Analog CMOS Circuits for Artificial Neural Networks

by

Christian R. Schneider

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

ISBN 0-315-77845-8

ANALOG CMOS CIRCUITS FOR ARTIFICIAL NEURAL NETWORKS

BY

CHRISTIAN R. SCHNEIDER

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

DOCTOR OF PHILOSOPHY

Abstract

CMOS circuits implementing analog neural networks with built-in Hebbian and contrastive Hebbian learning have been designed, fabricated and tested. These circuits employ capacitive synaptic weight storage. The Hebbian learning circuit was incorporated into a 600 synapse, 28 000 transistor neural network to evaluate its performance in a medium-sized system. A discussion of CMOS synaptic circuits motivated by invertebrate biology explores the relationship between certain aspects of learning in the marine mollusc *Aplysia* and Hebbian learning. Because adaptive circuits, such as neural networks with built-in learning, can compensate for imperfections in the components from which they are constructed, it is possible to build this type of system using simple, silicon area-efficient analog circuits. Typical analog computational elements, such as multipliers and adders, are far more compact than their digital counterparts: consequently analog neural networks have tremendous computational potential. A $3\mu m$ CMOS $1 cm^2$ implementation of the circuits described in this work has a throughput of more than 6 billion analog multiplications per second, at a conservative $1 MHz$ operating frequency.

Acknowledgements

I would like to thank my advisor, Professor H. C. Card for his encouragement and many useful suggestions throughout this project. I would also like to thank my brother, Roland Schneider, for performing system-level simulations, constructing test hardware, and many helpful discussions.

Financial support from Micronet and the Natural Sciences and Engineering Research Council of Canada, as well as equipment loans from the Canadian Microelectronics Corporation, are gratefully acknowledged.

Table of Contents

Glossary

AI: Artificial intelligence

algorithmic computation:
Computation in which a procedure consisting of a sequence of steps (the algorithm) is followed to transform a set of inputs into a set of outputs.

analog circuits:
Circuits in which quantities are represented as continuous-valued voltages, currents, etc.

ANN: Artificial neural network.

artificial intelligence:
Artificial systems (computers) that perform operations normally associated with biological intelligence.

artificial neural network:
A computer designed to perform computation in a manner similar to that of a biological neural network (biological brain).

back-propagation learning:
A supervised learning technique for feed-forward ANNs.

biological neural network:
The brain of a human or lower animal.

bipolar signals:
Signals which can take on both positive and negative values.

BNN: Biological neural network.

BP: Back-propagation.

chip: Integrated circuit.

CHL: Contrastive Hebbian learning.

CMOS: A common IC fabrication technology -- Complementary Metal Oxide Semiconductor.

contrastive Hebbian learning:
A supervised learning technique for fully-connected (Hopfield-type) ANNs.

digital circuits:
Circuits in which quantities are represented as discrete-valued voltages, currents, etc.

EEPROM:
Electrically Erasable Programmable Read Only Memory. Typically used in ANNs for synaptic weight storage.

Hebbian learning:
An ANN learning technique, typically used for unsupervised learning applications.

Hopfield ANN:
An ANN architecture with synaptic connections between each pair of neurons (ie. fully connected).

HSPICE:
Circuit-level simulation program.

IC:	Integrated circuit
Manhattan CHL:	Contrastive Hebbian learning variant, in which fixed-size weight change steps are used.
neuron:	Basic information processing unit in biological and artificial neural networks.
non-algorithmic computation:	Computation which does not follow an algorithm.
SPICE:	Circuit-level simulation program.
supervised learning:	ANN network weight training scheme in which the network is presented with pairs of inputs and outputs from a set of training data which it is supposed to learn to associate.
synapse:	Connection point between neurons. The synaptic weight determines the strength of the synaptic connection.
synaptic weight:	The synaptic weight determines the strength of the synaptic connection between two neurons.
transconductance circuit:	A circuit with voltage inputs and current outputs.
unipolar signals:	Signals which can only take on positive values.
unsupervised learning:	ANN network weight training scheme in which the network is presented with data from which it is supposed to extract common features, discover regularity, patterns, etc.
XNOR:	Exclusive NOR
XOR:	Exclusive OR

CHAPTER 1

Introduction

Digital computation systems have evolved over the past few decades into the ubiquitous, inexpensive electronic computers of the 1990's. These machines perform a wide variety of tasks, such as arithmetic and sorting, exceptionally well: far better in fact than their human creators. Conversely, there are a large number of tasks which humans perform with ease which digital computers perform rather poorly. Some of the most important examples are found in the field of artificial intelligence (AI), including 3-D scene interpretation and unconstrained speech recognition. What is the fundamental difference between digital electronic and biological computation systems which makes them suitable for such disparate computational tasks? One of the most fundamental differences is system architecture. Electronic computers perform digital computations, typically in a small number of complex processing units, whereas biological computers (brains) consist of a very large number of simple processing units which perform analog computations in parallel. The basic premise of artificial neural network (ANN) research is that when we can identify and synthesize the features of biological computers which enable them to perform tasks such as 3-D scene interpretation, then we can construct artificial systems which can perform the same operations.

A general definition of artificial neural networks is given by Kohonen: ''Artificial neural networks are massively parallel interconnected networks of simple (usually adaptive) elements and their hierarchical organizations, which are intended to interact with objects of the real world in the same way as biological nervous systems do'' [Kohonen-1]. There is a fundamental difference in the way that conventional digital computers and biological neural networks (BNNs) process information: in a digital computer, information is manipulated according to an *algorithm*, or sequence of instructions, whereas BNNs perform non-algorithmic computation, in which computation does not involve executing a sequence of instructions [Kohonen-1]. Thus conventional computers are suited to applications whose solutions may be cast into the form of an algorithm. Interestingly, for such an application, a simple digital computer can out-perform the human brain, a BNN with vastly greater computational power.

1.1 The Failure of Algorithmic AI

BNNs excel at the operation of pattern matching and completion, which is thought to be the basis of much of the non-algorithmic information processing in the brain. Pattern matching and completion of the type found in BNNs is extremely difficult to implement efficiently as an algorithm. An interesting case, which exemplifies this difficulty, arose during the development of AI *expert systems*, which attempt to capture the knowledge of a human expert as a complex database of rules. Although expert system research has been moderately successful, their performance frequently resembles that of an "expert novice", rather than a true expert. The difference is that a novice tends to apply rules that he has been taught, while a true expert uses the non-algorithmic pattern matching at which the brain excels, analyzing a new scenario by comparing (matching) it with situations which were encountered previously. As a result of this difference between expert systems and human experts, the initially encouraging performance of an expert system can not easily be improved to the level of a human expert, simply by adding more rules to its database. Significantly, when human experts were interviewed by expert systems developers, and were asked to elucidate the rules by which they made their decisions, more often than not they gave examples, rather than rules. The reason for this is simple: their expert knowledge is stored as a complex collection of examples, rather than as a set of rules.

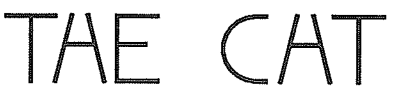

Figure 1.1 *Example of a difficult character recognition problem.*

Outside the expert systems field, AI is replete with problems which are simple for humans to solve, and are very difficult for conventional computers to solve. An example which demonstrates a few of the typical difficulties is illustrated in Fig 1.1 [Rumelhart-1]. When presented with these six symbols, the human brain will readily recognize the sentence fragment "THE CAT", even though the "H" and the "A" are represented by the same symbol in the two words. This deceptively simple recognition process is actually quite complex: the individual line segments are grouped into symbols, the symbols into words, and lastly semantic knowledge of English is used to complete the decoding of the sentence fragment. Unfortunately, this type of multi-level processing is exceptionally difficult to implement in a conventional computer system: there is a fundamental difficulty in attacking a problem at a number of different levels (line, symbol, word, and English semantic in this example) simultaneously to arrive at a solution. Clearly even this simple recognition problem cannot be solved without looking at the problem from a number of different levels of abstraction at one time.

This example illustrates another of the fundamental weaknesses of conventional computers: they suffer from the "worm's eye view" syndrome, where the minutiae of a problem are analyzed, but there is no good, general overview of the entire problem. Again, it appears that this deficiency and other related ones are very fundamental, resulting from the basic structure, or architecture, of digital computers.

The failure of expert and other AI systems to achieve performance approaching that of a BNN using algorithmic information processing resulted in a renewed interest in ANN research in the 1980's. Artificial neural network research is an attempt to solve these problems by designing systems structured in the same way as the brain, an organ which can solve these types of problems with ease. Non-algorithmic pattern matching and completion is the basic operation that ANNs implement. Thus we have an existence proof (the biological brain) that this type of problem can be solved efficiently: what remains is to extract the salient features of the biological brain, and find a way to implement these features in an artificial system.

1.2 Neural Network Models

The complexity of biological neural systems has resulted in the development over the years of a highly disparate collection of artificial neural systems, each of which models certain aspects of biological neural anatomy, and ignores many others. There is currently no consensus as to which characteristics of neural anatomy are important for computation, and which are merely the result of design restrictions imposed on the biological system by the basic chemistry and physics of biological systems.

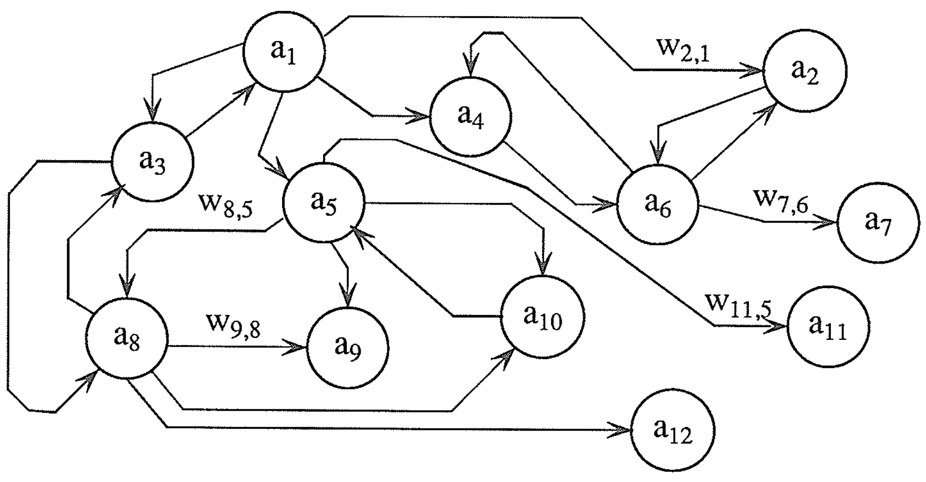

Figure 1.2 *Portion of a artificial neural network.*

Despite considerable variation, most current research employs the architecture of Fig. 1.2. The circles represent processing units, or nodes (roughly analogous to neurons), and the lines between processing units represent some form of communication path, or link, between the nodes (roughly analogous to axons, synapses and dendrites).

Prevailing neural network models impose several restrictions on this general architecture: typically the processing node stores a single activation state $a_i(t)$, and performs some sort of weighted summation aggregation operation, using the current output values of connected nodes to update its own state. Typical equations governing the operation of an ANN are [Rumelhart-1]:

$$a_i(t+1) = F\left[\sum_{j=1}^{N} w_{ij}\, o_j(t),\ a_i(t)\right] \tag{1.1}$$

$$\Delta w_{ij} = g(a_i(t),\ t_i(t))\ h(o_j(t),\ w_{ij}) \tag{1.2}$$

where

$o_i = f(a_i)$:	current output of node i
a_i:	current value at node i
w_{ij}:	strength of node j's influence on node i
t_i:	external teaching input for node i

$f(\cdot), F(\cdot)$:	activation functions
$g(\cdot), h(\cdot)$:	learning functions

Learning in these networks involves modifying the weights w_{ij} (1.2), and problem solving is accomplished by running the network through a number of time-steps until activation states stop changing from one time-step to the next (1.1). This neural network model omits a great many biological details: in a sense it is the minimal implementation of an artificial neural system which can still be used to perform arbitrary computation. However, it is not clear that the omission of biological details from this model improves the *efficiency* of the resulting ANN system, either in terms of processing speed or circuit size.

This interesting question of how closely biological neurons should be modeled for maximum system efficiency is one of the issues which was examined as part of this research program. Below, a general outline of the approach we have taken to ANN implementation using CMOS VLSI circuits is presented, followed by a discussion of other researchers' work which is most relevant to this investigation.

1.3 CMOS VLSI and Artificial Neural Networks

CMOS VLSI has been used extensively in the implementation of neural networks; the high integration density of modern MOS technologies is very attractive for the construction of large neural systems. Most MOS neural network implementations consist predominantly of digital circuitry, and are designed using the same methods that are used for conventional computers. Digital implementations of ANNs were recently reviewed by Atlas and Suzuki [Atlas]. Digital systems have many desirable properties -- high accuracy, reliability, repeatability, etc. However, a substantial price is paid in terms of circuit complexity, and consequently silicon area, for these features.

In this work we present analog circuits for neural networks which are far more compact than their digital counterparts. Analog implementations of ANNs have been reported by several authors, for example [Graf, Mead, Raffel, Schwartz-1, Satyanarayana, Hollis, Murray, Sage, Shoemaker, Holler, Card-1, Furman, Alspector-1, Schwartz-2, Clark, Arima], and a review of analog neural networks up to 1988 was included in [Card-1].

By using simple analog circuits, we can build systems with tremendous computational power: for example, a neural network chip that can perform more than 6 billion analog multiplications per second ($1cm^2$ die, $3\mu m$ CMOS, $1MHz$ operation). Unfortunately, this computational potential is difficult to exploit, because the architecture of an analog VLSI system must be tolerant of imperfections in the components from which it is constructed -- for example, two identical multipliers may have outputs that differ by 20% or more for the same input values. Digital systems design relies upon identical components having identical behavior: thus most digital system architectures cannot be implemented using analog components.

Neural networks with learning capability, as well as other adaptive networks, are suitable candidates for implementation using simple, highly imperfect analog components. If designed properly, these systems can "learn" to take component variations into account. Examples of this property of neural networks, as it relates to our work, are discussed in subsequent chapters.

The chapters which follow describe analog circuits for a variety of ANN applications. Despite their differences, several important design characteristics are shared by all circuits. Firstly, all circuits are constructed from compact low-accuracy analog computational components. As discussed above, this makes it possible to put tremendous computational power in a single integrated circuit; in addition, low-accuracy components are biologically plausible. The second common characteristic is that synaptic weights are stored as electric charge on a capacitor in all ANN implementations developed as part of this work. Capacitive weight storage is a silicon-area efficient way to store an analog weight value, and can be implemented in the IC fabrication processes available to our university. Lastly, our circuits all incorporate *in situ* learning; that is, weight adaptation circuitry is built into each synaptic circuit. *In situ* learning provides a way for an ANN system to compensate for variations in the low-accuracy analog components from which it is constructed, as well as avoiding the problems of long learning times and weight downloading, which plague many ANN systems with off-chip learning. It is anticipated that the present synaptic circuits will find applications in a wide variety of ANNs, operating in either supervised [Hopfield, Sivilotti, Peterson-2, Hinton-1, Rumelhart-2, Jacobs, Hinton-3] or unsupervised [Linsker, Kohonen-1, Hinton-2] learning modes. The suitability of our circuits for these applications is discussed in chapter 3 (unsupervised) and chapter 5 (supervised). Below, a survey of other electronic ANN implementation research efforts, which incorporate some of the features of this work, is presented.

The previous studies which are most relevant to our discussion are [Schwartz-1, Satyanarayana, Card-1, Furman, Alspector-1, Schwartz-2, Clark, Arima, Ismail-1, Ismail-2]. Papers [Schwartz-1, Satyanarayana, Card-1, Furman, Schwartz-2, Arima] represent synaptic weights as charge on capacitors, as opposed to digital storage of weights [Raffel, Hollis, Alspector-1], or EEPROM weights [Sage, Shoemaker, Holler]. Papers [Card-1, Furman, Alspector-1, Schwartz-2, Clark, Arima] report circuits which incorporate *in situ* learning; that is, the synapses contain circuitry which performs local computations of weight updates. A variety of analog ANN circuit designs, as well as general system-level considerations are discussed in [Ismail-1, Ismail-2].

Schwartz et al [Schwartz-1] describe an ANN synaptic circuit in which the weight is represented as the difference between the voltages stored on two capacitors. Synaptic weights are modified by transferring charge from one capacitor to the other using charge injection, and thus the total charge on the two capacitors remains constant. The authors do not address the issue of learning (as opposed to setting weight values), since their interest was in evaluating their weight storage circuit, and not in constructing a complete ANN system.

In [Satyanarayana], a reconfigurable feed-forward ANN implementation consisting of 1 024 distributed-neuron synapses with analog data processing is described. Capacitive weight storage is employed, with a weight resolution of 1% of the maximum weight value. Analog weight resolution is limited by charge injection and leakage at the storage node, and analog weights are periodically refreshed from an off-chip digital weight memory.

Furman et al have developed an analog CMOS implementation of the back propagation algorithm [Furman], where each IC consists of 48 input and 10 output neurons, and a fully connected synaptic matrix. Again, weights are stored as electric charge on capacitors, and *in situ* learning is performed by a three analog multiplier implementation of the generalized delta rule. Chip test data is not reported.

The paper [Schwartz-2], discusses a special-purpose VLSI ANN circuit, which can learn to approximate a mathematical function of a small number of variables from discrete data points. A simple one-layer linear network with capacitive weight storage is used in this specialized application.

Clark [Clark] describes a CMOS current mode ANN with *in situ* Hebbian learning which is intended for a self-organizing network, such as the Kammen-Yuille orientation selective network [Kammen]. Learning in the circuit is based on a linear switched capacitor integrator circuit using an operational amplifier.

In [Alspector-1], a 32 neuron 496 bi-directional synapse ANN which implements both Boltzmann and mean field networks is described. Weights are stored as 5 bit digital values, and *in situ* learning circuitry is incorporated into each synapse. Aside from weight storage, chip operation is analog. 32 uncorrelated pseudo-random noise sources are used to implement the simulated annealing process of the stochastic Boltzmann machine; the signals are summed along with the weighted post-synaptic signals at the input to each neuron. The authors report simulations comparing the performance of back propagation, Boltzmann and mean field architectures. They concluded that these three approaches have approximately the same recognition accuracy in partity and replication problems. They also studied the effect of the limited dynamic range of their 5 bit digital weights, and concluded that in most cases the use of 5 bit weights had no effect on performance. An exception was the mean field architecture, in which performance degradation occurred for large replication problems. This is consistent with our simulations, in which we found that small weight increments are essential for stable learning in mean field networks [R. Schneider]. No IC test data comparing chip performance to simulation results is reported in [Alspector-1].

Arima et al [Arima] describe a combined analog and digital IC implementation of a Boltzmann ANN, with 125 neurons and 10 000 synapses. Analog capacitive weight storage is used, together with binary neuron activations. The product between binary neuron outputs and analog weights is implemented with a six transistor circuit. The network is organized into a fully connected Boltzmann ANN with symmetrical weights ($W_{ij} = W_{ji}$). Network weight adaptation is achieved with a modified version of Boltzmann learning -- the learning algorithm is implemented with a weight modifier circuit, consisting of two charge-pump circuits connected in series: the weight modifier circuit is designed to change weights in increments of approximately 10% of their full range value. The authors report training the IC to memorize 15 patterns (25 input, 100 output, no hidden units). Our simulations of mean field ANNs, a similar neural network architecture, and discussions in [Schwartz-1] indicate that a weight change of far less than 10% (perhaps 100 to 1000 times smaller) is required for stable weight convergence. Also, Arima et al's circuit has poor matching between the two phases of the Boltzmann learning algorithm, which our simulations indicate will degrade network performance. Perhaps significantly, no test results are reported in which hidden units are employed, as this is the situation in which these problems will become evident.

The brief descriptions of related work presented above illustrates the tremendous variety of approaches to electronic ANN implementation which are currently under investigation. The remainder of this document presents four CMOS ANN implementations. Chapter 2 discusses a CMOS synaptic circuit with Hebbian learning and unipolar weights. In chapter 3, a CMOS ANN implementation with bipolar Hebbian synapses is described and IC test results are reported. Circuits motivated by synaptic function of the marine mollusc *Aplysia*, which model habituation, sensitization and classical conditioning, appear in chapter 4. Lastly, chapter 5 describes our work in developing a fully analog mean field ANN.

CHAPTER 2

Unipolar Hebbian Synaptic Circuits

2.1 Introduction

In the development of the circuits described in this chapter, the primary goal was silicon-area efficiency. To achieve this goal, very simple circuits are used to implement arithmetic operations, such as multiplication and summation. Although many current ANN architectures require more accurate computation than these simple circuits provide, in the long term it is important that the properties and capabilities of the hardware that is used to implement an ANN be taken into account in the system design process. Thus ANN architectures must be developed that are suited to implementation using low-accuracy computational components. This also makes sense from the biological plausibility point of view, because most current ANN models require far more computational accuracy than is found in a biological neural network. Below, a compact analog CMOS synaptic circuit which uses the ''natural'' capabilities of CMOS VLSI is described, and its performance is evaluated.

2.2 Basic MOS Synapse

Artificial neural networks consist of neurons and synapses; the area of electronic implementations is normally dominated by the synapses, as there are many more of these than there are neurons (Fig. 2.1). In the approach shown in Fig. 2.2, the synaptic weights W_{ij} between neurons are represented by the channel conductance of MOS transistors, with weight voltages V_C stored on capacitors. The output of a neuron corresponds to an analog voltage V_i $(-V < V_i < +V)$ which is a function of a weighted sum of its inputs.

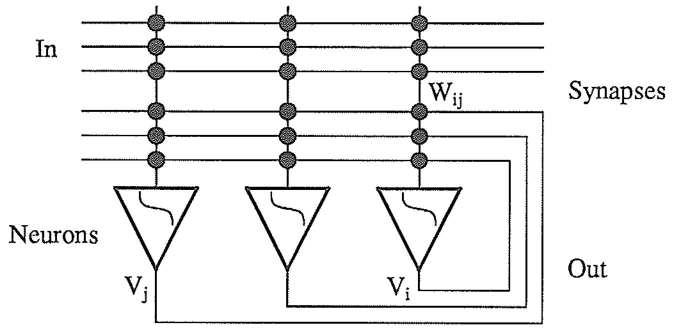

Figure 2.1 *Model of an artificial neural network.*

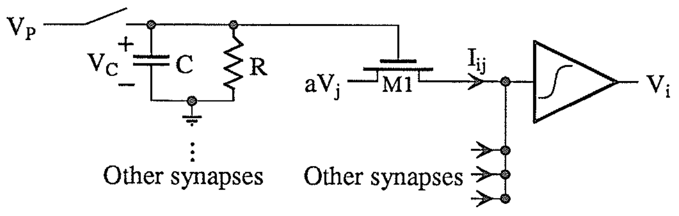

Figure 2.2 *Synaptic weights (W_{ij}) in analog MOS neural networks are commonly represented as charge storaged on a capacitor, which in turn controls the conductance of an MOS transistor.*

Provided the artificial neurons produce both $+V_j$ and $-V_j$, an inhibitory weight may be realized by employing $-V_j$. In this way, all weights W_{ij} can be positive (unipolar weights). In Fig. 2.2 the weight W_{ij} is adjusted according to a learning algorithm which is normally performed off-chip, and applied as V_p. In some cases the weight storage (which refreshes the capacitor voltage) is on-chip but the learning algorithm runs off-chip. In this chapter, we present circuits which perform *in situ* weight adjustments as a continuous analog process while the network is operating. We do not switch learning signals on and off; learning proceeds along with the circuit dynamics, although at a much slower rate.

To increase or decrease the synaptic weight one adds or removes charge from the capacitor C. A common analytical form of Hebbian learning is [Kohonen-1]:

$$\frac{dW_{ij}}{dt} = AV_iV_j - BW_{ij} \tag{2.1}$$

where A is usually taken to be a constant term to represent the learning process; B may in general be dependent upon V_i and V_j, and represents a weight decay or forgetting term. Equation (2.1) is consistent with Hebb's original qualitative statement of the learning rule [Hebb]:

> ''When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased''

Let us consider how the capacitor voltage V_C controls the weight W_{ij} of the i,j synapse. If one employs aV_j as the drain voltage of the MOS transistor M_1, where a is a small constant, then $V_D \ll V_G - V_T$ and this device operates in its triode region. In the simple case of threshold voltage $V_T = 0$ the drain current through this device, which contributes to the total current into neuron i, is given, for $V_C > V_T$, by

$$I_{ij} = \beta \left[(V_G - V_T)V_D - \frac{V_D^2}{2} \right] = \beta a (V_C - V_T)V_j = W_{ij}V_j \tag{2.2}$$

where the $V_D^2 / 2$ term is assumed to be negligible, $\beta = \mu C_{OX} W/L$ is a constant known as the conduction parameter of the transistor, and V_C and aV_j represent the gate and drain voltages V_G and V_D respectively. For $V_C < V_T$ the transistor M_1 is in the cutoff region and the current decreases exponentially with further decreases in V_C. The weight W_{ij} remains positive (but very small) for negative V_C.

2.3 Unipolar Hebbian Synapses and MOS Learning Rules

Hebbian learning depends upon local variables V_i and V_j, and there are penalties associated with implementing learning externally in VLSI; these penalties involve the amount of wiring and the number of pins required, which in turn impact the achievable speed and resolution of the circuit. It is therefore desirable to compute the weight changes locally at each synapse. One method is to employ an analog circuit such as a Gilbert multiplier [Gilbert] with inputs V_i and V_j to provide a current I_C to the capacitor to represent the first term of (2.1), and a

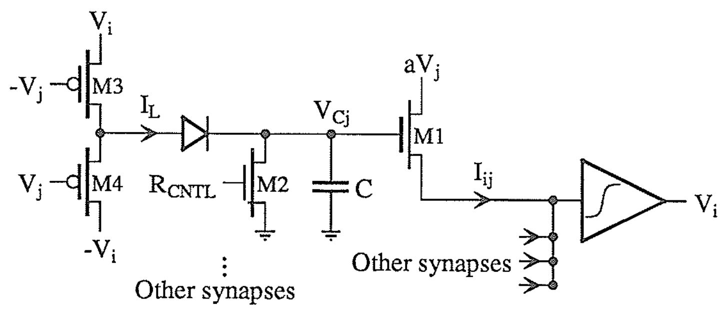

Figure 2.3 *Analog synapse with low transistor count which qualitatively approximates multiplicative Hebbian correlation of input and output activities V_i and V_j using a two transistor multiplier.*

leakage resistance R to represent the forgetting term. The leakage resistance R leads to a modulation of V_C and hence W_{ij} in accordance with the analytical form of (2.1). To reduce the complexity of the synaptic circuitry the multiplier may be replaced with the simpler circuit of Fig. 2.3. The leakage resistor R is approximated by the transistor M_2. The rate of synaptic weight change (the learning rate) is then given by

$$\frac{dW_{ij}}{dt} = \beta a \frac{d(V_C - V_T)}{dt} = \frac{\beta a}{C} I_L - \frac{\beta a}{RC} V_C \tag{2.3}$$

and the learning current I_L in (2.3) is

$$I_L = \beta V_i (V_j + \frac{V_i}{2} - V_T) \tag{2.4}$$

when V_i and V_j are both positive or both negative and $V_C = 0$. When V_i and V_j have opposite signs, $I_L = 0$. It is apparent from (2.4) that this circuit implements a form of Hebbian learning. Eqns. (2.3) and (2.4) are approximations of the behavior of the circuit of Fig. 2.3: they are intended as a guide to understanding the simulations below.

I_L is the current charging the weighting capacitor (the learning signal) which for $V_C = 0$ is approximately proportional to the product $V_i V_j$ (2.4). As the capacitor charges (V_C and W_{ij} increase) the current I_L will decrease and eventually become zero when the output voltage from the (M_3, M_4) pair is equal to $V_C + V_D$, where V_D is the forward diode voltage drop. This is in contrast to (2.1) where the learning signal AV_iV_j is independent of W_{ij}. The diode in Fig. 2.3 isolates the charge stored on the capacitor when V_iV_j is low (when V_i and V_j are uncorrelated or anticorrelated). In other words, the diode decouples learning and forgetting so that their rates may be independently adjusted. Normally the learning rate will be considerably larger than the forgetting rate so that the charge on the capacitor remembers a large number of training patterns. To summarize, (2.3) is a crude approximation of (2.1); however, learning governed by (2.3) is still consistent with Hebb's original qualitative statement, that weights increase according to the correlation between V_i and V_j. Quantitatively it departs from the simple form of (2.1) because of the dependence of the learning rate is on the previous weight value. High weight values discourage further learning. The present MOS learning rules are motivated by the desire to achieve low synaptic circuit complexity in VLSI neural networks, rather than being derived from established theoretical principles. Also, weight saturation is likely a feature of biological synaptic learning.

2.4 Circuit Layout and Performance of CMOS Synapses

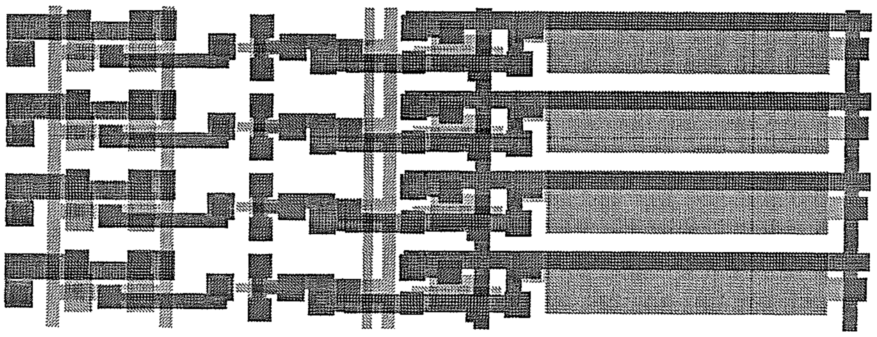

Figure 2.4 *Four CMOS synapses with in situ* MOS learning rules.

Fig. 2.4 presents a layout of four synapses of the type shown in Fig. 2.3. In $1.2\mu m$ CMOS the dimensions of a single synapse are $115{\times}11\mu m^2$, so a neural network chip of $1cm^2$ area could contain approximately 70 000 synapses, assuming that the synapses dominate the chip area. SPICE simulations of these layouts have been performed using level 3 transistor models. The multiplication $aV_j(V_C - V_T)$ (transistor M_1), representing the contribution I_{ij}

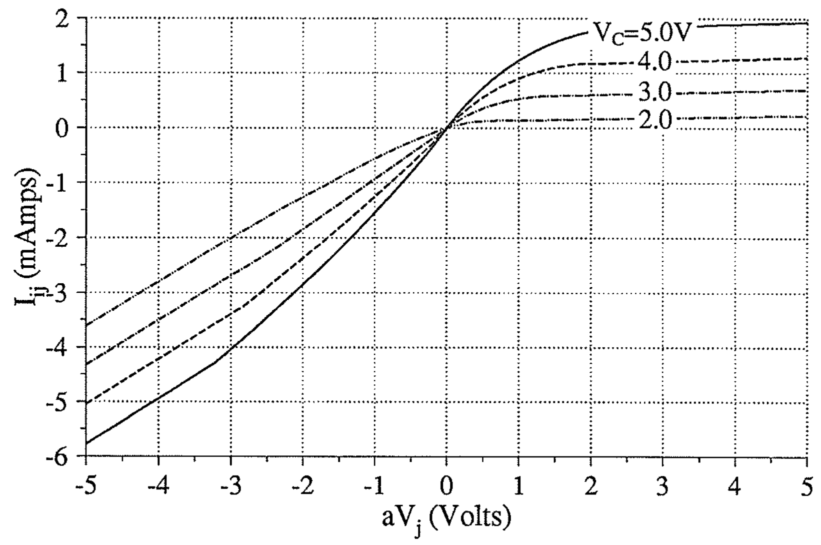

Figure 2.5 *Multiplication of aV_j by V_{Cj} in transistor M_1 of Fig. 2.3.*

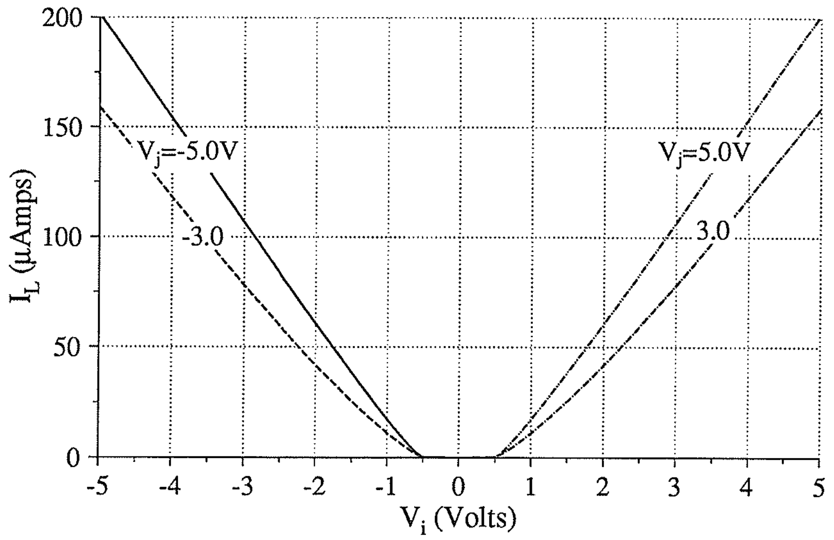

Figure 2.6 *Current I_L from (M_3, M_4), which performs Hebbian correlation of V_i and V_j.*

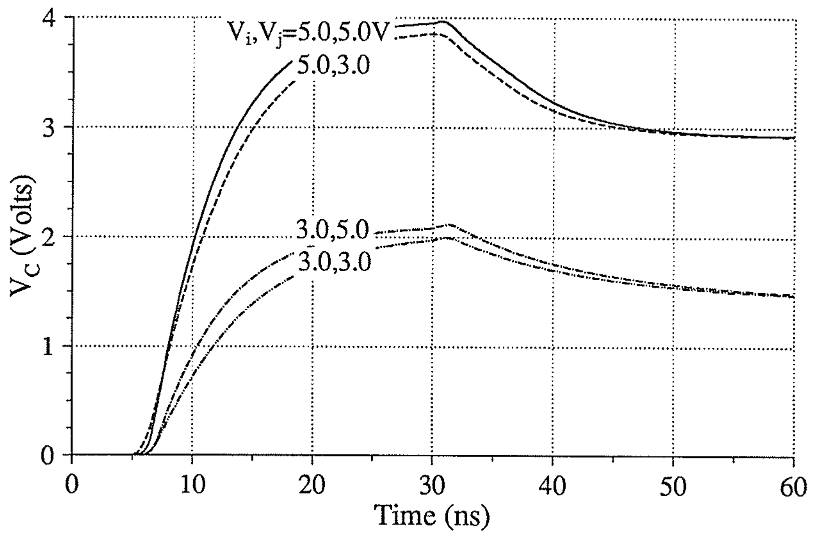

Figure 2.7 *Voltage on capacitor C due to current from* (M_3, M_4)*, corresponding to learning various positive correlations between* V_i *and* V_j*. At* $t = 32ns$*, learning is turned off, and* V_C *discharges through* M_2*.*

from synapse j to the total current into neuron i, is shown in Fig. 2.5. For $|aV_j| < 0.5V$ this circuit provides a respectable approximation to an analog multiplication. Fig. 2.6 presents the approximation to the multiplication required for Hebbian weight changes, or learning. This figure corresponds to the case $V_C = 0$ and therefore represents the maximum learning rate. Note that in this case we cannot employ the above trick of driving the learning from aV_i because we require large signal voltage excursions at the output of the (M_3, M_4) pair. There is a pronounced departure from linear multiplication near the origin, so small correlations between V_i and V_j are ignored by this circuit. The effects of weight saturation are clearly observed in Fig. 2.7. As discussed in the previous section, this represents the major departure from the traditional analytical form of Hebbian learning given by (2.1).

Weight decay has been greatly exaggerated in this circuit by employing a small R (implemented using M_2 in Fig. 2.3) in order to be able to observe its influence in Fig. 2.7. Actual implementations of neural network models would employ weight decay rates which would not be discernible in Fig. 2.7. This is readily achieved with the circuit of Fig. 2.3. If extremely low decay rates are desired (to remember large training sets) the chip could be cooled (perhaps

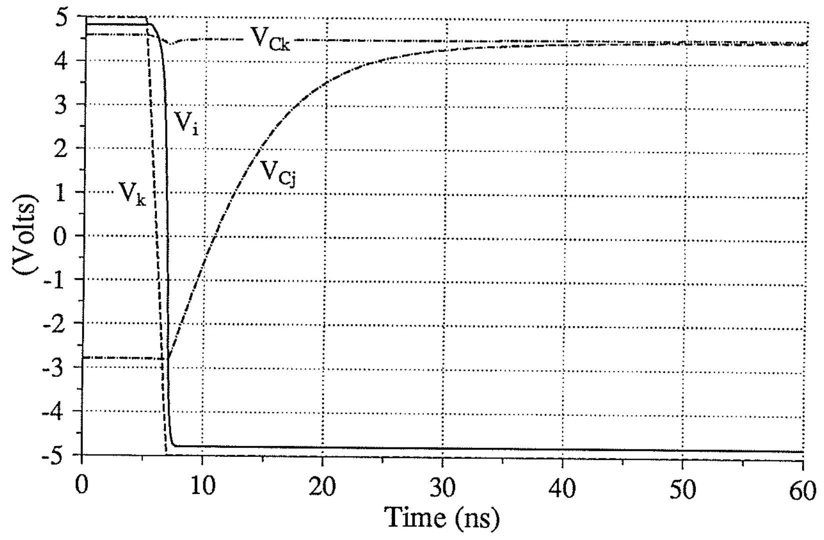

Figure 2.8 *Dynamic operation of two synapses j and k driving neuron i.*

even to cryogenic temperatures such as liquid nitrogen 77K) in order to reduce p-n junction leakage currents. Finally, in Fig. 2.8 we show an example of the dynamic behavior of a simple neural circuit of two synapses j and k driving one neuron i. V_j is negative throughout the simulation. V_i and V_k have both initially been high for a considerable time and the weight of the k synapse V_{Ck} has thus achieved its maximum value. V_k is then switched to its extreme negative value, and since V_{Cj} is negative, (ie. $W_{ij} = 0$, so the j^{th} neuron's state does not affect neuron i) neuron output V_i follows V_k and also swings negative. Since V_i and V_j are now both at their extreme negative values, they are fully correlated, and V_{Cj} now begins to rise, eventually reaching its saturation value near 5V. V_{Ck} began to decay when V_k was set to $-5V$, but as V_i soon followed, V_i and V_k were again correlated, and V_{Ck} returned to its former value.

2.5 Discussion

On the basis of the above simulations it appears that a mechanism consistent with the spirit of Hebbian learning is operating properly at the level of single synapses and neurons. It should be emphasized however that we do not have a theory at the systems level for the behavior of networks whose synapses employ these MOS learning rules. Depending upon the architecture of the network (feed-forward, feed-forward with crosstalk, or fully-connected feedback network) and the learning mode (supervised or unsupervised) the objective function to be optimized by the learning process will differ. For example in feed-forward supervised nets one minimizes a sum of squared errors by gradient descent [Rumelhart-2], whereas in an unsupervised feed-forward net with crosstalk one may maximize mutual information between several neighboring hidden units [Hinton-2]. Different measures are optimized by fully-connected supervised mean field networks [Hinton-1] and by unsupervised linear networks [Linsker] even though both of these latter cases employ versions of Hebbian learning rules at the synapses.

In this work we have only briefly discussed the issue of weight decay or forgetting rates. Forgetting rates determine the number of training cases remembered by the capacitor charge, and are dependent upon the number of weights that are mutually dependent during the learning process. For unsupervised learning based on optimization of local objective functions, the desired forgetting rates are expected to be larger than in supervised networks with global error signals during training. In the supervised case all the weights in the network are mutually dependent. The forgetting rate will normally be orders of magnitude lower than the learning rate so that many past training cases (on the order of the number of weights) can be remembered. To control the learning rate, one can adjust the *length/width* ratio L/W of the transistors M_3 and M_4 in Fig. 2.3. Learning rates are inversely proportional to both L/W and to the capacitance C so that by increasing L/W and C one can decelerate the learning process to the desired rate. One could alternatively introduce a series resistor in Fig. 2.3 to reduce the learning rate. All of these adjustments increase the silicon area per synapse.

One of the limitations of the present approach to *in situ* learning circuits is the inherent volatility of capacitive weight storage. Synaptic weights represent the knowledge of the network, and the network must be continuously exposed to relevant inputs or training data. Irrelevant or null inputs would otherwise corrupt the weights. One cannot simply disconnect the power supply and have the network retain its present state. To circumvent this problem either a refresh mechanism could be incorporated as in dynamic RAM or nonvolatile storage may be employed.

2.6 Summary

The suitability of VLSI-efficient learning algorithms based on the above MOS learning rules, as compared to previous analytical forms of Hebbian learning, can only be ascertained by simulation of a range of network models employing these synapses attempting a variety of learning tasks. These simulation studies are important in view of the potential benefits from reductions in silicon area in the realization of synaptic circuits, since the area of these synapses determines the VLSI complexity of the neural network itself. Hebbian learning has become popular in neural network research because it has achieved success in associative tasks and because it represents a simple mathematical approximation to what is believed to underlie biological neural systems. Recent work by Brown and his colleagues has established the presence of Hebbian synapses in the hippocampus of mammalian brains [Zador]. We believe it to be worthwhile to search for learning algorithms which achieve approximations to classical Hebbian rules based upon the simplest silicon implementations even though the corresponding mathematical models may be more complex.

CHAPTER 3

Bipolar Synaptic Learning Circuits

3.1 Introduction

In this chapter we describe a set of analog circuits which we have used to implement an ANN with bipolar weights and neuron activations. These circuits are considerably more complex than those developed in the previous chapter, and implement better approximations to the computational functions required by ANN theory. These circuits were developed because we wanted to implement existing ANN models in analog CMOS; ANN models which we were interested in implementing would have required extensive modifications to be compatible with the highly approximate computation of the circuits presented in the previous chapter. Furthermore, due to time constraints, we chose circuit designs which were robust and therefore likely to function properly the first time that they were fabricated. Of course the analog circuits presented in this chapter are still far more compact than their digital counterparts, so the computational benefits of analog computation have not been lost.

We have designed, fabricated and tested a fully analog neural network architecture with *in situ* Hebbian learning at each synapse. Synaptic weights are stored as voltages on capacitors. Below, we begin by describing the ideal neural network model which our analog VLSI implementation approximates. Next, our system architecture is discussed, followed by a detailed analysis of the behavior of the components from which the system is constructed. The ramifications of using imperfect components is analyzed, followed by a description of chip test results.

3.2 Ideal Neural Network Model

Our circuits implement a common neural network model, where both synaptic weights and neuron activations can take on values in the range $[-V, V]$. Network operation is governed by the following equations:

$$V_i = f(\sum_j W_{ij} V_j) \tag{3.1}$$

$$\frac{dW_{ij}}{dt} = AV_i V_j - BW_{ij} \tag{3.2}$$

where V_i and V_j are the current activations of the i^{th} and j^{th} neurons, W_{ij} is the weighting factor determining the effect that the j^{th} neuron has on the i^{th} neuron's activation, and A and B are small constants. $f(\cdot)$ is a sigmoidal saturating non-linear function.

The current activation of the i^{th} neuron is computed from a weighted summation of the current activations of all neurons which synapse on neuron i (equation 3.1). Network learning, or adaptation, is governed by (3.2), a common form of the Hebbian learning rule [Kohonen-1]. According to the first term in (3.2), when the activations of the i^{th} and j^{th} neurons are both positive or both negative, (ie. they are correlated), the weight W_{ij} will slowly increase over time, and when they have opposite signs (ie. they are anti-correlated), W_{ij} will decrease. The second term in (3.2) is a weight decay term, which causes all weights to decay towards zero over time.

Note that these two equations governing network behavior operate on very different time scales -- network learning (3.2) typically occurs at $\frac{1}{100}th$ to $\frac{1}{10000}th$ the rate of network operation, governed by (3.1). This makes intuitive sense, because the values of network weights represents the aggregate effect of many hundreds or thousands of network activations.

3.3 Analog CMOS Implementation

The circuit of Fig. 3.1 implements our analog CMOS approximation of the ideal neural network described by (3.1) and (3.2). In a typical neural network, many copies of the synaptic circuit (left half of Fig. 3.1) are connected to each neuron circuit, one for each neuron which affects the i^{th} neuron. Thus, there can be as many as $N(N-1)$ synapses in an N neuron system, in the case of a fully connected network, and synaptic circuitry occupies most of the silicon area of a typical neural network chip.

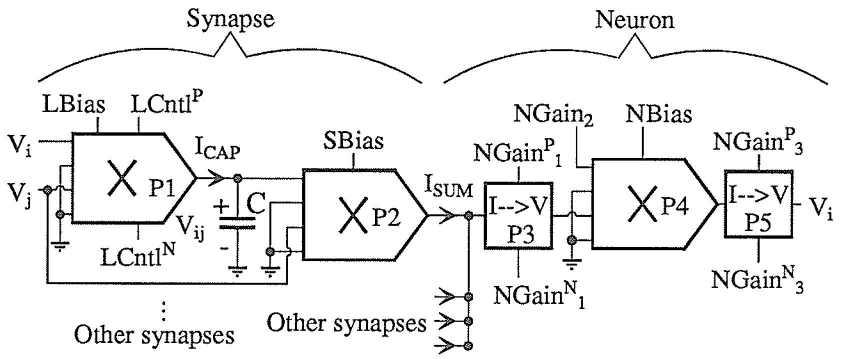

Figure 3.1 *Synapse and neuron circuits.*

The circuit of Fig. 3.1 implements (3.1) as follows. The synaptic weight W_{ij} is represented as the voltage V_{ij} on capacitor C, and the i^{th} and j^{th} neuron activations are represented by the voltages V_i and V_j respectively. The three multipliers, P_1, P_2 and P_4 multiply two differential input voltages, producing an output current proportional to the product of the inputs, independent of the voltage at the multiplier output. Thus the product $W_{ij}V_j$ in (3.1) is computed by P_2, and the summation operation is implemented as current summation at the input to P_3. P_3 converts the resulting net current into a voltage signal which is then input to the neuron multiplier P_4. The saturating non-linearity $f(\cdot)$ in (3.1) is produced using the saturating behavior of the neuron multiplier P_4, with signals *NGain* and *NBias* controlling neuron gain.

Hebbian learning (3.2) is implemented by P_1:

$$\frac{dW_{ij}}{dt} \propto \frac{dV_{ij}}{dt} = \frac{I_{CAP}}{C} \propto V_i V_j \tag{3.3}$$

since P_1 produces an output current I_{CAP} proportional to $V_i V_j$, independent of V_{ij}, the voltage on the capacitor C. The weight decay term in (3.2) has been set to zero ($B = 0$) in this circuit.

Since any practical neural network contains many hundreds or thousands of synapses, it is essential that the components of Fig. 3.1 be as simple as possible. For this reason, the components in Fig. 3.1 deviate considerably from the ideal case represented by (3.1) and (3.2). The behavior of these components is analyzed in detail below.

3.4 Circuit Components

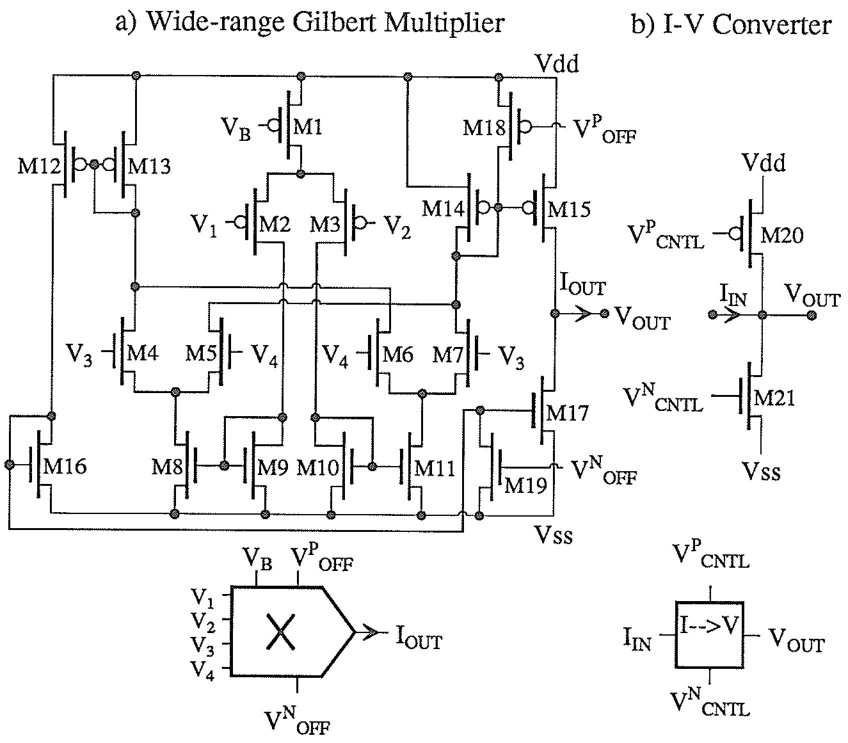

Figure 3.2 *Schematics of circuit components.*

In keeping with our goal to use the simplest and most robust circuit design possible, our neural network architecture is constructed from the components of Fig. 3.2 (with some minor specializations, depending on application).

3.4.1 Multiplier Circuit

The multiplier circuit is a variant of a wide-range Gilbert multiplier [Mead], implementing the operation $I_{OUT} = a(V_1 - V_2)(V_3 - V_4)$. Note that unlike in [Mead], transistors in our multiplier circuit are operating above threshold ($V_{GS}>V_T$). The six transistors which perform the basic multiplication operation (Fig. 3.2, M_2 - M_7) are three source-coupled differential pairs, of the form shown in Fig. 3.3. Using the simplified square-law equation $I_{DS} = \frac{\beta}{2}(V_{GS}-V_T)^2$, where $\beta = \mu C_{OX}\frac{W}{L}$ to represent the behavior of an N-channel transistor in saturation,

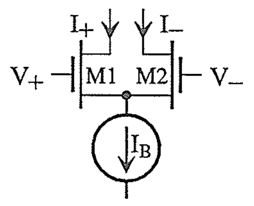

Figure 3.3 *MOS differential pair.*

$$I_+ = \frac{2I_B + \sqrt{\beta^N} V_X \sqrt{4I_B - \beta^N {V_X}^2}}{4} \tag{3.4}$$

$$I_- = \frac{2I_B - \sqrt{\beta^N} V_X \sqrt{4I_B - \beta^N {V_X}^2}}{4} \tag{3.5}$$

$$I_+ - I_- = \frac{\sqrt{\beta^N} V_X \sqrt{4I_B - \beta^N {V_X}^2}}{2} \tag{3.6}$$

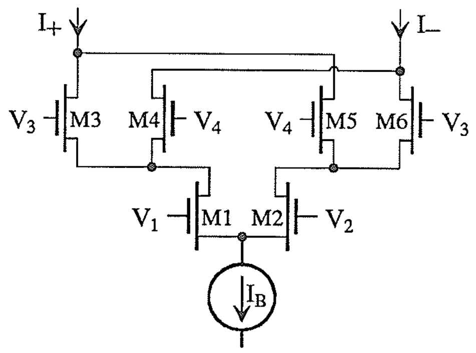

Figure 3.4 *Basic MOS Gilbert circuit.*

where $V_X = V_+ - V_-$, provided $|V_+ - V_-| < \sqrt{\frac{2I_B}{\beta^N}}$. Equations 3.4, 3.5 and 3.6 may be used to derive an expression for the output current of the basic Gilbert multiplier of Fig. 3.4, which consists of three differential pairs:

$$\begin{aligned} I_{OUT} &= I_+ - I_- = (I_3 + I_5) - (I_4 + I_6) \\ &= (I_3 - I_4) - (I_6 - I_5) \end{aligned} \tag{3.7}$$

which is the difference between the output currents of two differential pairs of the form (3.6), where the current source I_B is replaced by expressions (3.4) and (3.5), representing I_1 and I_2. Thus,

$$\begin{aligned} I_{OUT} = \frac{\sqrt{\beta^N} V_Y}{2} &\left(\sqrt{\frac{1}{2}\left[\sqrt{4I_B - \beta^N V_Z{}^2} + \sqrt{\beta^N} V_Z\right]^2 - \beta^N V_Y{}^2} \right. \\ &\left. - \sqrt{\frac{1}{2}\left[\sqrt{4I_B - \beta^N V_Z{}^2} - \sqrt{\beta^N} V_Z\right]^2 - \beta^N V_Y{}^2} \right) \end{aligned} \tag{3.8}$$

where $V_Y = V_3 - V_4$ and $V_Z = V_1 - V_2$. The multiplier circuit of Fig. 3.2 also consists of three differential pairs, connected in the same way as in the basic Gilbert multiplier. The difference between the two circuits is that current mirrors (M_8, M_9, and M_{10}, M_{11}) have been placed between the two stages to allow the restriction $max(V_3,V_4) > min(V_1,V_2)$ to be relaxed. Assuming that these current mirrors are ideal (ie. $I_{IN} = I_{OUT}$), then the transfer function remains unchanged. Introduction of the current mirrors also requires that the two stages be implemented with complementary transistors -- in this case the first stage is implemented with p-channel transistors, and the second stage with n-channel transistors. Finally, the current source I_B is implemented with transistor M_1 (operating in saturation), and three more current mirrors (M_{12} - M_{17}) are used to perform the subtraction $I_+ - I_-$, and to make I_{OUT} independent of V_{OUT}.

The multiplier transfer function is given by:

$$\begin{aligned} I_{OUT} = \frac{\sqrt{\beta^N} V_Y}{2} &\left(\sqrt{\frac{1}{2}\left[\sqrt{4I_B - \beta^P V_Z{}^2} + \sqrt{\beta^P} V_Z\right]^2 - \beta^N V_Y{}^2} \right. \\ &\left. - \sqrt{\frac{1}{2}\left[\sqrt{4I_B - \beta^P V_Z{}^2} - \sqrt{\beta^P} V_Z\right]^2 - \beta^N V_Y{}^2} \right) \end{aligned} \tag{3.9}$$

and when $V_1 - V_2$ and $V_3 - V_4$ are small,

$$I_{OUT} \approx \frac{\sqrt{\beta^N} V_Y}{2} \left[\sqrt{\frac{1}{2}\left[\sqrt{4I_B - \beta^P V_Z{}^2} + \sqrt{\beta^P} V_Z\right]^2} - \sqrt{\frac{1}{2}\left[\sqrt{4I_B - \beta^P V_Z{}^2} - \sqrt{\beta^P} V_Z\right]^2} \right] \tag{3.10}$$

$$= \frac{\sqrt{\beta^N \beta^P}}{\sqrt{2}} V_Y V_Z$$

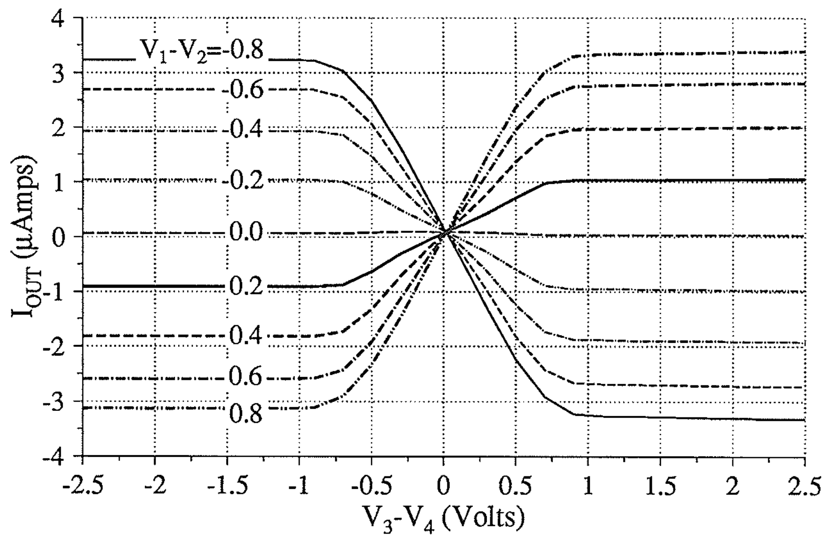

Figure 3.5 *I_{OUT} as a function of V_3–V_4 for V_1–V_2 from* $-0.8V$ *to* $0.8V$ *(3μm CMOS test chip, $Vdd = 2.5V$, $Vss = -2.5V$, $V_B = 0.9V$, data collected using HP4145A).*

Measurements from our test chip (Fig. 3.5) confirm the behavior predicted by (3.9) and (3.10): for small inputs the multiplier is nearly linear, with saturation occurring when $max(|V_1 - V_2|, |V_3 - V_4|) > 0.8V$. Fig. 3.6 shows that for V_{OUT} in the multiplier's operating range of $\pm 0.8V$, there is a variation in I_{OUT} of 3% of $max(I_{OUT})$. This is achieved by using long transistors ($L = 14\mu m$) to minimize channel-length modulation effects for the output current mirrors (M_{14} - M_{17}).

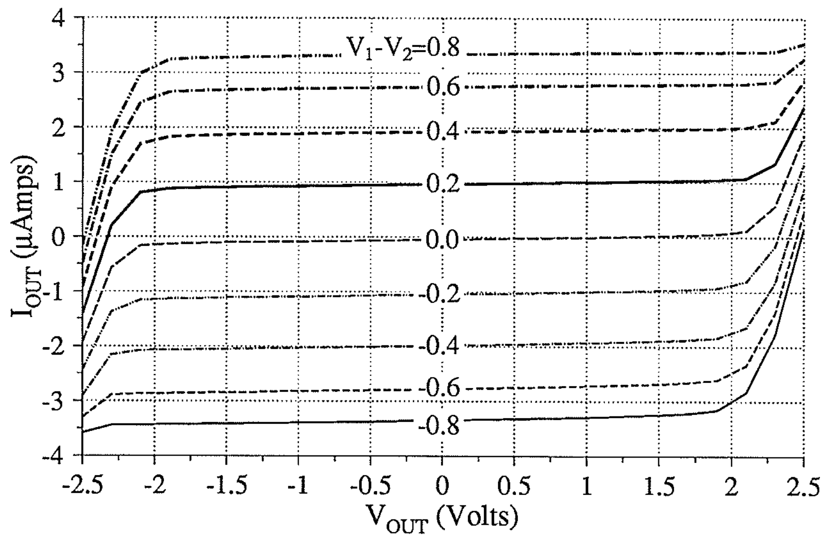

Figure 3.6 *I_{OUT} as a function of V_{OUT} for $V_3-V_4 = -2.5V$, V_1-V_2 from $-0.8V$ to $0.8V$ (3μm CMOS test chip, $Vdd = 2.5V$, $Vss = -2.5V$, $V_B = 0.9V$, data collected using HP4145A).*

The design of this multiplier circuit involved two important tradeoffs: low power consumption versus circuit speed, and wide linear input voltage swing versus good matching between identical multiplier circuits. An operating current in the $3\mu A$ range was chosen as a compromise between power consumption and circuit operating speed, giving a worst-case DC power consumption of $55\mu W/multiplier$, and a typical switching time of $70ns$. To achieve good matching between identical transistors, V_{GS} should be well above V_T, to minimize the effects of threshold variations. Since $I_{DS} \propto (V_{GS}-V_T)^2$, the sensitivity of I_{DS} with respect to V_T, $S_{V_T}^{I_{DS}} = \frac{-2V_T}{V_{GS}-V_T}$, so clearly good matching is impossible for V_{GS} near V_T. However, increasing $Vdd-V_B$ to achieve good matching between current sources (implemented with M_1) reduces the linear input range ($V_B \geq V_1, V_2 \geq -V_B$ is the linear range). A compromise $Vdd-V_B$ value of $1.6V$ ($V_T = 0.77V$) was chosen. Similar calculations were made to ensure that the input voltages to the current mirrors in this circuit are also well above threshold.

Increasing supply voltages is another way that the linear input swing can be improved: however, impact ionization effects limit the supply voltage of the $3\mu m$ CMOS technology that we are using to 5V.

3.4.2 I-V converter

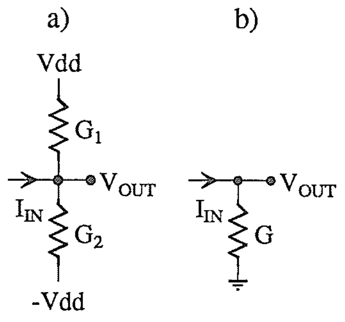

Figure 3.7 *Linear I-V converter. a) linear resistors to Vdd and Vss ; b) simplest case, when* $G_1 = G_2 = \frac{1}{2}G$

For our application, accurate current-to-voltage conversion is not necessary, so the simple circuit of Fig. 3.2b was chosen. This circuit performs the same operation as its linear equivalent (Fig. 3.7a). Here,

$$V_{OUT} = \frac{I_{IN}+Vdd(G_1-G_2)}{G_1+G_2} \tag{3.11}$$

(assuming $Vss = -Vdd$), and when $G_1 = G_2 = \frac{1}{2}G$ this circuit behaves like the simplest I-V converter, a resistor to ground (Fig. 3.7b).

Linear resistors require too much silicon area, so G_1 and G_2 are implemented using M_{20} and M_{21}. When $V^N_{CNTL} \approx Vdd$ and $V^P_{CNTL} \approx Vss$, M_{20} and M_{21} are operating in the triode region ($V_{DS}<V_{GS}-V_T$), an approximate expression for I-V converter behavior may be calculated, using $I_{DS} = \beta((V_{GS}-V_T)^2-V^2_{DS}/2)$. Assuming that the transistors are matched ($\beta^N = \beta^P = \beta$), and $Vss = -Vdd$,

$$V_{OUT} = \frac{I_{IN}+\beta Vdd(V^N_{CNTL}+V^P_{CNTL})}{\beta(V^N_{CNTL}-V^P_{CNTL}-2V_T)} \tag{3.12}$$

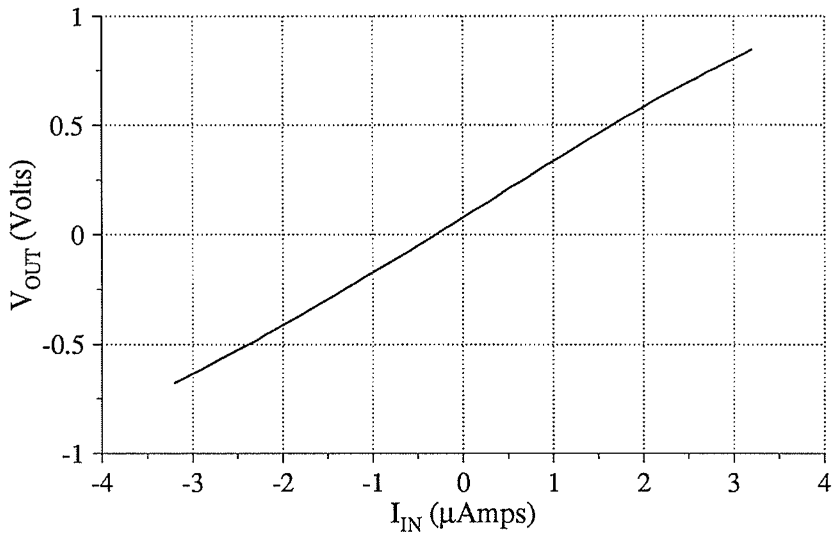

Figure 3.8 *Measured I-V converter performance (3μm CMOS test chip, $Vdd = 2.5V$, $Vss = -2.5V$, $V^N_{CNTL} = 2.50$, $V^P_{CNTL} = -2.27$, data collected using HP4145A).*

which is in the same form as (3.11) (note that V^P_{CNTL} is negative). In practise it is difficult to match M_{20} and M_{21}, but V^N_{CNTL} and V^P_{CNTL} can be adjusted to correct for transistor mismatch. Fig. 3.8 shows a typical I-V converter characteristic.

3.5 Effect of Imperfect Components

As mentioned previously, the reason that neural networks with learning capability are suitable for implementation using simple, low accuracy analog computational elements is that their architectures enable them to compensate for these imperfections. For example, assume that P_2 in Fig. 3.1 produces a larger than average I_{OUT} for a given weight value, V_{ij}. Simply by learning a slightly different weight, the circuit can compensate for substantial variations in P_2. Similarly, an offset introduced into the neuron output V_i by mismatches in P_3, P_4 and P_5 can also be cancelled by adjusting the synaptic weights of the system. As might be expected, neural networks can have critical components: our investigations using a modified version of these synaptic circuits to implement a Deterministic Boltzmann network [Hinton-1] demonstrate that architectures typically have certain operations which must be performed accurately.

Another important source of error is variation in temperature. As the neural network chip is operating, increasing temperature will alter component characteristics, perhaps non-uniformly across the chip's surface. Since our circuits are intended for neural networks which learn continuously, either from each input pattern or by interspersing training patterns between unclassified patterns, weights will change with temperature, reflecting the effect of temperature changes on each component in the circuit.

As the preceding discussion indicates, learning is not just important from the point of view of training a neural network to perform a certain task; it also serves as a form of slow feedback which allows the neural network to compensate for a wide variety of component imperfections. Qualitatively, learning feedback is performing the same correcting role in a neural network that a negative feedback path performs in a linear amplifier.

3.6 Implementation

The circuits described in sections 3.3 and 3.4 were fabricated using a $3\mu m$ double metal, P-well CMOS process (Northern Telecom CMOS3). In this combined analog/digital process, linear capacitors are created between heavily doped P-diffusion regions and polysilicon. Fig. 3.9 is a photomicrograph of a single synaptic cell, which implements the synaptic portion of the circuit of Fig. 3.1. The majority of the cell area is occupied by the two multipliers, with the remainder of the area occupied by control circuitry and the weight storage capacitor. For our test circuits, $C = 1.1pF$ (the capacitor is located in the lower right corner of the cell). Fig. 3.10 shows the layout of the neuron circuit. For this cell, layout efficiency was not an issue, because neuron cells occupy an insignificant fraction of total silicon area of a typical neural network chip.

Approximately 3 000 synaptic cells will fit on a $1cm^2$ die. Such a network can perform more than 6 000 multiplications simultaneously, giving a throughput of 6 billion analog multiplications per second (assuming a conservative $1MHz$ operating speed). These calculations demonstrate that analog computation is far more area-efficient than its digital equivalent. As mentioned previously, the key to using analog components for computation is that the system architecture must be tolerant of imperfect component behavior.

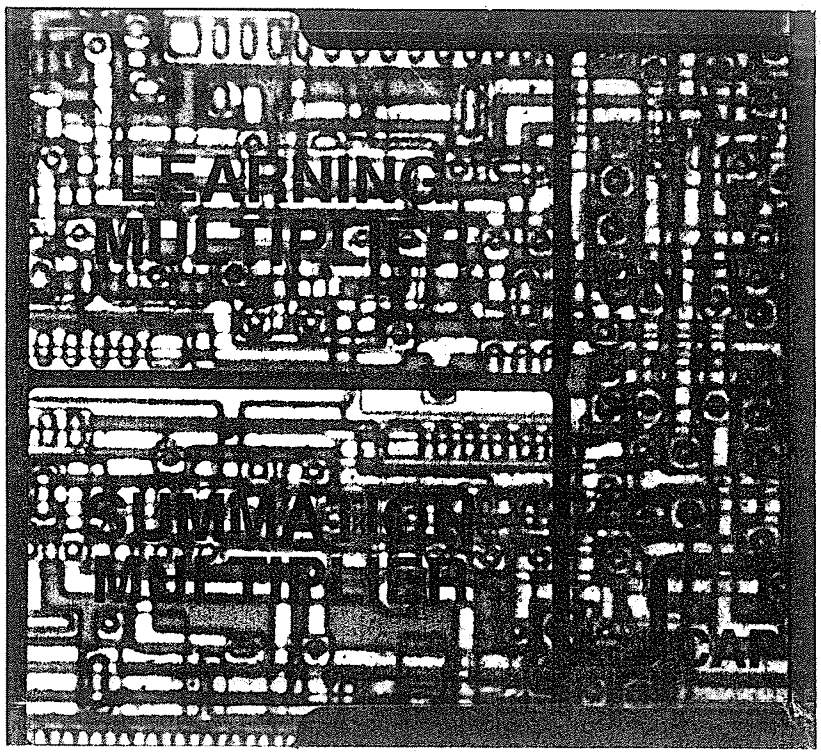

Figure 3.9 *Photomicrograph of synaptic circuit (*$32\,440\mu m^2$ *per synapse). Boundaries of subcircuits are indicated by broad, dark lines.*

To test our synaptic and neuron circuits in a complete neural network system, a fully-connected (each neuron synapses with every other neuron) Hopfield-type configuration was chosen. We fabricated two neural networks: a 3 neuron, 6 synapse network (Fig. 3.11), and a larger 25 neuron 600 synapse network containing more than 28 000 transistors (Fig. 3.12). The diagonal of missing synaptic cells in Figs. 3.11 and 3.12 arises from the fact that neurons do not synapse with themselves ($W_{ii} = 0$). In the following section, test results for these networks are reported.

3.7 Circuit Performance

The DC behavior of circuit components was discussed in section 3.4 and compared with simplified analytical expressions for component behavior. In this section the results of a variety of transient tests (both simulated and measured) are reported. In some cases, such as

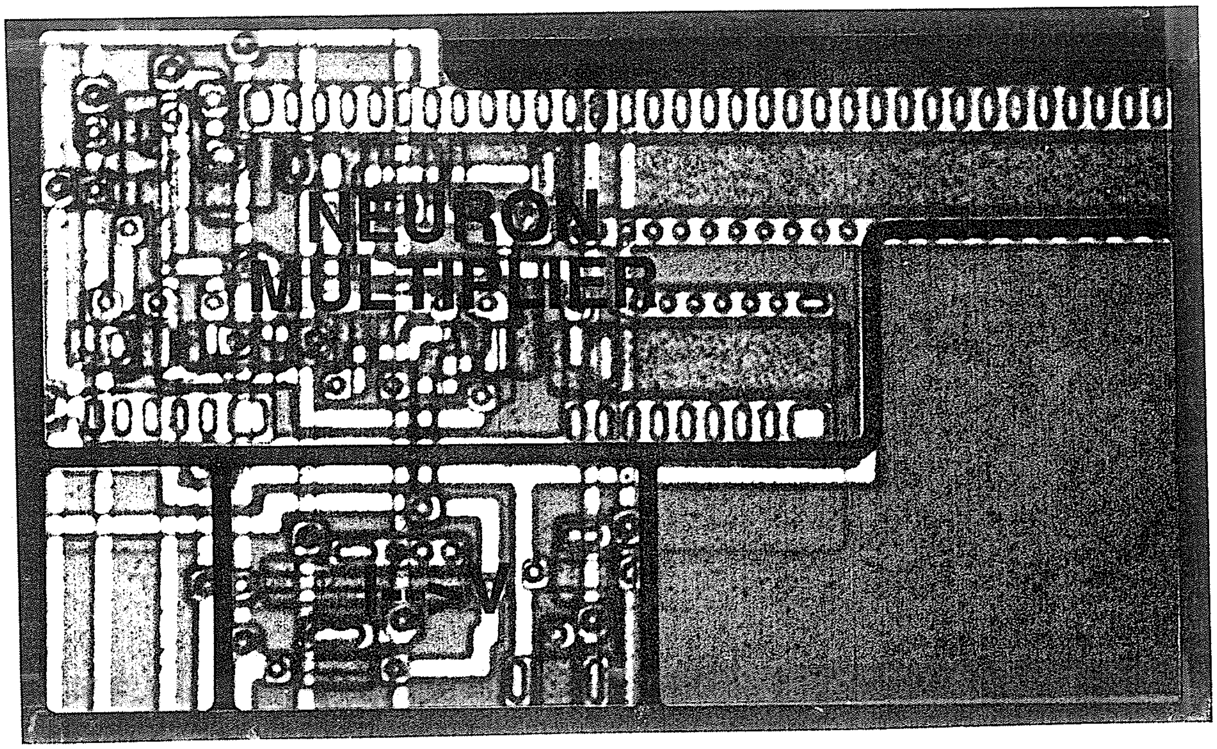

Figure 3.10 *Photomicrograph of neuron circuit* (44 430μm^2 *per neuron*). *Boundaries of subcircuits are indicated by broad, dark lines.*

the learning test below, simulation data, rather than measured data is used, because attaching measurement equipment alters circuit behavior too drastically. For example, the $10M\Omega$ input impedance of a typical oscilloscope will drain the weight storage capacitor in $10\mu s$. By presenting simulation results, as well as measurements, a clearer picture of circuit operation is obtained. Table 3.1 summarizes signal characteristics. Note that voltage signals (V_i, V_j, V_k, V_{ij}, V_{ik}, and V_{SUM}) may saturate at the supply voltages $\pm 2.5V$, but their active range is as stated in Table 3.1.

3.7.1 Three Neuron Learning Test

The test setup consists of the i^{th} neuron of a neural network, with two synapses connected to it, from neurons j and k (Fig. 3.13). The learning behavior of this circuit is shown in Fig. 3.14 and Table 3.2.

At $t = 0$, the system is stable, since V_i and V_j are anti-correlated and $V_{ij} = Vss$ (maximum negative weight value); V_i and V_k are correlated, and $V_{ik} = Vdd$ (maximum positive

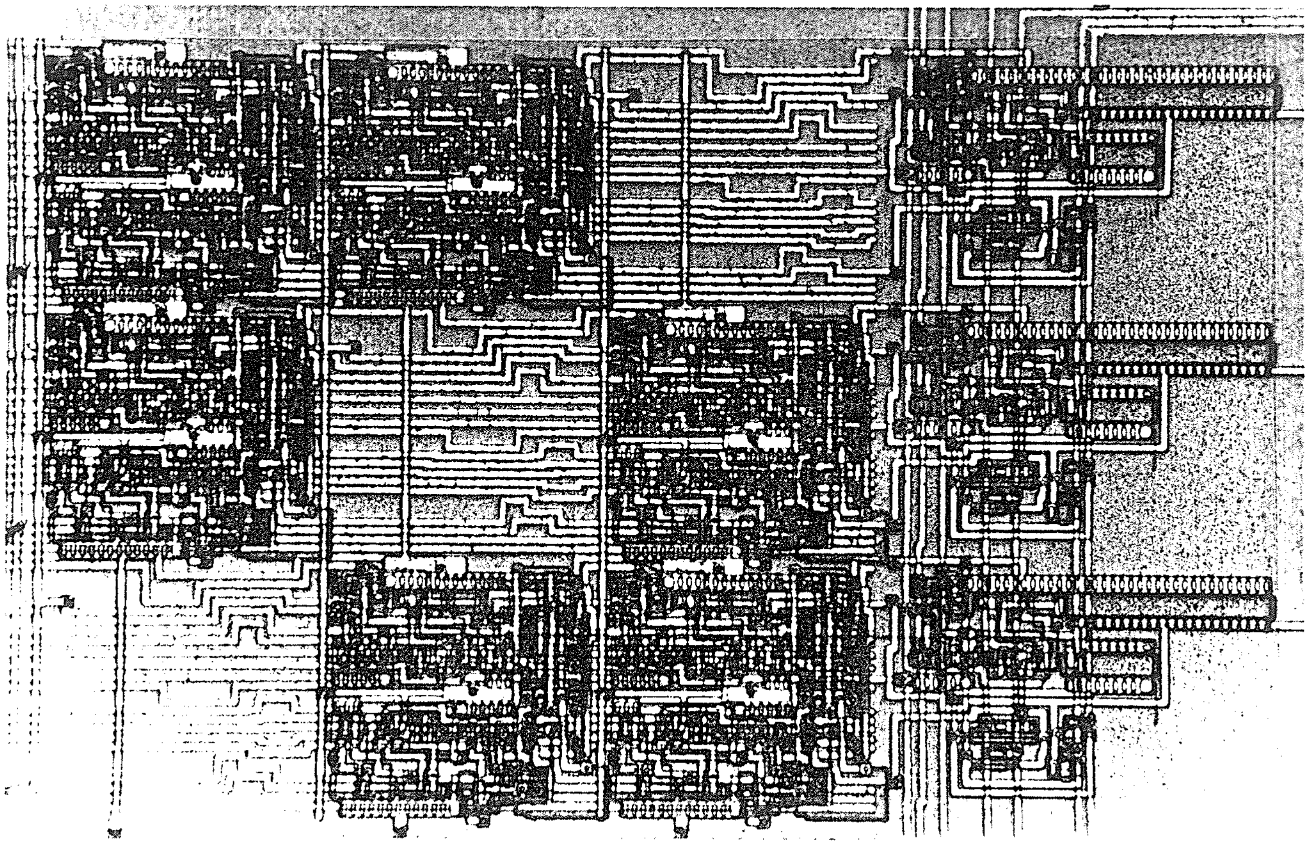

Figure 3.11 *Photomicrograph of 3 neuron, 6 synapse network ($1.0 \times 0.5mm^2$).*

weight value). At $t = 0.9\mu s$, we reverse the signs of inputs V_j and V_k. As a result, V_i also switches sign, because the circuit has learned that V_i is correlated with V_k, and anti-correlated with V_j. Thus, the circuit is using the relationships it has learned between V_i, V_j and V_k to *predict* a new value for V_i. At $t = 2.9\mu s$, V_j is set to a moderate negative value. Since V_i and V_j are anti-correlated, V_i begins to increase. However, V_i and V_k are correlated, and V_k is still negative, so the net result is that V_i settles to a smaller negative value ($| V_k | > | V_j |$, so $V_k W_{ik} + V_j W_{ij} < 0$, and hence $V_i < 0$). Now V_i and V_j are weakly correlated, and V_{ij} gradually becomes positive as this new condition is learned. Note that for purposes of illustration, a higher than typical learning rate was used: normally adaptation to new relationships between neuron activations occurs much more slowly than shown in Fig. 3.14.

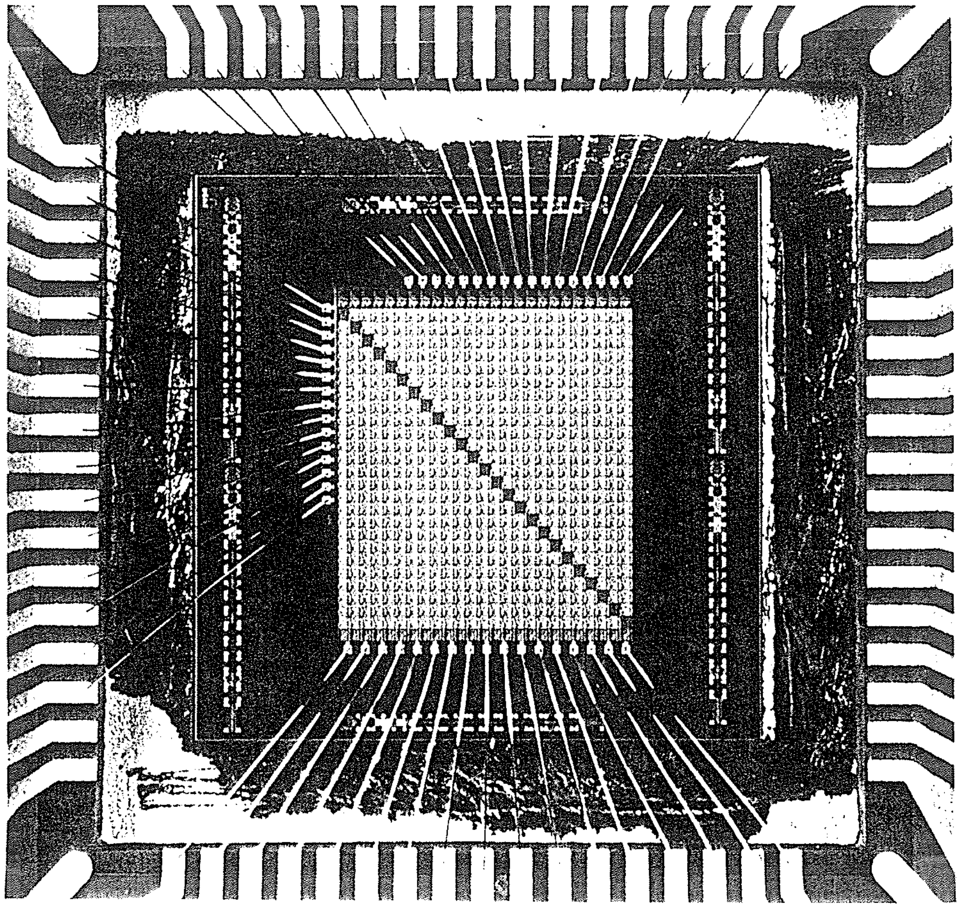

Figure 3.12 *Photomicrograph of 25 neuron, 600 synapse network (5.5 × 4.6mm^2).*

This learning test demonstrates that this circuit behaves in a way that is consistent with (3.1) and (3.2), the equations describing an ideal neural network with Hebbian learning. Details, such as the precise form of the product $W_{ij}V_j$ may differ, but the basic operation of learning correlation and anti-correlation, and performing a weighted summation are implemented with sufficient accuracy by our circuits. This learning test was also performed using the 3-neuron neural network on our test chip (Fig. 3.11), yielding the same results as the simulation discussed above.

Table 3.1 *Summary of signals.*

Signal	Type	Range	Function
V_i, V_j, V_k	voltage	$[-0.8V, 0.8V]$	neuron activation
V_{ij}, V_{ik}	voltage	$[-0.8V, 0.8V]$	synaptic weight
V_{SUM}	voltage	$[-0.8V, 0.8V]$	I_{SUM} after I-V converter
I_{SUM}	current	$[-3.5\mu A, 3.5\mu A]$	sum current

Table 3.2 *Learning test.*

	t=0.0	t=0.9μs	t=2.9μs
V_i	+	−	
V_j	−	+	$-\frac{1}{2}$
V_k	+	−	
V_{ij}	−		$\rightarrow +$
V_{ik}	+		

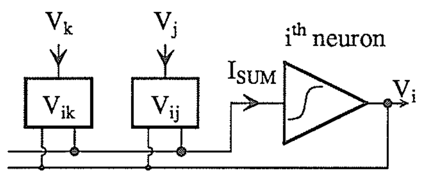

Figure 3.13 *Learning test configuration.*

3.7.2 Dynamic Circuit Tests

A number of dynamic circuit tests were performed to characterize the behavior of our circuits. Fig. 3.15 illustrates the behavior of the 3-neuron network when V_k is held at $0.4V$, and V_j is switched at $25kHz$. Fig. 3.15 shows that V_i follows V_j (the network has learned that V_i

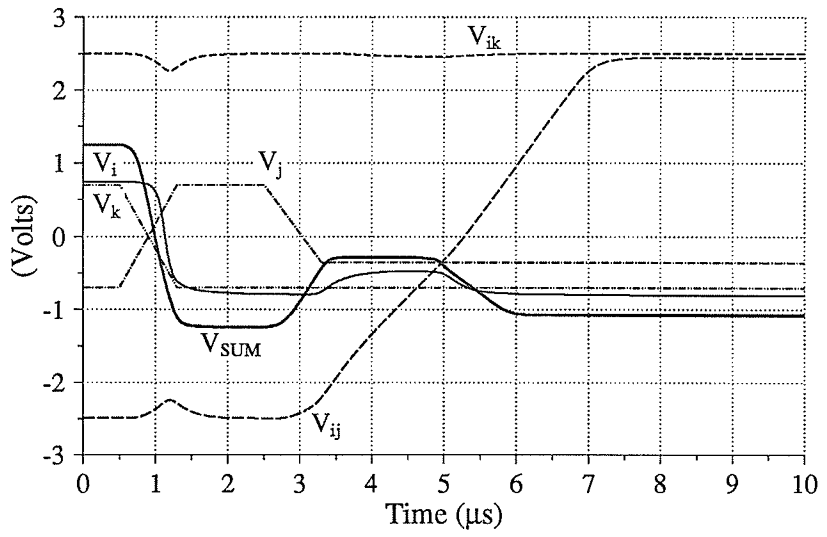

Figure 3.14 *Learning test results (HSPICE MOSFET model level 3 simulation).*

and V_j are correlated). However, because V_k is held at $0.4V$ while V_j switches (V_i and V_k are also correlated), V_i does not saturate at its maximum negative value (about $-1.0V$), because V_k remains positive while V_j is negative; thus the value of V_i generated by the network reflects the influence of both V_j and V_k. The network could also have learned an anti-correlation between V_i and V_j, and V_i and V_k, with similar results. In our test setup, the relation learned between V_i and V_j can be changed by briefly forcing V_i either high or low. If V_i is released when V_j has the same sign as V_i (either positive or negative), then a correlation is learned; otherwise an anti-correlation is learned. The same comments apply to the relation between V_i and V_k. Note that the V_i rise and fall times seen in Fig. 3.15 reflect the large capacitance that the neuron output has to drive in our test setup, rather than system learning dynamics.

Fig. 3.16 shows the effect of a high learning rate. In this test, V_j and V_k both switch at $25kHz$, and therefore V_i should also switch each time V_j and V_k switch. However, because the learning operation (3.2) is *faster* than system dynamics (3.1), at $t = 5\mu s$ when V_j and V_k switch, V_i begins to follow, but learns a new relation (anti-correlation) before V_i rises above $0V$. Notice that the learning rate chosen in Fig 3.16 is just slightly too high, so the network operates correctly when V_j and V_k switch at $t = 25\mu s$ and $t = 45\mu s$, and fails again at

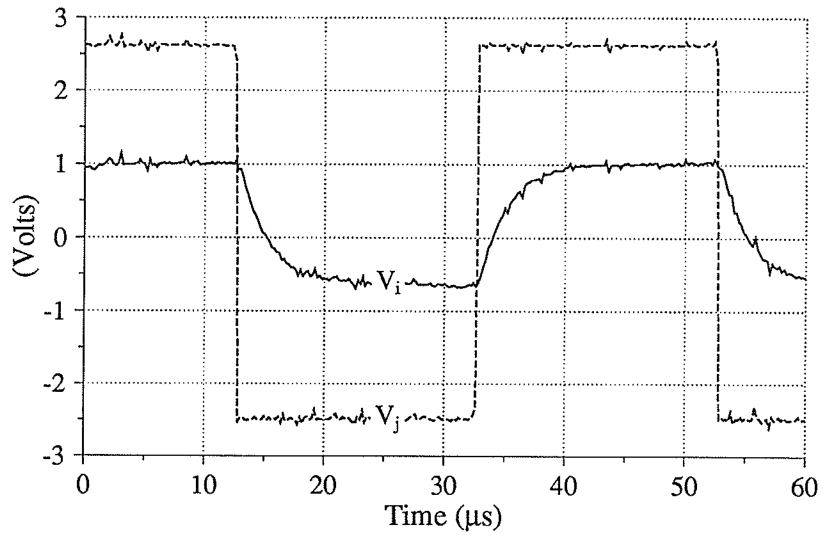

Figure 3.15 *Dynamic test of 3-neuron network, response of V_i when $V_k = 0.4V$, V_j switching at 25kHz (test data generated using ASIX-2 IC tester, analog data collected using Tektronix 2232 digital storage oscilloscope).*

$t = 65\mu s$. Noise determines whether V_i rises above $0V$ before V_{ij} and V_{ik} change sign (in Fig. 3.16 V_i rises to almost $0V$ before falling again in the cases that fail).

3.7.3 Power Dissipation

Power dissipation is a critical issue in VLSI systems, limiting both speed and integration density. Since neural networks have to be large to be useful, circuit designs with large power requirements are not practical. Synaptic circuitry occupies most of the chip area of a typical neural network, so power dissipation requirements may be estimated by considering synaptic power consumption. The worst-case DC power consumption of our synaptic circuit is $110\mu W$ (two multipliers at $55\mu W/multiplier$), giving a chip power dissipation of $0.34W/cm^2$ ($3\mu m$ CMOS), which is achievable using conventional air cooling.

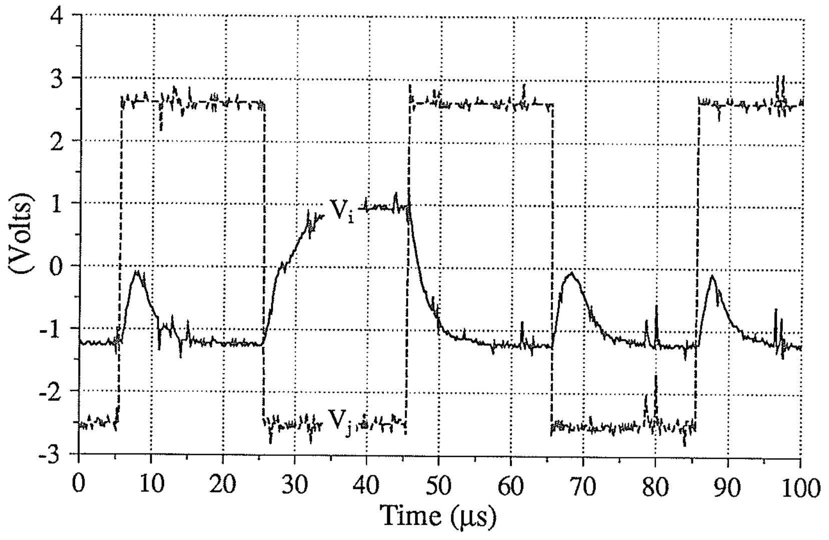

Figure 3.16 *Dynamic test of 3-neuron network, effect of high learning rate (test data generated using ASIX-2 IC tester, analog data collected using Tektronix 2232 digital storage oscilloscope).*

3.8 Discussion

Although the synaptic circuits described in this chapter have been incorporated into a fully-connected Hopfield-type architecture to evaluate their performance in a complete system, they are suitable for a variety of other ANN architectures as well. In assessing the suitability of our analog circuits for a particular learning model or architecture, the most important consideration is whether the architecture is tolerant of low-accuracy computation. This tolerance may be achieved in a number of ways, with weight adaptation (learning) being the most effective. As was discussed in previous sections, an ANN can compensate for a wide variety of component flaws by learning appropriate weight values.

3.8.1 Back-propagation ANNs

Studies have shown that the successful and popular back-propagation (BP) ANN [Rumelhart-2] requires between 8 and 16 bits of accuracy for gradient descent weight adaptation. Although it is difficult to achieve this level of accuracy using analog computation, this accuracy may only be required to permit small weight changes during learning, as is the case with contrastive Hebbian learning (see Chapter 5). Support for this point of view comes from informal discussions with other researchers, whose investigations seem to indicate that lower weight resolutions are sufficient for solving many classification problems. Thus, variants of our analog circuits may well be suitable for implementing back-propagation ANNs.

3.8.2 Self-Organizing Perceptual Map ANNs

There are a variety of ANNs which are well suited to the circuits described in this chapter. Generally, they use learning rules that depend only on local interactions, which simplifies communication in a VLSI system. Below, representative approaches which use *unsupervised* learning are discussed.

Linsker describes a self-organizing network which learns to extract features from an image in a way that is reminiscent of feature extraction in the first stages of the mammalian visual system [Linsker]. The self-organizing principle used by this system is based on information theory: the network connections (weights) develop such that the amount of information that is preserved at each processing stage is maximized. Linsker's system uses *linear* neurons and no feedback, as his goal was to use as simple a system as possible to model early vision.

The equations governing system operation are:

$$V_i = A + B\sum_j W_{ij} V_j \tag{3.13}$$

$$\Delta W_{ij} = C V_i V_j + D V_j + E V_i + F \tag{3.14}$$

where A, B, C, D, E and F are arbitrary constants, $C > 0$ and weights W_{ij} are constrained to lie in the range $-V \le W_{ij} \le V$. The similarity between (3.13) and (3.14), and (3.1) and (3.2) is obvious, and thus implementation of this ANN with our circuits should pose no major problems. Nonlinear extensions of Linsker's network could also be implemented with our analog circuits. This is an important consideration, because practical ANNs must have nonlinear information processing: strictly linear systems are only suitable for a restricted class of applications.

Linsker found that after passing through two layers of this linear network, governed by equations (3.13) and (3.14), center-surround cells develop, and after three layers, orientation-selective cells. *Center-surround* [Kandel-1] cells respond to bright spots surrounded by a dark background, and *orientation-selective* cells respond to bright bars with a particular orientation.

A further modification of Linsker's approach, following [Oja], results in a network which can perform a form of *principle-component analysis*, a method used in statistics for extracting unanticipated structure from high-dimensional data sets. Oja found that a Hebb-type learning rule which maximizes the output variance, subject to the constraint $\sum W_{ij}^2 = 1$ implements principle-component analysis.

3.8.3 Self-Organizing Feature Map ANNs

Kohonen has done considerable work in the area of unsupervised ANN learning, with networks called self-organizing feature maps [Kohonen-2]. Here, we discuss Kohonen's work with respect to a particular application, an experimental speech recognition system. The ANN architecture employed consists of a 2-dimensional hexagonal arrangement of neurons, each of which has an adaptive synaptic connection to an external input. The network has lateral feedback synaptic connections between neurons in the network, with the strength of the feedback connections following a ''Mexican hat'' [Kohonen-2] function, in which connection strengths decrease with distance from the objective neuron, alternating between excitatory and inhibitory. Self-organizing feature maps are a form of competitive learning network, in which the network activity develops into clusters or ''bubbles'' as the lateral feedback connections suppress activity in neurons near strongly activated neurons. Thus, neurons ''compete'' with one another, and the winner suppresses its neighbors through lateral inhibition in a ''winner take all'' fashion.

The following equations describe network operation:

$$V_i = f\,(\sum_j W_{ij} V_j) \tag{3.15}$$

$$\frac{dW_{ij}}{dt} = AV_i V_j - \beta(V_i) W_{ij} \tag{3.16}$$

where $\beta(\cdot)$ is a scalar function with a Taylor expansion in which the constant term is zero, $f\,(\cdot)$ is a sigmoidal function, and A is a positive constant. Typically, $\beta(\cdot)$ has a simple form that would be straightforward to implement in analog CMOS. It is apparent from (3.15) and (3.16) that this ANN's weight learning rule is a variation on Hebbian learning, and is therefore realizable using the analog CMOS circuits described above. Developing circuits to implement the lateral feedback connections poses no major problem, and as was the case with Linsker's ANN architecture, weight adaptation will compensate for analog component imperfections.

3.8.4 Unsupervised Discovery of Spatially Coherent Higher-order Features

Hinton and Becker [Hinton-2] describe a self-organizing ANN for perceptual learning, whose objective is to extract higher-order features that are coherent across time or space. Their procedure maximizes the explicit mutual information between pairs of parameters from adjacent but non-overlapping parts of the input [Hinton-2]. Although the objective function that Hinton and Becker propose is not ideal for direct implementation with analog hardware, Linsker's development of Hebb-type learning rules for information theoretic objective functions suggests that Hebb rules exist for a variety of information theoretic measures, including the one used by Hinton and Becker. As discussed previously, Hebbian learning rules are well suited to VLSI implementation because they require only local information for weight adaptation, and computation typically involves only multiplication, addition and subtraction.

3.8.5 Scalable Architectures

Very large ANN systems are required to tackle real-world problems: thus, a critical issue in evaluating network architectures is how well a very large implementation of a particular ANN will perform. Increased network size leads to additional redundancy, and hence additional fault tolerance. Conversely, large systems will have additional problems: for example, difficulties related to timing of global signals may require unclocked or self-timed circuits. Very large ANN systems involve many additional considerations beyond the scope of the present work.

Evidence from neurobiology indicates that there are upper limits on the size of tightly-coupled systems required for artificial intelligence: biological brains consist of many subsystems with a high level of internal connectivity, and relatively less interaction between subsystems. An example of an ANN which has a *hierarchical* structure of this sort is discussed in chapter 5.

3.9 Summary

We have designed, implemented and tested an analog CMOS neural network architecture with *in situ* Hebbian learning and capacitive weight storage. Because analog CMOS implementations of common computational operations, such as multiplication and addition, are far more compact than their digital counterparts, analog systems have tremendous computational potential. The difficulty arises in designing a system architecture tolerant of component variations. Our investigations show that other neural networks are well suited to implementation using simple, inaccurate analog components [R. Schneider]. The adaptive ability of a neural network with on-chip learning makes it possible for the network to compensate for imperfections in the analog components from which the system is constructed. Computational performance in excess of 6 billion multiplications per second is achievable ($1MHz$, $1cm^2$ die, $3\mu m$ CMOS).

CHAPTER 4

Synaptic Circuits Motivated by Invertebrate Biology

4.1 Introduction

In chapters 3 and 5 implementations of artificial neural networks (ANNs) using low-accuracy analog CMOS transconductance circuits are described. These ICs implement conventional ANN architectures, with Hebbian [Hebb] and mean field [Peterson-2] learning circuitry at each synapse. In this chapter, we take a slightly different approach to the implementation of ANNs: rather than designing circuits to implement an ANN architecture, we propose circuits which are suggested by the behavior of a *biological* neural network. These biologically motivated analog CMOS circuits include functions which, although important for the operation of biological neural networks, are typically omitted from ANN models. We incorporate only certain abstractions of the biological processes into our circuits, since our goal is to develop efficient ANNs for computational artificial intelligence, rather than modeling neural biology in a literal way. This is an important distinction, because there are computationally important differences between analog CMOS VLSI and neural biochemistry. Operations which are impossible to implement in one may be easy to implement in the other.

Our circuits model three learning paradigms present in biological synapses: habituation, sensitization and classical conditioning. Specifically, we are modeling the behavior of the marine mollusc *Aplysia*, which has been studied extensively -- see for example, [Kandel-1, Kandel-2, Hawkins]. Investigation of mammalian synaptic function is complicated by the small size of neurons and synapses in higher-order animals, and consequently less data is available. It must be emphasized that, although our circuits are more biologically plausible than most current ANNs, they still present a highly simplified picture of biological neural networks.

More complex aspects of biological synapse operation, such as those involving extinction and spontaneous recovery [Hawkins], are are omitted from the circuits described below. In doing so we achieve a balance between accurate modeling of neural biology, and mainstream ANNs.

This work draws upon two previous investigations, [Card-2], in which CMOS circuits implementing habituation, sensitization and classical conditioning with EEPROM weight storage are described, and chapter 3, which deals with analog CMOS Hebbian learning circuits using capacitive weight storage. Other approaches to ANN implementation with learning circuitry at each synapse have been reported recently, including [Schwartz-2, Alspector-1]. In section 4.2, biological synaptic learning is reviewed. Section 4.3 presents a simplified mathematical model of this learning behavior, which is followed by a description of CMOS components which implement these operations in section 4.4. The proposed analog CMOS synaptic circuit, constructed from these components, as well as simulation results, are discussed in section 4.5.

4.2 Basic Biological Synaptic Learning Mechanisms

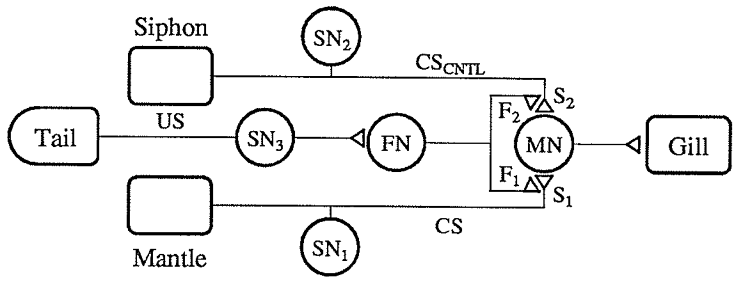

Figure 4.1 *Simplified schematic diagram of the portion of Aplysia's nervous system responsible for gill withdrawal reflex.*

Kandel et al [Kandel-2], in their study of the marine mollusc *Aplysia*, have shown that chemical changes in individual synapses are responsible for three important types of learning: habituation, sensitization and classical conditioning. In this section, biological synaptic function is described at the behavioral level; for a description of the electro-chemical processes involved, see [Kandel-2]. Fig. 4.1 shows a simplified schematic diagram of the portion of

Aplysia's 10^5 neuron nervous system responsible for its protective gill withdrawal reflex. The neural network connecting the gill (*Aplysia's* respiratory organ), siphon (a small spout for expelling sea water), mantle and tail allows the mollusc to withdraw its gill when a stimulus is applied to the tail, siphon or mantle. In nature, this reflex has obvious evolutionary advantages, as it allows *Aplysia* to protect its sensitive gill at the first sign of attack. *Aplysia's* response to tail, mantle and siphon stimuli may be altered dramatically by what it has learned about these stimuli in the past. In Fig. 4.1, the three sensory neurons SN_1, SN_2 and SN_3 receive stimuli from the mantle, siphon and tail, respectively. SN_1 and SN_2 synapse directly with motor neuron MN (S_1 and S_2), so siphon and mantle stimuli can trigger gill withdrawal. In addition, SN_3 synapses with the facilitating interneuron FN, which in turn synapses with *synapses* S_1 and S_2 (through F_1 and F_2). These synapses on synapses, F_1 and F_2, play a critical role in learning, as described below. For the purposes of neural modeling, the pairs S_1 and F_1, and S_2 and F_2 may be regarded as single *ternary* synapses (synapses between three neurons), in which the synaptic weight is determined by the interaction of two external signals. This is the approach that we take in the following section. Note that the schematic of Fig. 4.1 is highly simplified; in particular, each neuron illustrated represents approximately 6 to 24 neurons operating in parallel.

Habituation may be defined as "a decrease in the strength of a behavioral response that occurs when an initially novel eliciting stimulus is repeatedly presented" [Kandel-1]. Kandel et al have shown that repeated mild tactile stimuli to the mantle cause the gill withdrawal reflex to habituate, as the animal learns that the stimuli pose no danger. This learning behavior has been traced to chemical changes in the synaptic connection between mantle sensory neurons and gill motor neurons (S_1): the effective weight of the synapse S_1 is reduced.

The sensitization mechanism is somewhat more complex. *Sensitization* is defined as, "the enhancement of an animal's reflex response as a result of the presentation of a strong or noxious stimulus" [Kandel-1]. In the case of *Aplysia*, noxious stimuli to the tail result in enhanced subsequent gill withdrawal in response to mild mantle stimuli. Again the locus of learning is the synapse S_1, in this case through *presynaptic facilitation* from synapse F_1. The mechanism is as follows: tail sensory neuron firing (SN_3) causes the facilitating interneurons (FN) to fire, which in turn cause F_1 to alter the chemistry of S_1, such that the synaptic weight of S_1 is increased. Not surprisingly, sensitization can reverse the effects of habituation.

"In classical conditioning, an initially weak or ineffective conditioned stimulus (*CS*) becomes highly effective in producing a behavioral response after it has been paired in time with a strong unconditioned stimulus (*US*)" [Kandel-1]. Classical conditioning is a form of associative learning, by which an organism learns a *predictive* relationship between two stimuli. *Aplysia* can be classically conditioned by applying a mild tactile stimulus to the mantle (*CS*), followed approximately $\frac{1}{2}$ second later by a strong stimulus to the tail (*US*). Again F_1 plays an important role. In classical conditioning, recent activity in S_1 due to the *CS* results in *activity-dependent enhancement of presynaptic facilitation.* Thus classical conditioning uses the same mechanism as sensitization (presynaptic facilitation), the effect of which is enhanced by the arrival of the *CS* $\frac{1}{2}$ second before. In classical conditioning, the time between the *CS* and the *US* is critical: if the *US* arrives *before* the *CS*, no classical conditioning occurs. This makes good survival sense, since *Aplysia* is concerned about learning when a mild stimulus (*CS*) predicts a potentially threatening one (*US*). In Kandel's experiments with *Aplysia*, CS_{CNTL} (siphon stimulus) was used to differentiate between the effects of sensitization and classical conditioning.

As the above discussion shows, classical conditioning is a form of associative learning, and is therefore related to Hebbian learning [Hebb] and its variants. However, classical conditioning is *non-commutative*, because the *CS* must precede the *US* by a critical interval.

Kandel's investigation of synaptic learning shows that *Aplysia* uses learning rules which are much more elaborate than those used by most ANNs. *Aplysia* employs non-commutative timing-critical associative learning, as well as two forms of non-associative learning, habituation and sensitization. Work by others demonstrates that these mechanisms are important in a wide variety of organisms. Studies using the fruit fly *Drosophila* indicate that similar chemical mechanisms are involved in learning. In human studies, it was found that the ability of a one year old baby to habituate to repeated visual stimuli correlates well with measured intelligence at four years of age. The implication is that simple non-associative forms of learning may be an important part of biological intelligence, and should therefore be considered for inclusion in ANN models. In the following section, an abstraction of these aspects of learning in *Aplysia* is presented. Note that the above description of biological synaptic function omits many interesting aspects of learning in *Aplysia*; see [Hawkins] for a more detailed discussion.

4.3 Model of Habituation, Sensitization and Classical Conditioning

As a starting point for the development of this learning model, we use a standard ANN model with Hebbian learning described in chapter 3. In this model, both synaptic weights and neuron activations can take on values in the range $[-V,V]$. Network operation is governed by the equations (3.1) and (3.2).

The neural network model presented in chapter 3 differs from neuron biology in two important ways. Neuron activations are represented as analog values in the model, rather than as neuron firing rates. Secondly, biological neurons only have positive (unipolar) activations and synaptic weights, whereas the above model allows both positive and negative (bipolar) activations and weights. Although at first glance the use of bipolar weights and activations violates the principle of biological plausibility, it allows both excitatory and inhibitory synaptic connections to be treated in a unified way, and follows the principle of implementing functions in a way that is ''natural'' for the medium being used to build the system. In our case, the medium is analog CMOS circuitry, in which bipolar operation is easily achieved.

System behavior in the model of biological synaptic function proposed in this chapter is governed by the equations:

$$V_i = f(\sum_j \sum_k W_{ijk} V_j) \tag{4.1}$$

$$\frac{dW_{ijk}}{dt} = C\,\mathrm{d}(V_j)V_k - D \mid V_j \mid W_{ijk} + E \mid V_k \mid W_{ijk} \tag{4.2}$$

The ternary nature of these synapses is evident in (4.1) and (4.2): W_{ijk} is the weighting factor determining the effect that correlation between the activations of the j^{th} and k^{th} neurons will have on the activation of the i^{th} neuron. Hebbian learning, as described by (3.2), may be regarded as a special case of (4.2), in which $i = j$. Using the notation of the previous section, the unconditioned stimulus $US = V_{US} = V_k$, the conditioned stimulus $CS = V_{CS} = V_j$, and (4.2) may be rewritten (for motor neuron MN) as:

$$\frac{dW}{dt} = C\,\mathrm{d}(V_{CS})V_{US} - D \mid V_{CS} \mid W + E \mid V_{US} \mid W \tag{4.3}$$

where the synaptic weight W is a shorthand notation for $W_{ijk} = W_{MN,CS,US}$, and indices CS and US represent *neurons* SN_1 and FN_1 respectively, rather than their activations. The term $C\,\mathrm{d}(V_{CS})V_{US}$ implements classical conditioning, where $\mathrm{d}(V_{CS})$ is a delayed, time-averaged version of V_{CS}, representing the presynaptic facilitation function of F_1 in Fig. 4.1. Note that (4.3) differs from Hebbian learning, as it involves a correlation of two inputs, rather than an input and an output activation. V_{CS} must precede V_{US} by a critical interval for classical conditioning to cause a change in synaptic weight W, as in *Aplysia*, and the size and direction of the weight change are determined by the signs and magnitudes of V_{CS} and V_{US}, as in the system described by (3.2). Thus the system can learn *predictive* relationships between V_{CS} and V_{US}. Note that in addition to creating a delay, the function $\mathrm{d}(V_{CS})$ is *gated* by $V_{US} \neq 0$, such that classical conditioning will not occur when V_{US} precedes or coincides with V_{CS} (absence of reverse conditioning).

The second term in (4.3) represents habituation. Habituation occurs when V_{CS} is non-zero, and always causes W to decay towards zero. This extension of habituation to the bipolar case makes intuitive sense, since it causes non-associative weight decay when V_{CS} is active, as in habituation in *Aplysia.* The form of the expression is also appropriate, in light of Mead's observation, ''A great deal of inhibitory feedback in biological systems depends on activity in sensory input channels, but does not depend on the *sign* of the input'' [Mead].

Finally, the third term in (4.3) $E \mid V_{US} \mid W$ is the sensitization term. V_{US} activity causes an increase in the magnitude of the weight, resulting in increased sensitivity to subsequent V_{CS} signals. Notice that unlike in *Aplysia,* classical conditioning and sensitization are implemented by different mechanisms in (4.3). This is due to the generalization to bipolar weights and activations. In the unipolar case, weights can increase in only one direction, whether through associative learning or sensitization, and therefore classical conditioning and sensitization can use the same biochemical machinery. However, this is a special case: (4.3) gives a clearer picture of the fundamental difference between the operations of sensitization and classical conditioning.

In general, $C > E > D$, so associative learning dominates, and sensitization and habituation result in smaller non-associative weight changes. The role of habituation as a form of inhibitory (negative) feedback is apparent from (4.3). Sensitization causes weight values to increase: however, the natural saturating behavior of any practical realization of (4.3) will limit the effects of sensitization.

Next we describe analog CMOS components which will be assembled into a system to implement this model of biological synaptic learning.

4.4 Analog CMOS Components

The analog CMOS circuits described below are compact, low-accuracy amplifiers, multipliers, absolute value circuits, etc. The decision to use analog, rather than digital, computation results in large silicon chip area savings, at the expense of computational accuracy and repeatability. However, the biological machinery of *Aplysia* is even less precise, so any neural network architecture that requires high accuracy components is not biologically plausible, and therefore not of interest in the present work.

The components which we are using are *transconductance* circuits, where the input signals are voltages and the output is a current. Many of these circuits are variants of the circuits used in [Mead], but unlike in Mead's work, the transistors in our circuits are operating above threshold ($V_{GS} > V_T$). Our goal at this stage is not to use the most compact circuits possible; rather we wanted to use circuits that are reasonably silicon-area efficient, but still robust, and therefore requiring only one fabrication iteration.

Below, circuits are discussed in detail, approximate analytic expressions for their operation are derived, and SPICE MOSFET level 3 simulations of their behavior are presented. As in the previous chapter, analytic expressions are derived using the simplified square-law equation for the drain current of a MOSFET in saturation: $I_{DS} = \frac{\beta}{2}(V_{GS} - V_T)^2$, where $\beta = \mu C_{OX}\frac{W}{L}$.

4.4.1 Transconductance Amplifier

The transconductance amplifiers of Fig. 4.2 perform the operation $I_{OUT} = a(V_1 - V_2)$. In Fig. 4.2a, M_2 and M_3 form a differential pair, and M_4 and M_5 implement a current mirror, so I_{OUT} is the difference between I_+ and I_- from (3.4) and (3.5):

$$I_{OUT} = I_+ - I_- = \frac{\sqrt{\beta^P} V_X \sqrt{4I_B - \beta^P V_X{}^2}}{2} \tag{4.4}$$

where $V_X = V_1 - V_2$ and the current source I_B is implemented by M_1 operating in saturation. This amplifier has a limitation which restricts its usefulness: for the circuit to operate as described in (4.4), V_{OUT} must be more negative than at least one of the input voltages. This restriction causes problems in many system applications, in which the output voltage cannot be guaranteed.

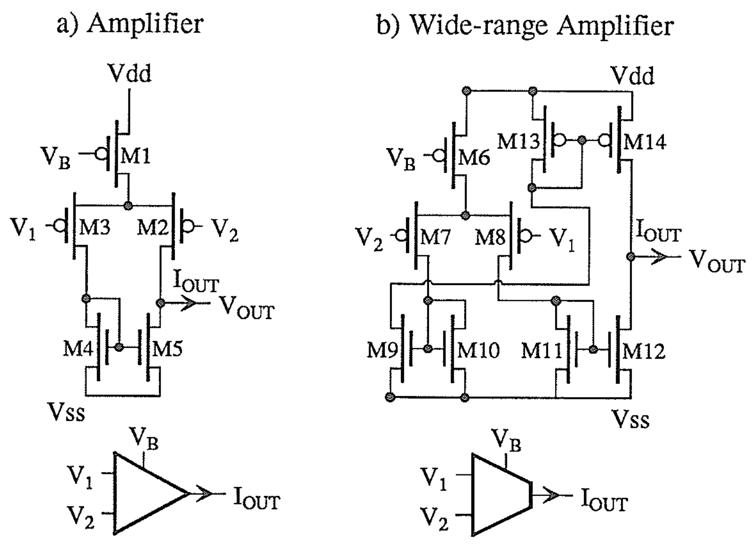

Figure 4.2 *Transconductance amplifiers, $I_{OUT} = a(V_1 - V_2)$; a) Basic amplifier, where V_{OUT} is limited by inputs V_1, V_2; b) Wide-range amplifier, where V_{OUT} is limited only by Vdd, Vss.*

The amplifier circuit of Fig. 4.2b eliminates this restriction by adding two current mirrors (M_9, M_{10}, and M_{13}, M_{14}) which mirror the current through M_7 twice, and thereby isolate input and output without altering the amplifier's transfer function. Fig. 4.3 shows that the amplifier circuits behave as predicted by (4.4): for small values of $V_1 - V_2$ the amplifier is linear, with saturation occurring when $|V_1 - V_2| > 0.75V$.

4.4.2 Transconductance Multiplier

The transconductance multipier performs the operation $I_{OUT} = a(V_1 - V_2)(V_3 - V_4)$. It is basically the same circuit that is described in section 3.4.1.

4.4.3 Absolute Value Circuit

To implement (4.3), an absolute value function is required. The circuit of Fig. 4.4a [Mead] works as follows: P_1 produces a current $I_1 = a(V_1 - V_2)$, which is mirrored by the current mirror M_1, M_2. However, this current mirror can only mirror $I_1 > 0$, and thus produces a half-wave rectified version of $V_1 - V_2$. Similarly, P_2, M_3 and M_4 produce a

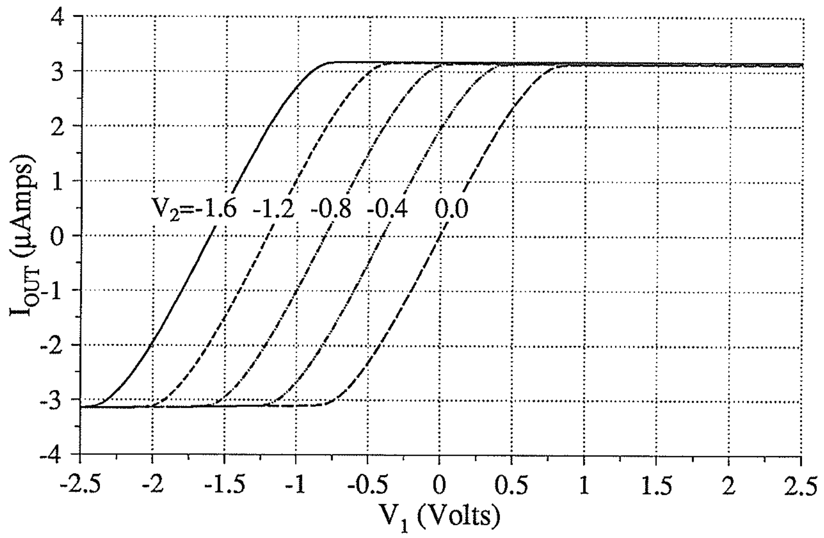

Figure 4.3 *Wide-range amplifier I_{OUT} as a function of V_1, for various V_2.*

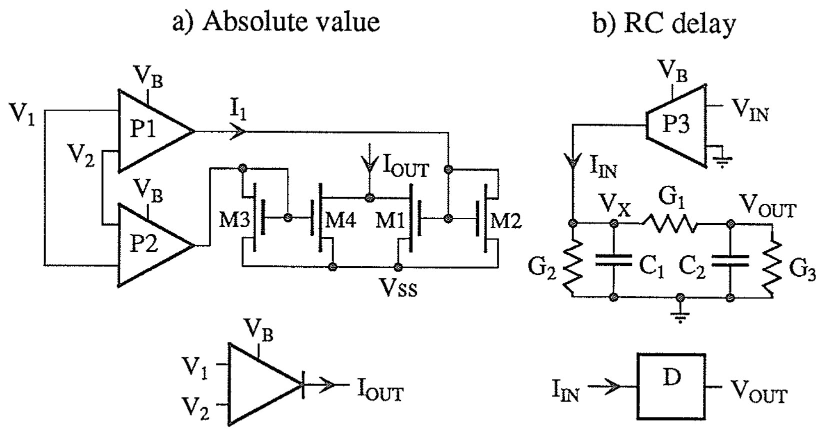

Figure 4.4 *a) Absolute value circuit, $I_{OUT} = a \mid V_1 - V_2 \mid$; b) RC delay circuit.*

half-wave rectified version of $-(V_1 - V_2)$, and $I_{OUT} = \text{pos}(a(V_1 - V_2)) + \text{pos}(-a(V_1 - V_2))$ $= a \mid V_1 - V_2 \mid$, as required. The transfer function of this circuit is

$$I_{OUT} = \frac{\sqrt{\beta^P} \mid V_X \mid \sqrt{4I_B - \beta^P V_X^2}}{2} \tag{4.5}$$

4.4.4 RC Delay Circuit

The delay circuit of Fig. 4.4b is used for the presynaptic delay function d(·) in (4.3). Its transfer function in the Laplace domain is:

$$\frac{V_{OUT}}{I_{IN}} = \frac{-G_1\tau_1\tau_2}{C_1C_2(\tau_1 s + 1)(\tau_2 s + 1)} \tag{4.6}$$

and its impulse response:

$$V_{OUT} = \frac{G_1\tau_1\tau_2}{C_1C_2(\tau_2 - \tau_1)}(e^{-t/\tau_2} - e^{-t/\tau_1}) \tag{4.7}$$

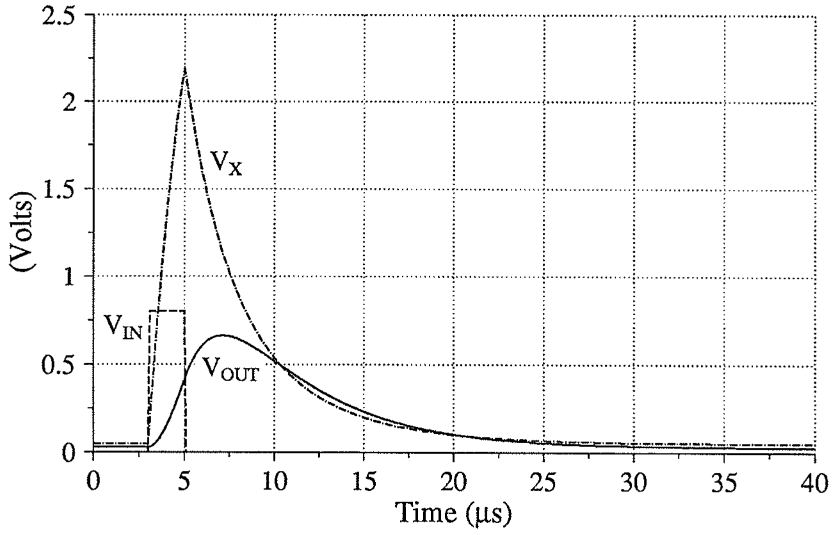

Figure 4.5 *RC delay circuit response to pulse input.*

where τ_1 and τ_2 are determined by G_1, G_2, G_3, C_1 and C_2. In response to a current impulse, assuming that $\tau_1 < \tau_2$, and that τ_1 and τ_2 are well separated, the rise time of V_{OUT} will be approximately determined by τ_1 and its fall time by τ_2. Fig. 4.5 illustrates the circuit's response to a short input pulse: τ_1 and τ_2 have been chosen such that V_{OUT} has a short rise time and a long decay time ($\tau_1 = 2.0\mu s$, $\tau_2 = 4.9\mu s$). Insufficient biological data is available to determine the appropriate shape of d(·) precisely.

4.4.5 Absolute Value Multiplier

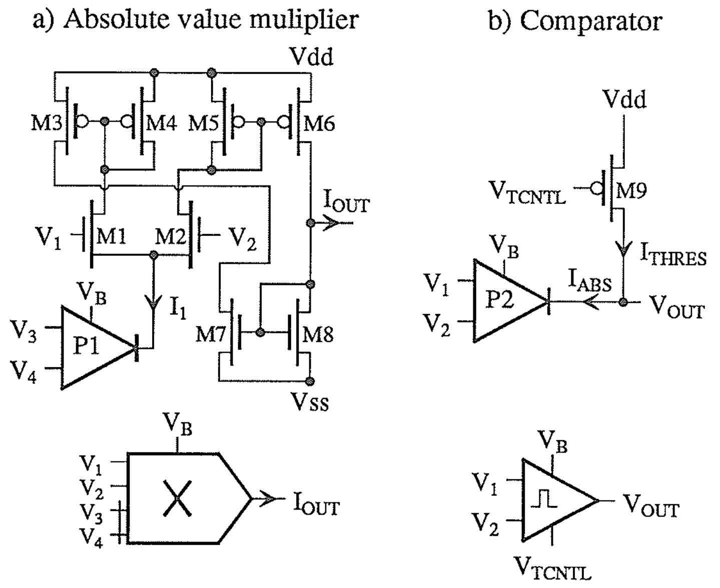

Figure 4.6 *a) Absolute value multiplier,* $I_{OUT} = a(V_1 - V_2)\,|\,V_3 - V_4\,|$ *; b) Comparator,* $V_{OUT} \approx Vdd$*, when* $|\,V_1 - V_2\,| < V_{THRES}$*,* $V_{OUT} \approx Vss$*, when* $|\,V_1 - V_2\,| \geq V_{THRES}$*.*

The circuit of Fig. 4.6a performs the calculation $I_{OUT} = a(V_1 - V_2)\,|\,V_3 - V_4\,|$, which is required to implement habituation and sensitization, as follows. P_1 takes the absolute value $I_1 = a\,|\,V_3 - V_4\,|$, which serves as the bias current I_B for the wide-range transconductance amplifier, M_1 - M_8. Fig. 4.7 shows the behavior of this circuit -- notice that it has approximately the same characteristics as the wide-range multiplier (Fig. 3.5), with the exception of the absolute value operation. Its transfer function is:

$$I_{OUT} = \frac{\sqrt{\beta^N} V_Y}{2}\left[\sqrt{\left[\sqrt{4I_B - \beta^P V_Z^2} + \sqrt{\beta^P}\,|\,V_Z\,|\,\right]^2 - 4I_B - \beta^N V_Y^2}\right] \tag{4.8}$$

This circuit is assembled from two sub-circuits: the absolute value circuit and the wide-range amplifier. The transconductance circuits that we are using are easily combined in this way, resulting in efficient implementation of complex functions. In this case, the functions of an absolute value circuit (14 transistors), and a multiplier (17 transistors) can be implemented with a 22 transistor circuit which combines these functions, resulting in a 30% saving in

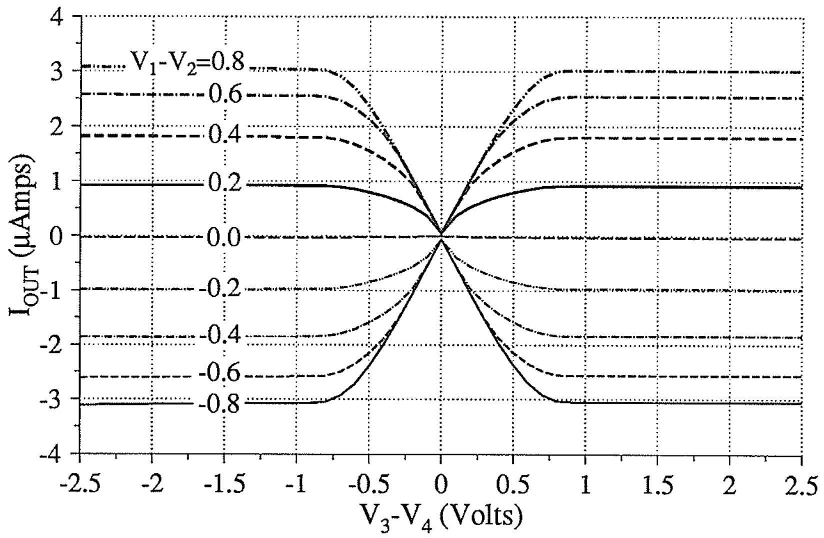

Figure 4.7 *Absolute value multiplier, I_{OUT} as a function of $V_3 - V_4$ for $V_1 - V_2$ from $-0.8V$ to $0.8V$.*

silicon area.

4.4.6 Comparator Circuit

The final circuit we need to build our analog CMOS synapse is the comparator circuit of Fig. 4.6b. It performs the presynaptic facilitation inhibition operation when V_{US} is non-zero.

V_{TCNTL} is used to set the threshold voltage, as shown in Fig. 4.8. When the input $|V_1 - V_2| < V_{THRES}$, $V_{OUT} \approx Vdd$, and when $|V_1 - V_2| \geq V_{THRES}$, $V_{OUT} \approx Vss$. A comparator circuit is conveniently built from a transconductance circuit, by connecting its output to a circuit with infinite input impedance (such as a MOSFET gate) -- the output voltage will saturate at either Vdd or Vss. Is this case, when $a|V_1 - V_2| = I_{ABS} < I_{THRES}$, V_{OUT} will saturate at Vdd.

4.5 Synaptic Learning Circuit

Fig. 4.9 is a bipolar synaptic learning circuit, built from the components described in the previous section. This circuit approximates (4.3), incorporating classical conditioning,

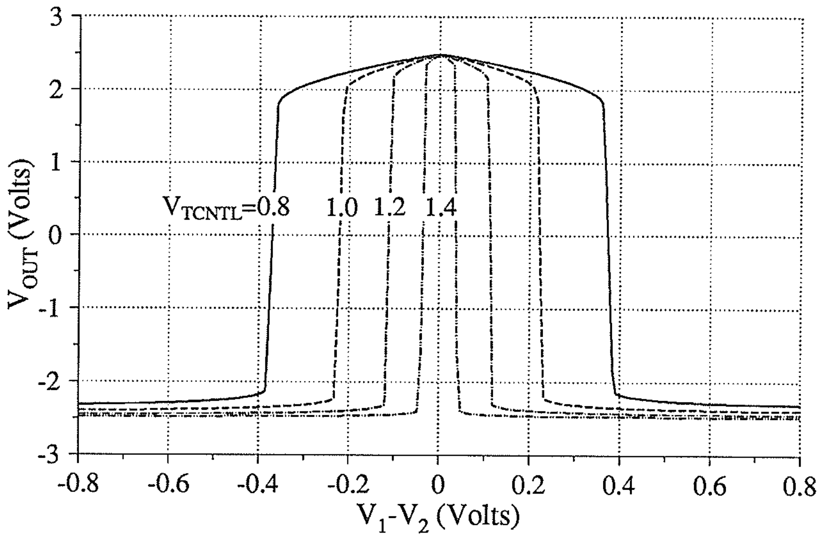

Figure 4.8 *Comparator, V_{OUT} as a function of $V_1 - V_2$, for various V_{TCNTL}.*

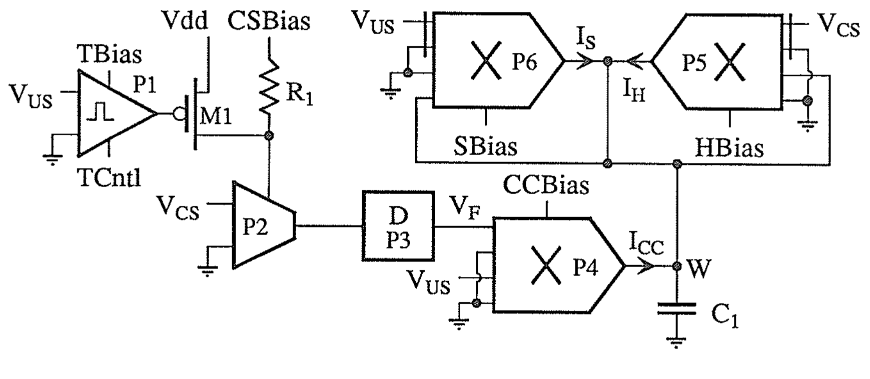

Figure 4.9 *Synaptic circuit.*

habituation, and sensitization. The synaptic weight is stored as the voltage W on capacitor C_1, and weight changes are governed by:

$$\frac{dW}{dt} = \frac{I_{CC} + I_H + I_S}{C_1} \tag{4.9}$$
$$= \frac{cV_F V_{US} - d \mid V_{CS} \mid W + e \mid V_{US} \mid W}{C_1}$$

where I_{CC}, I_H and I_S are, respectively, the classical conditioning, habituation and sensitization weight change components. P_1 - P_4, M_1 and R_1 implement classical conditioning. Inhibition of presynaptic facilitation (prevention of reverse conditioning) is achieved as follows: when $\mid V_{US} \mid > V_{THRES}$, P_1 turns on M_1, which turns off P_2 by setting its bias current to zero. As a result, when $\mid V_{US} \mid > V_{THRES}$, V_{CS} has no effect on V_F, and therefore no classical conditioning can occur. Conversely, when $\mid V_{US} \mid < V_{THRES}$, P_2 is active, and classical conditioning can occur.

Absolute value multipliers P_5 and P_6 approximate habituation and sensitization. Note that W is connected to the negative input of P_5, since habituation drives W towards zero. Below, three examples are presented which illustrate the learning behavior of this synaptic circuit. For the purposes of illustration, higher than typical learning rates are used, so that weight changes are evident over the relatively short interval of these simulations.

4.5.1 Classical Conditioning

Fig. 4.10 shows the synaptic circuit's response to a typical classical conditioning event. At $t = 3\mu s$, a $2\mu s$ conditioned stimulus (V_{CS}) pulse initiates a presynaptic facilitation pulse (V_F). At $t = 7\mu s$, the unconditioned stimulus (V_{US}) is presented, resulting in an increase in the weight W, as the correlation relation between V_{CS} and V_{US} is learned. The critical period for classical conditioning, that is, the time after V_{CS} occurs in which V_{US} must be presented for conditioning to occur, is approximately 2 to $8\mu s$. The shape of V_F can easily be changed by choosing different component values for the delay circuit, P_3.

At $t = 30\mu s$, V_{US} is briefly pulsed negative. The presynaptic facilitation signal V_F has decayed to almost zero, and thus no classical conditioning occurs. However, V_{US} does cause a slight increase in W, through the sensitization mechanism. Notice that despite V_{US} being negative at $t = 30\mu s$, W *increases*, since sensitization is independent of the sign of V_{US}. The slow decay of W from 13 to $30\mu s$ is due to a small component mismatch in P_4, P_5 and P_6.

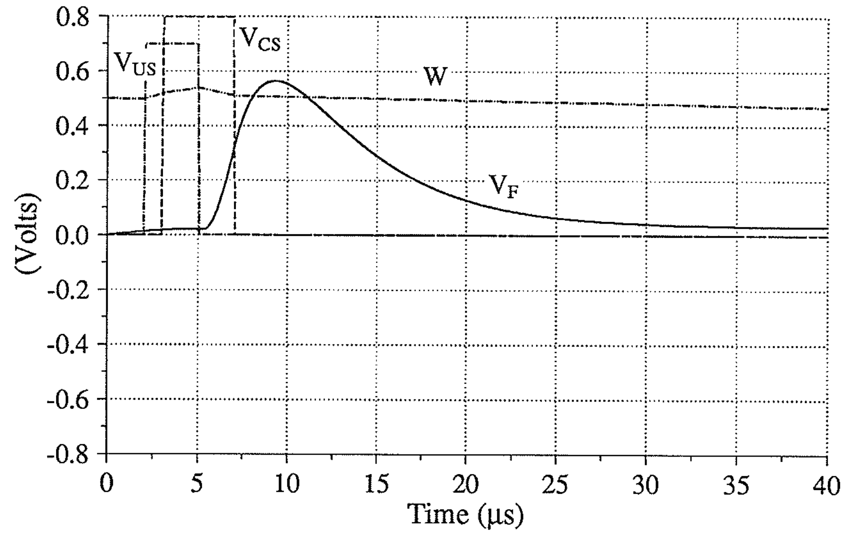

Figure 4.11 *Reverse conditioning. V_{US} occurs before V_{CS}, so no classical conditioning occurs.*

4.5.3 Bipolar Circuit Behavior

Thus far, we have shown classical conditioning examples in which both V_{CS} and V_{US} are positive. However, our circuits are bipolar, as illustrated in Fig. 4.12. In this example, the V_{CS} pulse presented is negative, resulting in a negative V_F. A positive V_{US} arrives during the critical period for classical conditioning, and the synapse learns an anti-correlation relationship between V_{CS} and V_{US}. When a second negative V_{CS} is presented at $t = 30\mu s$, habituation occurs, causing W to decay towards zero.

4.6 Implications for ANNS

The development of our biology-motivated learning model raises a number of interesting questions, including, whether the biological learning details presented in this chapter are important in ANN models. We began with a conventional Hebbian model, generalizing and extending it to incorporate the functions of synaptic learning found in *Aplysia*. Thus Hebbian learning (3.2) is a special case of (4.3), in which the delay of d(·) is zero, $i = j$, and reverse conditioning is allowed.

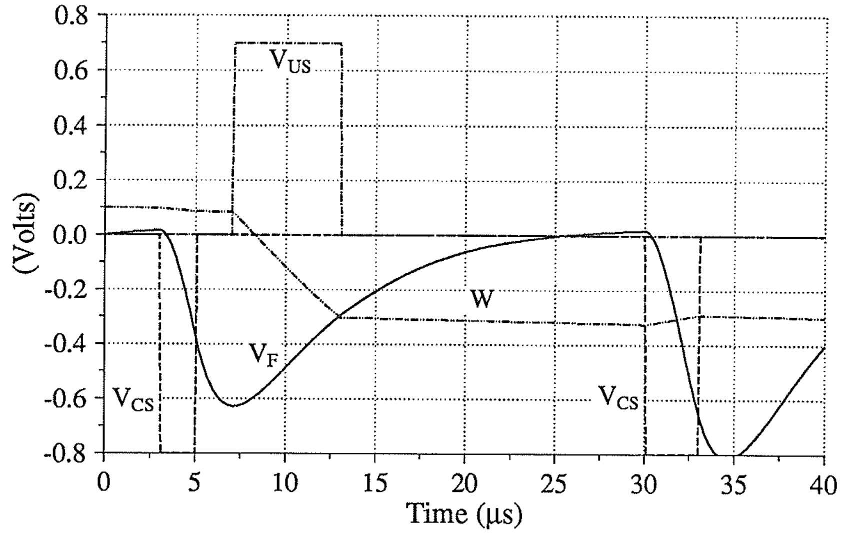

Figure 4.12 *Bipolar circuit behavior. Synapse can be classical conditioned to learn anti-correlation.*

The effect of non-zero delay in d(·), together with inhibition of reverse conditioning, is that learning becomes non-commutative. This ''symmetry breaking'' would result in non-symmetric weights ($W_{ijk} \neq W_{ikj}$) in the case of a fully-connected, Hopfield-type network. This type of non-commutative learning could be incorporated into networks using contrastive Hebbian learning [Peterson-2], again resulting in non-symmetric weights. Further work is planned to investigate this possibility.

The non-associative learning found in biological systems, habituation and sensitization, may also have a role to play in ANNs. Their importance, and how to go about incorporating them into an artificial system is less clear than with non-commutative learning, because non-associative learning is not used in many ANNs. However, they are obviously important in biological neural networks, as shown by the study of a variety of animals, and therefore must be considered for inclusion in ANNs.

An important issue is whether the biological synaptic learning presented in this work depends upon a certain degree of ''hard-wiring''. For example, *Aplysia* has been ''wired'' so that it can learn a predictive relationship between mild mantle stimuli and noxious tail stimuli. If the neural network of Fig. 4.1 were missing the facilitating interneuron *FN*, then neither sensitization nor classical conditioning could occur. There are two interpretations of this ''pre-wiring'': it is evidence that the biology of *Aplysia* is not of interest to ANN researchers, because it represents a special case in which pre-determined network architecture plays a major role; or, it is an indication that tailoring a network architecture for a particular purpose is an essential part of neural networks, whether natural or artificial. Our tendency is towards the latter view, although the issue is far from settled.

4.7 Summary

This work has demonstrated that biological details, which are omitted from most current ANN models, may be efficiently implemented with analog CMOS circuits. Three types of learning, habituation, sensitization and classical conditioning, are incorporated in the synaptic learning circuit which we have developed. Habituation and sensitization are types of non-associative learning, and are not often included in ANN models. Classical conditioning is a form of associative learning, related to Hebbian learning. It differs from Hebbian learning in temporally correlating two inputs at ternary synapses (synapses between three neurons), rather than the input and output of binary synapses. Classical conditioning is also more complex, as it is non-commutative, resulting in a type of ''symmetry breaking'' in the neural network system. Our work shows that biological synaptic learning is substantively different from current ANN learning, both in terms of learning paradigms employed, and in terms of the extensive use of ''hard-wiring'' of connections in biological neural networks. Further research is required to determine whether these aspects of neural biology are a critical part of information processing in biological and artificial neural networks.

CHAPTER 5

Circuits for Contrastive Hebbian ANNs

5.1 Introduction

In this chapter we describe an analog CMOS implementation of a fully connected ANN with *contrastive* Hebbian learning [Movellan] circuitry at each synapse. This work is an extension of the simple Hebbian learning ANN implementation presented in Chapter 3. Networks with contrastive Hebbian learning are intended for supervised learning applications, where a set of training patterns is used for weight learning. A key feature of contrastive Hebbian learning is that it can be used to train weights in networks with *hidden neurons*, that is, neurons whose activations are neither network inputs nor outputs. Implementations similar to those of this chapter appear in [Alspector-1, Arima]. See section 1.3 for a description of their work.

Below, we begin by outlining the theoretical basis of contrastive Hebbian learning. Then we present a description of our analog CMOS realization of contrastive Hebbian theory, with a discussion of approximations that were required for efficient implementation, and circuit testing. Lastly, other applications of these analog circuits are explored.

5.2 Theory of Network Operation

The architecture of the ANN under consideration is a fully-connected Hopfield-type arrangement of neurons and synapses, in which there is a synaptic connection between each pair of neurons. Thus, a network of N neurons has $N(N-1)$ synapses, and synaptic circuitry occupies the majority of silicon chip area (Fig. 5.1). In this section, we present some elements of the theory of operation of a fully-connected ANN with contrastive Hebbian weight learning

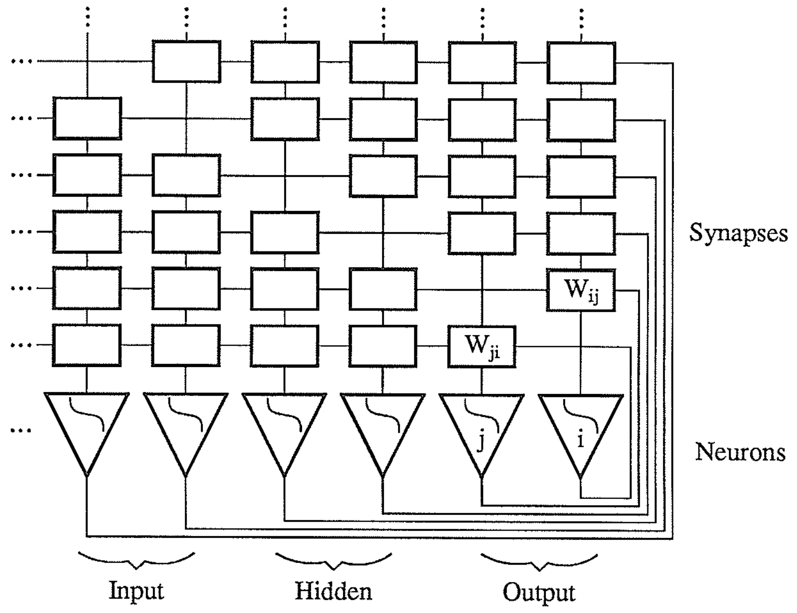

Figure 5.1 *Architecture of fully connected network.*

[Movellan, Peterson-1]. We begin by describing the well studied Hopfield network [Hopfield], extending the discussion to include contrastive Hebbian learning.

Hopfield ANNs are typically used for associative memory applications, in which the network is presented with a set of inputs, for which it generates a set of outputs. The input-output relationship is determined by the system's synaptic weights, W_{ij}. These networks are a variety of *relaxation network*, in which the network *settles* into a minimum energy state, subject to the constraints imposed by inputs and synaptic weights. Theory indicates that symmetric network weights ($W_{ij} = W_{ji}$) are required to ensure that Hopfield networks will settle into a stable final state. However, in practise weight symmetry is not essential [Galland].

In a Hopfield network, the stable neuron activations after settling are determined by the relation:

$$V_i = f(\sum_j W_{ij} V_j) \tag{5.1}$$

where V_i is the activation of the i^{th} neuron, W_{ij} is the synaptic weight determining the effect that the j^{th} neuron has on the i^{th} neuron, and $f(\cdot)$ is a sigmoidal nonlinear saturating function. The optimum synaptic weights, W_{ij}, to represent a particular set of input-output associations, may be readily calculated [Hopfield].

The major limitation of Hopfield networks is that they have no *hidden neurons*, that is, neurons whose activations are neither network inputs nor outputs. Networks without hidden neurons can not represent a wide variety of input-output associations, including the exclusive OR (XOR) function and its generalizations [Rumelhart-1]. The contrastive Hebbian network architecture, which we describe below, takes the basic Hopfield network and incorporates hidden neurons. In principle, this is an easy task, because hidden neurons are simply neurons whose activations are not set to particular values during network training and operation. The difficulty arises in finding a weight learning scheme which makes use of the hidden neurons to represent input-output relationships: contrastive Hebbian learning is one such method.

Contrastive Hebbian learning (CHL) is a two-phase weight learning process, governed by:

$$\Delta W_{ij} = a\,(V_i^+ V_j^+ - V_i^- V_j^-) \tag{5.2}$$

where V_i^+ and V_j^+ are the activations of the i^{th} and j^{th} neurons in the *clamped* phase, V_i^- and V_j^- are the activations of the i^{th} and j^{th} neurons in the *unclamped* phase, and a is a small positive constant which determines the weight learning rate. During the clamped phase, input and output neuron activations are held at the desired values (ie. *clamped*), and the network is allowed to settle into its minimum energy state. After settling, the activation values V_i^+ are recorded for all neurons. In the unclamped phase, only inputs are fixed, and the network determines the activations of both output and hidden neurons as it settles into its minimum energy state. After settling in the unclamped phase, (5.2) is used to modify network weights. This two phase procedure is repeated for each pair of input-output associations in the training set. Typically, several hundred passes through the training data set are required for CHL weight training. This weight training procedure is also called deterministic Boltzmann or mean field learning, because it is a non-stochastic version of the Boltzmann weight learning algorithm [Hinton-1]. As mentioned previously, the key feature of CHL is its ability to use hidden neurons to represent complex associations.

CHL is a form of gradient descent learning in the *contrastive function* J [Movellan]:

$$J = F^+ - F^- \tag{5.3}$$

where F^+ and F^- are the energy functions of the network after settling in the clamped and unclamped phases. Here we present a brief outline of the theoretical basis of CHL: for a more detailed account, see [Movellan]. As we are dealing with a fully connected Hopfield-type architecture, F^+ and F^- are minimum energies, subject to the constraints of weights and clamped inputs and outputs. Assume the F has a unique minimum over allowed neuron activations. Then, since F^+ and F^- share the same free parameters (the activations of the hidden neurons), and F^- also has the activations of the unclamped neurons as free parameters, $F^+ \geq F^-$. Given the assumption that the minimum energy state of the network is unique, when $J = 0$, $V_i^+ = V_i^-$ for all output neurons, and therefore J may be used as a gradient descent function for weight learning. For Hopfield-type networks [Movellan],

$$\frac{\partial F}{\partial W_{ij}} = -V_i V_j \qquad i \neq j \tag{5.4}$$

and from (5.3)

$$\frac{\partial J}{\partial W_{ij}} \propto V_i^- V_j^- - V_i^+ V_j^+ \tag{5.5}$$

so CHL (5.2) descends in the contrastive function J.

The difficulty with CHL arises from the assumption that F has a unique minimum. In the case where the network has no hidden neurons a unique minimum exists. However, when hidden neurons are introduced, there need no longer be a unique minimum, and thus $F^+ \geq F^-$ is not guaranteed. Fortunately, if the training procedure maximizes the probability that the clamped phase F^+ settles to the same activations as the unclamped phase F^-, CHL performs well. Two approaches are commonly used. If the unclamped phase is performed first, and the settled unclamped activations are used as the initial activations during the clamped phase (as opposed to random activations), then the network tends to settle into the same state [Movellan]. The fixing of output neuron activations when the clamped phase follows the unclamped phase may be regarded as an imposition of additional constraints on the permissible energy function, which perturbs the values of the unclamped phase activations. As the network learns correct weight values, the difference between clamped and unclamped output activations is reduced, and these perturbations tend towards zero. A second technique that is used to promote settling is a form of annealing, in which neuron gains are gradually increased while the network activations are settling. This annealing procedure reduces the probability that the network will settle into spurious local minima [Movellan]. We have implemented both these techniques in our VLSI implementation of CHL.

Peterson and Hartman [Peterson-2] describe a *Manhattan* updating variant of CHL,

$$\Delta W_{ij} = a \cdot \mathrm{sgn}(V_i^+V_j^+ - V_i^-V_j^-) \tag{5.6}$$

where the magnitude of the fixed-size weight changes is determined by the constant a, and the sign of the change by $\mathrm{sgn}(V_i^+V_j^+ - V_i^-V_j^-)$. It is apparent that (5.2) and (5.6) perform gradient descent in the same function. Peterson and Hartman found that Manhattan updating can result in more stable learning, because situations arise in practise in which (5.2) produces weight changes which vary greatly in magnitude, making it difficult to choose a suitable learning rate [Peterson-2]. Our circuits implement a further modification of Manhattan learning, where weight changes are governed by:

$$\Delta W_{ij} = a(\mathrm{sgn}(V_i^+V_j^+) - \mathrm{sgn}(V_i^-V_j^-)) \tag{5.7}$$

Simulations show that weight adaptation governed by (5.7) results in learning performance approaching that of conventional CHL, provided that Gaussian noise is added to neuron activations during the learning process. For more details regarding the performance of this modified form of CHL, see [R. Schneider]. This form of Manhattan updating is well suited to implementation using analog CMOS circuits. In the next section, we describe an analog CMOS ANN with modified Manhattan contrastive Hebbian learning circuitry at each synapse.

5.3 Overview of Implementation

Fig. 5.2 is a block diagram of the ANN system: the circuit is an analog CMOS approximation to (5.1) and (5.7). Both neuron activations and synaptic weights may take on analog values in the range $[-V,V]$. Multipliers P_3 and P_5 are transconductance multipliers, whose inputs are voltages and output is a current. P_4 and P_6 are linear current to voltage converters, functioning as a resistor to $(Vdd + Vss)/2$. The comparator P_7 outputs either Vdd or Vss, depending on whether its input is above or below its threshold reference, and P_8 and P_9 are conventional digital gates. Lastly, P_1 and P_2 are modified Manhattan CHL circuits, containing weight update circuitry and weight storage capacitors. Synaptic weights are represented by the differential voltage $V_{CL} - V_{UN}$; storing synaptic weights as differential voltages has a number of advantages, which are discussed below.

The circuit of Fig. 5.2 approximates equations (5.1) and (5.7), representing ideal network behavior, as follows. The synaptic weight W_{ij} is represented by the differential voltage, $V_{CL} - V_{UN}$. Then, the product $W_{ij}V_j$ in (5.1) is implemented by P_3 as $I_{SUM} = b(V_{CL} - V_{UN})V_j$, and the summation operation is performed as current summation at

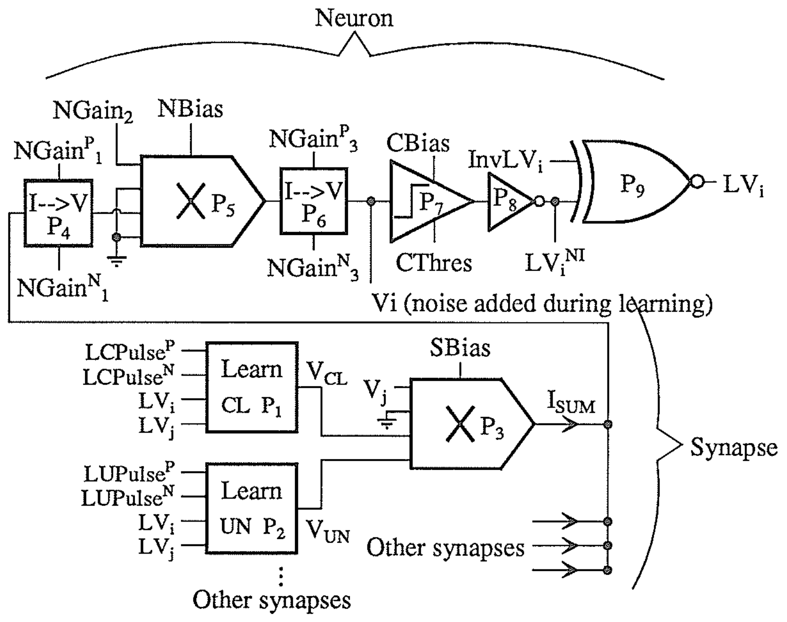

Figure 5.2 *System block diagram.* $N-1$ *synaptic circuits are connected to each neuron circuit.*

the input to P_4. A true summation is performed, despite the fact that the input to P_4 is not held at virtual ground, because multiplier P_3 produces an output *current*, independent of the output voltage. This is an important consideration, because simpler schemes, which perform averaging, rather than summation, such as that employed in [Hopfield], will not function properly in a CHL network [R. Schneider]. P_4 - P_6 realize a variable gain neuron, whose transfer function may be adjusted in a variety of ways, using the *NGain* control signals.

Since we implement modified Manhattan learning, only the *sign* of neuron activations are required for learning. P_7 - P_9 generate LV_i, a binary version of each neuron activation for learning. XNOR gate P_9 is used to invert LV_i via control signal *InvLVi*: details concerning the function of LV_i and its non-invertible counterpart LV_i^{NI} are provided in section 5.7.

5.3.1 Network Operation

Network operation proceeds as follows. First, the network inputs are set for a particular training pattern, while output and hidden neurons remain free (ie. unclamped). The network is allowed to settle to its unclamped phase minimum energy state, LV_i^- and LV_j^- (the binary learning activations generated by the i^{th} and j^{th} neurons) settle, and control signals $LUPulse^P$ and $LUPulse^N$ are pulsed briefly. The binary product $LV_i^- LV_j^-$ determines whether a small quantity of charge is added to, or removed from, the storage capacitor connected to the unclamped learning circuit. Next, network outputs are set to training pattern values, the network settles in the clamped phase, and learning control signals $LCPulse^P$ and $LCPulse^N$ are pulsed briefly. As before, the binary product $LV_i^+ LV_j^+$ determines whether charge is added to, or removed from the clamped storage capacitor. In the case where the network has learned the current training pattern perfectly, $LV_i^- = LV_i^+$ for all neurons, $LV_i^- LV_j^- = LV_i^+ LV_j^+$, V_{CL} and V_{UN} are both increased or decreased by the same amount, and the synaptic weight $W_{ij} = V_{CL} - V_{UN}$ remains unchanged. In all other cases, the net effect of the two phase procedure will be a small weight change. This two phase procedure is repeated for each training pattern. Typically, many passes through the entire set of training data is required to learn network weights.

The network operation detailed above deviates somewhat from the ideal case described by (5.1) and (5.7). In particular, weights are updated after the unclamped phase according to the product $LV_i^- LV_j^-$ and then again after the clamped phase, according to $LV_i^+ LV_j^+$. As a result, weights may take a small unnecessary step after the unclamped phase, which is corrected in the subsequent clamped phase. In addition, since noise is added to neuron activations during weight learning, the learning process is non-deterministic, which introduces additional spurious weight adjustment steps. However, system level simulations [R. Schneider] indicate that these deviations from CHL theory do not seriously affect network learning.

5.3.2 Fabrication

To test our design, we had several circuits fabricated in Northern Telecom's $1.2\mu m$ double metal twin-tub CMOS process. In this process, linear capacitors are formed between two layers of polysilicon. Fig. 5.4 is a photomicrograph of the multi-project die containing a 19 neuron, 342 synapse test network, and Fig. 5.3 illustrates a 3 neuron network. Figs. 5.5 and 5.6 show individual synapse and neuron circuits. In addition, a variety of test circuits were fabricated. It is evident from these photomicrographs that capacitors (the apparently unoccupied areas of Figs. 5.3 and 5.5) take up a significant portion of total IC area. However, this design will still function properly when constructed with smaller capacitors: large capacitors are used in this preliminary design to facilitate testing.

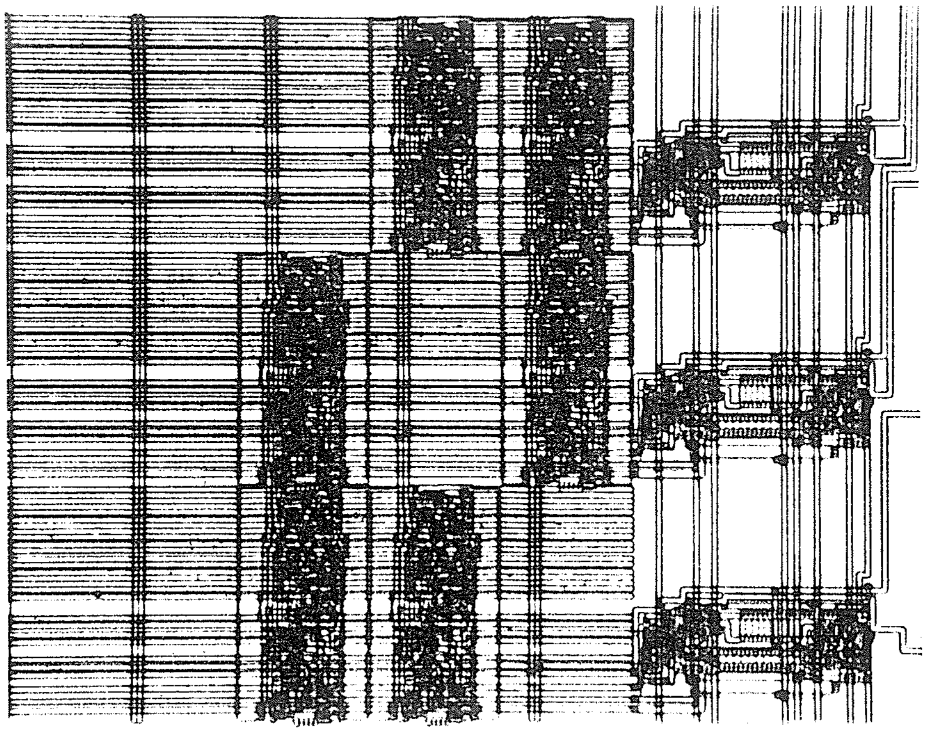

Figure 5.3 *Photomicrograph of 3 neuron, 6 synapse network* ($0.66 \times 0.78 mm^2$).

5.4 Analog Implementation Considerations

The primary reason for using simple analog components is that these circuits are far more compact than their digital counterparts, and hence systems with substantial information processing capability may be integrated on a single chip. An ANN is generally well suited to implementation with low-accuracy analog components, because many operations are not highly critical. Thus, low accuracy computation, in the form of non-ideal multipliers and adders, may be used in their construction. In addition, weight learning allows the network to compensate for a wide variety of component imperfections.

Despite the fact that many operations in an ANN are non-critical, simulations [R. Schneider] indicate that there are two operations in CHL which must be implemented accurately. First, there must be good matching between the clamped and the unclamped learning phases: in particular, in the case of Manhattan weight updating, the size of the weight increment and decrement steps must be the same. It is easy to see why this is important in the

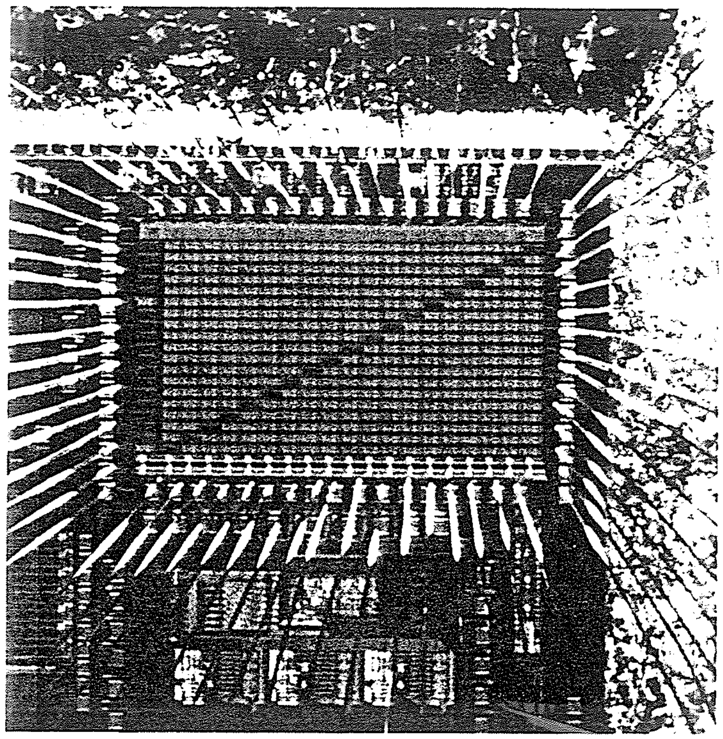

Figure 5.4 *Photomicrograph of 19 neuron, 342 synapse network ($4.16 \times 2.76mm^2$).*

special case when the network has learned a particular training pattern perfectly. Then, the clamped and unclamped phases should cancel one another, and there should be no weight change. If there is a mismatch between the two phases, then the synaptic weight will be changed from its correct value. Simulations show that the size of the mismatch between learning phases is of the same order as the minimum attainable mean squared error in learning associations [R. Schneider]. Clearly, phase mismatch must be kept to at most a few percent.

The second critical function is the weight learning rate. Simulations show [R. Schneider] that a small learning step size (ie. a small ΔW_{ij}) is essential for reliable weight learning. In a high-dimensional weight space, in which the energy function typically has a complex topology, convergence is impossible with large weight change steps. Our results are consistent with those reported by Peterson [Peterson-2] in his discussion of the advantages of Manhattan weight updating. Large sets of training data require small learning steps in any case, to avoid biasing the network weights towards the training patterns presented most recently. The design

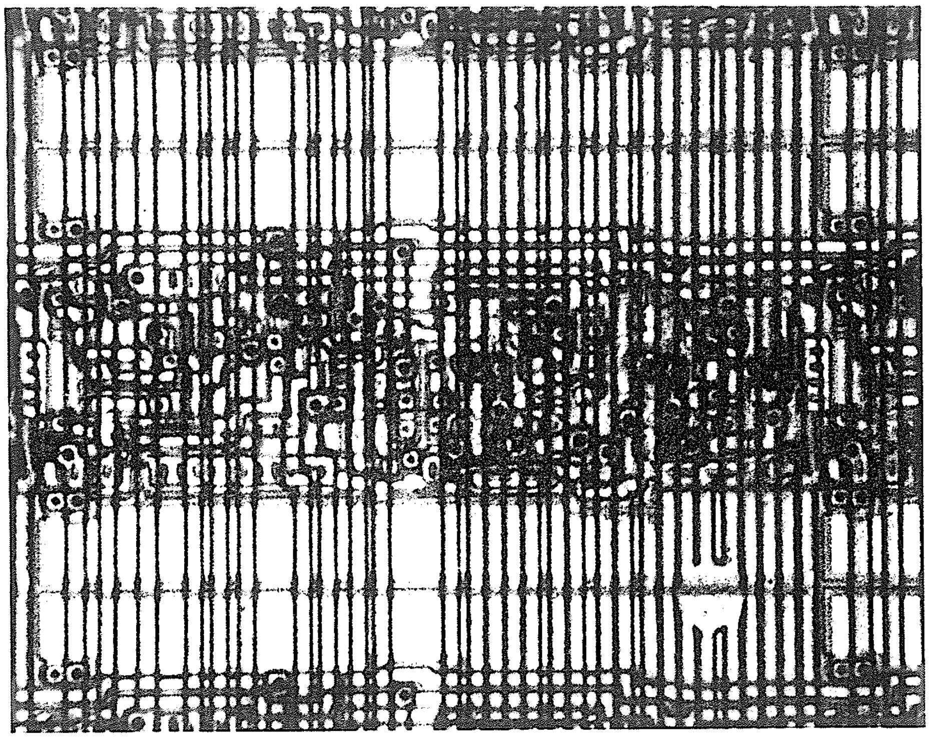

Figure 5.5 *Photomicrograph of synaptic circuit* (27 012μm^2 *per synapse*).

ramifications of these two critical operations are discussed in section 5.7. In the following sections, we describe circuit components in detail, illustrating their behavior with measured data from our test chips.

5.5 Transconductance Multiplier

The analog multiplier circuit that we use to implement P_3, Fig. 5.2 is illustrated in schematic form in Fig. 5.7. It is a Gilbert transconductance multiplier [Gilbert, Mead], whose inputs are voltages and output is a current. This circuit is well suited to our application: it has differential voltage inputs with a near-infinite input resistance, which is what is required to read the differential weight voltage $V_{CL} - V_{UN}$ without discharging the weight storage capacitors. As well, multiplier current outputs I_{SUM} may be summed directly with no additional hardware. As with many analog multiplier circuits, this multiplier does not require accurate matching between N and P channel transistors: only matching between identically sized same-polarity

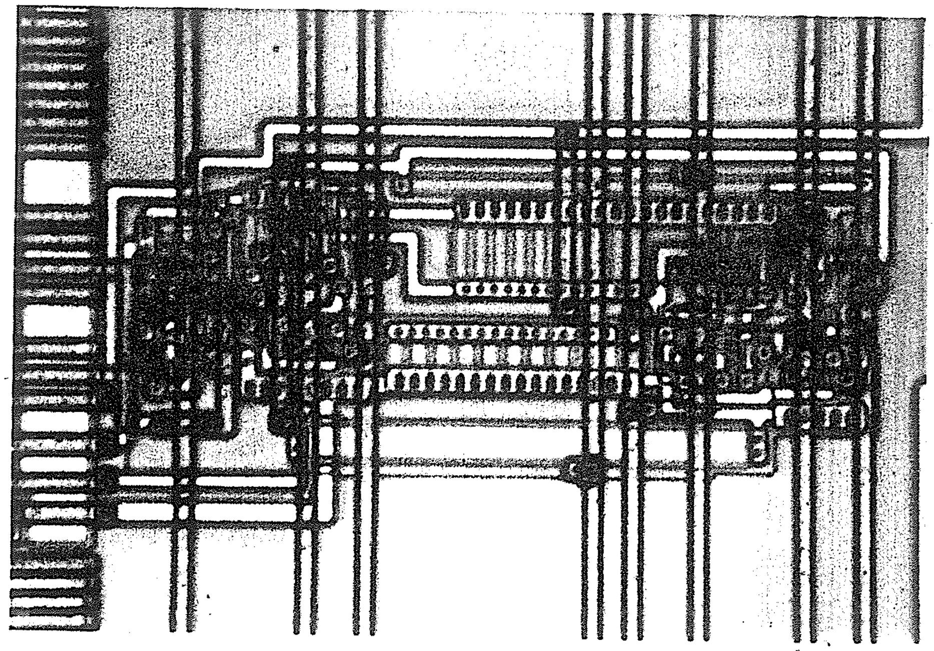

Figure 5.6 *Photomicrograph of neuron circuit* (41 847μm^2 *per neuron).*

transistors is necessary.

Fig. 5.8 shows the transfer characteristic of the summation multiplier. Its linear range is fairly narrow: about 0.75V for the $V_{CL} - V_{UN}$ input, and 1.6V for the V_j input, but our simulations indicate that this is sufficient for our synaptic circuit. Similarly, the 'step' in the characteristic of Fig. 5.8 for $V_{CL} - V_{UN} > 0$ is not large enough to matter for our application. It is due to a mismatch between the transistors making up the current mirror (M_{10}, M_{11}): M_{11} was rotated by 90° in the layout to save space.

Fig. 5.9 illustrates the range of common mode weight voltages for which this multiplier circuit functions properly: V_{CL} and V_{UN} must be kept in the range [–2.5, 1.0] volts. In designing this circuit, there was a tradeoff between common mode voltage swing and the linear range of $V_{CL} - V_{UN}$.

At this stage of our investigation it was not a high priority to use the most compact and fastest circuit designs possible; rather, we opted for circuits which were likely to function after

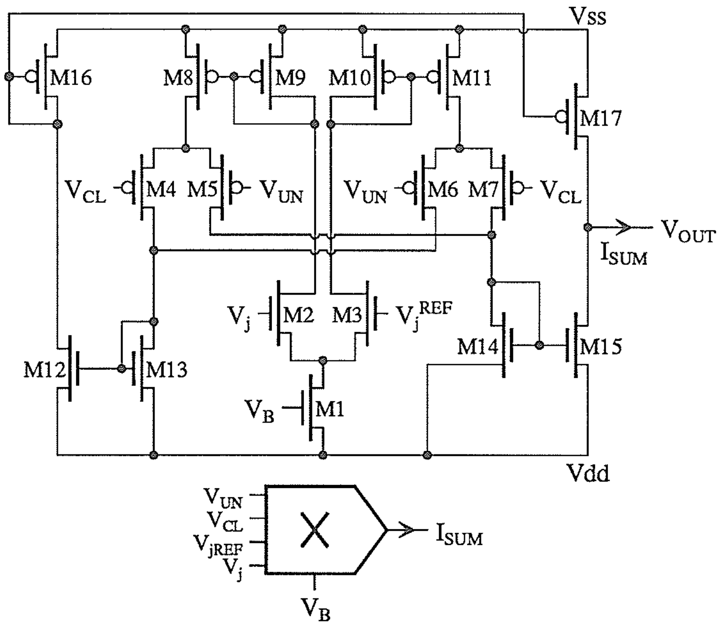

Figure 5.7 *Synaptic transconductance multiplier,* $I_{SUM} = a(V_{CL} - V_{UN})V_j$

one design and fabrication cycle. We also wanted to build our system out of as few different circuits as possible, primarily to reduce design time.

5.6 Other Circuits: Comparator and I-V Converter

The comparator (Fig. 5.10a) consists of a transconductance amplifier driving an infinite input resistance circuit: V_{OUT} will saturate at either *Vdd* or *Vss*. We use this comparator to create a sharp, adjustable breakpoint between positive and negative activations for modified Manhattan learning.

Fig. 5.10b is the current to voltage converter circuit P_4, Fig. 5.2. When the conductances of M_{10} and M_{11} are matched, this circuit functions as a resistor to ground: Fig. 5.11 shows this circuit performs a near-linear current to voltage conversion. Highly linear behavior is not essential, since V_{SUM} is passed through the highly nonlinear neuron amplifier. For greater testing flexibility, the second I-V converter, P_6 at the neuron output, has been implemented off-chip.

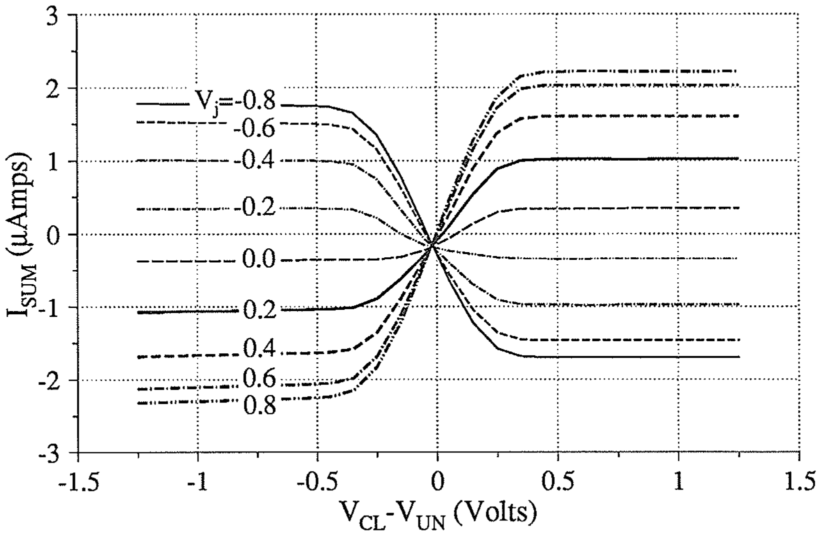

Figure 5.8 *Synaptic multiplier, I_{SUM} as a function of $V_{CL} - V_{UN}$ for V_j from $-0.8V$ to $0.8V$ (1.2μm CMOS test chip, $Vdd = 2.5V$, $Vss = -2.5V$, $V_B = -0.9V$, data collected using HP4145A).*

5.7 Learning Circuit

Fig. 5.12 is a schematic of the learning circuits P_1 and P_2, Fig. 5.2. We begin by describing the typical operation of this circuit. XNOR gate P_1 implements the binary product LV_iLV_j. Assume that $LV_iLV_j > 0$, so M_8 and M_{10} are turned on and M_7 and M_9 are turned off. After settling, learning pulses $LPulse^P$ (from its resting value of Vdd to $Vdd - V_P$) and $LPulse^N$ (from its resting value of Vss to $Vss + V_P$) are pulsed briefly. Because we are dealing with the case where $LV_iLV_j > 0$, M_8 is on, $LPulse^P$ causes M_1 to be turned on briefly, charge flows onto the weight storage capacitor, and V_{CAP} is increased. In the case where $LV_iLV_j < 0$, M_2 is briefly activated, and V_{CAP} is decreased. Note that transistors M_5 and M_6 are normally on, and M_3 and M_4 are normally off: these four transistors are used to set capacitor values at the beginning of the training process. The capacitor access transistors M_1 and M_2 operate in saturation, serving as current sources, and consequently, ΔV_{CAP} will be the same, independent of V_{CAP}.

Differential weight storage schemes have been described by other authors, for example, [Schwartz-1]. However, our approach is substantially different from the techniques employed

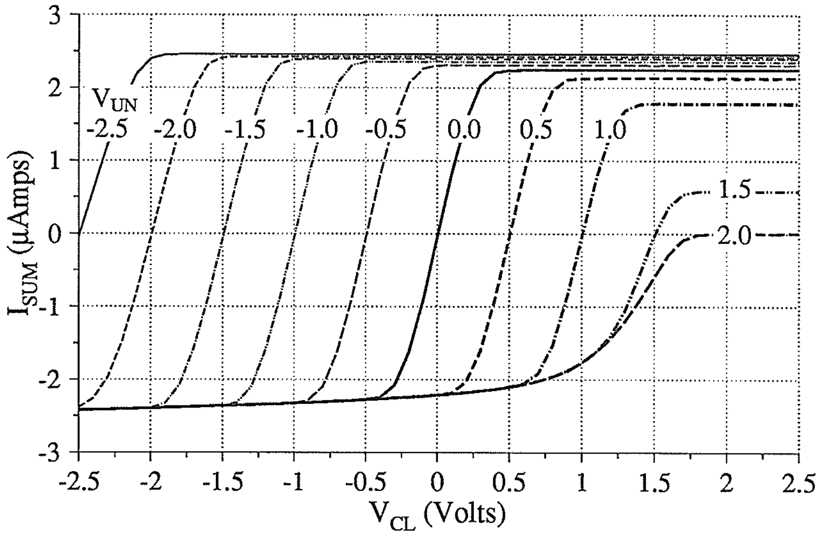

Figure 5.9 *Synaptic multiplier, I_{SUM} as a function of V_{CL} for V_{UN} from −2.5 to 2V (1.2μm CMOS test chip, $V_j = 2.5V$, $Vdd = 2.5V$, $Vss = -2.5V$, $V_B = -0.9V$, data collected using HP4145A).*

in [Schwartz-1]: see section 1.3 for a discussion of Schwartz's work.

5.7.1 Small Learning Step

System-level simulations of CHL ANNs indicate that a learning step that is from 10^{-4} to 10^{-3} of the full range of the weights is required for stable weight learning [R. Schneider]. The only way to achieve such a small weight change with the circuit of Fig. 5.12 is to use the *subthreshold* characteristics of the access transistors M_1 and M_2.

A MOSFET is operating in its subthreshold region when $V_{GS} < V_T$. When $V_{GS} < V_T$, no channel is formed between source and drain and I_{DS} is the result of diffusion of electrons from source to drain. Thus, a MOSFET operating in its subthreshold region behaves like a bipolar transistor, with I_{DS} exponentially dependent on V_{GS}. The N-channel MOSFET drain current expression is derived in the same way as the expression for the collector current in a bipolar $n-p-n$ transistor with homogeneous base doping [Sze]:

$$I_{DS} = I_0 \frac{W}{L} e^{\beta(V_{GS}-V_T)}(1-e^{-\beta V_{DS}}) \tag{5.8}$$

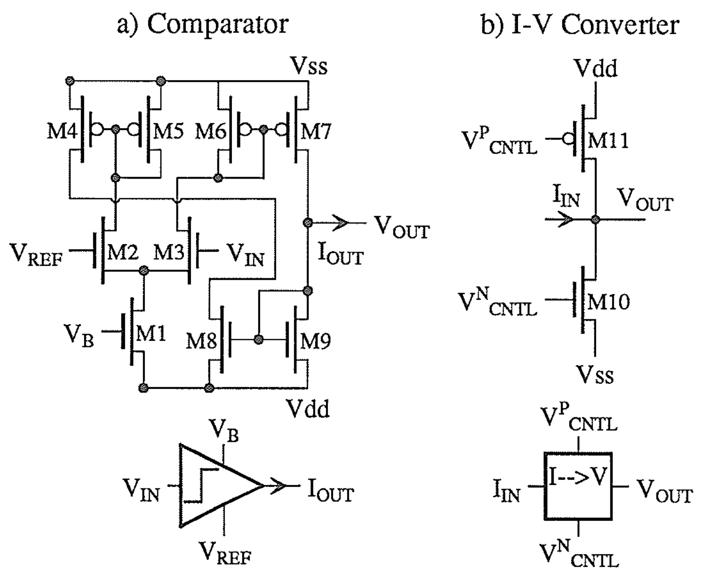

Figure 5.10 *a) Comparator,* $V_{OUT} = Vdd$ *when* $V_{IN} > V_{REF}$, $V_{OUT} = Vss$ *when* $V_{IN} \leq V_{REF}$*; b) Current to voltage converter,* $V_{OUT} = cI_{IN}$.

and for large V_{DS},

$$I_{DS} = I_0 \frac{W}{L} e^{\beta(V_{GS}-V_T)} \tag{5.9}$$

where $V_{GS} < V_T$, the constant I_0 absorbs a number of process parameters, and $\beta = \frac{q}{kT}$.

Figs. 5.13 and 5.14 plot I_{CAP} as a function of *LPulse* for access transistors M_1 and M_2. The exponential relationship between the access transistors' gate voltages and I_{CAP} appears as a straight line on these log plots. We require a learning current I_{CAP} in the $2 - 20nA$ range to achieve a learning rate in the 10^{-4} to 10^{-3} of full weight swing range. The exponential subthreshold MOSFET characteristic has the advantage of providing a tremendous range of learning rates: however, care must be taken to achieve adequate matching between the clamped and unclamped learning phases.

5.7.2 Capacitive Coupling Effects

During weight learning the access transistors' gate voltage is switched rapidly in each learning cycle: thus there is a potential problem with capacitive coupling between the gates of the access transistors and the weight storage capacitor. Even if the storage capacitor is much

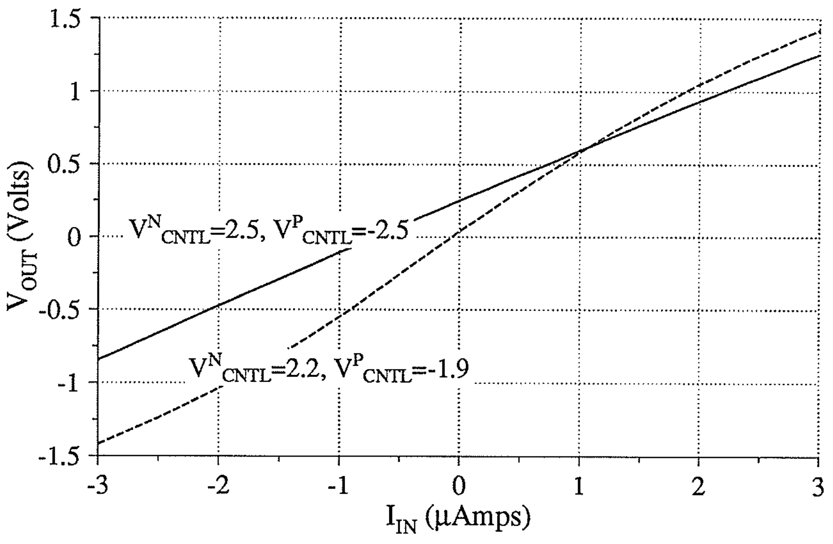

Figure 5.11 *Summation current to voltage converter, V_{OUT} as a function of I_{IN} (1.2μm CMOS test chip, $Vdd = 2.5V$, $Vss = -2.5V$, data collected using HP4145A).*

larger than the access transistors' gate capacitances, the capacitive coupling effect can still be large when compared to the small weight change. Fortunately, the small *LPulse* that is required for subthreshold transistor activations also prevents capacitive coupling between the drain and the gate (Fig. 5.15). In the subthreshold operation, most of the gate capacitance is between the bulk and the gate, and as long as the transistor is in saturation, the remainder is between the gate and the source. Only a small linear overlap capacitance couples the gate and the drain.

5.7.3 Matching Between Phases

Considerable care was taken to ensure that learning in the clamped and unclamped phases is well matched. If matching was not important, then a single weight storage capacitor could be used, and the unclamped phase would simply involve negating the product LV_iLV_j. A single capacitor weight storage scheme relies on matching between transistors M_1 and M_2 to achieve matching between the learning phases. Unfortunately, it is not possible to reliably match N and P channel transistors, particularly for operation in the subthreshold region. Thus

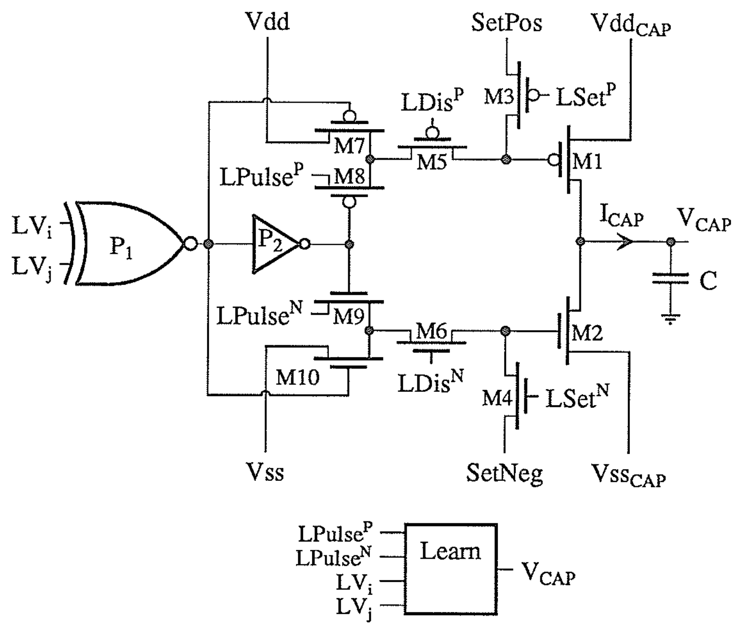

Figure 5.12 *Modified Manhattan synaptic learning circuit.*

we adopted a two capacitor differential voltage scheme, in which only matching between same-polarity, same-size transistors is required. The access transistors are also all oriented in the same direction, and have large channel regions to further improve matching. Figs. 5.13 and 5.14 show the matching between four transistors from two different test chips. Typical mismatch between access transistors on the same chip is 5% (N-channel) and 3% (P-channel). Between chips, the N-channel mismatch is 40% and the P-channel, 30%.

Differential weight storage causes an additional complication: we must keep the common mode weight voltage $(V_{CL} + V_{UN})/2$ in the proper range for the summation multiplier (Fig. 5.9). The following scheme is used. Capacitor voltages are initialized to approximately $-1V$. A training pass is made through the entire set of training data. Before the second pass through the training set, *InvLVi* is used to invert the learning product LV_iLV_j. This is possible because the non-inverted learning activation LV_i^{NI} is the actual source of LV_j: otherwise, if both terms in LV_iLV_j were inverted, then the sign of the product LV_iLV_j would remain unchanged. In this inverted mode, clamped weight changes are applyed to the *unclamped* capacitor, and visa-versa. *InvLVi* is used in this manner for every second pass through the training set; the effect

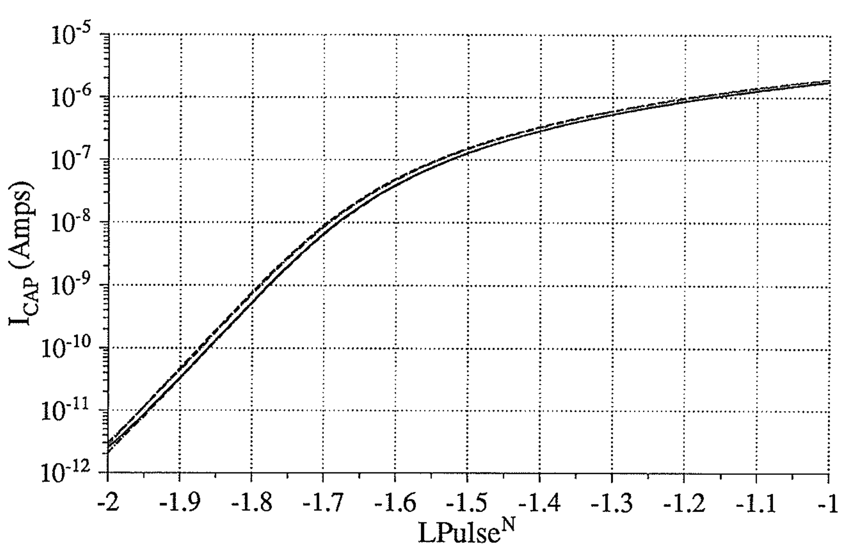

Figure 5.13 *Learning current I_{CAP} as a function of $LPulse^N$; four measurements from two test chips illustrated. Typical variation in I_{CAP} ($LPulse^N = 1.725V$): 5% on same chip, 40% between chips. (1.2µm CMOS test chip, $LPulse^P = 2.5V$, $Vdd = 2.5V$, $Vss = -2.5V$, data collected using HP4145A).*

of this procedure is to keep the common mode weight voltages roughly centered. Weight decay may also be used to keep common mode voltages in range.

5.7.4 Charge Leakage

Leakage from weight storage capacitors is less than $1pA$ at room temperature, and even modest cooling reduces the leakage current dramatically [Schwartz-2]. Differential weight storage provides some additional immunity from leakage, as the stored capacitor voltages will decay simultaneously, and their difference will be maintained. We anticipate that these circuits will be used in ANNs where training data is interspersed with patterns that the network is supposed to classify: in this case, weights will be continually refreshed by the occasional training patterns, and weight decay becomes a less serious problem. An additional benefit of this continuous learning process is that it provides a way for the network to compensate for component drift, as the temperature of the circuit changes.

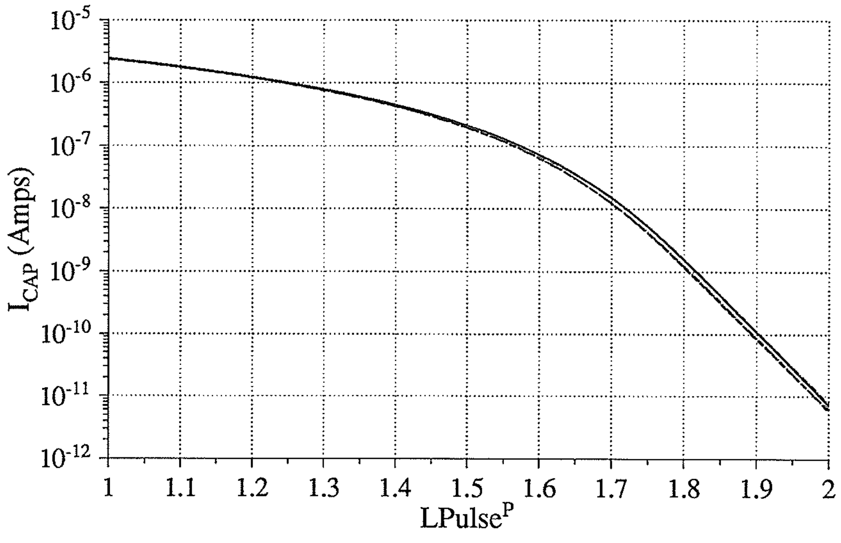

Figure 5.14 *Learning current I_{CAP} as a function of $LPulse^P$; four measurements from two test chips illustrated. Typical variation in I_{CAP} ($LPulse^P$ = 1.75V): 3% on same chip, 30% between chips. (1.2μm CMOS test chip, $LPulse^N = -2.5V$, $Vdd = 2.5V$, $Vss = -2.5V$, data collected using HP4145A).*

5.8 Power Dissipation

Power dissipation is an important consideration for VLSI circuits. Typical DC power consumption for these circuits is in the $54\mu W/synapse$ range, or about $200mW/cm^2$. The digital circuits which implement modified Manhattan learning have a high transient power consumption, but as this circuit runs at a fairly low clock rate, the average total power consumption is approximately 50% more than the DC power consumption.

5.9 System Testing

We used an ASIX-II IC tester to perform a series of circuit tests. Two typical tests are described below. Fig. 5.16 illustrates three learning test runs, in which a $100kHz$ learning pulse frequency with $3\mu s$ pulse widths was used. The weight storage capacitor voltage was read using the summation multiplier operating in its linear region, as connecting measurement equipment directly to the capacitor storage node would alter circuit behavior drastically. For

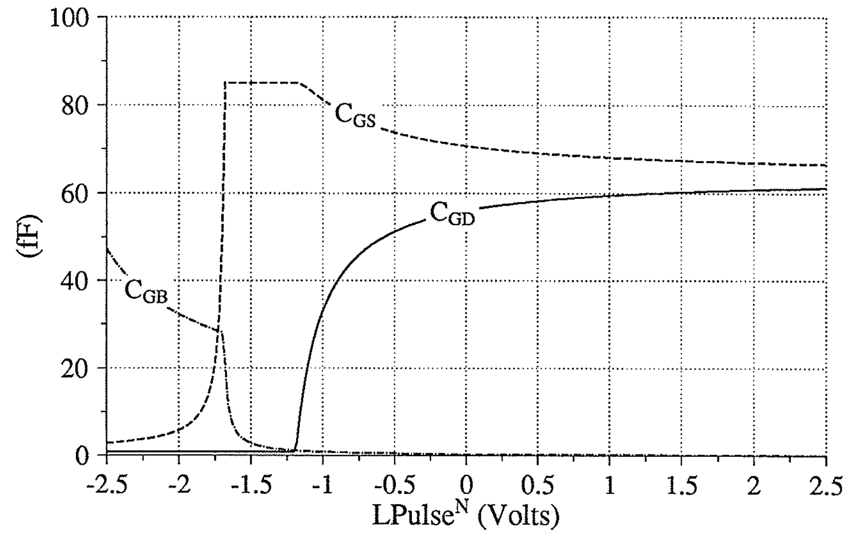

Figure 5.15 *C_{GB}, C_{GS}, C_{GD} as a function of $LPulse^N$ for $V_{CAP} = -2V$. (HSPICE MOSFET model level 3 simulation).*

these tests, V_{UN} was held constant, and $LCPulse^P$ pulses were applied as shown in Fig. 5.16. Three cases are illustrated, for $LCPulse^P = 1.7$, 1.75, and $1.8V$, giving learning rates of 3.2×10^{-3}, 1.7×10^{-3} and 6.7×10^{-4} respectively. Fairly high learning rates were used for these tests so that weight changes could be measured above background noise. The low learning pulse rate ($100kHz$) was used because the summation multiplier does not have sufficient drive to supply large off-chip capacitances at rates of more than a few hundred kilo-Hertz.

In Fig. 5.17, the same test was repeated, in this case with a $400kHz$ pulse rate and $50ns$ pulse widths. Since the pulses were narrow, a larger $LPulse^P$ was used to give learning steps that are visible over background noise. The test of Fig. 5.17 illustrates that the learning circuit operates properly with short learning pulses. Combining the results shown in Figs. 5.16 and 5.17, it is evident that this circuit can achieve very low learning rates, by using small, narrow learning pulses.

Similar tests were performed to evaluate the remainder of the synaptic and neuron circuitry: all components functioned properly. However, a defect in the neuron circuit prevented testing of the 19 neuron network: the neuron operated correctly in a small test circuit, but had insufficient drive in its learning circuitry to drive a full array of synapses. This problem may

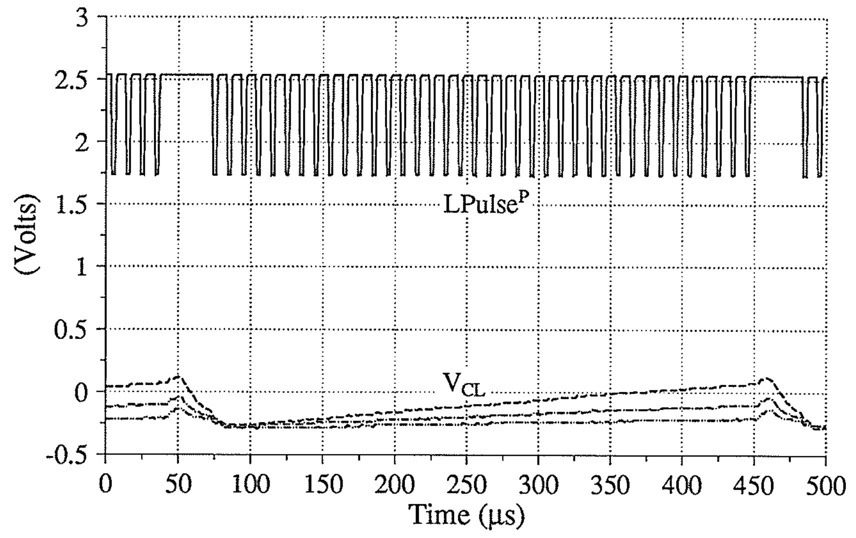

Figure 5.16 *Weight up learning test,* 100*kHz pulse rate,* 3μ*s VPulse*^{P} *pulses, VPulse*^{P} = 1.7, 1.75 and 1.8*V, giving learning rates of* 3.2×10^{-3}, 1.7×10^{-3} *and* 6.7×10^{-4} *(test data generated using ASIX-2 IC tester, analog data collected using Tektronix 2232 digital storage oscilloscope).*

be readily solved by increasing the drive of inverter P_8 in Fig. 5.2, a change that requires very little circuit redesign. Our testing indicates that despite this flaw, these analog circuits are suitable for implementing an analog CMOS contrastive learning ANN.

5.10 Discussion

The basic mean field implementation described above may be extended in a number of ways. The papers discussed below each take basic supervised learning ANNs and improve their performance by modifying their structure. These types of enhancements may be readily incorporated into our circuit designs, and may in fact address some of the problems associated with implementing ANNs with analog hardware.

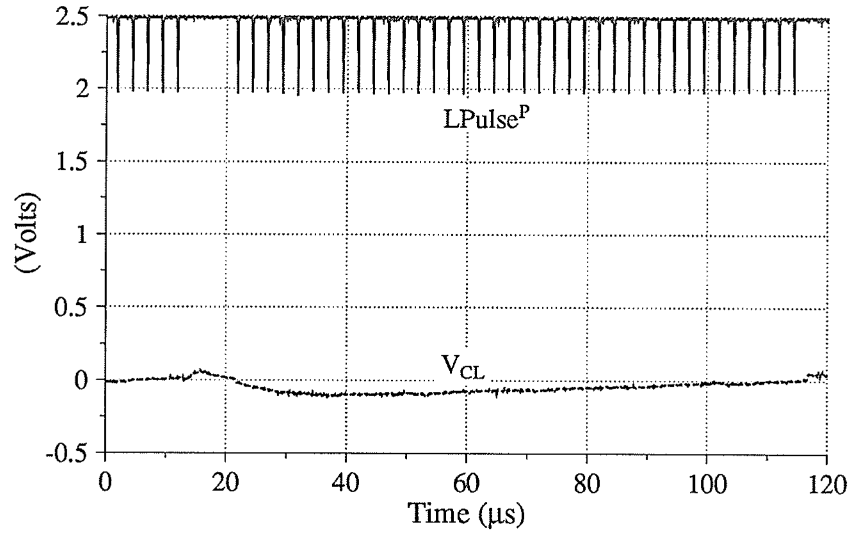

Figure 5.17 *Short pulse weight up learning test,* 400*kHz pulse rate,* 50*ns VPulse^P pulses, VPulse^P* = 1.4*V, learning rate:* 1.2×10^{-3} *(test data generated using ASIX-2 IC tester, analog data collected using Tektronix 2232 digital storage oscilloscope).*

5.10.1 Fast and Slow Synaptic Weights

In standard ANN models synaptic weights are represented by a single weight value, W_{ij}. Hinton and Plaut [Hinton-3] describe interesting properties of ANNs with *two* weights: a slowly-changing weight which stores long-term knowledge, and a fast-changing weight with a short decay time which stores temporary knowledge. The effective synaptic weight is the sum of fast and slow weights. Supervised learning affects both weights, but the learning rate of the fast weights is higher than that of the slow weights, and consequently the fast weights are strongly biased towards recent training data.

The temporary overlaying of fast weights gives the ANN a *temporary context*, resulting in more flexible information processing [Hinton-3]. Fast and slow weights have a number of applications. If an ANN learns a set of associations, which are subsequently "blurred" by later training data, *all* the original associations may be "deblurred" by rehearsing with a small number of patterns from the original training set. The rehearsal process results in learning fast weight values which compensate for the later training of the slow weights. Thus, a temporary context of the earlier associations is created.

Other benefits of fast weights include the ability to perform a type of recursive processing, in which the fast weights are used to store a temporary context during recursive computation. The state of the system is not stored as the activation of a set of neurons, and thus the same neurons can participate in computation at multiple levels of recursion.

Fast and slow weights may be implemented in a variety of ways using analog CMOS circuits. Fast weights can be added to the circuits described in this chapter by incorporating a second set of weight storage capacitors and summation multipliers in each synaptic circuit. Analog EEPROM weight storage [Shoemaker] presents an intriguing possibility. In this case, EEPROMs would store the slow weights, and capacitors the fast weights. This approach has the advantage that it eliminates the need for constant weight refreshing, since an EEPROM will maintain its charge for years. The disadvantage of this approach is that a special IC fabrication technology with EEPROM capability is required, and that adjusting EEPROM weights is an involved procedure [Holler].

5.10.2 Hybrid Network Architectures

It is natural to decompose a large problem into several sub-tasks, and then combine the solutions of each of the sub-tasks to arrive at a solution of the original problem. In the case of ANNs, this approach implies that a problem may be solved more efficiently by some sort of hierarchically structured network, in which small neural networks solve specific aspects of the entire problem, and then the individual results are combined into a final solution. This hierarchical approach is consistent with human brain function: the brain is partitioned into subsystems which perform a specific operation, thereby contributing to the solution of a larger problem. Whether this hierarchical structure should be at most a few levels deep, with bottom-level sub-systems performing complex computation, or whether the hierarchical structure should be very deep, with bottom-level sub-systems consisting of only a few neurons, is an unresolved issue.

Jacobs et al [Jacobs] present a two-level hierarchical supervised learning ANN in which small ''local expert'' networks solve sub-problems, whose results are combined by a supervisory gating network. This system may be viewed as a competitive learning network in which each hidden unit consists of a small sub-network. The operation of both the local expert networks and the gating network is determined by the weight training procedure. The learning algorithm that Jacobs et al use decouples the training of the sub-networks: the goal of a local expert for a given training case is not directly affected by the weights within other local experts [Jacobs]. There is still some indirect interaction between the sub-networks, but the *sign* of the error that a local expert senses for a particular training pattern does not depend on other portions of the network. Jacobs et al report that this hierarchical network arrangement avoids the strong interference effects that occur when a single non-hierarchical multilayer network is used to perform different sub-tasks on different occasions.

As the size of a supervised learning ANN is increased, lower learning rates are needed, a requirement that is hard to satisfy using analog circuitry. Thus, an additional advantage of a hierarchical network structure is that a hierarchy of small networks is better suited to analog implementation than a single large network.

5.11 Summary

We have designed, fabricated and tested analog CMOS circuits for constructing fully connected ANNs with contrastive Hebbian learning at each synapse. In developing these circuits, system-level simulations of CHL networks were used to discover which operations had to be implemented accurately, and which operations were tolerant of low-precision calculation. Thus, we were able to use compact analog circuits for most computations, and care was taken to implement critical operations accurately. We believe that this approach to analog ANN implementation, where only a few key computations are implemented accurately, may be used for a variety of other ANN architectures.

CHAPTER 6

Conclusion

The investigations presented in this work all address the basic question of what types of ANN architectures may be implemented with low precision analog circuitry. The reason for this interest in analog implementations, as we have stated previously, is that analog computations can be implemented far more efficiently than their digital counterparts. Although our work is restricted to a particular implementation medium, namely CMOS VLSI, this question of analog versus digital computation has far-reaching consequences. The same principles and tradeoffs will likely apply to future implementation technologies as well. The history of artificial computation shows that the rivalry between digital and analog computers is not new.

6.1 Analog Computation and ANNs

Before the microelectronics revolution began in the 1960's, analog computers were used for a variety of applications. With the advent of integrated circuit technology, digital systems became fast, reliable, and ultimately, inexpensive: as a result, digital computation has dominated for the past thirty years. An important reason for this dominance is that conventional computer architectures rely upon basic features of digital computation, such as its precision and repeatability, features which analog computation does not possess.

The resurgence of ANN research in the 1980's resulted in the development of a variety of ANN designs which are potential candidates for analog implementation. The common feature of these ANN architectures is that the system does not require precise, repeatable, error-free components: rather, some collective property of the entire system tolerates unreliable and inaccurate components. Our investigations, and those of other researchers, show that analog VLSI is suitable for the implementation of these types of artificial neural networks, but a great deal of additional research is required.

Our work with contrastive Hebbian learning ANNs demonstrates that system architectures that are tolerant of many component imperfections can still require that some operations be performed accurately. This raises the question of whether further ANN research will lead to architectures that do not have *any* critical operations. As always, biological neural networks provide an existence proof that such an architecture must exist. An important guide for this future ANN research may be the basic properties and underlying physics of the medium which will ultimately be used to implement the ANN. Architecture and implementation research should be an interactive process, so that an attempt is made to use the 'natural' capabilities of the underlying hardware.

6.2 Analog Circuit Design and Testing

This work has demonstrated the importance of accurate simulation for analog circuit research. Although simulation is also a key element of digital circuit design, analog work places much more stringent requirements on simulation tools. Even second-order effects, such as channel-length modulation in MOSFETS, must be modeled accurately. Thus, an ongoing effort to improve simulation accuracy through fine-tuning of model parameters is a worthwhile undertaking. Our experience shows that with careful design, one design and fabrication cycle will result in functional analog chips.

Analog IC testing is complicated by the lack of analog-oriented test equipment. The ASIX-II digital IC tester provides some of the functions required for our investigations, but lacks the ability to generate a variety of analog signals. In addition, analog voltage measurement equipment with a very high input impedance would be an asset.

6.3 Analog VLSI versus Neural Biology

The basic goal of ANN research is to develop artificial systems which will perform the functions associated with biological intelligence. Thus, it is interesting to make comparisons between artificial and biological systems. Below, we compare CMOS VLSI and neural biology [McClelland] in terms of *integration density*: this allows us to compare the complexity of artificial and biological systems.

The *neocortex* is the center of high-level information processing in the mammalian brain. It is a highly convoluted sheet of neurons, between 1.5 and $5mm$ thick, with a total area of approximately $2\,000cm^2$ in humans, and $200cm^2$ in macaque monkeys. Interestingly, the neocortex is basically a two dimensional structure, even though neurons are stacked vertically. This is an encouraging indication that full three dimensional connectivity is not required for artificial neural systems. The density of neurons in the neocortex is approximately 80 000 per mm^2, independent of the sheet thickness, for a wide variety of species [McClelland]. The one exception to this rule appears to be the striate cortex of primates, where there are approximately 200 000 neurons per mm^2. Each neuron in the neocortex typically receives inputs from 1 000 to 10 000 synapses.

The $1.2\mu m$ CMOS VLSI technology which we used to implement the circuits of chapter 5 has an integration density of approximately 63 000 transistor-sized cells per mm^2, or about 0.78 transistor cells per neocortex neuron. For a $0.5\mu m$ technology, this figure rises to approximately 4.5 transistor-sized cells per neuron. An area of at least 100 of these cells is required to make a simple circuit.

This analysis ignores the 1 000 to 10 000 synapses connected to each neuron: when these are taken into account, it is apparent that a biological brain has an integration density of *at least* 10^5 to 10^6 times that of a $0.5\mu m$ VLSI technology. Finally, if we compare a single $1cm^2$ chip to the $2\,000cm^2$ neocortex, we conclude that the brain has at least $2{\times}10^8$ to $2{\times}10^9$ times the complexity of a state of the art IC. The actual figure may be higher still if current ANN models grossly oversimplify computationally important aspects of neural behavior.

Although these calculations are highly approximate, they give an indication that ANNs will not rival the complexity of mammalian brains, given current IC technology. However, even extremely simple nervous systems, such as that of *Aplysia*, perform remarkable computational feats. Thus we conclude that although we cannot hope to build an artificial human brain, we can certainly construct ANNs for a host of important artificial intelligence applications.

APPENDIX 1

HSPICE Simulation Parameters

This appendix contains HSPICE simulation parameters for Northern Telecom's 3μm CMOS3 and 1.2μm CMOS4S fabrication processes. These parameters were used for all circuit simulations conducted as part of this work.

Ap-1.1 CMOS3 Parameters

```
.MODEL N.1 NMOS (
+     LMIN=0E-6
+     LMAX=6E-6
+     WMIN=0
+     WMAX=1
+     UO=800
+     THETA=0.05
+     GAMMA=1.11
+     KAPPA=0.8
+      LEVEL=3
+      VTO=0.702
+      NFS=1.541E+11
+      TPG=1.0
+      TOX=5.048E-08
+      NSUB=2.022E+16
+      VMAX=2.306E+05
+      XJ=1.132E-07
+      LD=2.693E-07
+      DELTA=0.6
+      ETA=0.1
+      PB=0.800
+      IS=1.000E-16
+      JS=1.000E-04
+      CJ=4.090E-04
+      MJ=0.498
+      CJSW=4.780E-10
+      MJSW=0.363
+      CGSO=2.910E-10
+      CGDO=2.910E-10
+      FC=0.500
+     )
.MODEL N.2 NMOS (
+     LMIN=6E-6
+     LMAX=12E-6
+     WMIN=0
+     WMAX=1
+     UO=740
+     THETA=0.05
+     GAMMA=1.1325
+     KAPPA=2.8
+      LEVEL=3
+      VTO=0.702
+      NFS=1.541E+11
+      TPG=1.0
+      TOX=5.048E-08
+      NSUB=2.022E+16
+      VMAX=2.306E+05
+      XJ=1.132E-07
+      LD=2.693E-07
+      DELTA=0.6
+      ETA=0.1
+      PB=0.800
+      IS=1.000E-16
+      JS=1.000E-04
+      CJ=4.090E-04
+      MJ=0.498
+      CJSW=4.780E-10
+      MJSW=0.363
+      CGSO=2.910E-10
+      CGDO=2.910E-10
+      FC=0.500
+     )
.MODEL N.3 NMOS (
+     LMIN=12E-6
+     LMAX=18E-6
+     WMIN=0
+     WMAX=1
+     UO=700
+     THETA=0.045
+     GAMMA=1.155
+     KAPPA=7.533
+      LEVEL=3
+      VTO=0.702
+      NFS=1.541E+11
+      TPG=1.0
+      TOX=5.048E-08
+      NSUB=2.022E+16
+      VMAX=2.306E+05
+      XJ=1.132E-07
+      LD=2.693E-07
+      DELTA=0.6
+      ETA=0.1
+      PB=0.800
+      IS=1.000E-16
+      JS=1.000E-04
+      CJ=4.090E-04
+      MJ=0.498
+      CJSW=4.780E-10
+      MJSW=0.363
+      CGSO=2.910E-10
+      CGDO=2.910E-10
+      FC=0.500
+     )
.MODEL N.4 NMOS (
+     LMIN=18E-6
+     LMAX=24E-6
+     WMIN=0
+     WMAX=1
+     UO=660
+     THETA=0.04
+     GAMMA=1.1775
+     KAPPA=12.27
+      LEVEL=3
+      VTO=0.702
+      NFS=1.541E+11
+      TPG=1.0
+      TOX=5.048E-08
+      NSUB=2.022E+16
+      VMAX=2.306E+05
+      XJ=1.132E-07
+      LD=2.693E-07
+      DELTA=0.6
+      ETA=0.1
+      PB=0.800
+      IS=1.000E-16
+      JS=1.000E-04
+      CJ=4.090E-04
+      MJ=0.498
+      CJSW=4.780E-10
+      MJSW=0.363
+      CGSO=2.910E-10
+      CGDO=2.910E-10
+      FC=0.500
+      )
.MODEL N.5 NMOS (
+     LMIN=24E-6
+     LMAX=1
+     WMIN=0
+     WMAX=1
+     UO=620
+     THETA=0.035
+     GAMMA=1.2
+     KAPPA=17
+      LEVEL=3
+      VTO=0.702
+      NFS=1.541E+11
+      TPG=1.0
+      TOX=5.048E-08
+      NSUB=2.022E+16
+      VMAX=2.306E+05
+      XJ=1.132E-07
+      LD=2.693E-07
+      DELTA=0.6
+      ETA=0.1
+      PB=0.800
+      IS=1.000E-16
+      JS=1.000E-04
+      CJ=4.090E-04
+      MJ=0.498
+      CJSW=4.780E-10
+      MJSW=0.363
+      CGSO=2.910E-10
+      CGDO=2.910E-10
+      FC=0.500
+      )
.MODEL P.1 PMOS (
+     LMIN=0E-6
+     LMAX=6E-6
+     WMIN=0
+     WMAX=1
+     UO=280
+     KAPPA=3.5
+      LEVEL=3
+      VTO=-0.769
+      NFS=4.121E+11
+      TPG=1.0
+      TOX=5.048E-08
+      NSUB=3.843E+15
+      VMAX=1.61E5
+      XJ=3.091E-07
+      LD=1.686E-07
+      DELTA=.9
+      THETA=.1
+      ETA=1.2
+      PB=0.800
+      IS=1.000E-16
+      JS=1.000E-04
+      CJ=1.440E-04
+      MJ=0.621
+      CJSW=3.360E-10
+      MJSW=0.434
+      CGSO=2.370E-10
+      CGDO=2.370E-10
+      FC=0.500
+      )
.MODEL P.2 PMOS (
+     LMIN=6E-6
+     LMAX=12E-6
+     WMIN=0
+     WMAX=1
+     UO=250
+     KAPPA=35
+      LEVEL=3
+      VTO=-0.769
+       NFS=4.121E+11
+       TPG=1.0
+       TOX=5.048E-08
+       NSUB=3.843E+15
+       VMAX=1.61E5
+       XJ=3.091E-07
+       LD=1.686E-07
+       DELTA=.9
+       THETA=.1
+       ETA=1.2
+       PB=0.800
+       IS=1.000E-16
+       JS=1.000E-04
+       CJ=1.440E-04
+       MJ=0.621
+       CJSW=3.360E-10
+       MJSW=0.434
+       CGSO=2.370E-10
+       CGDO=2.370E-10
+       FC=0.500
+       )
.MODEL P.3 PMOS (
+      LMIN=12E-6
+      LMAX=18E-6
+      WMIN=0
+      WMAX=1
+      UO=250
+      KAPPA=73.33
+       LEVEL=3
+       VTO=-0.769
+       NFS=4.121E+11
+       TPG=1.0
+       TOX=5.048E-08
+       NSUB=3.843E+15
+       VMAX=1.61E5
+       XJ=3.091E-07
+       LD=1.686E-07
+       DELTA=.9
+       THETA=.1
+       ETA=1.2
+       PB=0.800
+       IS=1.000E-16
+       JS=1.000E-04
+       CJ=1.440E-04
+       MJ=0.621
+       CJSW=3.360E-10
+       MJSW=0.434
+       CGSO=2.370E-10
+       CGDO=2.370E-10
+       FC=0.500
+       )
.MODEL P.4 PMOS (
+      LMIN=18E-6
+      LMAX=24E-6
+      WMIN=0
+      WMAX=1
+      UO=250
+      KAPPA=111.67
+       LEVEL=3
+       VTO=-0.769
+       NFS=4.121E+11
+       TPG=1.0
+       TOX=5.048E-08
+       NSUB=3.843E+15
+       VMAX=1.61E5
+       XJ=3.091E-07
+       LD=1.686E-07
+       DELTA=.9
+       THETA=.1
+       ETA=1.2
+       PB=0.800
+       IS=1.000E-16
+       JS=1.000E-04
+       CJ=1.440E-04
+       MJ=0.621
+       CJSW=3.360E-10
+       MJSW=0.434
+       CGSO=2.370E-10
+       CGDO=2.370E-10
+       FC=0.500
+       )
.MODEL P.5 PMOS (
+      LMIN=24E-6
+      LMAX=1
+      WMIN=0
+      WMAX=1
+      UO=250
+      KAPPA=150
+       LEVEL=3
+       VTO=-0.769
+       NFS=4.121E+11
+       TPG=1.0
+       TOX=5.048E-08
+       NSUB=3.843E+15
+       VMAX=1.61E5
+       XJ=3.091E-07
+       LD=1.686E-07
+       DELTA=.9
+       THETA=.1
+       ETA=1.2
+       PB=0.800
+       IS=1.000E-16
+       JS=1.000E-04
+       CJ=1.440E-04
+       MJ=0.621
+       CJSW=3.360E-10
+       MJSW=0.434
+       CGSO=2.370E-10
+       CGDO=2.370E-10
+       FC=0.500
+       )
```

Ap-1.2 CMOS4S Parameters

```
.MODEL N.1 NMOS (
+       capop=2
+       LMIN=0E-6
+       LMAX=2.4E-6
+       WMIN=0
+       WMAX=1
+       UO=566.3
+       KAPPA=2.000E-17
+        CGDO=1.973E-10
+        CGSO=1.973E-10
+        CJ=2.900E-4
+        CJSW=3.3E-10
+        DELTA=0.3551
+        ETA=9.814E-2
+        FC=0.500
+        IS=1.000E-16
+        JS=1.00E-4
+        LD=1.157E-7
+        LEVEL=3
+        MJ=0.486
+        MJSW=0.330
+        NEFF=1.000
+        NFS=5.764E+11
+        NSUB=1.4E+16
+        PB=0.8
+        RD=26.77
+        RS=26.77
+        THETA=6.574E-2
+        TOX=2.502E-8
+        TPG=1.000
+        UCRIT=1.000E+4
+        VMAX=1.651E+5
+        VTO=0.7572
+        XJ=1.165E-7
+        XQC=1.000
+        )

.MODEL N.2 NMOS (
+       capop=2
+       LMIN=2.4E-6
+       LMAX=4.8E-6
+       WMIN=0
+       WMAX=1
+       UO=587.4
+       KAPPA=.5714
+        CGDO=1.973E-10
+        CGSO=1.973E-10
+        CJ=2.900E-4
+        CJSW=3.3E-10
+        DELTA=0.3551
+        ETA=9.814E-2
+        FC=0.500
+        IS=1.000E-16
+        JS=1.00E-4
+        LD=1.157E-7
+        LEVEL=3
+        MJ=0.486
+        MJSW=0.330
+        NEFF=1.000
+        NFS=5.764E+11
+        NSUB=1.4E+16
+        PB=0.8
+        RD=26.77
+        RS=26.77
+        THETA=6.574E-2
+        TOX=2.502E-8
+        TPG=1.000
+        UCRIT=1.000E+4
+        VMAX=1.651E+5
+        VTO=0.7572
+        XJ=1.165E-7
+        XQC=1.000
+        )

.MODEL N.3 NMOS (
+       capop=2
+       LMIN=4.8E-6
+       LMAX=7.2E-6
+       WMIN=0
+       WMAX=1
+       UO=608.5
+       KAPPA=1.1429
+        CGDO=1.973E-10
+        CGSO=1.973E-10
+        CJ=2.900E-4
+        CJSW=3.3E-10
+        DELTA=0.3551
+        ETA=9.814E-2
+        FC=0.500
+        IS=1.000E-16
+        JS=1.00E-4
+        LD=1.157E-7
+        LEVEL=3
+        MJ=0.486
+       MJSW=0.330
+       NEFF=1.000
+       NFS=5.764E+11
+       NSUB=1.4E+16
+       PB=0.8
+       RD=26.77
+       RS=26.77
+       THETA=6.574E-2
+       TOX=2.502E-8
+       TPG=1.000
+       UCRIT=1.000E+4
+       VMAX=1.651E+5
+       VTO=0.7572
+       XJ=1.165E-7
+       XQC=1.000
+       )

.MODEL N.4 NMOS (
+      capop=2
+      LMIN=7.2E-6
+      LMAX=9.6E-6
+      WMIN=0
+      WMAX=1
+      UO=629.6
+      KAPPA=1.7143
+       CGDO=1.973E-10
+       CGSO=1.973E-10
+       CJ=2.900E-4
+       CJSW=3.3E-10
+       DELTA=0.3551
+       ETA=9.814E-2
+       FC=0.500
+       IS=1.000E-16
+       JS=1.00E-4
+       LD=1.157E-7
+       LEVEL=3
+       MJ=0.486
+       MJSW=0.330
+       NEFF=1.000
+       NFS=5.764E+11
+       NSUB=1.4E+16
+       PB=0.8
+       RD=26.77
+       RS=26.77
+       THETA=6.574E-2
+       TOX=2.502E-8
+       TPG=1.000
+       UCRIT=1.000E+4
+       VMAX=1.651E+5
+       VTO=0.7572
```

```
+       XJ=1.165E-7
+       XQC=1.000
+       )

.MODEL N.5 NMOS (
+      capop=2
+      LMIN=9.6E-6
+      LMAX=1
+      WMIN=0
+      WMAX=1
+      UO=640
+      KAPPA=2
+       CGDO=1.973E-10
+       CGSO=1.973E-10
+       CJ=2.900E-4
+       CJSW=3.3E-10
+       DELTA=0.3551
+       ETA=9.814E-2
+       FC=0.500
+       IS=1.000E-16
+       JS=1.00E-4
+       LD=1.157E-7
+       LEVEL=3
+       MJ=0.486
+       MJSW=0.330
+       NEFF=1.000
+       NFS=5.764E+11
+       NSUB=1.4E+16
+       PB=0.8
+       RD=26.77
+       RS=26.77
+       THETA=6.574E-2
+       TOX=2.502E-8
+       TPG=1.000
+       UCRIT=1.000E+4
+       VMAX=1.651E+5
+       VTO=0.7572
+       XJ=1.165E-7
+       XQC=1.000
+       )

.MODEL P.1 PMOS (
+      capop=2
+      LMIN=0E-6
+      LMAX=2.4E-6
+      WMIN=0
+      WMAX=1
+      UO=185
+      KAPPA=4
+       CGDO=3.284E-10
+      CGSO=3.284E-10
+      CJ=4.100E-4
+      CJSW=3.4E-10
+      DELTA=0.4598
+      ETA=6.729E-2
+      FC=0.500
+      IS=1.000E-16
+      JS=1.00E-4
+      LD=1.364E-7
+      LEVEL=3
+      MJ=0.540
+      MJSW=0.300
+      NEFF=1.000
+      NFS=5.864E+11
+      NSUB=2.011E+16
+      PB=0.8
+      RD=58.90
+      RS=58.90
+      THETA=0.1376
+      TOX=2.502E-8
+      TPG=1.000
+      UCRIT=1.000E+4
+      VMAX=2.823E+5
+      VTO=-0.8307
+      XJ=1.742E-7
+      XQC=1.000
+      )

.MODEL P.2 PMOS (
+     capop=2
+     LMIN=2.4E-6
+     LMAX=4.8E-6
+     WMIN=0
+     WMAX=1
+     UO=230
+     KAPPA=23
+      CGDO=3.284E-10
+      CGSO=3.284E-10
+      CJ=4.100E-4
+      CJSW=3.4E-10
+      DELTA=0.4598
+      ETA=6.729E-2
+      FC=0.500
+      IS=1.000E-16
+      JS=1.00E-4
+      LD=1.364E-7
+      LEVEL=3
+      MJ=0.540
+      MJSW=0.300
+      NEFF=1.000
+      NFS=5.864E+11
+      NSUB=2.011E+16
+      PB=0.8
+      RD=58.90
+      RS=58.90
+      THETA=0.1376
+      TOX=2.502E-8
+      TPG=1.000
+      UCRIT=1.000E+4
+      VMAX=2.823E+5
+      VTO=-0.8307
+      XJ=1.742E-7
+      XQC=1.000
+      )

.MODEL P.3 PMOS (
+     capop=2
+     LMIN=4.8E-6
+     LMAX=7.2E-6
+     WMIN=0
+     WMAX=1
+     UO=230
+     KAPPA=41.8
+      CGDO=3.284E-10
+      CGSO=3.284E-10
+      CJ=4.100E-4
+      CJSW=3.4E-10
+      DELTA=0.4598
+      ETA=6.729E-2
+      FC=0.500
+      IS=1.000E-16
+      JS=1.00E-4
+      LD=1.364E-7
+      LEVEL=3
+      MJ=0.540
+      MJSW=0.300
+      NEFF=1.000
+      NFS=5.864E+11
+      NSUB=2.011E+16
+      PB=0.8
+      RD=58.90
+      RS=58.90
+      THETA=0.1376
+      TOX=2.502E-8
+      TPG=1.000
+      UCRIT=1.000E+4
+      VMAX=2.823E+5
+      VTO=-0.8307
+      XJ=1.742E-7
+      XQC=1.000
+       )

.MODEL P.4 PMOS (
+       capop=2
+       LMIN=7.2E-6
+       LMAX=9.6E-6
+       WMIN=0
+       WMAX=1
+       UO=230
+       KAPPA=60.6
+        CGDO=3.284E-10
+        CGSO=3.284E-10
+        CJ=4.100E-4
+        CJSW=3.4E-10
+        DELTA=0.4598
+        ETA=6.729E-2
+        FC=0.500
+        IS=1.000E-16
+        JS=1.00E-4
+        LD=1.364E-7
+        LEVEL=3
+        MJ=0.540
+        MJSW=0.300
+        NEFF=1.000
+        NFS=5.864E+11
+        NSUB=2.011E+16
+        PB=0.8
+        RD=58.90
+        RS=58.90
+        THETA=0.1376
+        TOX=2.502E-8
+        TPG=1.000
+        UCRIT=1.000E+4
+        VMAX=2.823E+5
+        VTO=-0.8307
+        XJ=1.742E-7
+        XQC=1.000
+       )

.MODEL P.5 PMOS (
+       capop=2
+       LMIN=9.6E-6
+       LMAX=1
+       WMIN=0
+       WMAX=1
+       UO=230
+       KAPPA=70
+        CGDO=3.284E-10
+        CGSO=3.284E-10
+        CJ=4.100E-4
+        CJSW=3.4E-10
+        DELTA=0.4598
+        ETA=6.729E-2
+        FC=0.500
+        IS=1.000E-16
+        JS=1.00E-4
+        LD=1.364E-7
+        LEVEL=3
+        MJ=0.540
+        MJSW=0.300
+        NEFF=1.000
+        NFS=5.864E+11
+        NSUB=2.011E+16
+        PB=0.8
+        RD=58.90
+        RS=58.90
+        THETA=0.1376
+        TOX=2.502E-8
+        TPG=1.000
+        UCRIT=1.000E+4
+        VMAX=2.823E+5
+        VTO=-0.8307
+        XJ=1.742E-7
+        XQC=1.000
+       )
```

References

[Alspector-1]
J. Alspector, R. B. Allen and A. Jayakumar, "Relaxation Networks for Large Supervised Learning Problems", *Proc. NIPS-90*, paper VLSI-6, in press.

[Alspector-2]
J. Alspector and R. B. Allen, "A Neuromorphic VLSI Learning System", *Advanced Research in VLSI: Proc. of 1987 Stanford Conf.*, P. Losleben, editor, Cambridge: MIT Press, 1987, pp. 313-349.

[Arima]
Y. Arima, K. Mashiko, K. Okada, T. Yamada, A. Maeda, H. Kondoh, S. Kayano, "A Self-Learning Neural Network Chip with 125 Neurons and 10K Self-Organization Synapses", *IEEE J. Solid St. Ccts.*, 1991, Vol. 23, No. 4, pp. 607-611.

[Atlas]
L. E. Atlas and Y. Suzuki, "Digital Systems for Artificial Neural Networks", *IEEE Circuits and Devices Mag.*, pp. 20-24, Nov. 1989.

[Babanezhad]
J. N. Babanezhad and G. C. Temes, "A 20-V Four-Quadrant CMOS Analog Multiplier", *IEEE J. Solid-State Circuits*, Vol. SC-20, No. 6, 1985, pp. 1158-1168.

[Bult]
K. Bult and H. Wallinga, *A Class of Analog CMOS Circuits Based on the Square-Law Characteristic of an MOS Transistor in Saturation, IEEE J. Solid-State Circuits*, Vol. SC-22, No. 3, 1987, pp. 357-365.

[Card-1]
H. C. Card and W. R. Moore, "VLSI Devices and Circuits for Neural Networks", *Int. J. Neural Systems*, Vol. 1, pp. 149-165, 1989.

[Card-2]
H. C. Card and W. R. Moore, "Silicon Models of Associative Learning in *Aplysia*", *Neural Networks*, Vol. 3, pp. 333-446, 1990.

[Clark]
J. J. Clark, "An Analog CMOS Implementation of a Self Organizing Feed-forward Network", *1990 Int. Joint. Conf. on Neural Networks, Washington D. C.*, Vol. II, pp. 118-121, 1990.

[Furman]
B. Furman, J. White and A. A. Abidi, "CMOS Analog IC Implementing the Back Propagation Algorithm", *First Ann. INNS Mtg.*, Sept 6-10, 1988, Boston, MA. Abstract also published in *Neural Networks*, Vol. 1, Supp. 1, p. 381, 1988.

[Galland]
C. Galland and G. E. Hinton, "Deterministic Boltzmann Learning in Networks with Asymmetric Connectivity", *Connectionist Models: Proceedings of the 1990 Summer School*, D. S. Touretzky et al eds., 1990, pp. 3-9.

[Gilbert]
B. Gilbert, "A high-performance monolithic multiplier using active feedback", *IEEE J. Solid-State Circuits*, 1974, SC-9, pp. 364-373.

[Graf]
H. P. Graf and L. D. Jackel, "Analog Electronic Neural Network Circuits", *IEEE*

Circuits and Devices Mag., Vol. 5, No. 4, pp. 44-55, July 1989.

[Hawkins]
R. D. Hawkins and G. H. Bower, *Computational Models of Learning in Simple Neural Systems*, Academic Press, San Diego, CA, pp. 65-108, 1989.

[Hebb]
D. O. Hebb, *Organization of Behavior*, 1949, New York: John Wiley.

[Hinton-1]
G. E. Hinton, ''Deterministic Boltzmann Learning Performs Steepest Descent in Weight Space'', *Neural Computation*, Vol. 1. No. 1. 1989, pp. 143-150.

[Hinton-2]
G. E. Hinton and S. Becker, ''An Unsupervised Learning Procedure that Discovers Surfaces in Random Dot Stereograms'', *Proc. Int. Joint Conf. on Neural Networks, IJCNN-WASH 90*, M. Caudill, editor, 1990, Vol. 1, pp. 218-222.

[Hinton-3]
G. E. Hinton and D. C. Plaut, ''Using Fast Weights to Deblur Old Memories'', *Proc. Cognitive Sciences Conf., Seattle, WA*, 1987, pp. 177-186.

[Holler]
M. Holler, S. Tam, H. Castro and R. Benson, ''An Electrically Trainable Artificial Neural Network with 10240 Floating Gate Synapses'', *1989 Int. Joint Conf. on Neural Networks*, pp. II-191-196, 1989.

[Hollis]
P. W. Hollis and J. J. Paulos, ''Artificial Neural Networks Using MOS Analog Multipliers'', *IEEE J. Solid St. Ccts.*, Vol. 25, pp. 849-855, June 1990.

[Hopfield]
J. Hopfield, ''Neurons with Graded Response Have Collective Properties Like Those of Two-state Neurons'', *Proc. Natl. Acad. Sci.*, 1984, 81, pp. 3088-3092.

[Ismail-1]
S. Bibyk and M. Ismail, ''Issues in Analog VLSI and MOS Techniques for Neural Computing'', in *Analog VLSI Implementation of Neural Systems*, C. A. Mead and M. Ismail (eds.) Kluwer Academic Publishers, 1989.

[Ismail-2]
S. Bibyk and M. Ismail, ''Neural Network Building Blocks for Analog MOS VLSI'', in *Analogue IC design: the current mode approach*, C. Toumazou, F. J. Lidgey and D. G. Haigh (eds.) Peter Peregrinus Ltd., London, 1990.

[Jacobs]
R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton, ''Adaptive Mixtures of Local Experts'', *Neural Computation*, Vol. 3, No. 1, 1991.

[Kammen]
D. M. Kammen and A. L. Yuille, ''Spontaneous Symmetry Breaking Energy Functions and the Emergence of Orientation Selective Cortical Cells'', *Biological Cybernetics*, Vol. 59, 1988, pp. 23-31.

[Kandel-1]
E. R. Kandel and J. H. Schwartz, eds., *Principles of Neural Science*, 2nd edition, Elsevier, New York, 1985.

[Kandel-2]
E. R. Kandel, M. Klein, B. Hochner, M. Shuster, S. A. Siegelbaum, R. D. Hawkins, D. L. Glanzman, V. F. Castellucci and T. W. Abrams, Synaptic Modulation and Learning:

New Insights into Synaptic Transmission From the Study of Behavior. In G. M. Edelman, W. E. Gall and W. M. Cowan (Eds.), *Synaptic Function* (Chap. 19, pp. 471-518). John Wiley, New York, 1987.

[Kohonen-1]
T. Kohonen, "An Introduction to Neural Computing", *Neural Networks*, 1988, 1, pp. 3-16.

[Kohonen-2]
T. Kohonen, "The 'Neural' Phonetic Typewriter", *Computer*, March 1988, pp. 11-22.

[Kub]
F. J. Kub, K. K. Moon, I. A. Mack and F. M. Long, "Programmable Analog Vector-Matrix Multipliers", *IEEE Journal of Solid State Circuits*, Vol. 25, No. 1, 1990, pp. 207-214.

[Linsker]
R. Linsker, "Self-organization in a Perceptual Network", *Computer*, March 1988, pp. 105-117.

[McClelland]
J. L. McClelland and D. E. Rumelhart, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models*, MIT Press, Cambridge, MA., 1986.

[Mead]
C. A. Mead, *Analog VLSI and Neural Systems*, 1988, Reading: Addison-Wesley.

[Movellan]
Movellan, J. R., "Contrastive Hebbian Learning in the Continuous Hopfield Model", *Connectionist Models: Proceedings of the 1990 Summer School*, D. S. Touretzky et al eds., 1990, pp. 10-17.

[Mundie]
D. B. Mundie and L. W. Massengill, "Weight Decay and Resolution Effects in Feedforward Artificial Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 2, No. 1, 1991, pp. 168-170.

[Murray]
A. F. Murray and A. V. W. Smith, "Asynchronous VLSI Neural Networks Using Pulse Stream Arithmetic", *IEEE J. Solid St. Ccts.*, 1988, Vol. 23, pp. 688-697.

[Oja]
E. Oja, "A Simplified Neuron Model as a Principle Component Analyzer", *J. Math Biology*, Vol. 15, 1982, pp. 267-273.

[Peterson-1]
C. Peterson and J. R. Anderson, "A Mean Field Theory Learning Algorithm for Neural Networks", *Complex Systems*, No. 1, 1987, pp. 995-1019.

[Peterson-2]
C. Peterson and E. Hartman, "Explorations of the Mean Field Theory Learning Algorithm", *Neural Networks*, Vol. 2, pp. 475-494, 1989.

[Qin]
S. Qin and R. L. Geiger, *A* ±5-V CMOS Analog Multiplier, *IEEE J. Solid-State Circuits*, Vol. SC-22, No. 6, 1987, pp. 1143-1146.

[R. Schneider]
R. Schneider, Ph. D. Dissertation, University of Manitoba, 1991.

[Raffel]

J. Raffel, J. Mann, R. Berger, A. Soares and S. Gilbert, ''A Generic Architecture for Wafer Scale Neuromorphic Systems'', *Proc. IEEE Int. Conf. Neural Networks*, Vol. III, pp. 501-513, 1987.

[Rumelhart-1]

D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, MIT Press, Cambridge, MA., 1986.

[Rumelhart-2]

D. E. Rumelhart, G. E. Hinton, and R. J. Williams, ''Learning Representations by Back-Propagating Errors'', *Nature*, Vol. 323, 1986, pp. 533-536.

[Sage]

J. P. Sage, K. Thompson and R. S. Withers, ''An Artificial Neural Network Integrated Circuit Based on MNOS/CCD Principles'', in *AIP Conf. Proc. 151, Neural Networks for Computing, Snowbird*, J. S. Denker, editor, New York: American Institute of Physics, 1986, pp. 381-385.

[Satyanarayana]

S. Satyanarayana, Y. Tsividis and H. P. Graf, ''A Reconfigurable Analog VLSI Neural Network Chip'', *Adv. in Neural Info. Proc. Systems 2*, D. Touretzky, editor, Morgan Kaufmann, 1990.

[Schwartz-1]

D. B. Schwartz, R. E. Howard and W. E. Hubbard, ''A Programmable Analog Neural Network Chip'', *IEEE J. Solid St. Ccts.*, Vol. 24, pp. 313-319, April 1989.

[Schwartz-2]

D. B. Schwartz and V. K. Samalam, ''An Analog VLSI Splining Circuit'', *Proc. NIPS-90*, paper VLSI-4, in press.

[Shoemaker]

P. A. Shoemaker, I. Lagnado and R. Shimabukuro, ''Artificial Neural Network Implementation with Floating Gate MOS Devices'', in *Hardware Implementations of Neuron Nets and Synapses, NSF/ONR Workshop*, San Diego, CA, P. Mueller, editor, Jan 14-15, 1988, pp. 114-119.

[Sivilotti]

M. Sivilotti, M. Emerling and C. A. Mead, ''A Novel Associative Memory Implemented Using Collective Computation'', *Proc. Chapel Hill Conf. on VLSI*, 1985, pp. 329-342.

[Sze]

S. M. Sze, *Physics of Semiconductor Devices*, John Wiley and Sons, New York, 1981.

[Zador]

A. Zador, C. Koch, and T. H. Brown, ''Biophysical Model of a Hebbian Synapse'', *Proc. Int. Joint Conf. on Neural Networks, IJCNN-WASH 90*, M. Caudill, editor, 1990, Vol. 1, pp. 138-141.