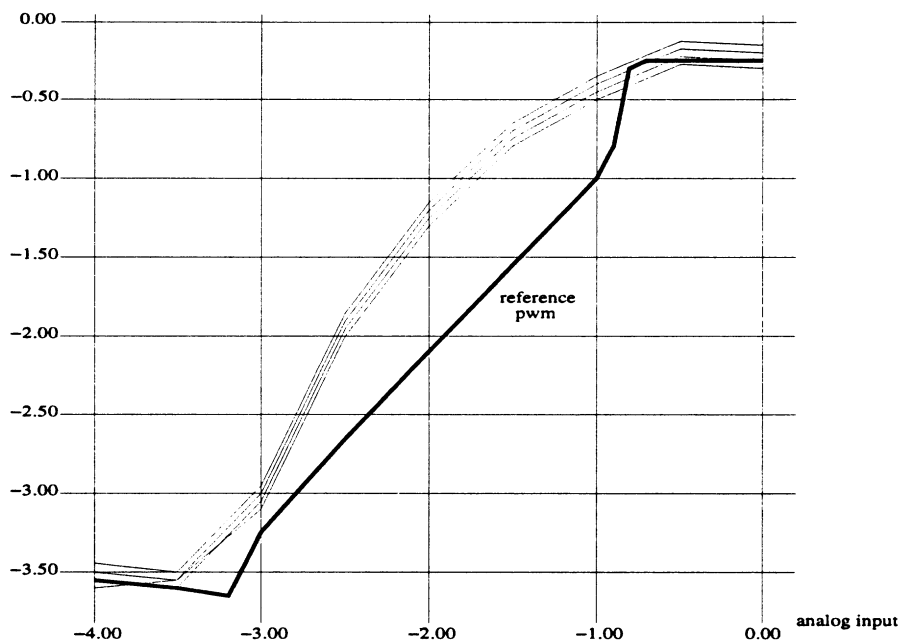


Figure 8. Time averaged product of pulses with both multiplicands equal.

The pulse width modulator circuit is an extension of the current controlled neuron circuit which uses a positive feedback mechanism to generate pulses [Mead 89]. This circuit compares the input analog value to a triangular wave, the midpoint of which corresponds to the midpoint of the range of weights ($w_{ji} < 1$). Thus, half of the maximum allowable analog value generates a pulse signal with a 50% duty cycle. This implementation proved to generate pulse width modulated signals quite accurately (Figure 7, reference signal

PWM marked with a thick line).

The random bit streams are generated by a shift register sequence, based on irreducible polynomials, using specifically the polynomial $(x^{16} + x^{12} + x^3 + x + 1)$. We have already commented on the relevant properties of such bit sequences. The product pair gets time averaged using a simple RC circuit.

This scheme has been tested both with one multiplicand fixed and both multiplicands equal (Figure 6 and Figure 7, respectively). In both of these the chip has performed very robustly over frequency ratios ranging from 0.1 to 10. This has been demonstrated by time averaging the pulse trains through an RC filter with a fixed time constant (0.1 sec.). The frequency of activity levels are adjusted by changing the ratio of frequencies between the triangular wave generating the pulse-width-modulated representation and the clock generating the uncorrelated bit sequences which multiplex between them.

CONCLUSIONS

We have described a digital neural network architecture and reported results from the VLSI implementation of the multiplication scheme embedded in it. As can be inferred from the test plots, the circuit is robust and quite accurate over a wide range of clock frequencies. We believe this robustness to be congenial to the exploration of novel training algorithms in such a system. One could, by reducing the time constant of the time averaging circuit and thus averaging the pulse train over a smaller number of clock cycles, get significant variations in output activity for the same input. The use of the statistical properties of these variations among computing units could have novel implications in weight update decisions during training.

ACKNOWLEDGEMENTS

This work was supported in part by an ONR contract, number N00014-92-J-1860. G. E. Salam is supported in part by an NSF Graduate Fellowship.

References

- [Tomlinson 91] Tomlinson, M. S., Sivilotti, M. A., A High Performance, Scalable Digital Neural Network Architecture for VLSI, *Advanced Research in VLSI 1991*, UC Santa Cruz pp:262-273.

- [Tomlinson 90] Tomlinson, M. S., Walker, D. J., Sivilotti, M. A., A Digital Neural Network Architecture for VLSI, *IJCNN Proceedings 1990*, Vol II pp 545-550.
- [Murray 91] Murray, A. F., Del Corso, D., Tarassenko, L., Pulse Stream VLSI Neural Networks Mixing Analog and Digital Techniques, *IEEE Transactions on Neural Networks*, Vol 2, No.2, March 1991, pp 193-204.
- [Alspector 91] Alspector, J., Gannett, J. W., Haber, S., Parker, M. B., Chu, R., A VLSI-Efficient Technique for Generating Multiple Uncorrelated Noise Sources and Its Application to Stochastic Neural Networks. *IEEE Transactions on Circuits and Systems*, Vol 38, No.1, January 1991.
- [Golomb 67] Golomb, S. W., Shift Register Sequences, Holden-Day Inc., 1967.
- [Gaines 69] Gaines, B. R., Stochastic Computing Systems, *Advances in Information Systems Science*, J.T. Tou, editor. Plenum Press, 1969.
- [Mead 89] Mead, C., Analog VLSI and Neural Systems, Addison Wesley, 1989.

AN UNSUPERVISED NEURAL PROCESSOR

James Donald and Lex A. Akers

*Center for Solid State Electronics Research
Arizona State University, Tempe AZ, 85287*

ABSTRACT

Real-time control requires fast and adaptive processing. VLSI offers the speed and density required to implement such a processor. A controller must continuously adapt to the data sensed from a dynamic environment. The first stage of such a controller is a system implementing an unsupervised learning rule that generates linearly separable representations that capture important statistical information from the inputs.

We describe the design and test results of two adaptive VLSI processing chips [1,2]. These chips use pulse coded signals for communication between processing nodes and analog weights for information storage. The unsupervised weight modification rule, implemented on chip, uses concepts developed by Oja [3], and later extended by Leen [4] and Sanger [5]. Experimental results demonstrate that the network produces linearly separable outputs that correspond to dominant features of the inputs. Such representations allow for efficient additional neural processing. The adaptation rule also includes a variable lateral inhibition mechanism. Experimental results from the first chip show the operation of function blocks that make a single processing node. These function blocks include forward transfer function, weight modification, and inhibition. Experimental results from the second chip

show the ability of an array of processing elements to extract important features from sensory input data.

INTRODUCTION

Many applications exist for real-time control systems that can adapt to changing control environments. Neural networks implemented in VLSI can provide solutions for some of these applications. However, many issues relating to neural networks for adaptive control remain open. Several researchers have examined the performance of neural network control systems using various types of input-output mapping units [6,7]. These systems use multi-layers of neurons with various learning rules that modify the weights to reduce error in the output. These systems require several layers of processing nodes to produce a nonlinear mapping [8,9]. Representations generated by neurons in the first layers determine the robustness and generalization capabilities of the network [10]. While our application interest is real-time adaptive control, this chapter concentrates on the VLSI implementation of an adaptable processing node. The processing node is unsupervised and acts as a pre-processing stage to reduce the dimension of its inputs while generating efficient representations for additional layers of processing elements [1,2].

Figure 1 shows a block diagram of our complete analog input-output mapping module. First, a set of tapped delay lines and level sensitive neurons expand the dimensionality of the analog signal. Tapped delay lines time-embed input signals converting time information to a spatial representation required by the feed-forward transfer function of our processing nodes. Level sensitive neurons convert amplitude information to a spatial representation. This is required if the individual processing nodes in the dimensional reduction layer have insufficient accuracy to code the analog signal. The next layer combines multiple inputs to reduce the dimensionality of the first layer. It must also provide an efficient representation for the output layer. The output layer maps the internal representation to the correct output value. A layer of neurons using a supervised training algorithm performs the final output-mapping task [8]. For the output layer to operate correctly, the representation provided by the dimensional reduction layer must be linearly separable [9].

An adaptive system must continuously adjust to the nonstationary statistics provided by the controlled environment. To maintain an optimal representation for the output layer, the dimensional reduction unit must continuously adapt through weight modification. Several researchers have shown that Hebbian weight modification rules can form efficient representations of the input statistics when used in conjunction with a method to force each processing node to extract a different feature from the inputs

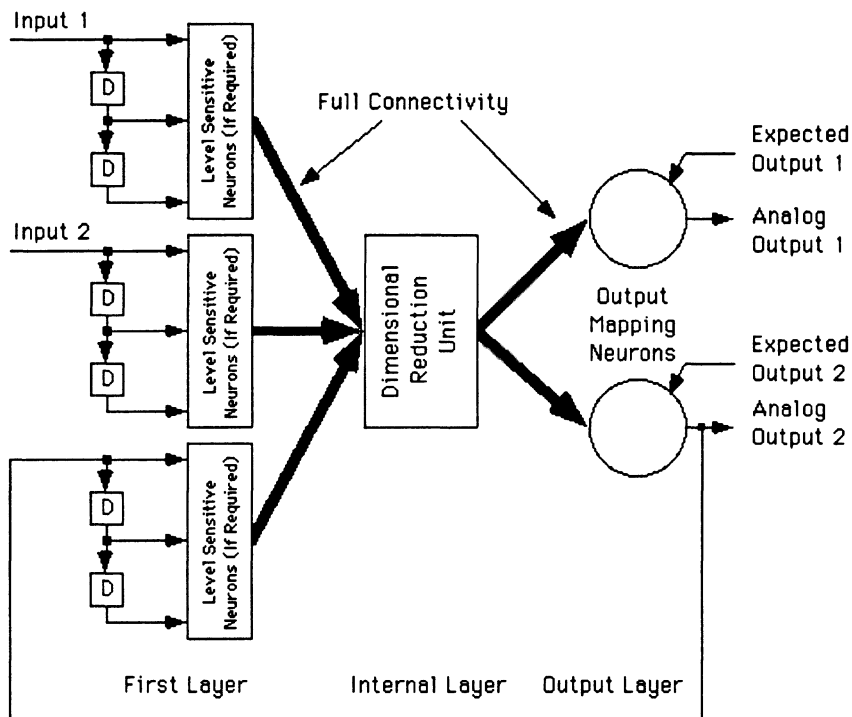


Figure 1. A complete nonlinear input-output mapping system

[1-5,10,11]. For this continuous time VLSI implementation, we desire the magnitude of the weights be controlled. The weight modification rule proposed by Oja uses negative feedback from the outputs to control the magnitude of the weights [3]. Our design utilizes the concept of negative output feedback to control the weight magnitude. This rule combined with a variable lateral inhibition mechanism forces each processing node to extract a different feature from the inputs while learning.

To test the feasibility of a VLSI internal layer based on our weight modification rule, we designed, fabricated, and tested two integrated circuit chips. The first chip contained one processing node connected so the state of individual function blocks could be monitored and controlled. The second chip contained an array of four processing nodes. In the array, each processing node shared nine inputs connected to adaptable synapses and two inputs connected to nonadapting synapses. Five of these inputs had defective write circuits caused by a layout error. Both synapse types can be addressed and programmed from an external analog source. Additionally, weights stored as

analog values on a capacitor require a refresh mechanism which has not been implemented here to negate the effects of charge leakage [13]. The nonadapting synapses provide two quadrant multiplication, while the adaptable synapses provide only first quadrant multiplication. Adaptable synapses also contains additional circuitry for weight modification. The adaptable synapse measures $105\mu\text{m} \times 168\mu\text{m}$, including interconnects, using a $2\mu\text{m}$ CMOS MOSIS process with two layers of polysilicon and two layers of metal. The tapped delay lines were not implemented on this chip. However, including the tapped delay lines on chip would significantly reduce the pin count for large arrays. The output layer was also not implemented on this chip. It is planned to have the output layer implemented on a separate chip. Since digital pulses communicate analog values between chips, EMI type noise will have little effect on the accuracy. Our longer range goal is to build large scale sensory processing chips and real-time control chips, hence large chips capable of multi-sensor integration will be necessary.

In this chapter, we present the design and test results in four sections. The first three sections describe the design and test of the function blocks. The final section shows results from a complete classifier array chip. The three function blocks described include the forward transfer function, weight modification, and inhibition. Tests results from each of these blocks were obtained from the single processing node chip. Tests of the array shows the system extracting features from two inputs. These tests allow us to quantify limits to the array's capabilities in distinguishing between two similar features.

FORWARD TRANSFER FUNCTION

The feed-forward function in most synthetic neural systems consists of a sum of products between an input vector and weight vector [3-5,6,7,11,12]. This provides a measure of the match between the weight vectors and input [6,7]. Additionally, some type of output nonlinearity, typically a sigmoidal function, splits the output probability density into two groups indicating a decision. In our system, the dimensional reduction layer performs a linear sum-of-products computation. Nonlinearities producing a decision are provided by the next processing layer. Figure 2 shows a block diagram of the feed-forward function for our complete nonlinear mapping system. The figure shows both the dimensional reduction layer, implemented on the chip described here, and the output mapping layer. Pulses generated by the dimensional reduction layer enable a voltage source for each input from that layer to charge a capacitor that averages and filters outputs from all enabled voltage sources. During supervised training, the output mapping layer adjusts these voltage sources to produce the optimal Z out with minimum error

given a specific training input. When an input is applied to the dimensional reduction layer, the voltage on the averaging capacitor approaches the average of the voltage sources corresponding to the firing neurons. The absolute frequency of the firing neurons changes only the rate that the voltage on the averaging capacitor approaches the correct value. This allows us to remove the dependency on the absolute frequency of the outputs from the dimensional reduction layer. We will restrict our discussion to the operation of the linear sum-of-products function.

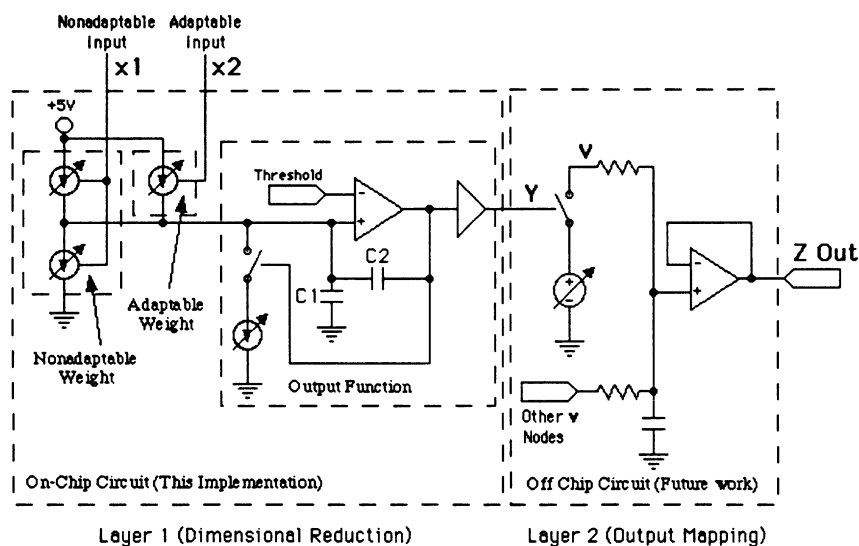


Figure 2. Adaptable nonlinear function generator

In our implementation, the problem of inter-processor communication is addressed using digital pulses. Pulse coded information allows both robust information transfer and space efficient computation [14-15]. Regeneration and crosstalk problems associated with analog communication are greatly reduced by using digital signals [16]. Several other researchers have also utilized pulses to communicate analog values in a neural network [17-21]. In this implementation, inputs and outputs of a processing node are represented by a sequence of pulses. Analog inputs have values proportional to the duration of the pulse divided by the period between pulses. This type of signal allows most computations to be performed using MOS switches.

Figure 3 shows the circuit used for the sum of products function which includes one synapse, and the output function. Summation of the product of the weight and input occurs by injecting a current, proportional to the synaptic

weight, into a summing node for the duration of each input pulse. The summing node receives the current produced by a complete row of synapses producing summation through Kirchhoff's current law. An integrating capacitor (C1) converts the current pulses to a voltage change at the summing node. The neuron output function compares the voltage on the summing node to a threshold voltage, V_{thresh} , and produces an active high digital output if the threshold voltage is exceeded. The output increases the summing node voltage by positive feedback through C2. If the digital signal switches from 0 to V_{dd} , the voltage shift produced is:

$$V_{\Delta\text{th}} = V_{\text{dd}} \frac{C2}{C1 + C2} \quad (1)$$

In our implementation, each output function block requires one C1-C2 set. When implemented in a two layer polysilicon process, C1 and C2 have areas of $9792\mu\text{m}^2$ and $2496\mu\text{m}^2$ respectively. C1 is approximately 5pF, and when combined with C2 provides a hysteresis of 1V in the switching voltage.

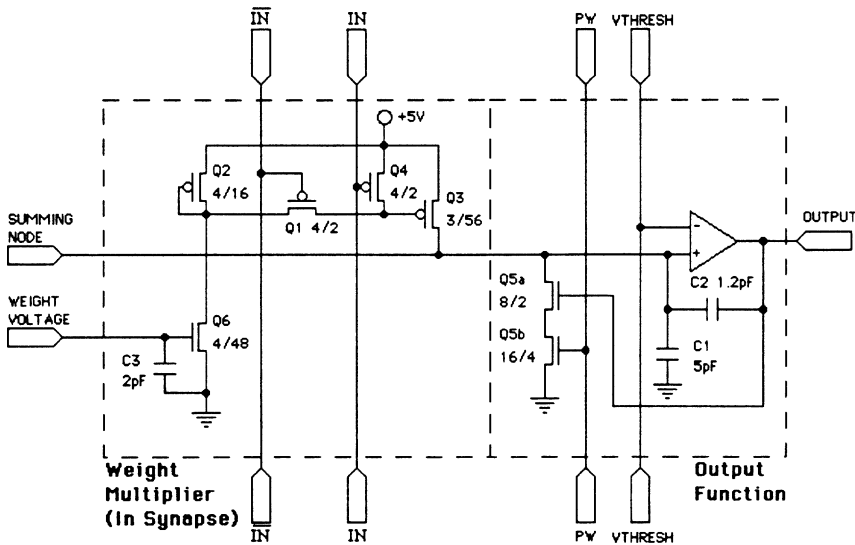


Figure 3. Single synapse and summation circuitry

Input to the synapse circuit, Figure 3, consists of an active high signal $\overline{\text{IN}}$, and an active low signal IN generated by the column driver. An input pulse enables the transmission gate formed by Q1 to transmit the voltage present on the drain connected transistor Q2, to the gate of transistor Q3. When the input is inactive (IN low), Q4 shorts the gate of Q3 to V_{dd} to reduce leakage into the

summing node through Q3. This structure forms a simple current mirror that can be switched on for the duration of the input pulse. The current delivered by the current mirror depends on the voltage on C3, the weight capacitor. This dependency, shown in the Spice simulation in Figure 4, illustrates the current injected into the summing node while the input is active.

When active the digital output from the output function block enables a current sink to remove charge from C1. The digital output becomes inactive when the voltage on C1

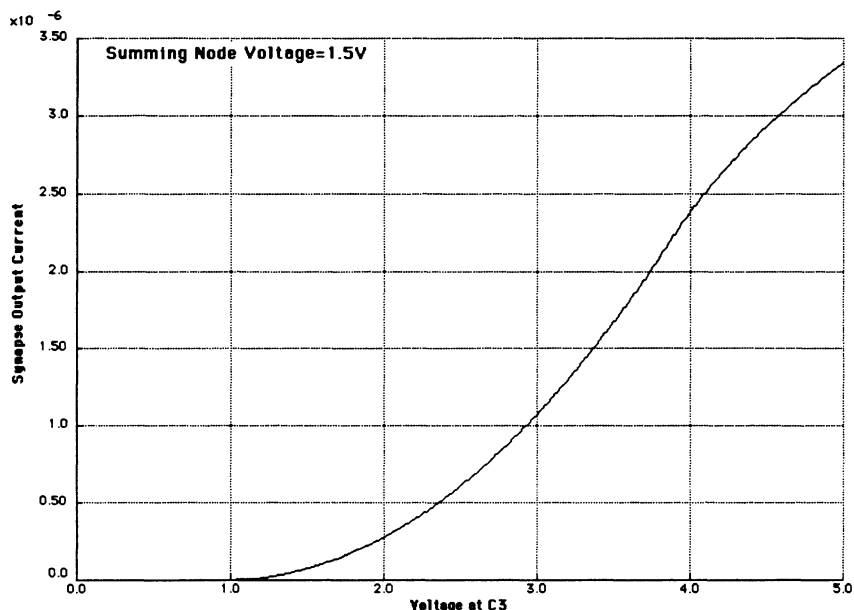


Figure 4. Theoretical current into summing node during active input

falls below the threshold voltage. Transistor Q5a enables the current sink formed by Q5b. For tests presented in this section, the current through Q5b was adjusted so pulse widths of 250ns were obtained. The theoretical value of the current through Q5b is 24.8 μ A for this pulse width. This action produces a series of pulses with the average period between the pulses equal to the sum of products of the weights and inputs. Mead [22] has implemented this type of circuit using CMOS inverters. In our system, we use a differential comparator instead of CMOS inverters to generate the threshold function. The comparator allows us to externally control the threshold voltage which allows more control of the operation of the circuit.

A simpler weight multiplication structure consisting of a MOS switch and MOS transistor for a current source has been proposed by several researchers

[14, 18]. Compared to the two transistor structure, our weight multiplication circuit reduces perturbation of the summing node voltage when the weight is near zero and an input pulse occurs. The only coupling between the input and summing node occurs through the gate-drain capacitance of Q3. When the weight is near zero, the change in voltage on the gate of Q3 approaches zero. In the two transistor structure, the source/drain-substrate capacitance between the two transistors must discharge from V_{dd} to the summing node voltage and the gate-drain capacitance of the switch transistor must charge to the gate voltage during the pulse. This can result in significant perturbation of the summing node voltage even if the weight is near zero.

The transistors sizes in the synapse current mirror, the size of C1, and the hysteresis of the output determine the maximum output frequency of the processing node. A nominal pulse width of 250nS and maximum frequency of 1 MHz was chosen for this design. CMOS VLSI circuits can easily drive these signals while maintaining pulse widths accurate for our application.

To determine the size of the transistors for the synapse, we need to calculate the contribution of each input to the output. If the weights are normalized such that:

$$\sum_{i=0}^{N-1} w_i^2 = 1$$

then let $w_i \in \{0, \alpha\}$, $x_i \in \{0, 1\}$,

where N is the total number of inputs,

w_i is the weights, x_i is the inputs,

(2)

and $w_i = \alpha$ if $x_i = 1$, and define $n = \sum_{i=0}^{N-1} x_i$ then:

$$n\alpha^2 = 1$$

$$\alpha = \sqrt{1/n}$$

where α is the size of the weight and the number of active inputs is defined as n. From this we can determine the dependence of the output on active inputs:

$$y = k \sum_{i=0}^{N-1} w_i x_i = k \sum_{i=0}^{N-1} \alpha x_i = k n \alpha = k \sqrt{n} \quad (3)$$

If $y = 1$ when the maximum number of possible inputs, L , are active:

$$\begin{aligned} k &= 1/\sqrt{L} \text{ and} \\ y &= \sqrt{\frac{n}{L}} \end{aligned} \quad (4)$$

In this design we set the time between pulses, $1/y$, to be $1\mu\text{S}$ for 9 simultaneously active inputs, i.e., $n=9$, $x_i=1$, and $L=9$. This requires that the time between output pulses for one active input, $n=1$, after weight modification, to be $3\mu\text{S}$. Figure 3 shows the transistor sizes used for a 5pF summing node capacitor and 1V hysteresis design.

For experiments testing the operation of the function blocks, a test chip with one adaptable input, one nonadaptable input, and one output was used. Internal analog voltages were monitored with source followers which exhibited maximum slew rates of about $5 \cdot 10^6 \text{ V/S}$ and were gain and offset compensated by measuring an accessible source follower on the chip. Figure 5 shows the measured-compensated summing node voltage during pulsing. Nine input pulses charge the summing node until the threshold voltage of 2.5V was reached, and the output fired raising the summing node voltage, producing a 1V hysteresis. Figure 6 demonstrates the weighting function by showing the measured output frequencies for a range of input levels and weights. The input frequency times the pulse width gives the input level (active fraction), except for the input level of 1, which was held constantly active. Figure 6 shows the maximum errors in the output linearity occur when the weight voltage was large and the input duty cycle was greater than 40%. The maximum error of approximately 5% occurred at an output frequency of 150KHz with the input at 60%. Using the previous analysis, eq. 4, the difference in output between two patterns that differ by a Hamming distance of H is:

$$\Delta y = \frac{\sqrt{L} - \sqrt{L-H}}{\sqrt{L}} = 1 - \sqrt{1 - \frac{H}{L}} \quad (5)$$

which shows the output accuracy required to distinguish between two binary patterns with L active inputs and a Hamming distance of H . The measured worst case accuracy of the output function is approximately 5%. This means a Hamming distance of 1 can be detected if $L=9$. Note that this accuracy can be significantly improved if an input duty cycle of less than 40% is used. Also, the minimum period between pulses for a continuously active input was $4\mu\text{S}$, not the $3\mu\text{S}$ design target. Note that mismatch between synapses may produce an additional source of error not considered here.

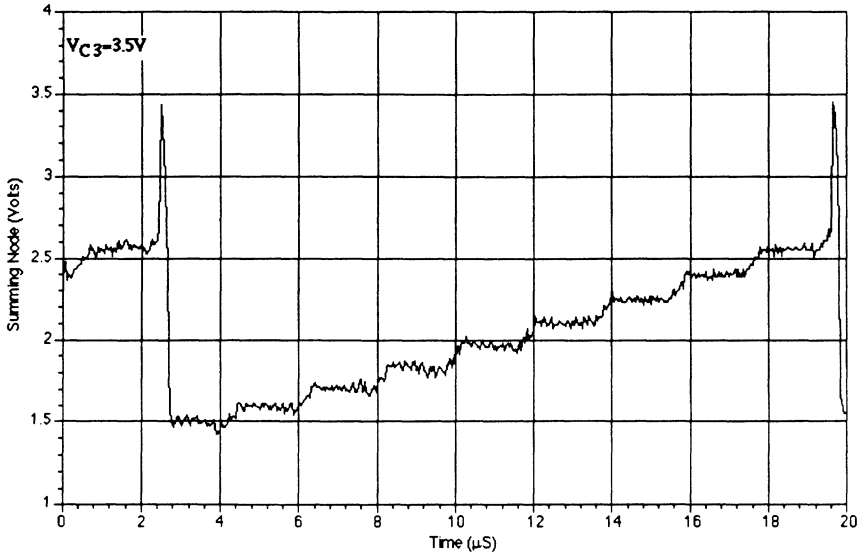


Figure 5. Measured summing node voltage when one input is pulsing at 1 MHz with a pulse width of 250ns

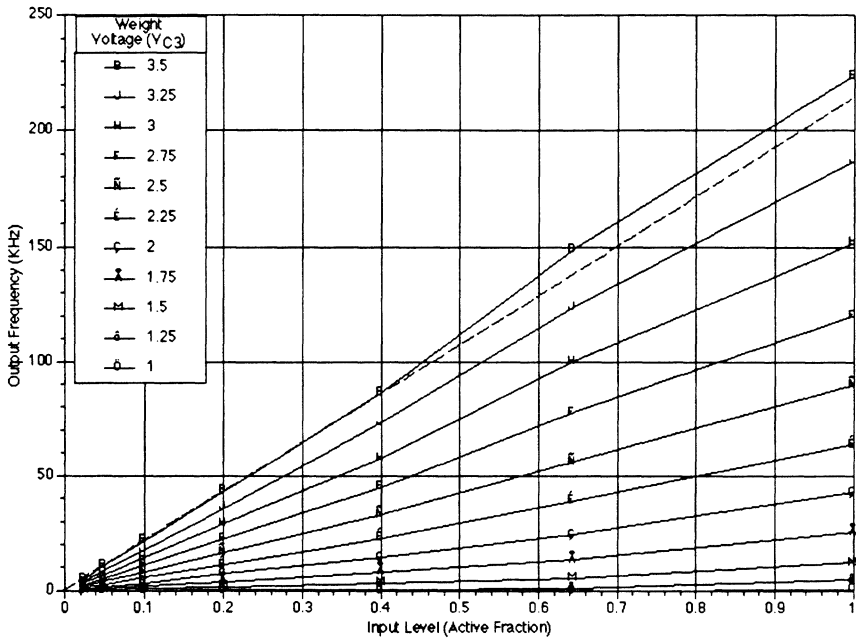


Figure 6. Measured output frequency for a range of weights and inputs

Since the outputs are based on relative periods between pulses, the lower maximum frequency does not pose a problem. These variations can be accounted for by decreased PMOS transistor transconductance and larger C1 capacitance. Our results show that a sum of products function can be constructed with characteristics of high dynamic range, and input linearity sufficient for binary feature detection. Nonlinearity in sum-of-products output function arises when the input current exceeds the current capability of the sinking transistor Q5b (Figure 3). When this condition occurs the pulse width will become wider until the output turns on constantly leading to a sigmoidal output characteristic. The chip will normally operate in the linear mode since the output needs to provide a measure of the weights for the normalization process.

WEIGHT MODIFICATION

We implement a modified unsupervised learning algorithm using concepts developed by Oja [3] to determine the change in the weight to best represent the input statistics. For a single synapse element, Oja's weight modification rule is:

$$w_i(t + \tau) = w_i(t) + \lambda y(t)(x_i(t) - y(t)w_i(t)) \quad (6)$$

where, $x_i(t)$ is the input, $w_i(t)$ is the weight, $y(t)$ is the output from the linear sum of products function, and λ is the learning rate or weight increment size.

One of the main advantages of Oja's weight modification rule is the utilization of feedback from the neuron output to normalize the weights. This has several significant advantages in a VLSI hardware implementation. First, if the system is stable, the weight equilibrium is independent of the weight increment size (λ). Second, negative feedback reduces the effects of nonlinearities in the weight to output transfer-function resulting from the implementation. In this implementation we take advantage of the insensitivity to the forward weight transfer function to simplify circuit design and allow continuous adaptation.

Figure 7 shows the synapse weight modification circuit used in this design. Basically, the weight modification circuit operates as follows. The weight update is calculated in two stages. The argument $(x - yw)$ is calculated by using pulses to enable current sources to charge and discharge a capacitor. The charging is proportional to x , and the discharging is proportional to yw . Then, a switch capacitor system produces the multiplication of the output, y , with the argument $(x - yw)$ [23].

The modification circuit generates a weight update using three phases as

shown in Figure 8. The first phase resets the capacitor C5 to a predetermined voltage (LRNLEV). During the second phase, called the statistical sampling phase, input pulses charge capacitor C3 at a rate that depends on $V_{PreGain}$ (see Figure 7) using the current source formed by Q7. Simultaneously, output pulses, when active, enable a current sink that discharges C5 at a rate that depends on the weight capacitor voltage, VC3, and the output. If Q7 and Q10 are operating in saturation with gain parameters β_7 and β_{10} , the voltage on C5 at time t after the reset phase is:

$$V_{C5}(t) = \frac{1}{C5} \left(\beta_7 x (V_{cc} - V_{PreGain} - |V_{tp}|)^2 - \beta_{10} y (V_{C3} - |V_{in}| - V_{LrnLev})^2 \right) t \quad (7)$$

where x is the active fraction of \overline{PRE} and y is the active fraction of POST during the statistical sampling phase (SAMPEN active). The signal SAMPEN is active during the statistical sampling phase and is used to gate \overline{POST} . The voltage pulses on the POST line are proportional to the output y during the statistical sampling phase. By controlling the period of the statistical sampling phase, (t) , we can control the final voltage on C5.

The current mirror for the weight multiplication function supplies current into the summing node that depends on the voltage on C3 and is proportional to the current through the drain of Q6. The gate of Q10 also connects to the weight storage capacitor, C3, which generates a drain current proportional to

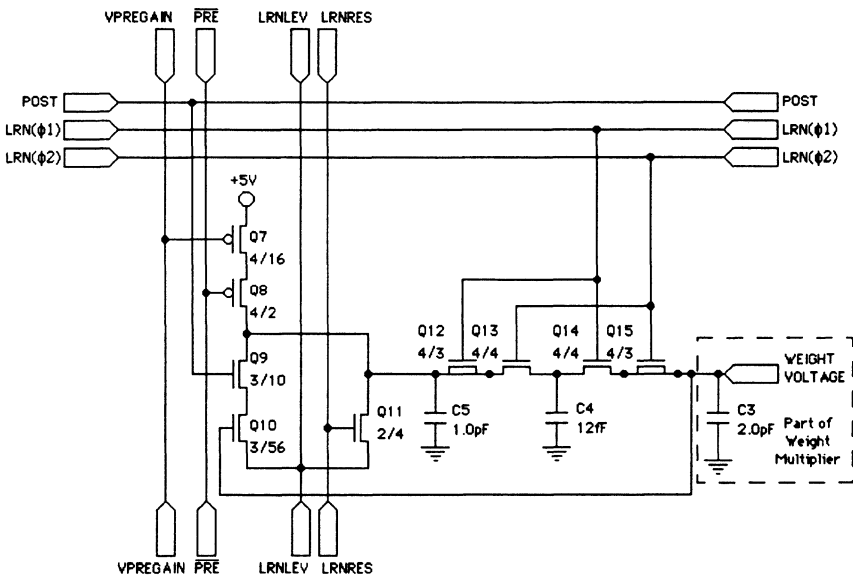


Figure 7. Weight modification circuit

the drain current of Q6. The largest error in this current source occurs when the voltage on C3 is maximum. This occurs when learning features which contain only one active input. Therefore, the drain current of Q10 is proportional to the weight with only slight error. From this simplification, and assuming that the initial voltage at $t=0$ was zero, the voltage on C5 at the end of the statistical sampling period is:

$$V_{C5} = v(d_{PreGain} x - y w) \quad (8)$$

where v is a proportionality constant with units of voltage, and $d_{PreGain}$ is an external gain constant that depends on $V_{PreGain}$.

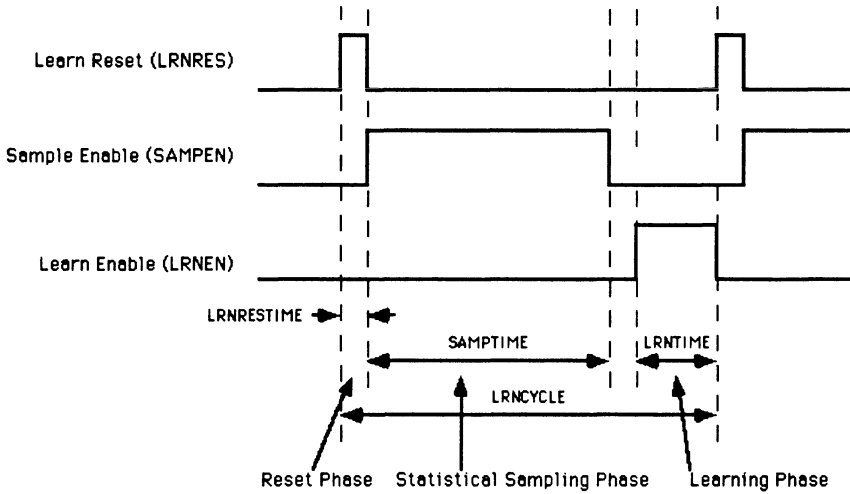


Figure 8. Learning control signals and timing

We use a switched capacitor approach to calculate the final multiplication. This allows us to use pulse computation to simplify the design. During the learning phase, each output pulse transfers charge from the statistical sampling capacitor C5 to the weight capacitor C3. The size of charge transfer must be as small as possible so the time constant of the weight filter is as long as possible (λ small). Since y is proportional to output pulse frequency, when the output operates in the linear region, the output can be used to set the frequency of the switched capacitor system. During the learning phase an output pulse generates a set of non-overlapping signals $LRN(\phi1)$ and $LRN(\phi2)$. These signals enable switches formed by Q13 and Q14 to first move charge from C5 to C4 and finally C3. This incrementally modifies the voltage on C3 for each output pulse. The ratio between C3 and C4, and the difference between the weight voltage V_{C3} and the voltage on C5 sets the size

of the weight increment.

For the longest weight filter time constant, C3 must be large and C4 must be small. To make C4 as small as possible, the diffusion region between Q13 and Q14 formed C4. The size of this capacitor is approximately 12fF and depends on the square root of the stored voltage. C3 is constructed using two layers of polysilicon and has a size of approximately 2.0pF. This combination of C3 and C4 gives a theoretical maximum increment size of one part in 170 when the difference between the voltage at C5 and C3 is maximum. The two MOSFETs with the source and drain shorted in Figure 7 is a standard analog technique to reduce clock feedthrough. However, for our case C3 and C5 are much larger than the gate capacitance of Q13 and Q14 therefore while we used this scheme, it is not required. Note the weight increment size effects only the time constant of the weight filter since negative feedback is used to stabilize the weight. Therefore, variations in the increment size do not effect the value of the weight equilibrium. Figure 9 shows experimental measurements of the weight counter. In this test the sampling capacitor was set to 4.8V, and the weight capacitor was set to 1.3V. Learning was then enabled and each processing element output pulse incremented the weight capacitor. In this test the increment size was approximately 6 mV per output pulse. When the weight is large, one can derive the weight modification used in our circuit as:

$$w_1(t+\tau_2) = w_1(t+\tau_1) + \lambda y(t+\tau_1) \{ (d_{preGain} x_1(t) - y(t)w_1(t)) - \delta w(t+\tau_1) \} \quad (9)$$

where d is a proportionality constant relating the voltage on C3 and the weight. Note our implementation of the training rule departs slightly from Oja's [1].

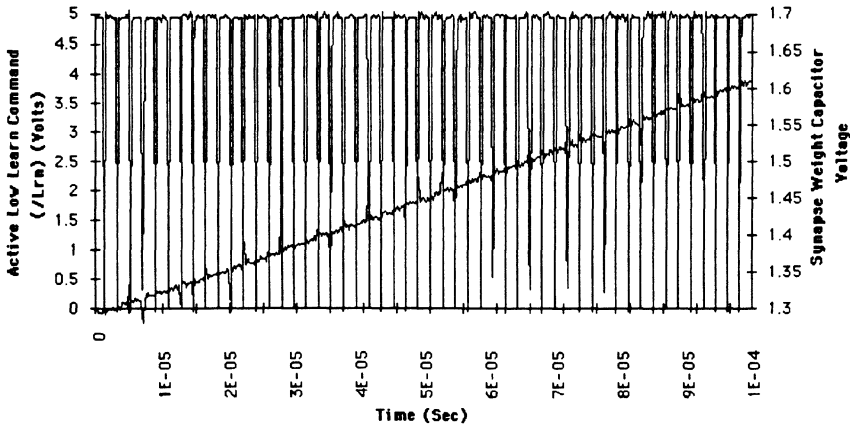


Figure 9. Measurements of weight voltage during weight increment

INHIBITION

An array of neurons using a modified Hebbian weight modification rule requires a mechanism to force each neuron to extract a different feature from the inputs [3-5,11,12]. Sanger [5] incorporates a Gram-Schmidt orthogonalization mechanism in the synapse to force the weights to be orthogonal. Leen [4] adds a matrix of weights between outputs that uses an inverse Hebbian modification rule to force the outputs to be uncorrelated. In our system, we use a variable lateral inhibition mechanism to force the outputs to be uncorrelated during learning and allow overlap between outputs during recall. Since the output blocks, which are $O(M)$ (where M is the number of outputs) in area, contain circuitry for the lateral inhibition, the area of a large array can be reduced over the previously cited networks. During learning, this inhibition mechanism acts as a winner take all circuit found in Hamming networks [9]. During recall, the inhibition mechanism allows several outputs to become active if the presented pattern lies between trained classes.

Pulse computation allows a simplified inhibition circuit. Our lateral inhibition circuit uses a global wire-or signal and a gate circuit to allow the neuron with the greatest sum of products to inhibit all other neurons in the array without inhibiting itself. This inhibition occurs only during the high output pulse of the active neuron. When the output pulse is high it resets all nonfiring neuron summing node voltages to the inhibition voltage (V_{inh}). After the active neuron's pulse occurs, synapses attached to each neuron charge the summing node capacitor at a rate that depends on the sum of products between the weights and inputs. The time for the summing node capacitor to charge to the threshold and cause the neuron to fire, depends not only on the sum of products, but also on the initial voltage of the summing node. Note that the active neuron also resets its summing node voltage below the threshold to $V_{thresh} - V_{\Delta th}$ after it fires. The relation between the initial voltage of the inhibited and the active neurons determines how much greater the sum of products must be for an inactive neuron to become active. If the inhibited summing node voltage of a neuron is less than the reset voltage of the active neuron, the inhibited neuron must have its sum of products greater than those of the active neuron for it to become active. Conversely, if the neuron's summing node voltage is inhibited to a value greater than the reset voltage, this neuron can become active if its sum of products is slightly less than the active neuron.

By varying the inhibition voltage, the sensitivity to changes in the input can be controlled. Normal operation of an array of neurons would first set the inhibition voltage higher than the reset voltage allowing neurons with close pattern matches to become active. The inhibition voltage would then be

reduced resulting in only one active neuron for the duration of the learning phase. Reducing the value of the inhibition voltage allows the network to reduce its sensitivity to changes in the input feature during synapse modification in a real-time control system. For tests presented here we fixed the inhibition voltage to a value slightly lower than the reset voltage using an external voltage source.

Figure 10 shows the circuit used for the lateral inhibition mechanism. One transistor, Q18, per chip acts as a load transistor for the inhibition line. Each neuron output block ties to the single inhibition line with a pull-up transistor Q17. This produces a wire-or computation that activates Q16 to set the summing node to the inhibition voltage, INHLEV, in all neurons except the currently firing neuron. When the neuron fires, transistor Q19 disconnects the gate of the inhibition transistor Q16 from the inhibition line and Q20 shorts the gate to ground to prevent spurious triggering of Q16. Just after the gate of Q16 is disconnected from the summing line, Q17 pulls the inhibition line active. The size of Q22 contributes to this delay. The inhibition line, INH, becoming active causes the inhibition transistor Q16 to connect the summing node to INHLEV on all neurons except the neuron(s) that have an active output pulse while the inhibition line is active.

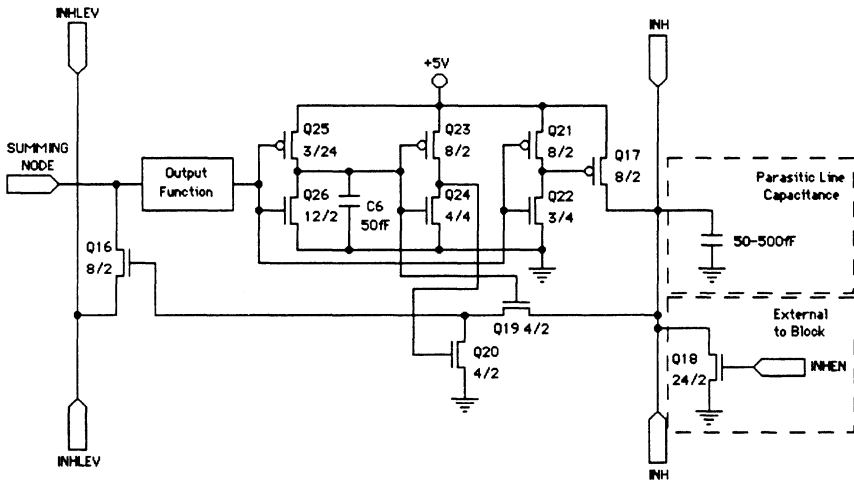


Figure 10. Inhibition circuit

The single node test chip allowed us to examine the summing node voltage and inhibit the neuron using an external signal. Figure 11 shows experimental results of the summing node being set to the inhibition voltage after firing twice. From the figure, the delay time between the inhibit signal and the summing node reaching the reset voltage is about 250nS. This time is

longer than the actual delay for the case when no off-chip signals were driven. The additional delay is due to the slew rate of the source follower used to monitor the summing node voltage, and the propagation time of the inhibit pulse. The estimated delay time from neuron firing to inhibition is 100nS. This limits the minimum overlap between patterns for any one array to 10% if the time between pulses is 1 μ S. Longer periods between pulses allow patterns with greater overlap to be recognized as belonging to different categories.

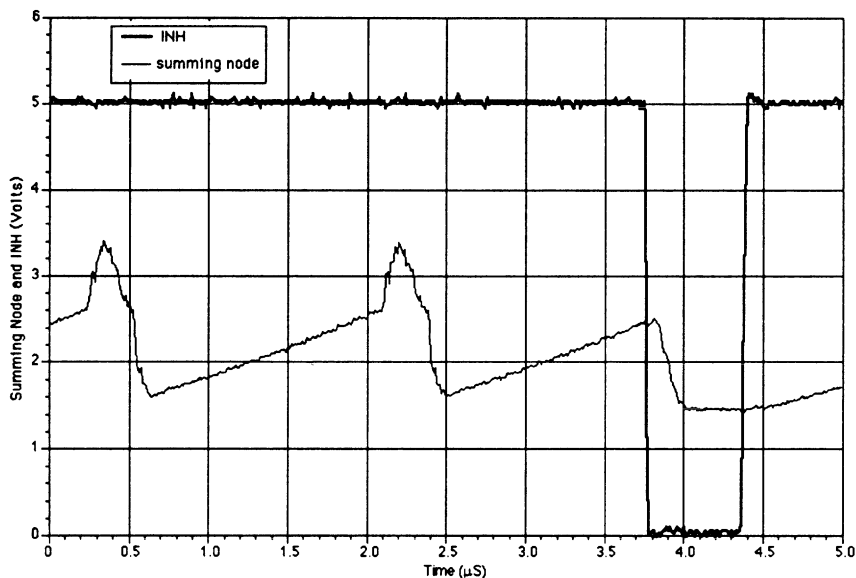


Figure 11. Measurements of the summing node voltage during firing and inhibit

PROCESS ARRAY TESTS

First, results illustrating the array chip extracting features from two input signals are presented. Two sinewaves with adjustable phase shift provided input data to the network. Before being applied to the network, the sinewaves were converted to pulses. Outputs from the pulse generators were applied to the 4 neuron classifier chip. Depending on the phase shift between the two inputs, one or two outputs became active and indicated specific features at the inputs.

Figure 12 shows a block diagram of the test fixture used in these experiments. An oscillator generated a 1KHz sinewave. A variable phase shifter and input converter provided two phase shifted and offset sinewaves from the single input. These two sinewaves were applied to two pulse

generators using optical coupling. Outputs from the pulse generators consisted of $1\mu\text{S}$ pulses whose frequency was proportional to the input level. First order low pass filters with a 6dB point of 8KHz provided a measure of the active fraction from the pulse generators. The two pulse inputs were applied to adaptable inputs 2 and 6 of the classifier array chip. Active outputs from the chip were also applied to 8KHz first order low pass filters. This increased the width of the pulses so they could be sampled over a period of 1mS by the digital storage scope. For this range, the digital scope had a sample time of $1.95\mu\text{S}$. Output waveforms were then digitally processed to extract the individual pulses. Further processing extracted the time between pulses used in the plots of the output levels. At 50KHz a maximum accuracy of about 10% can be obtained with accuracy decreasing as the time between pulses decreases.

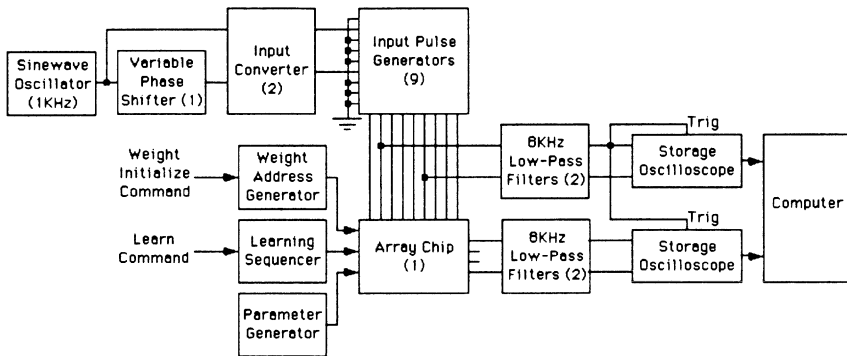


Figure 12. Block diagram of test fixture used for array tests

Table 1 shows external parameter inputs to the chip. All array tests presented here use these inputs. The input level and offset of the source sinewave was set for maximum level with the minimum amount of distortion as measured at the low pass filters. For each measurement, the phase shift of the two inputs was set to the desired level. Next, all weights were set to $+W_r$ by the weight write circuit. The weight write circuit sets the initial values to each of the synapses. The value of the write voltage $+W_r$ is set so the network has sufficient activity so that the synapse increment rate exceeds the decay rate. Then learning was enabled and the weights were allowed to come to equilibrium. Finally, the inputs and outputs were measured while the chip continued learning. Note that the inhibition voltage for measurements of the output was held constant at a level less than the reset voltage. This resulted in outputs that were completely uncorrelated.

LRNLEV: Precharge voltage for the capacitor Q5 in the weight modification circuit

VPREGAIN: Voltage on the gate of Q7

INHLEV: Inhibit level voltage.

PW : Output pulse width

THRESH: The minimum voltage to produce an active output (threshold)

INHEN: Gate voltage on Q18

+WR: Starting weights set by the weight write circuitry

SAMPTIME: The time the weight modification circuit samples the inputs and generates equation 8.

LRNRESTIME: The time that C5 is preset to LRNLEV

LRNTIME: The time where output pulses are enabled to transfer charge from the sampling capacitor to the weight capacitor.

LRNCYCLE: Time between learn resets.

Table 1. External parameter input values for array test where:

Eigenvectors of the input correlation matrix provide the best basis vectors for the representation of the input statistics [5]. The correlation matrix is a statistical measure used to determine the degree of similarity between elements. In our simple case, the matrix elements are just the time average of the two inputs, e.g., $X1*X1$, $X1*X2$, $X2*X1$, and $X2*X2$. The cross-correlation matrix of two sinewaves offset by half the amplitude of the sinewave have eigenvectors, $\{w_1, w_2\}$ of $\{\{0.707, 0.707\}, \{0.707, -0.707\}\}$. Since outputs from the system provide the only method of information exchange between nodes, the outputs should have activations that correspond to the eigenvectors. We expect one output to be maximum when the two inputs have the largest product and one output to be maximum when one input is maximum and the other input is minimum. Note that our system can only have positive weights, inputs, and outputs. Our system converges on the weights of $\{\{0.707, 0.707\}, \{1, 0\}\}$. This produces one output to be

maximum when the two inputs have the largest product and one output to be maximum when one input is maximum and the other input is minimum. Figure 13 shows this result using a theoretical plot of the activation levels for two outputs with the expected weights and a phase shift of 72° . When the inhibition voltage is less than the reset voltage, the outputs remain uncorrelated with the output having the largest sum-of-products active. This results in output 1 active for the first part and output 2 active for the second part of the cycle with the transition occurring at .3mS.

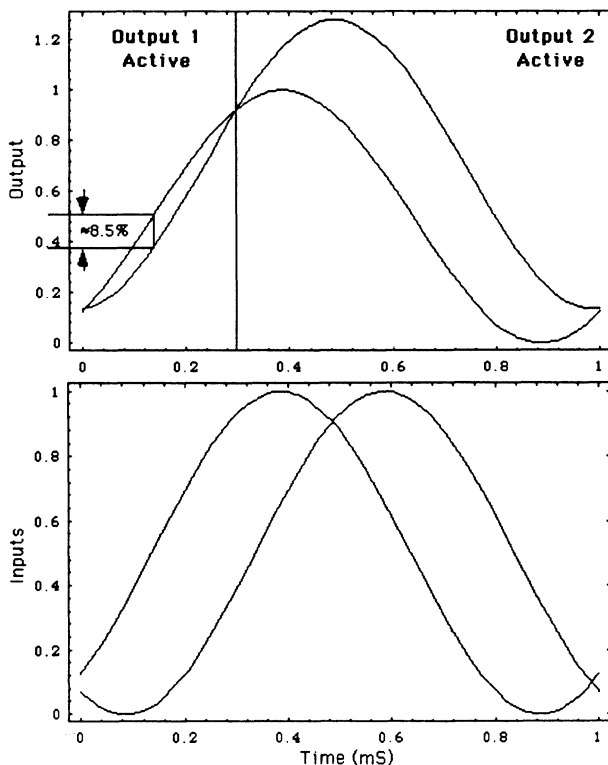


Figure 13. Inputs and theoretical outputs for a phase shift of 72° and weights of $\{.707, .707\}$, and $\{1,0\}$

Figures 14, 15, and 16 show results from experimental tests using three phase shifts. For the experimental test described, we only needed and therefore used 2 neurons, 2 inputs and 4 weights. The first graph shows activation of one output when the inputs were phase shifted by 36° . For a phase shift of 36° , the eigenvalue for the second eigenvector is too small for the network to extract, resulting in only one feature recognized by the network.

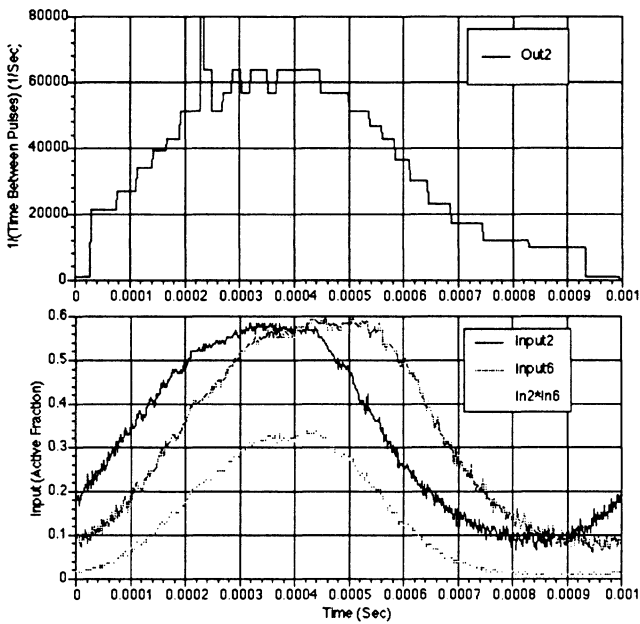


Figure 14. Measured outputs and inputs while learning inputs with 36° shift

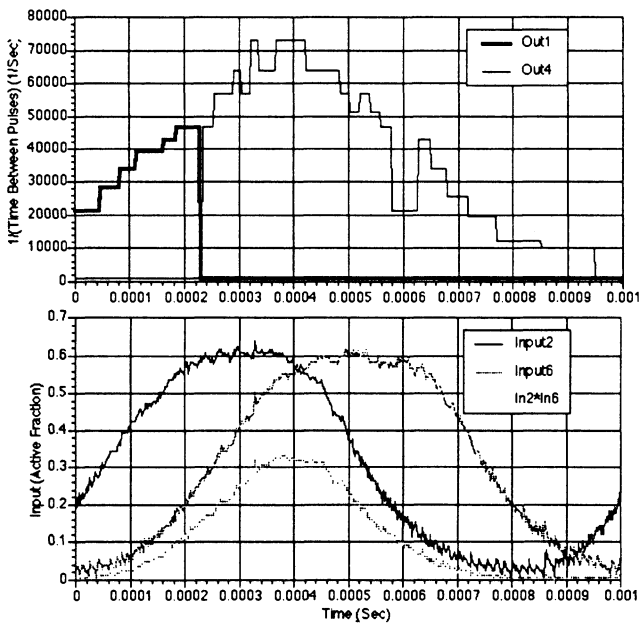


Figure 15. Measured outputs and inputs while learning inputs with 72° shift

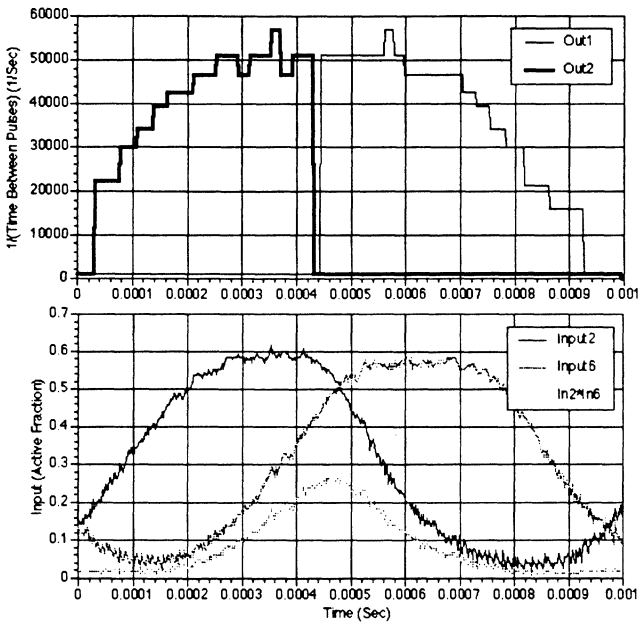


Figure 16. Measured outputs and inputs while learning inputs with 108° shift

At 72° phase shift, the network can detect two features in the input as shown in the figure. However, with an input of 72° , the network has two equally probable stable states. The other probable state was also detected. In this case the weights converged on the most dominate feature (like Figure 14). Figure 13 shows the theoretical transition point between two activations to be 0.3mS and Figure 15 shows a measured transition point of 0.225mS. Finally Figure 16 shows the case where the inputs have a 108° phase shift. This case has only the stable state shown, where two outputs are active indicating the two features were detected.

Several conclusions can be drawn from results obtained for inputs shifted by 72° . Since the network had two stable states, this is the point where errors in the two outputs force the two outputs to have nearly equal time between pulses. This results in one output completely inhibiting the other. The error in the time between pulses is equal to the maximum difference between the two theoretical outputs for the case where output 1 should be active. The maximum difference between the two theoretical outputs for this case is about 8.5% of full scale (see Figure 15). Additionally, the time to transition for output 1 active and output 2 active was theoretically 0.3mS and the measured transition time was 0.225mS. This also indicates an effective error in the output of about 10%. This effective error includes all network functions and

provides a measure of the system's ability to distinguish between two features.

Next, we demonstrate the ability of the system to generate representations from more complex inputs. Four second-order bandpass filters, with a center frequency of 23, 59, 159, and 408 Hz, generate four training waveforms from a single periodic triangle wave input. The positive part of these four training waveforms control the frequency of the four pulse generators that supply inputs to the chip. To allow our digital oscilloscope to record pulses at the inputs and outputs of the chip, the pulses were averaged with a 8kHz lowpass filter. This provides a signal that was proportional to the pulse density. Figures 17 and 18 show the triangle generator waveform and the four resulting waveforms as measured from the low-pass filter for two different input frequencies. Note the relative positions of the two inputs from the higher frequency bandpass filters. To generate the output waveforms shown in Figs. 19 and 20, the synaptic weights were first set to nearly equal values and learning was enabled. Figure 19 shows the outputs when learning inputs from Figure 17, and Figure 20 shows the outputs when learning inputs from Figure 18. Notice how the processor has grouped the inputs from Figure 17 into two features as shown in Figure 19. For a different input frequency, the processor grouped the inputs into three groups as shown in Figure 20.

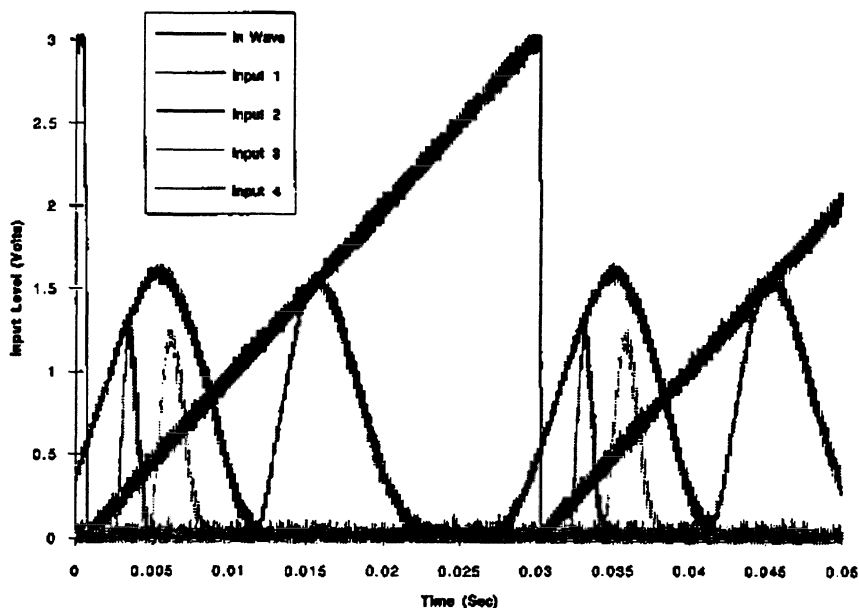


Figure 17. Training Set with 33Hz Triangle Input

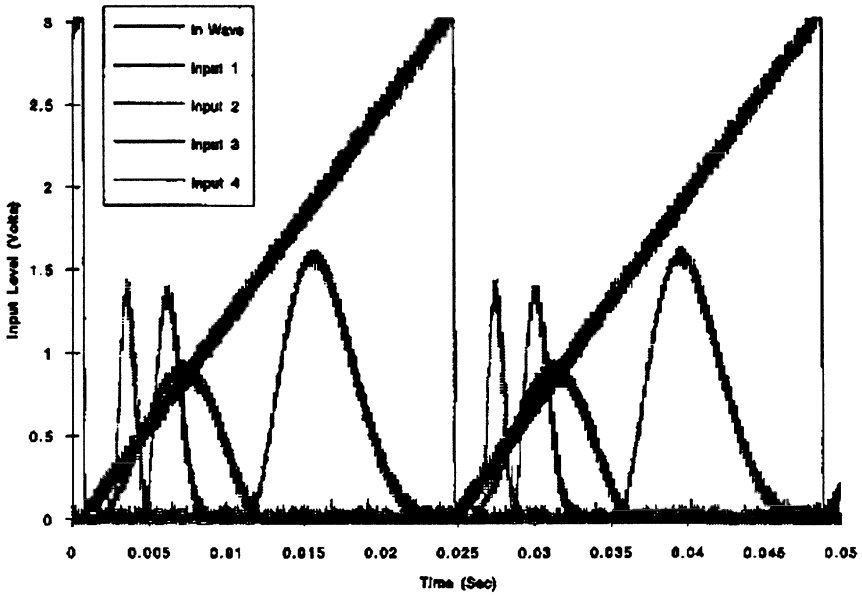


Figure 18. Training Set with 40Hz Triangle Input

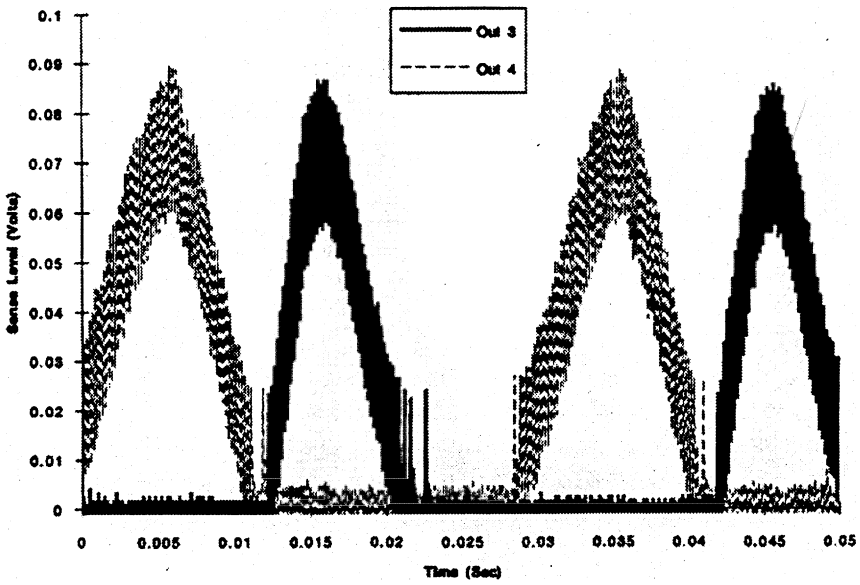


Figure 19. Measured Outputs after Learning 33Hz Training Set

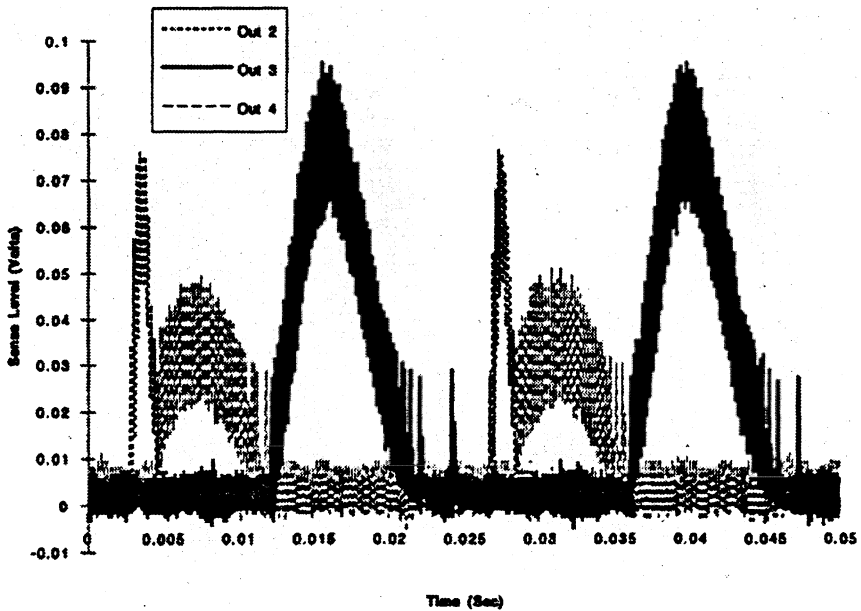


Figure 20. Measured Outputs after Learning 40Hz Training Set

CONCLUSIONS

We have constructed VLSI test chips to explore the operation of a processing node using pulsed communication, analog weights, and on-chip adaptation. We have implemented a variation of the modified Hebbian algorithm presented by Oja. To complete the learning rule, we have added a lateral inhibition mechanism that makes the output representation useful for a final layer of neurons for output mapping. The first chip allowed us to examine the internal operation of a single neuron and compare these results with the results predicted by theory and simulation. Analysis of the operation included the feed-forward transfer-function, weight adaptation, and inhibition functions. The array chip allowed us to test a complete classifier system's ability to distinguish between two similar features. This ability was further demonstrated with complex time-varying signals. The classifier grouped the inputs into classes determined by the statistics of the inputs.

This array system performed as a fuzzy classifier where the weights were normalized and the outputs had controlled overlap. Future experiments will duplicate in hardware simulations of learning nonlinear plant models. Further work will also demonstrate the ability to produce inverse plant models for adaptive control applications.

References

- [1] J. Donald, and L. A. Akers, "An Adaptive Neural Processing Node", IEEE Neural Networks, Vol. 4, No. 3, p.413, 1993
- [2] J. Donald, and L. A. Akers, "A Neural Processing Node with On Chip Learning", Proceedings of the IEEE International Conference on Circuits and Systems, Chicago, 1993, p. 2748
- [3] E. Oja, "A Simplified Neuron Model as a Principal Component Analyzer," Journal of Mathematical Biology vol.15, pp.267-273, 1982
- [4] T. Leen, M. Rudnick and D. Hammerstrom, "Hebbian Feature Discovery Improves Classifier Efficiency," Proc. Int. Joint Conference on Neural Networks, Washington, DC, June 1989
- [5] T. Sanger, "An Optimality Principle for Unsupervised Learning," Advances in Neural Information Processing Systems 1, 1989
- [6] K. Hunt, and D. Sbarbaro, "Neural Networks for Nonlinear Internal Model Control," IEE Proceedings, vol. 138, no. 5, September 1991
- [7] K. S. Narendra, and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," IEEE Transactions on Neural Networks, Vol. 1, no. 1, pp 4-26 March 1990
- [8] D. Rumelhart, and J. McClelland, "Parallel Distributed Processing," Massachusetts: The MIT Press, 1986
- [9] R. Lippmann, "An Introduction to computing with Neural Nets," IEEE ASSP Magazine, vol. 4, pp. 2-22, Apr. 1987
- [10] T. Sanger, "A Theoretical Analysis of Population Coding in Motor Cortex," Neural Networks Concepts, Applications, and Implementations Volume II, New Jersey: Prentice Hall, 1991
- [11] R. Linsker, "From basic network principles to neural architecture", Proc. Natl. Acad. Sci USA, Vol. 83 pp.7508-7512, Nov, 1986

- [12] R. H. White, "Competitive Hebbian Learning: Algorithm and Demonstrations," *Neural Networks*, Vol. 5, pp. 261-275, 1992
- [13] P. Hasler and L. Akers, "A Continuous time synapse employing a refreshable multilevel memory," *Proceedings of the IEEE International Conference on Neural Networks*, July 1991
- [14] A. F. Murray, "Pulse Arithmetic in VLSI Neural Networks," *IEEE Micro*, pp. 64-74, December 1989
- [15] A. Hamilton, A. F. Murray, D. J. Baxter, S. Churcher, H. M. Reekie, and L. Tarassenko, "Integrated Pulse Stream Neural Networks: Results, Issues, and Pointers," *IEEE Transactions on Neural Networks*, Vol. 3, no. 3, pp. 385-393, May 1992
- [16] B. Lathi "Modern Digital and Analog Communication Systems," New York: Holt, Rinehart and Winston, 1983
- [17] B. Linares-Barranco, E. Sánchez-Sinencio, A. Rodríguez-Vázquez, and J. L. Huertas, "A CMOS Implementation of FitzHugh-Nagumo Neuron Model," *IEEE Journal of Solid-State Circuits*, Vol.26, no. 7, pp. 956-965, July 1991
- [18] Y. Arima, K. Mashiko, K. Okada, T. Yamada, A. Maeda, H.Kondoh, and S. Kayano, "A Self-Learning Neural Network Chip with 125 Neurons and 10K Self-Organization Synapses," *IEEE Journal of Solid-State Circuits*, Vol. 26, no. 4, pp. 607-617, April 1991
- [19] L. W. Massengill, "A Dynamic CMOS Multiplier for Analog VLSI Based on Exponential Pulse-Decay Modulation," *IEEE Journal of Solid-State Circuits*, Vol. 26, no. 3, pp. 268-276, March 1991
- [20] N. E. Cotter, and O. N. Mian, "A Pulsed Neural Network Capable of Universal Approximation," *IEEE Transactions on Neural Networks*, Vol. 3, no. 2, pp. 308-314, March 1992

- [21] J. E. Tomberg, and K. Kaski, "Pulse-Density Modulation Technique in VLSI Implementations of Neural Network Algorithms," *IEEE Journal of Solid-State Circuits*, Vol. 25 no.5, pp. 1277-1286, October 1990
- [22] C. Mead, "Analog VLSI and Neural Systems," Reading: Addison-Wesley, 1989
- [23] K. Madani, P. Garda, E Belhaire and F. Devos, "Two Analog Counters for Neural Network Implementation," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 7, July 1991

Index

A

action potential 1, 3, 43, 64, 65, 82,
200, 204, 206, 209, 211, 223
amorphous-silicon 30, 173, 269

B

back-propagation 30, 171, 194
BAM 200, 232, 235, 237
bandpass filter 72, 103, 107, 285
body effect 20
Boltzmann machine 188, 189, 194

C

cable equation 41, 42
charge leakage 97, 130, 266, 268
chemical synapse 40, 42, 43, 45, 59
chemically gated channel 205, 209
competitive learning 80
current controlled oscillator 82

D

delay modulation 12, 114
delta rule 184, 186
differential amplifier 130, 132

E

EPSILON 10, 14, 24, 25
ETANN 27, 170

F

fault tolerance 114
FitzHugh-Nagumo model 213, 215,
216, 223, 224, 225
floating-gate 29, 268

G

Gram-Schmidt orthogonalization 277

H

Hamming distance 271
Hebb rule 46, 135, 186, 264, 277
highpass filter 71, 72
Hodgkin Huxley model 3, 211, 213
Hopfield network 80, 136, 141, 181,
188, 194, 232, 237, 238, 239

I

interpulse interval 43, 44
ionic channel 42, 43, 49, 204, 209,
212

L

lateral inhibition 79, 86, 263, 265,
277, 278
linear feedback shift register 249
lowpass filter 71, 72, 102, 103, 105,
107, 285

M

McCulloch-Pitts 2
membrane resting potential 43, 202,
215
MNOS 29, 172
monostable multivibrator 155
multiplex 50, 167, 168, 178, 183

N

Nernst potential 202, 203
neural-type 4, 101, 102, 104, 113,
141
neuristor 3, 4
neuromeme 2
neurotransmitter 42, 204, 205, 206,
209, 213, 215

P

phase modulation 12
post-synaptic potential 46, 65
pseudo-random number generator
189, 190, 249
pulse amplitude modulation 11, 114
pulse density modulation 166, 190,
193, 194
pulse duty cycle 113, 115, 128, 136
pulse frequency modulation 12, 82,
114, 121
pulse width modulation 12, 114, 250
Purkinje cell 41, 42

R

refractory period 4, 11, 67

S

Schmitt trigger 2, 3, 81, 92

self-organizing feature map 80, 190

sigma-delta modulator 171, 194

subthreshold 92

supervised training 264, 266

switched-capacitor 172, 173, 174,
175, 181, 190, 195, 275

T

target tracker 57

tetanus 48

transconductance amplifier 224, 230

transconductance multiplier 23, 235

transconductance-mode 216, 233

U

unsupervised learning 190, 263, 273

V

voltage controlled oscillator 14, 58,
115, 116

voltage gated channel 204, 206, 208

W

weak inversion 82, 152, 154, 158

winner-take-all 79, 80, 81, 86, 88, 92,
136, 141, 277