MULTILAYER PERCEPTRON NEURAL NETWORK IN ANALOG VLSI - A SYSTEM LEVEL STUDY

by
Arif Selçuk Öğrenci
B.S. in E.E., Boğaziçi University, 1992
B.S. in Mathematics, Boğaziçi University, 1992
M.S. in E.E., Boğaziçi University, 1995

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor
of
Philosophy

Boğaziçi University
1999

MULTILAYER PERCEPTRON NEURAL NETWORK IN ANALOG VLSI - A SYSTEM LEVEL STUDY

APPROVED BY:

Assoc. Prof. Dr. Günhan Dündar .....
(Thesis Supervisor)

Assoc. Prof. Dr. Sina Balkır .....
(Thesis Co-Supervisor)

Assoc. Prof. Dr. Ethem Alpaydın .....

Assoc. Prof. Dr. Hans-Heinrich Bothe .....

Prof. Dr. Ömer Cerid .....

DATE OF APPROVAL: 23.6.1999

ACKNOWLEDGEMENTS

I am very grateful to my thesis supervisors for their continuous support and guidance throughout my study. I wish to express my gratitudes to the following people and institutions:

all members of my thesis examination committee for their invaluable suggestions, specifically to Prof. Bothe for his comments from which I have benefited quite a lot;

Prof. Ömer Cerid for his inspiring and enthusiastic approach to electronics and circuit analysis which have been influencing my views so strongly;

Prof. Sina Balkır who introduced us the field of VLSI and supported me through my graduate studies both as a teacher and as an advisor;

Prof. Günhan Dündar for his guidance and tolerance as an advisor, and for his close friendship;

İsmet Bayraktaroğlu for his collaboration and for the simulator which I have utilized a lot;

Murat Becer for his help with LabView in the characterization of the chips;

Prof. Ethem Alpaydın for his collaboration and assistance in neural networks during my studies;

Dr. Arda Yurdakul who recently finished her PhD, for her efforts in the preparation of the style files which I also used;

TÜBİTAK for supporting my studies via the project EEEAG-183;

the Electrical and Electronic Engineering Department as a whole for its striking environment.

ABSTRACT

Analog neural networks exhibit a potential for proper/suitable hardware implementation of artificial neural networks. Their advantages such as small size, low power and high speed, however, are seriously questioned/confronted by the difficulty in the training of analog neural network circuitry. Especially, hardware training by software, i.e., training of the circuitry by software based on models, so as to avoid on-chip and chip in-the-loop training methods, is threatened by circuit nonidealities and variations at outputs of identical blocks. The performance of the analog neural network is severely degraded in the presence of those unwanted effects caused mainly by statistical variations in the production process. We propose a new paradigm for the backpropagation algorithm in hardware training of multilayer perceptron type analog neural networks. The variations at outputs of analog neural network circuitry are modeled based on the transistor level mismatches occurring between identically designed transistors. Those variations can be used as additive noise during the training, and it has been shown that this will increase fault tolerance of the trained neural network drastically. The method has been compared to the method of injecting random noise, and our method outperforms the latter where injecting random noise is seen to be inadequate for establishing a satisfactory level of fault tolerance in the presence of mismatch based variations. The concept of mismatch based variations has been verified by measurements on our test chip.

ÖZET

Analog sinir ağları, yapay sinir ağlarının donanım bazında kullanılabilir olarak gerçeklenmesi için potansiyel oluşturmaktadırlar. Küçük alan kaplamaları, düşük güç harcamaları ve hızlı çalışmaları gibi avantajları karşısında analog sinir ağı devrelerinin eğitimindeki zorluklar önemli bir engel oluşturmaktadırlar. Kırmık-üstü eğitim veya bilgisayar-kırmık döngüsü eğitimi yerine kullanılabilecek olan donanımın yazılım bazında eğitimi, devrelerin ideal olmayan karakteristiklerinden ve birbirinin aynı olarak tasarlanan devrelerin çıkışlarındaki sapmalardan kötü olarak etkilenmektedir. Üretim sürecindeki istatistiksel sapmalardan kaynaklanan bu istenmeyen etkiler, analog sinir ağı devrelerinin başarımını ciddi miktarda düşürmektedirler. Bu tezde çok katmanlı perseptron türü analog sinir ağı devrelerinin donanım bazında eğitiminde geriye yayılım yordamı için yeni bir yöntem önerilmektedir. Analog sinir ağı devrelerinin çıkışlarındaki sapmalar birbiriyle aynı tasarlanmış transistörlerdeki uyumsuzluklara dayandırılarak modellenmiştir. Bu sapmalar eğitim sırasında toplanır gürültü olarak kullanılmış ve bunun eğitilen sinir ağının hataya karşı dayanıklılığını önemli oranda artırdığı gösterilmiştir. Önerilen yöntem, rasgele gürültü katılımıyla karşılaştırılmış ve daha başarılı olduğu gözlemlenmiştir. Transistörlerdeki uyumsuzluklara bağlı olan sapmaların varlığında, rasgele gürültü katılımıyla yapılan eğitimin hataya karşı yeterince dayanıklı olmadığı görülmüştür. Sapmaların uyumsuzluklara bağlı olduğu ürettirilen kırmıklar üzerinde yapılan ölçümlerle doğrulanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

LIST OF SYMBOLS

A_{VT0}, A_β, A_γ	Process related constants for mismatch in $VT0$, β, and γ
a	Current in weight differential pair of Gilbert multiplier
b	Current in weight differential pair of Gilbert multiplier
c	Current in weight differential pair of Gilbert multiplier
c_i	Regression coefficients for the mean and variance of synapse current
D_x	Spacing between matched transistors
d	Current in weight differential pair of Gilbert multiplier
d	Dimension of output vector of the training set
d_i	Regression coefficients for the sigmoid and variance of sigmoid
E	Error
f	Function of threshold mismatches
h	Number of hidden units
I_1, I_2, I_3, I_4	Differential currents in Gilbert multiplier and sigmoid block
I_D	Drain current
I_{ss}	Bias current
K	Proportionality constant for sigmoid
L	Length of the transistor
M	MOS transistor
M_i	Factor of threshold voltage mismatch in current mirror i
n	Noise term as a subscript
net	Weighted sum of synapse outputs
O	Opamp output
p	Number of patterns in the training set
Q_i	Factor of current factor mismatch in current mirror i
R, R_L	Resistance
$\mathbf{r}$	Desired output vector of the training set
S_{VT0}, S_β, S_γ	Process related constants for mismatch in $VT0$, β, and γ
T	Current in Gilbert multiplier
T_i	Weights associated with the hidden unit
U	Current in Gilbert multiplier

V_1, V_2	Voltages at second differential pair of sigmoid
VDD, VSS	Positive and negative supply voltages
V_{GS}	Gate-to-Source voltage of the transistor
V_Q	Quiescent operating point voltage of sigmoid for zero input
V_{SB}	Source-to-Bulk voltage of the transistor
V_T	Threshold voltage of the transistor
V_{T0}	Zero bias threshold voltage of the transistor
V_{offset}	Off-set voltage at the opamp
V_{out}	Output voltage of sigmoid
W	Width of the transistor
W_i	Weights associated with the input layer
w	Weights of neural network or weight for synapse
$\mathcal{X}$	Training set
$\mathbf{x}$	Input vector of the training set
x	Input variable for neural network, building block or function
y	Output variable for neural network, building block or function
Z	Output current of Gilbert multiplier
β	Current factor of the transistor
γ	Substrate factor coefficient
Δ	Amount of update or amount of variation
Δx	Differential input voltage of Gilbert multiplier
Δy	Differential weight voltage of Gilbert multiplier
η	Learning rate
λ	Decay factor
μ	Synapse function
ϕ	Activation function, sigmoid, surface potential
σ	Standard deviation

ABBREVIATIONS

A/D	Analog-Digital
ANN	Artificial Neural Network(s)
ANNSiS	An Artificial Neural Network Simulation System
ANNSyS	An Analog Neural Network Synthesis System
BP	Backpropagation
cif	Caltech Intermediate Form for Data Exchange
GCPS	Giga Connections Per Second
MLP	Multilayer Perceptron
MOS	Metal-Oxide-Semiconductor
OSO	On-Software-Only
rms	root-mean-square
SOC	System-On-Chip
SAFANN	Silicon Assembler For Analog Neural Networks
SIMD	Single Instruction Multiple Data
VLSI	Very Large Scale Integration
XOR	Exclusive Or

1. INTRODUCTION

Designing intelligent machines which can *think* like a human being has been one of the most enthusiastic dreams of mankind. With the advance of science and technology in an exponential rate during the last 150 years, great steps have been taken towards that goal. Automatic machines, robots mimicking human motion, research in artificial intelligence developing methods for symbolic computation are examples of these efforts. During the course of incorporating intelligence into machines, almost all of the intelligent algorithms have been developed by considering the human brain:

- How does the brain receive data from outside world through sensors?
- How does the brain manipulate sensory data into *information*?
- How does the brain store and process the information to make assertions?

This naturally required a thorough understanding of how the brain actually works, which is still not achieved yet. However, studies on the structure of human brain revealed that the brain is composed of approximately 10^{11} processing nodes (neurons) connected massively via approximately 10^{15} synapses [1, 2]. This huge information processing network is known to operate based on pulses (electrical signals of magnitude in the order of 100 mV), where each neuron attains a relatively low activity level (10-100 spikes per second). However, the massive parallelism in the structure and the efficiency in power consumption result in a system which can perform six orders of magnitude more operations with eight orders of magnitude less energy with respect to a state of the art computer system of today. Beyond the evident superiority of human neural networks in terms of processing power and energy, they are also capable of performing *difficult* tasks such as classification and recognition easily. Cognitive problems like spoken phoneme recognition or classification of handwritten digits require that the system (brain) should be able to learn from examples, which is not an easy task for a sequential software code implementing a certain algorithm since such problems usually do not possess well-defined algorithms. The structure of brain as a neural network attracted interest for realizing artificial neural networks (ANN) which would

mimic/simulate/emulate the interconnection structure and dynamics of human neurons so as to enable learning for such *difficult* tasks.

The primary advantage in the realization of ANN is that they do not describe any algorithm for the direct solution of the problem at hand, whether it be a function approximation or classification problem. Rather, ANN learn from examples either supplied with a teacher indicating the correct solution (supervised) or the level of success (reinforcement) or supplied without any external feedback (unsupervised). In the latter case, ANN simply detect the inherent dynamics of the system from the sample data. The essential requirement for ANN is, however, that ANN should be able to function properly when new, previously unknown input data are applied to the network. This is the so called generalization property of ANN.

Based on three main criteria given below, several types of ANN models have been formulated in the literature [3, 4, 5]:

- presence of a teacher during learning {supervised, reinforcement, unsupervised};
- type of learning algorithm {Hebbian, gradient descent based, Winner-Take-All};
- type of connection structure {feedforward, feedback, bidirectional}.

In this thesis, feedforward neural networks with supervised, gradient descent based learning are taken into consideration; however, some of the results are equally well applicable to other structures. This will be highlighted where appropriate.

Formally, a neuron in ANN is a processing unit with several variable inputs $x_1, x_2, \cdots, x_n$ and (usually) one fixed bias (threshold) of value -1, denoted by x_0 (FIGURE 1.1). The output y of the neuron is a weighted sum of its inputs, passed through a (usually) nonlinear activation function, ϕ,

$$y(x_1, x_2, \cdots, x_n) = \phi\left(\sum_{i=0}^{n} w_i x_i\right) \tag{1.1}$$

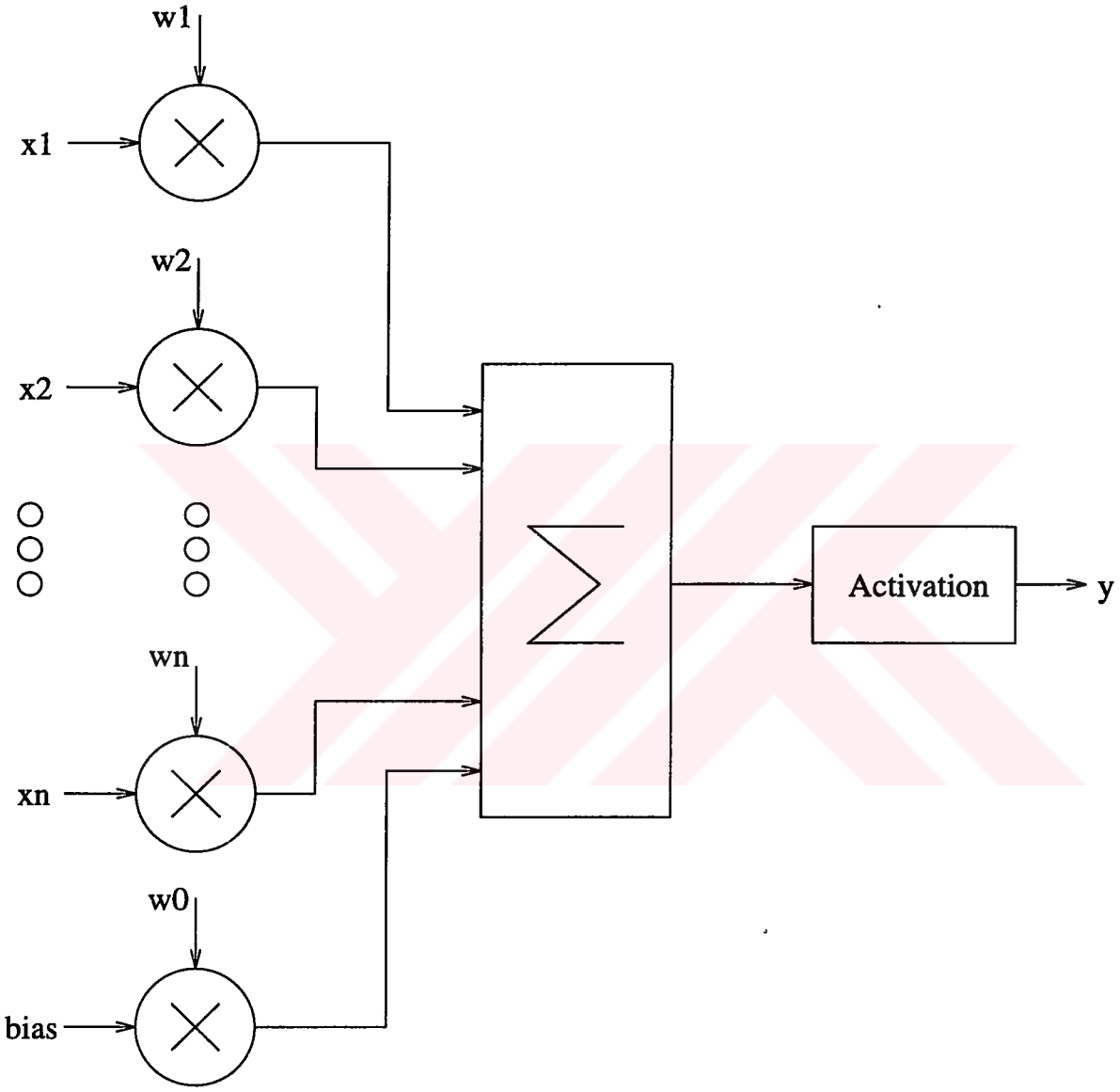

FIGURE 1.1. Structure of a neuron (single perceptron).

where w_i is the weight associated with the input x_i. This type of a neuron is also called a perceptron. A feedforward, multilayer perceptron (MLP) network is a layered structure consisting of several number of neurons in each layer (FIGURE 1.2) where the external inputs are applied to the neurons in the first layer and subsequently, outputs of each layer are *fed* to inputs of neurons in the next layer. Outputs of the last layer are the external outputs. Layers which do not have any direct connection to external world are called hidden layers. Even though there is no theoretical limitation on the number of hidden layers in a MLP, usually a maximum of two hidden layers are used in practice. In this thesis, fully connected MLP structures are considered, that is, outputs of each neuron in any layer except the output layer, are connected to to each neuron in the next layer. The overall output of the MLP depends on the inputs and the weights. Given a training set $\mathcal{X} = \{\mathbf{x}^p, \mathbf{r}^p\}_p$, where $\mathbf{r}^p$ is the desired output vector corresponding to the input vector $\mathbf{x}^p$, learning for the MLP denotes determination of the optimal set of values for the weights so as to achieve an acceptable level of approximation to desired values. The convergence criterion is usually expressed in terms of the rms (root-mean-square) error for the training set given as,

$$E_{rms} = \sqrt{\frac{1}{pd}\sum_{i=1}^{p}\sum_{j=1}^{d}(\mathbf{r}_j^i - \mathbf{y}_j^i)^2} \tag{1.2}$$

where d is the dimension of the output vector. Learning is based on applying the input samples sequentially to the MLP and calculating the outputs where the weights are initialized to small random numbers. Those outputs are compared to target values and the free parameters of the MLP (weights) are updated according to the gradient descent algorithm where sensitivity of output error with respect to each weight value is used in the update as follows,

$$\Delta w = -\eta \frac{\partial E}{\partial w} \tag{1.3}$$

where $E = (r^p - y^p)$ is the error at output of a single neuron for the p^{th} pattern and η is the learning rate. The essential principle underlying the backpropagation (BP) of the output error using gradient descent is that the error forms a hypersurface in the weight space and the global minimum of this surface (where all partial derivatives

are zero), is the optimal solution if it exists. It should be kept in mind that the error surface may also possess local minima where the above condition is met, hence, the backpropagation algorithm usually stops at a local minimum depending on the initial weight set. Since the full mathematical treatment of the error surface for a MLP with hundreds of weights is very elaborate, it is of little concern if a local minimum satisfying the convergence criteria is reached. As there is no guarantee that the algorithm will converge to an acceptable level of error, it may be necessary to restart the process with a new set of initial weights and/or training parameters (e.g. learning rate, momentum term, decay factor).

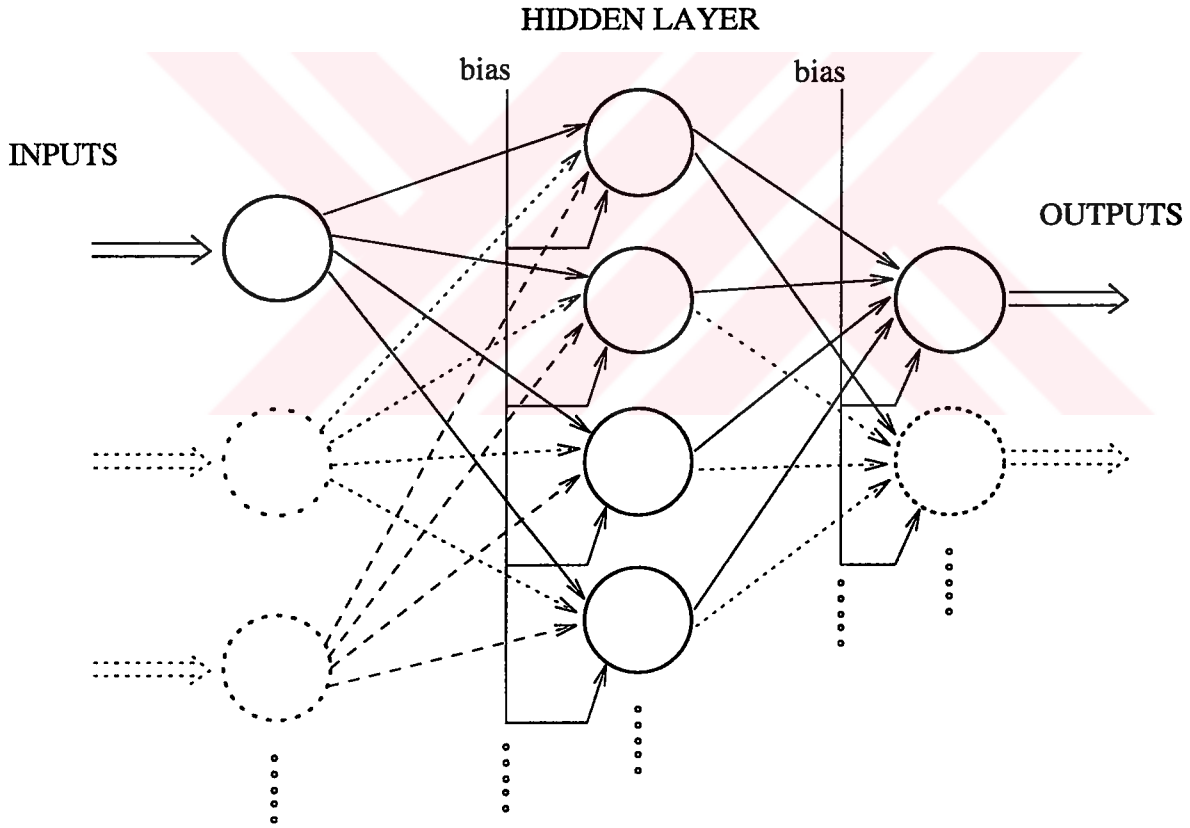

FIGURE 1.2. The general structure of the multilayer perceptron.

In this thesis, online weight update scheme is employed, that is, updates take place after every sample pattern is applied to the MLP. In the other alternative, batch update, the update terms are accumulated and the actual updates in the weight values are performed after the epoch is completed. An epoch designates the application of all

training samples to the MLP network once.

As models of information processing systems, ANN have been implemented by software or hardware in a variety of disciplines for many purposes. High energy physics (particle detection), engineering, financial markets (time series prediction, credit card fraud detection), medicine [6], computer science (handwritten digit recognition, speech processing [7]) and many other daily life applications [8] can be cited as examples of ANN usage. Software implementations of ANN, however, do not utilize the parallelism of processing unless special purpose multiprocessor systems are used [9]. As it happens, "simulation" of ANN models may require *long* training and/or execution times. The term *long* may designate periods of several minutes to several weeks on mainframe computers depending on the complexity of the problem which can be described by several factors as,

- number of independent variables (inputs);
- number of training samples;
- difficulty of the task to be learned.

On the other hand, the execution time (computation of outputs for a given, trained network) may also be considered as being long depending on the need: In a particle detector system or a fingerprint recognition system, classification of signals have to be achieved in real time, that is, a software implementation using a dedicated microprocessor would not be fast enough [10].

Use of dedicated hardware for ANN have also emerged as a solution offering the following advantages over simulation by general purpose computers:

- independence from a host computer;
- faster learning and forward operation in real time;
- easier interfacing with the outer world in case of analog implementations;

where several disadvantages also exist for dedicated ANN hardware implementations which adversely affect their use:

- necessity of large boards/chips for practical ANN applications;
- nonidealities in ANN hardware (especially in analog implementations) and limited resolution of building blocks;
- difficulty in training due to the nonidealities and the probable necessity of a host computer for training;
- storage of weights (digital storage requires memory, analog storage has programming and decay problems [11]).

There is an important decision to be made by choosing analog or digital hardware: The compactness and speed of analog systems versus the predictable accuracy and ease of programming of digital systems. On the other hand, the area of ANN hardware is not yet as commercialized as general purpose processors, that is why specific digital ANN hardware can not compete with the advances in the technology and architecture of general purpose computers. As digital hardware for ANN is usually specific for an algorithm, the designer can not utilize the flexibility of software available on a general computer [12]. Hardware realizations of ANN are, however, still very promising since they can exceed computational capabilities of software and advances in VLSI (Very Large Scale Integration) technology may offer low power, high density ANN chips operating at higher frequency so that ANN blocks may be integrated to the so called SOC (System-On-Chip) if problems of training, weight storage etc. are solved. The aim of this thesis is to make contributions to realization of analog ANN in VLSI by addressing system level problems in hardware training of analog ANN circuitry. The main attempt of the thesis is to suggest a new paradigm of hardware learning by software based on models of analog building blocks. The methodology suggested takes hardware related nonidealities and process dependent variations into account so that a robust training of analog ANN for a given problem can be achieved solely on software during the design phase of the network. The main emphasis will be on feedforward,

multilayer perceptron (MLP) type networks realized in MOS (Metal-Oxide Semiconductor) technology and trained by the backpropagation algorithm. CHAPTER 2 will consider hardware implementations of ANN where advantages and disadvantages of different approaches will be compared. Building blocks for a general analog ANN will be described in CHAPTER 3 where special emphasis will be put on the prototype analog neural network designed and produced during the course of this thesis. CHAPTER 3 also describes the silicon assembler for analog neural networks which forms an essential part of a complete synthesis system. Hardware training for analog ANN is discussed in CHAPTER 4 where different approaches are compared. It also describes BP algorithm as adapted for hardware training of analog ANN. Different modeling schemes for building blocks are compared. System level problems such as effects of mismatches, their modeling and incorporation into BP training will be studied in CHAPTER 5 where theoretical derivations will be verified by measurement results. Finally, CHAPTER 6 will conclude the thesis by summarizing the results and contributions of the thesis.

2. HARDWARE IMPLEMENTATIONS OF NEURAL NETWORKS

Neural networks have gained popularity in the last few years due to their success in diverse applications. Many applications require real time or very fast operation. This is possible only with dedicated neural network hardware. Due to the inherently parallel nature of neural networks, they are suitable for VLSI implementation. These implementations may be digital, analog, hybrid of the two, pulse based, or optical. This chapter will constitute a survey for the existing hardware realizations of ANN. Then, a comparison between the two major styles of analog and digital implementations will be done. The aim of the thesis is to make clear that analog ANN have the potential of realizing the SOC concept by solving the problems associated with the training of the analog circuitry.

2.1. Examples of Existing Hardware Implementations

Hardware implementations of ANN can be classified into the following categories:

- neurochips (digital, analog, mixed signal);
- accelerator boards for computers (using neurochips or special purpose processors such as digital signal processors);
- neurocomputers (using neurochips or general purpose processors).

Different realizations of neurochips suitable for MLP architecture in hardware are cited in TABLE 2.1 based on data from [1, 13, 14, 15] about commercially available chips and results of research papers [16, 17]. Moreover, examples of hardware realiza-

tions for different types of ANN other than MLP also exist. They include analog implementations of cellular neural networks [18, 19], bio-inspired processors implementing cellular neural networks [20], charge based [21] and digital [22] Hamming classifiers, charge based [23], analog [24], digital [25], and mixed signal [26] Kohonen map circuits, analog bidirectional associative memory [27], current mode Winner-Take-All [28] circuits, and analog ART1 systems [29].

TABLE 2.1. Commercially available neurochips suitable for MLP architecture.

Type	Name	Company	Precision (bits)	Neurons	Synapses
Analog	ETANN	Intel	6	64	10280
Digital	NLX-420	NeuraLogix	1-16	16	off-chip
	100-NAP	HNC	32	100	off-chip
	WSI	Hitachi	8	144	off-chip
	—	Hitachi	8	1024	10^6
	—	Lockheed	5	256	2048
	N64000	Adaptive Solutions	1-16	64	128K
	MT19003	MCE	13	8	off-chip
	MD-1220	Micro Devices	neuron: 1 weights: 16	1	8
	Lneuro-1	Philips	1-16	16	64
	MA-16	Siemens	16	16	256
Hybrid	TOTEM	—	neuron:16 weights:8	32	512
	ANNA	AT&T	neuron:3 weights:6	16-256	4096
	Neuro-classifier	Mesa Research	neuron:6 weights:5	6	426
	RN-200	Ricoh	—	16	256

Other examples of hardware ANN can be found in [30] for digital implementation and the works in [31, 32, 33, 34, 35, 36] represent analog realizations. Pulse coded (pulse stream) ANN hardware can be found in [37, 38, 39, 40, 41] which combine the advantages of analog and digital circuitry in an asynchronous operation mode yielding compact designs with simple interfaces. A recent special issue displays a wide interest in pulse coupled neural networks and their implementation [42]. Optical implementations

of analog ANN hardware have also appeared in the literature [43, 44, 45] which have been applied successfully to classification problems; however, the difficulty of designing optical systems and integrating them onto a chip, has prevented a wider usage of optical networks.

Accelerator boards are special purpose hardware systems suitable for implementing the ANN in conjunction with a host computer. They usually contain commercially available or custom designed neurochips mentioned in TABLE 2.1 for the forward operation whereas the data acquisition, storage, and training are carried out by the host computer. Examples of such boards have been used in several applications such as high-speed character recognition using ANNA chip [46], reverse modeling of microwave circuits using digital processors [47], and real time meteorological data acquisition using the TOTEM chips [17]. The only analog neural network chip which was commercially available is the ETANN chip from Intel [48]. There has been an extensive use of this chip in the field of particle physics for several applications: Drift chamber tracking [10, 49] and particle identification [50] where real time responce is of utmost importance. Accelerator boards using ETANN chips have been used for these applications. A cascadable VME-bus based analog neural network module of [51] has been one of the most successful applications of the ETANN chip. However, this chip has been plagued by limited resolution in storing the synapse weights in that the long time resolution of the weights is not more than five bits. Implementing Madaline Rule III [52] has been suggested for the ETANN chip; however, this requires a host computer and external hardware besides many timing problems which limit the performance of training. Problems like these have prevented the success of this chip on the market so that commercial applications using this chip and similar ones have been limited. Several applications reported in the literature have demonstrated successful operation, whereas other reported applications have suffered from the aforementioned problems. Another major deficiency of this chip is the issue of cyclability in the weight storing EAROM's.

Neurocomputers are dedicated systems made up of special neurochips and/or general purpose processors which form a complete training and operation system. They usually require specific programming environments and in that sense they are called computers. Commercial products exist [1]: CNAPS from Adaptive Solutions employing N64000 chips contains 64 processing nodes connected in SIMD (Single Instruction Multiple Data) mode; Siemens' SYNAPSE1 neurocomputer uses eight MA-16 chips in a systolic array architecture; SNAP from HNC utilizes two VME cards containing four 100-NAP chips each. Other examples of neurocomputers are also developed for specific purpose applications utilizing custom digital chips realizing ANN [53, 54, 55].

As neurocomputers and accelerator boards are alternatives to expensive super-computers or slow general purpose computers, they are usually developed for specific applications. There is no general methodology for hardware implementation of ANN. The problems of ANN chips such as difficulty of training due to nonidealities, large chip sizes of digital realizations and weight storage have to be solved so as to realize easy-to-use ANN blocks which can be incorporated as a sub-system into more complex systems.

2.2. Analog versus Digital Implementation

Many digital implementations have been reported in the literature owing to the fact that they offer several advantages such as predictable accuracy, high noise immunity, ease of multiplexing communication and computation, availability of well-established tools for digital design, and ease of interfacing with other digital systems [56]. Analog implementations, on the other hand, have many advantages such as small size, high speed, and straightforward interfacing with the outside world which is analog by nature [57]. In the following, analog and digital realizations of ANN will be compared based on several criteria: layout area (transistor count), power consumption, ease of training. Synapses, which are the most common elements in a neural network,

can be represented at the circuit level by multipliers. Parallel digital multipliers require a very large area compared to analog multipliers of comparable precision which use less than 20 transistors. For instance, parallel digital multipliers of 8x8 input word lengths have transistor counts on the order of at least several thousand [58]. Serial digital multipliers are smaller than their parallel counterparts; however, they are much slower. On the other hand, the speed of an analog multiplier is limited mostly by its settling time. When one looks at neurons, a similar picture can be seen. Again, an adder and the nonlinearity can be realized by less than 20 transistors in the analog domain, whereas the same operations require transistor counts which are at least an order of magnitude larger in the digital domain assuming that multiple input parallel adders and efficient look-up tables for the nonlinearity are utilized [48, 59, 60, 61]. A comparison carried out in [1] indicates that analog neural processors are much more superior than their digital counterparts in terms of power, area and transistor count. The details of the comparison are repeated in TABLE 2.2.

TABLE 2.2. Comparison of ANN realizations in software, digital and analog VLSI.

	microprocessor	digital	analog
technology	0.75μm CMOS	0.8μm CMOS	0.8μm CMOS
operational speed	200 MHz	60 MHz	2-5 MHz
accuracy(bits)	64	16	7-8
chip size (mm^2)	16.8x 13.9	12.38 x 12.90	10 x 10
power dissipation	30W @ 3.3 V	2.4W @ 5V	5W @+/-5V
# of transistors	1.68M	930K	160K
Target connection	32E12 Connections Per Second (32GCPS)		
# of chips required	60	12	1
power consumption	1800 W	28.8 W	5W
total silicon area	14011mm^2	1916mm^2	100mm^2
# of transistors	101M	11.2M	160K

Use of MOS transistors in subthreshold operation region for analog ANN constitute another advantage in terms of power dissipation [62]. Combining subthreshold operation with current-mode analog building blocks which also do not require transconductance amplifiers, neural systems have been constructed in a compact form [63, 64, 65]. The storage of weights as analog values on a floating gate transistor

structure also allows a high density implementation for ANN [32, 66, 67, 68]. Despite their evident advantages over digital implementations, analog implementations have not been accepted as a mature product which can be used as off-the shelf components. The main reasons for the lack of success in analog ANN implementations can be classified as,

- circuit based problems (circuit nonidealities, mismatches between identical chips, difficulty of full-custom analog design, weight storage and update, interfacing with other computational resources);
- training based problems (use of extra hardware and/or software for training, limitations on the algorithms due to circuit nonidealities).

It can also be said that the problems mentioned above can not be considered separately: The training algorithms have to be modified so as to incorporate circuit based features, whereas the circuits have to be designed such that they allow a meaningful training to be done.

In the literature, analog neural network implementations have been rather ad hoc in that very few, if any, have explored the constraints that circuit non-idealities bring about. Examples are nonlinear synapses [69, 70], neurons that deviate from ideal functions, or errors and limitations in storing weights [71, 72, 73]. It has previously been shown that multiplier nonlinearity can be a very severe problem even for nonlinearity factors of less than 10 per cent for many applications [68, 69, 70, 74, 75]. Limited precision in storing weights has also proven to be a crucial problem in analog neural network design. The work in this area has been mostly limited to predicting these effects either through simulation or through theoretical analysis and developing some methods to overcome these problems partially. In [70], the effects of some non-idealities have been studied through circuit simulation with SPICE and the importance of circuit level simulation in analog neural network design has been demonstrated. The aim of this thesis is to study the constraints on analog ANN training posed by circuit nonidealities and mismatches, and to formulate solutions for the training of hardware by software.

3. ANALOG NEURAL NETWORK BUILDING BLOCKS

Several representative circuits will be described in this chapter which can be used as building blocks for analog ANN. These circuits have also been implemented and tested in a prototype chip. The modeling and training issues for analog neural network circuitry will be investigated based on these circuits; however, the approach to be described in the thesis and the conclusions arrived are also valid for other synapse and neuron implementations.

3.1. The Synapse

The synapses in a neural network can be realized by analog multipliers if the inputs and the weights can be represented by voltages. The synapse circuit used in the chip is a modified version of the well known Gilbert multiplier [66, 76] as shown in FIGURE 3.1. The inputs are in the form of voltage differences and are denoted by $(x_2 - x_1)$ and $(y_2 - y_1)$. The output of the original Gilbert multiplier is a current difference $(U - T)$ and this difference is converted to a single ended current (Z) through current mirrors. This improves the linearity of the multiplier as well as providing easy interfacing to the following circuitry. Using voltage input - current output synapses for analog ANN is very suitable for VLSI implementation since the actual signals from outside are mostly in voltage form, and the summing operation on synapse outputs can be performed by connecting the synapse outputs together. The layout of the designed synapse is given in FIGURE 3.2 and the dimensions of transistors in the synapse are given in TABLE 3.1 where W and L represent the width and the length of the transistor respectively.

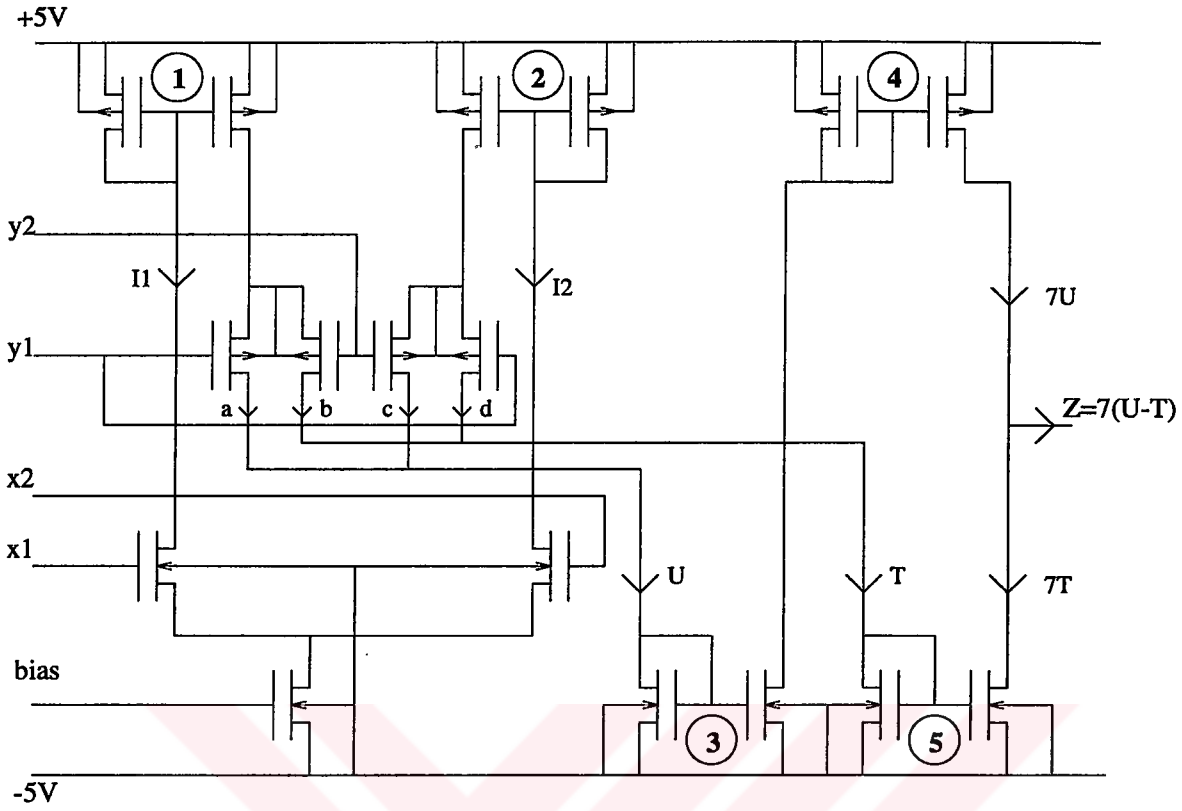

FIGURE 3.1. Circuit diagram of the synapse.

TABLE 3.1. Dimensions of the transistors in the synapse as drawn (values will be scaled by 0.8 on the mask).

	Mirror number					Differential pair		bias
	1	2	3	4	5	x	y	
W/L	6/3	6/3	3/3, 6/3	24/3, 60/3	3/3, 22/3	3/12	6/16	9/3

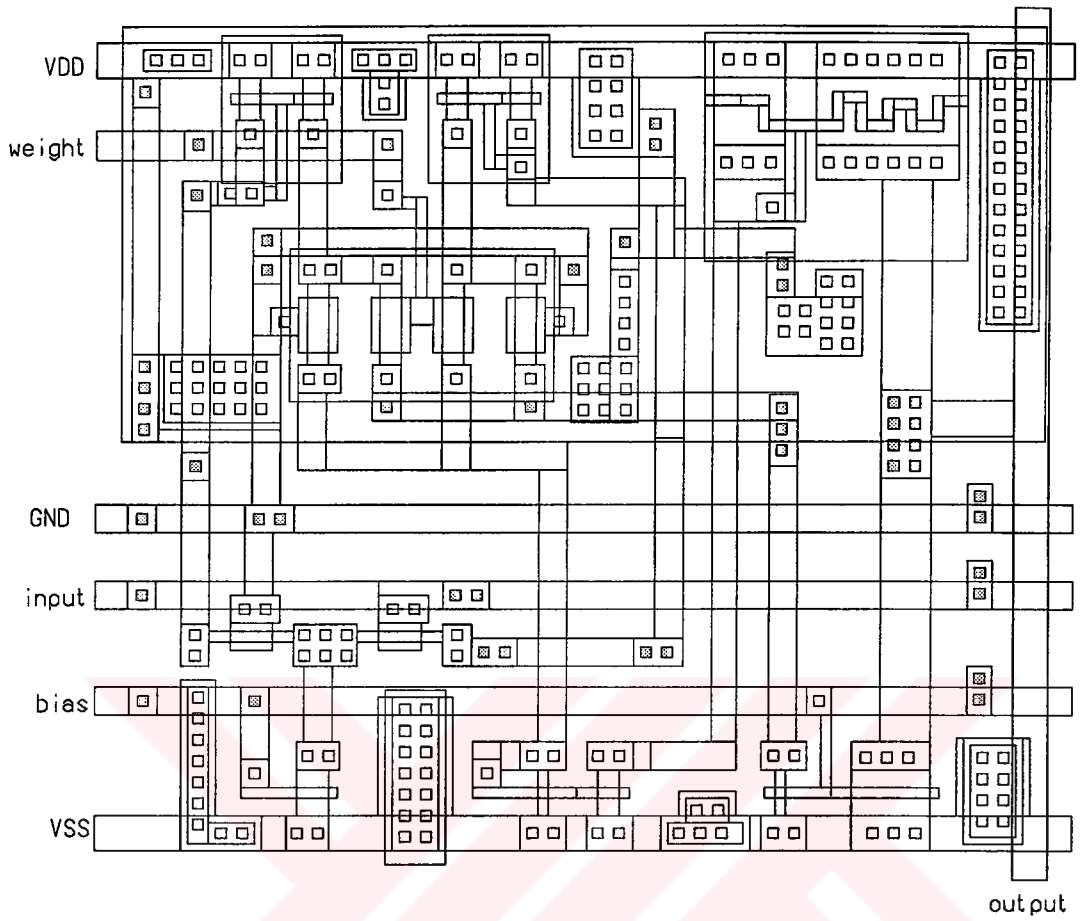

FIGURE 3.2. Layout of the synapse circuit.

The output current of the synapse can be derived assuming that the transistors operate in the saturation region for which the drain current, I_D, of the MOS transistor is given by

$$I_D = \frac{\beta}{2}(V_{GS} - V_T)^2 \tag{3.1}$$

where β is the current factor, V_{GS} is the gate-to-source potential and V_T is the threshold voltage. The differential pair for the input $\Delta x = (x_2 - x_1)$ causes the currents I_1 and I_2 to flow through the differential pairs for the weight, $\Delta y = (y_2 - y_1)$, after reflected by the current mirrors 1 and 2,

$$I_1 = \left(\frac{-\sqrt{\frac{\beta_w}{2}}\Delta x + \sqrt{-\frac{\beta_w}{2}\Delta x^2 + 2I_{ss}}}{2} \right)^2 \tag{3.2}$$

$$I_2 = \left(\frac{\sqrt{\frac{\beta_x}{2}}\Delta x + \sqrt{-\frac{\beta_x}{2}\Delta x^2 + 2I_{ss}}}{2} \right)^2 \tag{3.3}$$

where β_x is the current factor of the input differential pair and I_{ss} is the bias current. The differential current $I_1 - I_2$ simplifies to

$$I_1 - I_2 = -\sqrt{\beta_x}\sqrt{I_{ss}}\Delta x \tag{3.4}$$

if $\frac{\beta_x}{2}\Delta x^2$ can be neglected with respect to $2I_{ss}$. This differential current will be the *bias* current for the weight differential pairs. Hence, the differential currents $a - b$ and $c - d$ formed by the weight differential pairs, can be approximated by,

$$a - b = \sqrt{\beta_y}\sqrt{I_1}\Delta y \tag{3.5}$$

$$c - d = -\sqrt{\beta_y}\sqrt{I_2}\Delta y \tag{3.6}$$

$$(a + c) - (b + d) = \sqrt{\beta_y}\Delta y \left(\sqrt{I_1} - \sqrt{I_2} \right) \tag{3.7}$$

from which $U - T = (a + c) - (b + d)$ can be rewritten using (3.2) and (3.3) as,

$$U - T = -\sqrt{\frac{\beta x \beta y}{2}}\Delta x \Delta y. \tag{3.8}$$

The currents U and T are multiplied by a factor of seven by means of the current mirrors 3, 4 and 5 respectively. Finally, the output current Z becomes

$$Z = -7\sqrt{\frac{\beta x \beta y}{2}}\Delta x \Delta y. \tag{3.9}$$

FIGURE 3.3 displays the output current characteristics of the synapse circuit for different weight values, $w = \{-2, -1, 0, 1, 2\}$ V as simulated by HSPICE with the model parameters given in the APPENDIX, where the output node is connected to ground.

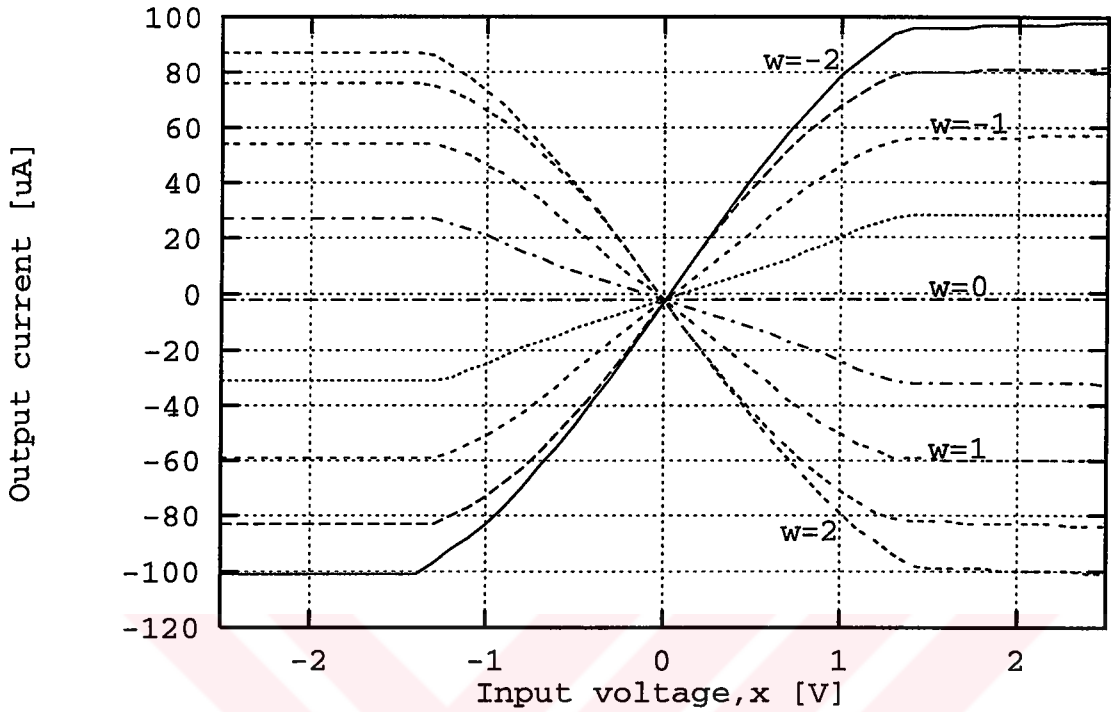

FIGURE 3.3. HSPICE simulation for the synapse with nominal model parameters.

Many multiplier circuits have been proposed in MOS technology in the literature. A large portion of the examples employs the same structure of Gilbert multiplier [26, 32, 66, 77, 78, 79] for four-quadrant operation where the transistors are operated in the saturation region. Current mode operation in subthreshold region [19, 64, 80] and in saturation region [81] have also been offered as low power synapse circuits. Some synapse circuits incorporated the weight storage element as a floating gate MOS device so as to allow very compact designs [67, 82] whereas some circuits used a single differential pair with digitally adjustable weight values [83]. Tunable MOS transistors operated in the triode region used as multipliers have also appeared in the literature [18, 62, 66, 84, 85, 86, 87]. In some implementations of analog neural networks, the square-law characteristics of the MOS transistor is utilized in order to form squaring circuits for multiplication [60, 88, 89]. Digital implementations of multipliers in analog neural networks include the works of [90] and [91] where in the latter case the multiplier is realized using an A/D converter and latches. It can be stated that the current

output Gilbert multiplier using floating gate structures for storing the weight values seems to be the most optimal design for the multiplier in terms of compactness and accuracy. However, in terms of programming the weight values, use of multiplying digital-analog converters offer the highest flexibility where the weight values have to be quantized. The implementations of analog ANN mentioned in CHAPTER 2 usually employed Gilbert type multipliers operated in the saturation region.

3.2. I-V Conversion

The use of current-output synapses enables the summation of those currents by simply connecting them together at the input of the neuron. This current sum can be converted to voltage by using an opamp and a resistor as a current-to-voltage converter. The circuit diagram of the opamp is given in FIGURE 3.4. This opamp consists of two stages, where the first stage is a differential amplifier whose differential current output is mirrored into the next stage and converted to a single ended output through circuitry very similar to the synapse circuit mentioned above. The layout of the I-V converter is given in FIGURE 3.5. As most of the applications employed current output multipliers as synapses, the summation is carried out as in our chip. The I-V conversion is usually required since the activation function blocks to be described in the next section, require voltage input. It should be mentioned, however, that current input activation blocks can also be designed which would not require I-V conversion [57]. There are several types of current-to-voltage converters [62, 72, 92, 93, 94] to be used where opamp based I-V conversion can also be employed [95]. The essential requirements for opamp based I-V conversion are that the opamp exhibits a large input impedance (in order to satisfy the virtual ground requirement), and the opamp can supply enough current by its output stage.

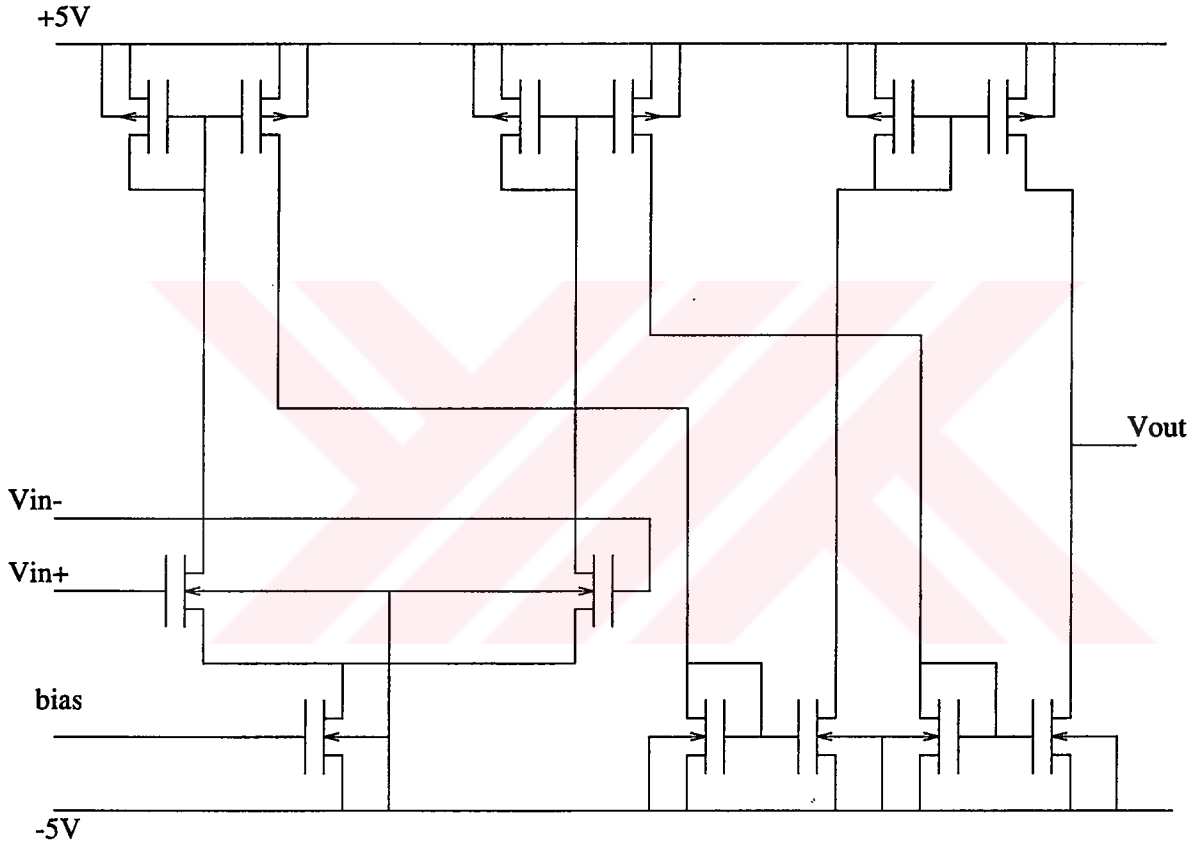

FIGURE 3.4. General purpose opamp for I-V conversion.

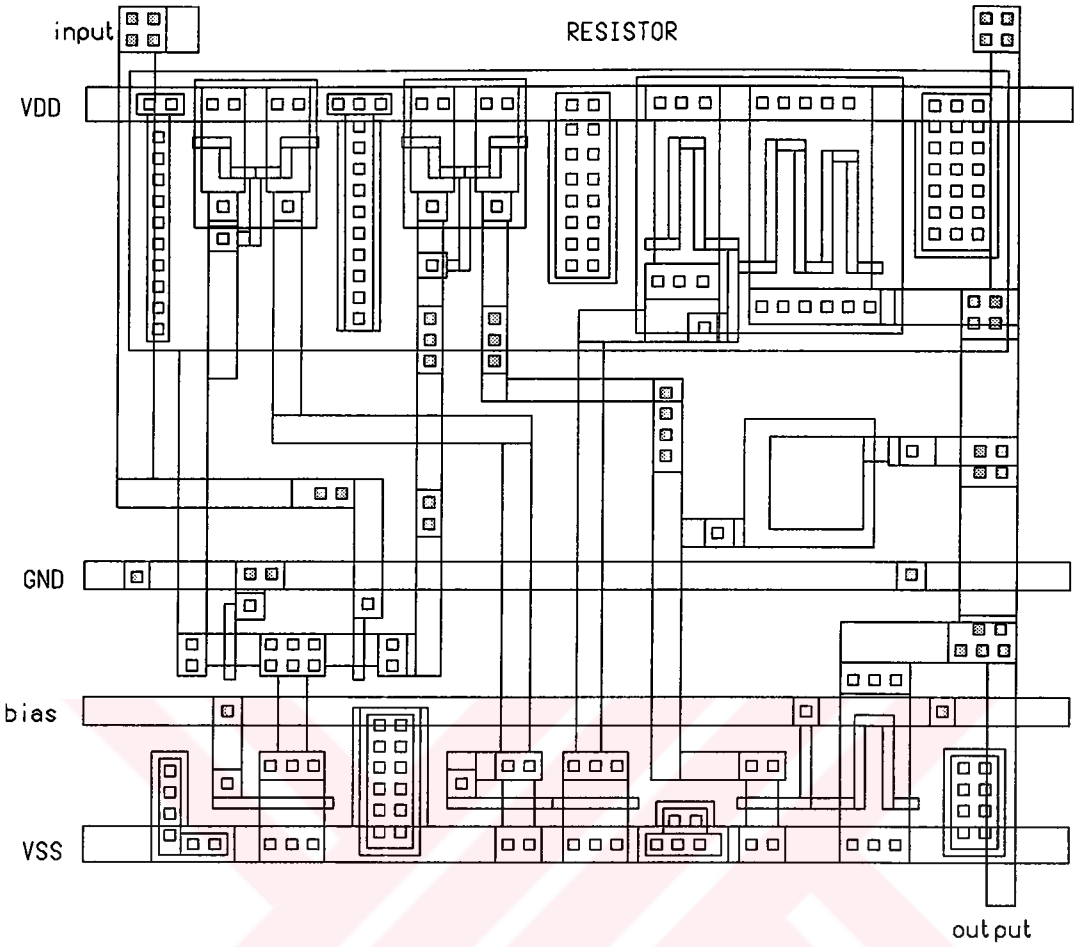

FIGURE 3.5. Layout of the opamp circuit.

3.3. The Sigmoid

A sigmoid generator introduced in [91] is used after the opamp to generate the activation function for the neuron. This generator is depicted in FIGURE 3.6 where the layout of the sigmoid block can be seen in FIGURE 3.7. For the sigmoid block, the output voltage can be derived in semi-analytic manner (FIGURE 3.6): The input differential pair is connected to the positive supply via two resistively connected

transistors, each having a value of R_L. The differential current

$$I_1 - I_2 = -\frac{\beta_1 x}{2}\sqrt{\frac{4I_{ss1}}{\beta_1} - x^2} \tag{3.10}$$

causes a differential voltage $(V_1 - V_2)$ at the gates of the second differential pair which also yields the differential current,

$$I_3 - I_4 = \frac{\beta_2}{2}\frac{\beta_1 x R_L}{2}\sqrt{\frac{4I_{ss1}}{\beta_1} - x^2}\sqrt{\frac{4I_{ss2}}{\beta_2} - (V_2 - V_1)^2} \tag{3.11}$$

where $\beta_1(\beta_2)$ and $I_{ss1}(I_{ss2})$ are the current factor and bias current for the first (second) differential pair respectively. The output of the sigmoid can be expressed in terms of the difference $(I_3 - I_4)$ as follows,

$$V_{out} = V_Q + K(I_3 - I_4) \tag{3.12}$$

where V_Q is the quiescent operating point voltage of the sigmoid output for the input of $0V$, and K is the proportionality factor which will be determined based on simulations. Finally, neglecting the $(V_2 - V_1)^2$ term in (3.11), the sigmoid output will be approximated by,

$$V_{out} = V_Q + \frac{1}{4}K\beta_1\beta_2 x R_L\sqrt{\frac{4I_{ss1}}{\beta_1} - x^2}\sqrt{\frac{4I_{ss2}}{\beta_2}}\,. \tag{3.13}$$

Transfer characteristics of the sigmoid block according to (3.13) is given in FIGURE 3.8. It should be noted, however, that (3.13) does not hold for input magnitudes larger than 2 V where the sigmoid output actually saturates.

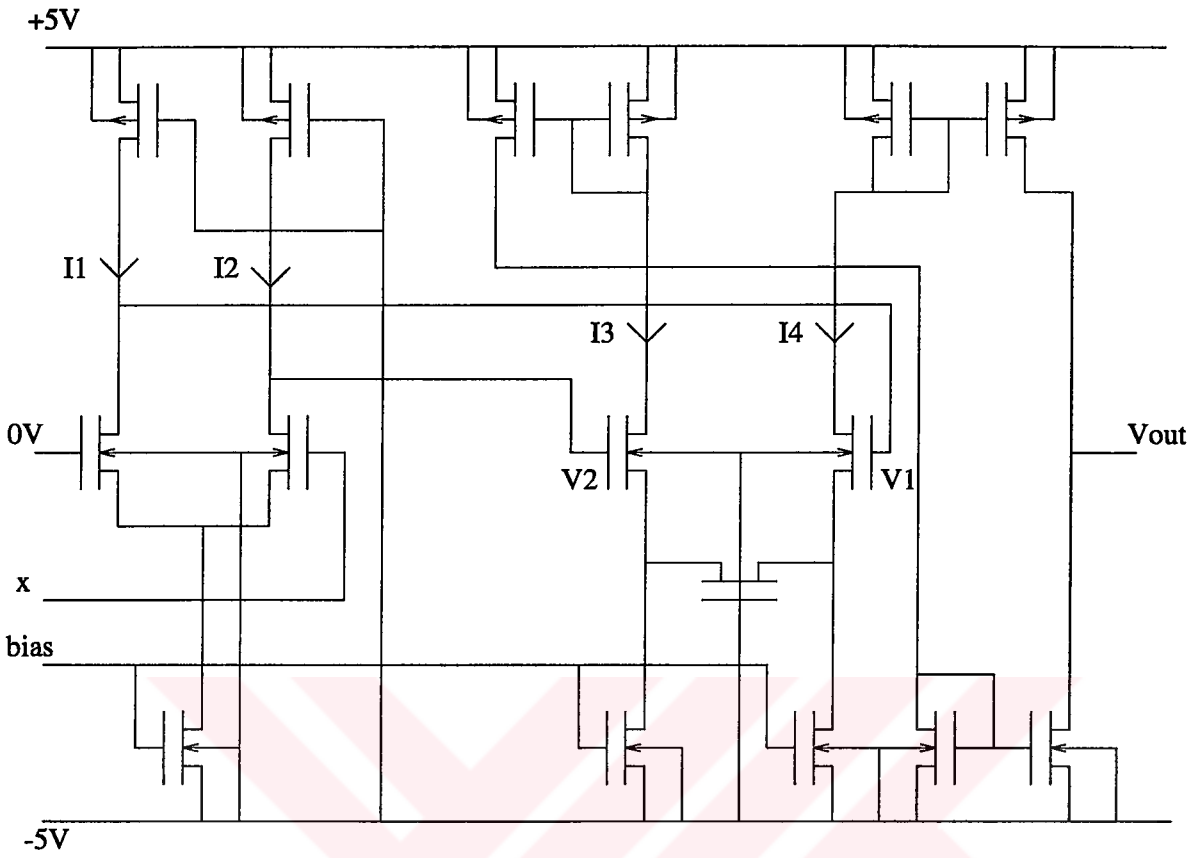

FIGURE 3.6. Sigmoid circuit.

Regarding the activation function, the sigmoid and the tangent hyperbolic are the two forms found in the literature. Most of the analog implementations have used differential pair input circuits for generating the required waveform where the outputs can be current or voltage [68, 91, 96, 97, 98]. A class of inverter based circuits can also be used for the activation function [57] where the nonlinearity can also be achieved by means of the synapse circuits described above: The saturation of the output current for large input magnitudes makes the fixed weight Gilbert multiplier a potential activation circuit [34]. Digital circuitry for the realization of the activation function based on piecewise linear approximation has also been reported in the literature [99].

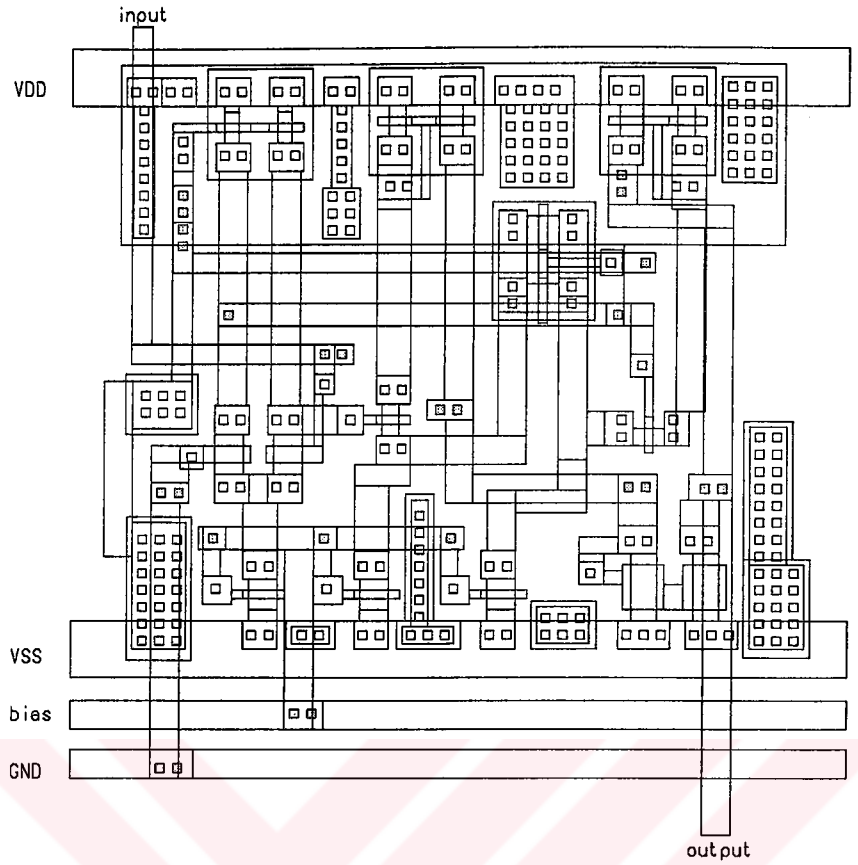

FIGURE 3.7. Layout of the sigmoid circuit.

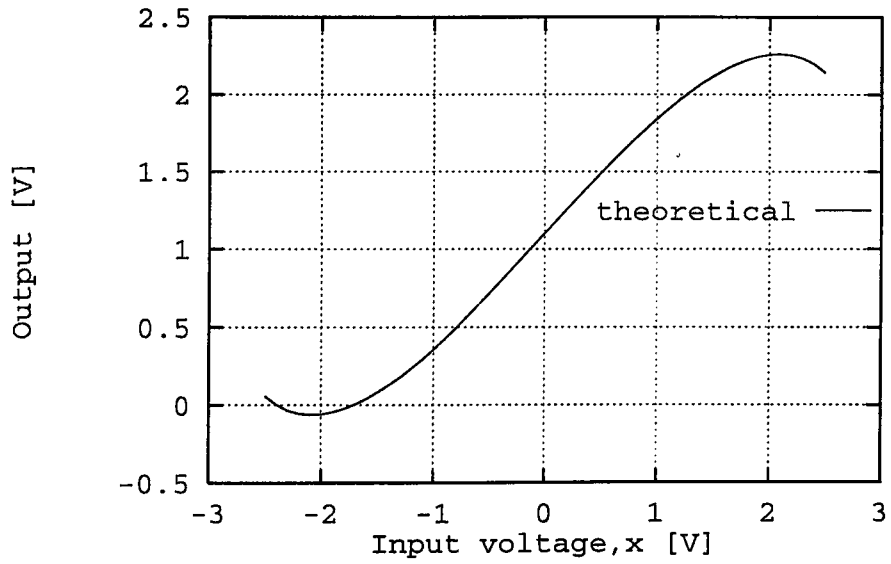

FIGURE 3.8. Transfer characteristics of the sigmoid circuit as computed.

3.4. SAFANN: Silicon Assembler For Analog Neural Networks

The straightforward structure of neural networks consisting of identical building blocks (mainly multipliers, adders and activation function) in a hierarchical manner may allow easy implementation on a single chip. The generation of the actual layout for a VLSI circuit is also a tedious task. There are several design methodologies employed like bottom-up design or top-down design along with different design styles like full-custom, semi-custom, gate-array or standard cell based designs. In this thesis, bottom-up design methodology is chosen because the individual building blocks have already been designed and the whole system exhibits a well defined regular structure. Hence, the realization of the layout can be seen as an assembly of those building blocks which can be considered as a semi-custom design style. The placement and routing of those blocks can be done automatically by means of a silicon assembler. Silicon assemblers are tools designed to generate compact and correctly connected (electrically) layouts which will exhibit high performance provided that the building blocks are designed accordingly. Using a silicon assembler for automatic layout generation guarantees that the layout is correct with respect to both the functional specification and the constraints of the specific technology used (layout design rules). Thus, using a silicon assembler enables fast and reliable generation of the layout where any modification in the design of any block can be reflected to the layout of the system easily [100].

The layout generated by a silicon assembler may consume more area than a layout generated manually (full-custom design); however, use of a silicon assembler for ANN allows a very fast layout generation due to the regular structure of the ANN consisting of several building blocks only. The layout becomes even more compact if the building blocks are designed such that abutment becomes possible; i.e., placement of neighboring blocks is achieved without any routing channel between them. Moreover, the silicon assembler can be used to implement different types of the same building block at different locations of the design. For this, the actual blocks have to be designed accordingly such that they would fit to the abutment property. Automatic generation

of the ANN layout is easily possible if there are suitably designed multiplier and neuron blocks. Those basic blocks (subcells) can be placed in arrays and the complete ANN will be obtained. Once there are the *cif* (Caltech Intermediate Form for Data Exchange: A special language that describes the layout which will be explained below in short), files for the building blocks, the final circuit can be obtained as a *cif* file by appending the placement information to the *cif* definitions of the blocks. In this thesis, a program in C-language (SAFANN) is devised to accomplish this task.

The starting point is the layout for a single ANN cell which consists of three types of subcells and some interconnections. A sample three-input cell's structure is given in FIGURE 3.9. It is mainly made up of three multipliers, a neuron and three of the so called channels. In fact, each channel consists of three subchannels employing two lines each. This type of a topology has its reasons: First of all, the design has to be modular, that is, it should support any number of inputs. Hence, the weights for the inputs are carried on channels whose number can easily be manipulated. It should be also noticed that only one weight; i.e., two weight lines should be connected to each multiplier, so that a decoding scheme is necessary. Next, the input lines have to travel throughout the cell in horizontal direction because that input will also be required in the next cell placed to the right of the first one. Finally, the output lines of the multiplier have to be aligned such that they are common in the vertical direction, so that they will be connected together which is also part of the abutment property. The supply and bias lines will also run through the cells so that they will be aligned.

The Caltech Intermediate Form for Data Exchange is a means of describing the layout which is a collection of 2-dimensional geometric features. Once the layout consisting of mask features describing the graphic items is created by some CAD tool like the IC Station, the *cif* file will specify those geometric items, their locations on a Cartesian coordinate system, and the layers on which those items reside in usual text format. A very common element of such a description is the box (B) which will be defined by its length, width, center and direction along with the layer specification. A very useful feature of the *cif* file is that groups of objects can be identified as symbols

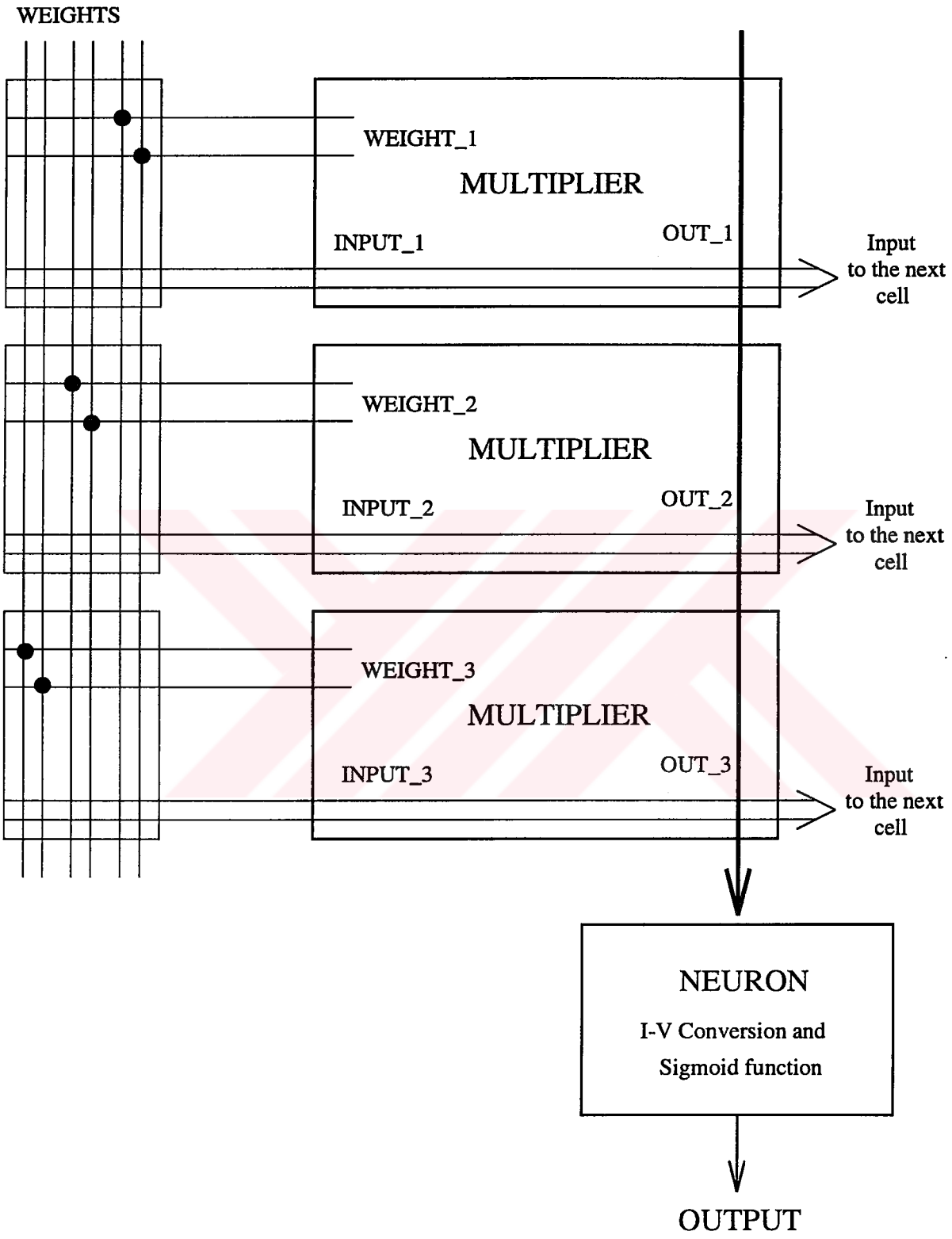

FIGURE 3.9. Topology of a single cell with three inputs.

(subcells) and their instances can be called (placed) in the design at specific locations in a hierarchical manner [101].

The SAFANN program achieves the automatic placement of ANN building blocks and routing between them by placing instances of symbols created for those blocks, namely the multiplier, opamp, sigmoid, and subchannel; and routing channels as collection of boxes into a final *cif* file. The structure of this methodology can be understood by considering a single layer of an ANN given in FIGURE 3.10. Here, the channels will be described by collection of boxes representing metal-1, metal-2 lines and vias. Neuron cells, however will be called by the symbols. A detailed explanation of the SAFANN program is given as an algorithm on the following page.

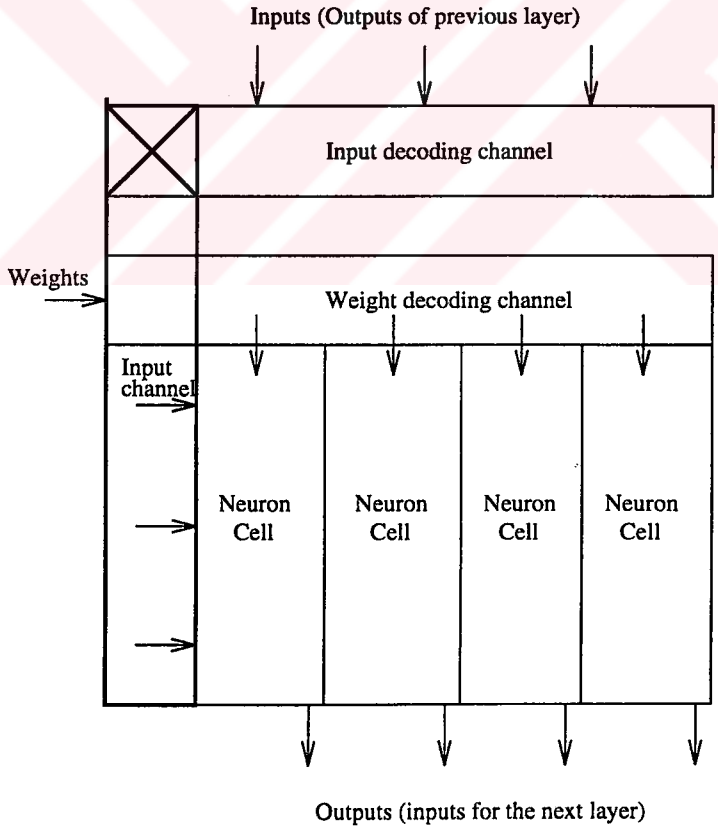

FIGURE 3.10. Topology of a single layer with three inputs and four neurons.

Algorithm of the SAFANN code:

i) Input names of the

input_file (containing the *cif* definitions of the basic building blocks),

data_file (containing definition of the ANN topology,

the names and dimensions of basic building blocks to be used),

output_file (will contain the *cif* description of the final layout).

ii) Read the data_file and obtain number_of_layers and

the corresponding number_of_neurons[layer].

iii) Initialize the position to $(0, 0)$.

iv) For index1=1, layer do

```
{
place_input_channel;
For index2=1, number_of_neurons[layer-1] do
        {
        place_neuron_cell;
        for index3=1, number_of_neurons[layer] do
                {
                place_neuron; /* I-V converter and sigmoid cell*/
                for index4=1, number_of_neurons[layer-1] do
                        {
                        for index5=1, number_of_neurons[layer-2] do
                                {       place_weight_channel; }
                        place_multiplier;
                        }
                }
        }
place_weight_decoding_channel;
place_input_decoding_channel;
}
```

v) Perform routing between layers.

vi) Append the resultant *cif* code to the input_file and obtain the output_file.

SAFANN is developed to realize automated layout generation for ANN. It is a compact, easy to use program which is, to the best of the author's knowledge, the first silicon assembler for ANN in the literature. Layouts of different ANN topologies can be generated within seconds so that their extraction, circuit level simulation, and reconfiguration can be realized easily. FIGURE 3.11 presents the layout generated by SAFANN for an XOR gate consisting of two inputs, three hidden units, and an output unit. Similarly, layout generated by SAFANN for a $1 \times 10 \times 1$ structure is given in FIGURE 3.12. The adaptive nature of SAFANN also displays itself in the fact that different types of multiplier, opamp, or sigmoid cells (circuits) can be placed at different synapse and neuron locations. This allows flexibility in the overall design; e.g., less accurate multipliers consuming less area can be used where the corresponding weights are relatively small such that their effects are negligible.

As future work, SAFANN can be extended to accomplish global routing and placement of nonvolatile elements, with weight decoding, for weight storage so that the design of complete ANN chips would be possible. For this, development of a library containing different types of multipliers and neurons is necessary. Moreover, routing of the layers can be optimized so that the resulting layout would be more compact, i.e., filling a rectangular region fully, instead of extending in length and leaving empty space in width An attempt for compaction of SAFANN has been carried out in [102].

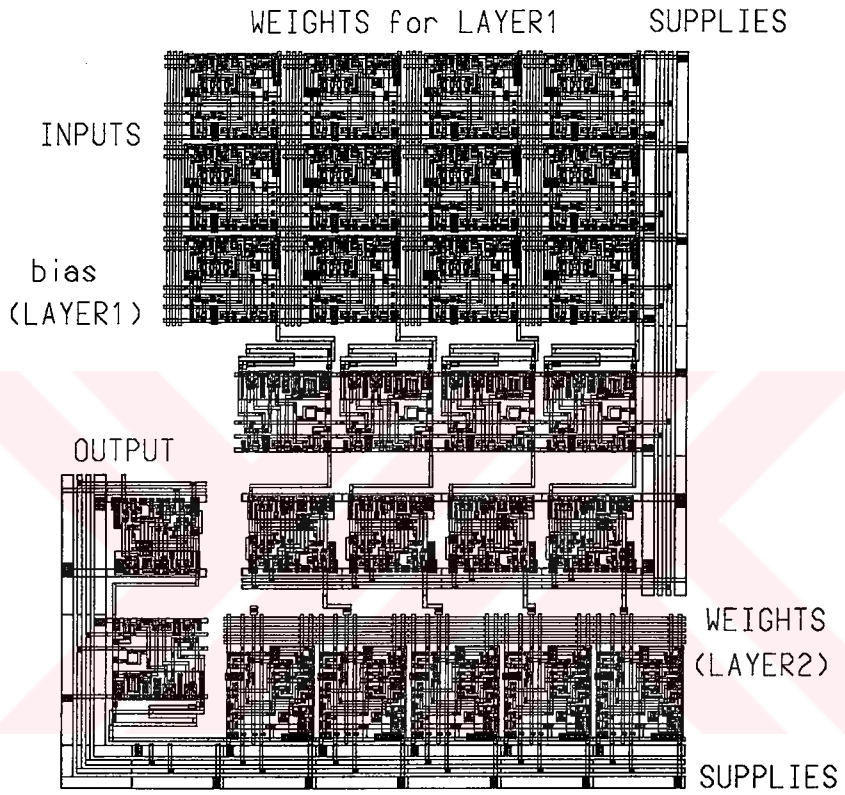

FIGURE 3.11. Layout of XOR generated by SAFANN.

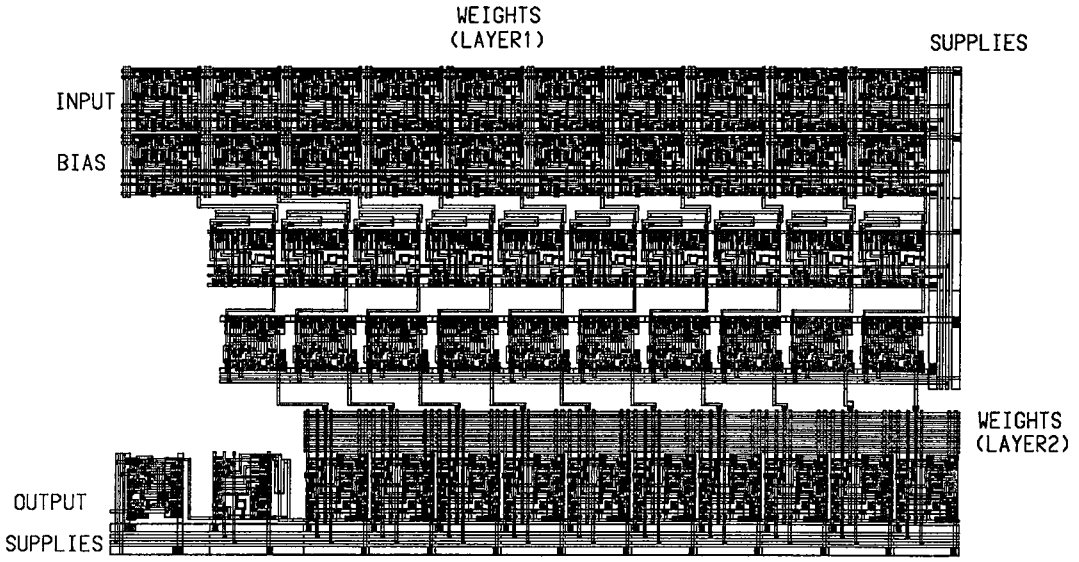

FIGURE 3.12. Layout of 1x10x1 structure generated by SAFANN.

3.5. The Chip

The prototype chip has been designed in Alcatel-Mietec 2.4μm technology using Mentor Graphics IC station program, in a full-custom style. The characteristics of the technology are given in the APPENDIX. The prototype chip contains four neurons each having five inputs, so that a total of 20 synapses are found in the chip. The dimensions of the building blocks are given in TABLE 3.2. The structure of the unit cell for the network is given in FIGURE 3.13 whereas the layout of the core (chip without the pads and the extra circuitry) can be seen in FIGURE 3.14.

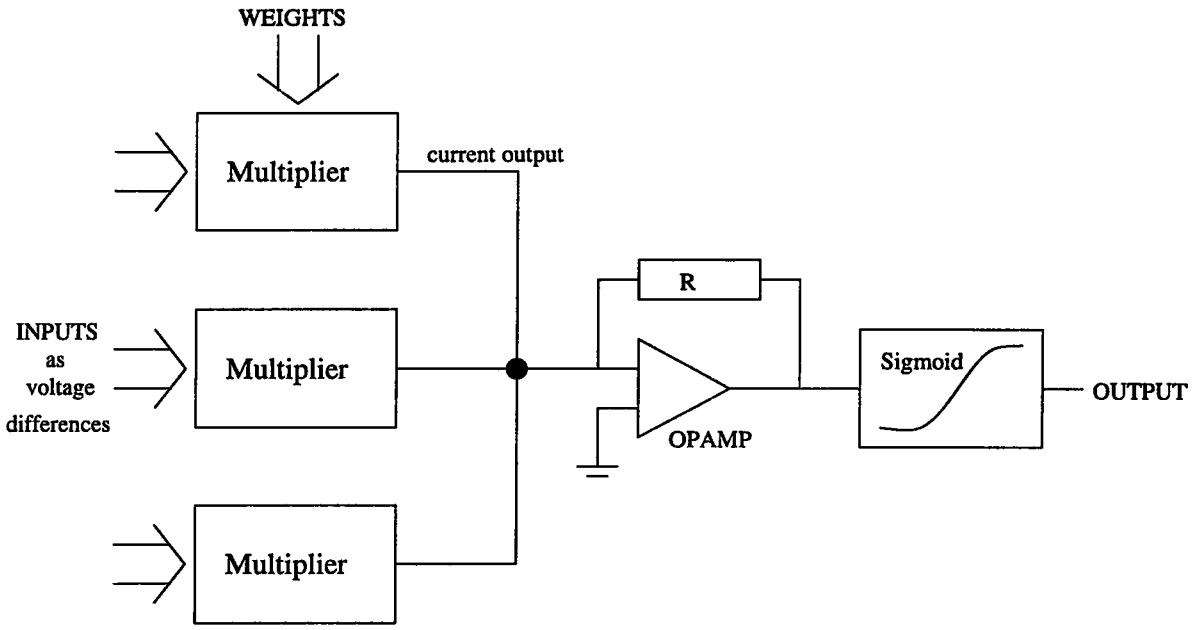

FIGURE 3.13. ANN unit cell.

TABLE 3.2. Dimensions of the building blocks in the prototype chip.

	transistor count	width μm	height μm	area mm^2
synapse	17	220	200	0.044
I-V converter	13	220	200	0.044
sigmoid	16	216	228	0.049
core	456	1280	1600	2.0
chip	—	3600	3740	13.45

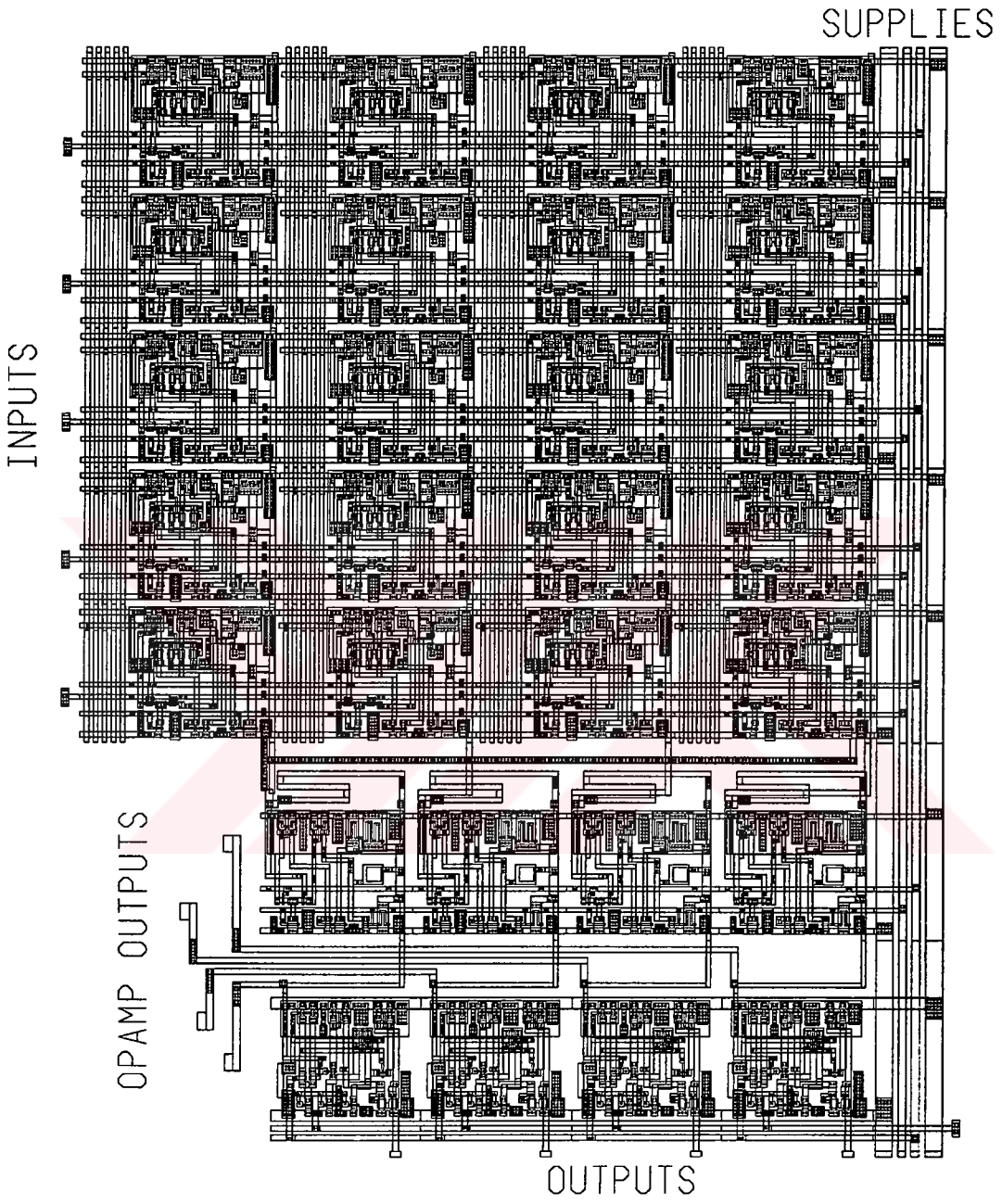

FIGURE 3.14. Core of the prototype ANN chip.

4. BACKPROPAGATION TRAINING OF NEURAL NETWORK HARDWARE

Different approaches are used to obtain the weights of an analog neural network which also dictate the implementation style and the architecture of the network [57]. In this chapter, hardware training for analog ANN will be investigated where the emphasis will be put on training On-Software-Only (OSO) method. OSO Training aims to eliminate the need for an on-line host computer (and the related interface circuitry) in the case of chip-in-the-loop training and the need for extra circuitry on the chip in the case of on-chip training methods. First, the training methods mentioned above will be examined and examples will be given. Meanwhile, several issues for hardware training for analog ANN will be highlighted. Then, the modified backpropagation algorithm will be explained which aims to perform the (initial) training on software using models of the analog neural network building blocks. Regarding the OSO hardware training, ANNSyS (An Analog Neural Network Synthesis System) will be investigated which uses the modified BP algorithm and weight perturbation (MADALINE Rule III) in circuit simulation, for the training. Several illustrative examples will be given for the OSO training which will demonstrate the potential of circuit simulation based training for analog ANN.

The most naive way to implement an analog ANN is the non-learning network. In this method, the weights are hardwired through the implementation of the fixed gain multiplier. The weights to be hardwired must be calculated before the design of the neural network layout. The calculations are done on a computer which uses the model of the analog neural network. The performance of this method depends heavily on the matching between the model and the real circuit, which is a task that is very difficult to achieve as will be shown in the following sections. An example of this type of system has been proposed in [103] which uses inverters as fixed weight multipliers where the transistor dimensions determine the weight associated with the multiplier. It is clear that this method does not find a wide-spread acceptance as a reliable system

due to the lack of flexibility and due to hardware mismatches.

4.1. Chip-in-the-Loop Training

Chip-in-the-loop training refers to training of analog (or digital) hardware where the weights can be adjusted externally during the training stage by means of a host computer. For this realization, the initial weights are usually computed on the host computer and downloaded to the chip. Then, the weights are fine tuned: The chip is used for forward pass, host computer is used for feedback (weight adaptation). The outputs of the neurons are sampled and fed to the host computer where the weight update computations take place. In this way, the matching of the model used for the initial training and analog hardware is considerably increased. The weight update may be based on the usual backpropagation algorithm using gradient descent or the weight perturbation algorithm both of which will be described in subsequent sections, in detail. Several applications of chip-in-the-loop training have appeared in the literature [68, 104, 105, 106, 107]. The main advantage of this method is that the host computer can monitor the actual outputs of the neurons and/or synapses so that the maximum available information is going to be used in the training algorithm. Chip-in-the-loop training is especially well tailored for perturbation based algorithms [52, 95, 108, 109] where the weights are perturbed by a small amount and the resulting change in the output error is used to update the weights. This is the so called MADALINE Rule III which, in theory, assumes that the synapses are linear; however, for chip-in-the-loop training, this assumption has to be left, hence, the weight update algorithm has to use a nonlinear model for the synapses. Chip-in-the-loop training implicitly compensates for the nonidealities of the hardware and for the mismatches occurring between identical blocks as the computations are carried out according to the actual outputs of the circuitry; however, the time required for the training heavily depends on the accuracy of the modeling of building blocks. A straight update rule of increasing or decreasing the weight by some fixed ratio based on the results of comparison between the desired

and actual outputs would require excessively long training times. A very safe way of getting successful results from chip-in-the-loop training is to obtain the models of the building blocks from measurements on the actual circuitry: The host computer may sweep the inputs of the building blocks so as to obtain the characteristics of the hardware components, so that the following training would really be based on the hardware [110].

Even though chip-in-the-loop training seems to be a reliable way of achieving hardware training of analog ANN, several disadvantages also exist:

- Chip-in-the-loop training requires a host computer for performing the weight updates and necessary interfacing circuitry with the actual chip for sampling the outputs. This leads to the fact that analog to digital and digital to analog converters have to be used which by themselves cause errors due to the quantization applied.
- There is a trade off between a large pin count of the chip (in order to sample as many block's outputs as possible), and the necessity of *good* modeling of the building blocks.
- Each individual analog ANN system has to be trained separately, that is, a trained chip can - most probably - not be changed by another sample since the training carried out does not take the mismatches between chips into account.
- Integration of analog ANN systems into larger complex systems-on-chip becomes very improbable, if not impossible, since the SOC paradigm would not allow extra area and extra pins for chip-in-the-loop training.

4.2. On-Chip Training

In analog neural networks with on-chip learning, both the feedforward structure and all circuitry required to adapt the weights are realized on the chip. As the backpropagation algorithm mainly uses simple arithmetic operations such as subtraction, multiplication, and addition along with taking derivatives, it can be realized by extra circuitry on the chip. A lot of implementations of on-chip training can be found in the literature using digital [84], analog [1, 24, 34, 35, 72, 79, 91, 111], optical [112], and mixed signal [27, 113] circuitry. The algorithm applied may be again either the gradient descent based backpropagation or weight perturbation. The main advantage of on-chip training is that the analog hardware does not need any external circuitry or computer for training. The user has to supply the training samples only. Moreover the nonidealities of the hardware and mismatches in the forward pass circuitry will implicitly be compensated whereas the mismatches of updating circuitry can not be prevented to affect the training adversely. The major disadvantage of this approach is that additional hardware is required which is used only at the training stage. The extra layout area brought by the learning circuitry can be very large in comparison to the actual forward propagation circuitry. Even though the actual ratio is not given in most of the references, data from [79] indicate that the synapse made up of a Gilbert multiplier and update circuitry occupies an area of $170 \times 140\mu m^2$ in a $0.7\mu m$ technology. The similar Gilbert multiplier based synapse in our chip occupies an area of $220 \times 200\mu m^2$ in a $2.4\mu m$ technology, which would be scaled down to approximately $70 \times 60\mu m^2$ in the $0.7\mu m$ technology. This reveals that approximately 85 per cent of the synapse is devoted to update circuitry. As an another example the value of 12 per cent can be cited from [35] which is the proportion of the synapse area in the die for a 64 synapse prototype chip. It is argued that the core efficiency increases for larger number of synapses on the chip. The extra layout area and power consumption due to the learning circuitry makes on-chip training difficult for SOC applications also. Another point of interest is the initialization of the weight values: Even though there is no clear reference for initialization of weight values prior to the training phase, an initial approximate training based on simple models of the analog ANN circuitry is

reported to be beneficial for on-chip learning [72, 112].

4.3. Modified Backpropagation Algorithm for Analog ANN Hardware

Training of the analog neural network hardware on software is another alternative. In this method, nonideal behaviour of analog neural network circuitry is modeled based on simulations or measurements, and the training software employs those models in the backpropagation learning [68, 72, 112, 114]. In fact, this type of modeling and training is usually required for obtaining a suitable initial weight set to be used in chip-in-the-loop or on-chip training. This section will investigate the modified backpropagation algorithm which is suitable for hardware training of analog ANN blocks which exhibit nonideal behaviour. For this purpose, consider a general multilayer perceptron structure depicted in FIGURE 1.2. The backpropagation algorithm as modified will be reviewed on this structure where for simplicity of notation, we limit ourselves to the scalar input-output case (Solid drawings in FIGURE 1.2). The synapse function will be denoted by $\mu(x, w)$ where x and w are the input and weight respectively. The output of the multilayer perceptron (y) is the opamp (neuron) output, $O(\mu)$ (a weighted sum of the outputs of a number (h) of hidden units, net_o), filtered through the nonlinear sigmoid function $\phi(x)$ as follows:

$$net_o = \sum_{i=1}^{h} \mu(T_i, H_i(x)) + \mu(T_0, 1) \tag{4.1}$$

$$y(x) = \phi(O(net_o)) \tag{4.2}$$

where x is the input, and T_i are the weights associated with the outputs of hidden units. T_0 is the bias weight. Output of each hidden unit is another similar expression, net_i being the net output of the synapses and O_h being the output of opamps at the hidden layer:

$$net_i = \mu(W_i, x) + \mu(W_{i0}, 1) \tag{4.3}$$

$$H_i(x) = \phi\left(O_h(net_i)\right) \tag{4.4}$$

where W_i are the weights associated with the hidden units. W_{i0} are the bias weights for the units in the hidden layer. Given a training set $\mathcal{X} = \{x^p, r^p\}_p$, where r^p is the desired output corresponding to the input x^p, the sum of squared errors is:

$$E(W,T) = \sum_p E^p = \sum_p [r^p - y(x^p)]^2. \tag{4.5}$$

In the backpropagation algorithm, parameters W, T are updated to minimize E using gradient-descent, where in the online version, for each pattern pair, the following updates are done:

$$\Delta T_i = \eta(r^p - y^p)\frac{d\phi}{dO}\frac{dO}{d\mu}\frac{\partial\mu(T_i, H_i(x^p))}{\partial T_i} \tag{4.6}$$

$$\Delta T_0 = \eta(r^p - y^p)\frac{d\phi}{dO}\frac{dO}{d\mu}\frac{\partial\mu(T_i, 1)}{\partial T_i} \tag{4.7}$$

$$\Delta W_i = \eta(r^p - y^p)\frac{d\phi}{dO}\frac{dO}{d\mu}\frac{\partial\mu(T_i, H_i(x^p))}{\partial H_i(x^p)}\frac{d\phi(O_h)}{dO_h}\frac{dO_h}{d\mu}\frac{\partial\mu(W_i, x^p)}{\partial W_i} \tag{4.8}$$

$$\Delta W_{i0} = \eta(r^p - y^p)\frac{\partial\mu(T_i, H_i(x^p))}{\partial H_i(x^p)}\frac{d\phi(x^p)}{dx^p}\frac{\partial\mu(W_{i0}, 1)}{\partial W_{i0}} \tag{4.9}$$

where η is the learning rate. So the learning method can be implemented using any synapse and nonlinearity if $\mu(x,w)$, $\partial\mu/\partial x$, $\partial\mu/\partial w$, O, O', ϕ, and ϕ' can be computed. Hence, the backpropagation learning can be realized if proper models of the building blocks can be obtained. Training of analog neural networks for analog hardware implementation utilizing the blocks discussed so far, requires special modifications: Given a training set $\mathcal{X} = \{x^p, r^p\}_p$ and a certain network topology, the input-output values need to be scaled such that they fall within the operational range of the analog circuitry. That is, input values have to be within the linear region ($-2V$ to $+2V$) of the multipliers, and output values have to be within the output range ($0V$ to $2.5V$) of the sigmoid. A further modification in the update rules is applied to favor small weight magnitudes so that the synapses operate in their linear region and the variations in the outputs due to mismatches are less severe as will be shown in CHAPTER 5. This is done by the *weight decay* technique which has shown to be effective in the improvement of generalization in the presence of noise [115, 116]. In weight decay, the error function

of (4.5) is modified to be:

$$E(W,T) = \sum_{p}[r^p - y(x^p)]^2 + \lambda \sum_{i} w_i^2 \tag{4.10}$$

where w_i are the weights in the network (including both W and T values) and λ is the decay factor. The first term is the usual sum of squared errors. The second term is the *penalty* term that penalizes weights that have large magnitude. Performing gradient-descent on this yields,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w} - \lambda w_i. \tag{4.11}$$

Thus at each iteration, there is an effect of pulling a weight towards zero unless pushed away to decrease the sum of squared errors. From a Bayesian perspective, weight decay corresponds to assuming that weights, w_i, are sampled from a Gaussian distribution with zero mean and variance $1/\lambda$. Minimizing (4.10) is the maximum a posteriori solution [117]. Using weight decay and a suitable choice of λ, weight values are found that minimize error and also stay in the linear region of the multiplier. This not only assures that the blocks operate in their "close to ideal" region, but also cause less variations as will be shown later. Moreover, it has also been reported in the literature that employing smaller weight magnitudes enhances the fault tolerance of the neural network by distributing the computation evenly to the neurons and synapses [118].

4.4. Modeling of Analog ANN Building Blocks for Backpropagation

As models of analog ANN building blocks are necessary for the backpropagation training of the hardware, macromodels for the synapse, opamp, and sigmoid circuits have been developed in this thesis. For modeling the blocks' behaviour, three different approaches are used, namely, approximation by neural networks, representation by look-up tables, and regression by analytical functions.

4.4.1. Modeling by Neural Networks

Approximators used are multilayer perceptrons for the synapse characteristics. Finite difference was used to calculate the derivatives of the approximators. The same technique was also used to approximate a partial derivative. The multiplier characteristics are given in FIGURE 4.1-b. The approximator is a multilayer perceptron with four hidden units. This neural network is trained using standard backpropagation. Data obtained from the HSPICE characterizations of the netlists extracted from the layout are used as training samples. Compared with the ideal multiplier, this exhibits nonlinear behaviour for large input x, with the nonlinearity getting pronounced for large weight values. The MLP can approximate the synapse characteristics well in the linear operation region whereas the deviations tend to increase for larger input values. The neural network approximator to the sigmoid circuit is a multilayer perceptron with one hidden unit; the model for the nonlinearity $\phi(x)$ approximates the sigmoid within 1.0 per cent accuracy.

4.4.2. Modeling by Table Look-Up

The synapses are modeled by a look-up table of size 81×81 where the inputs and weights varying between $-2V$ and $2V$ are quantized to 81 different values, each centered around $0V$ with 40 values residing on both positive and negative voltage axis. The closest value is chosen when the inputs or the weights do not match the values in the table. Derivatives are computed from finite differences. It is obvious that the training time required is inversely and the approximation performance is directly proportional to the size of the table. The 81×81 size was chosen as it gave superior performance to the previous method. It was observed that, larger look-up tables yielded longer simulation times with marginal improvements in approximation performance, whereas the weak approximation performance of smaller tables made this method inapplicable.

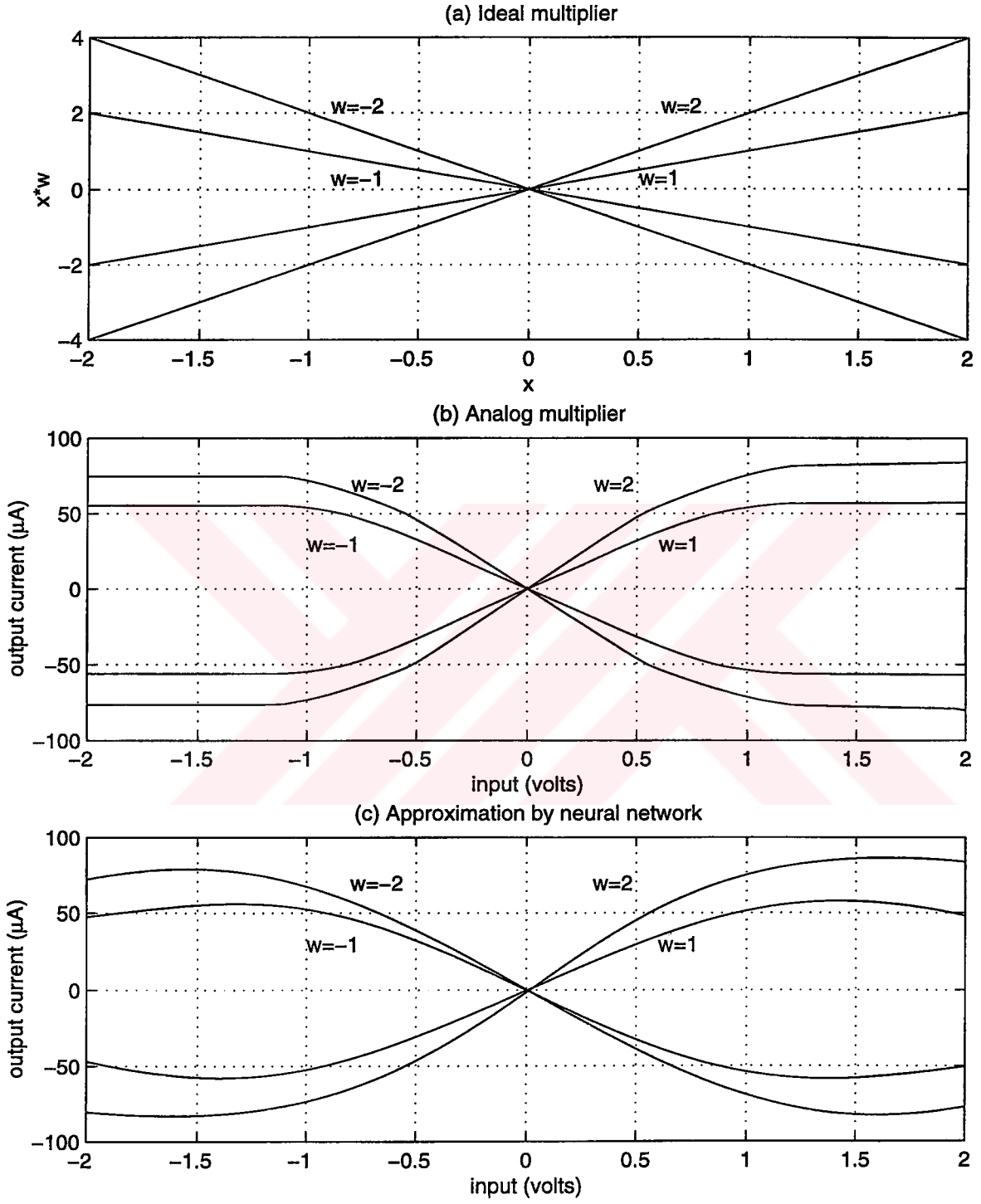

FIGURE 4.1. (a) Ideal multiplier (b) Multiplier data provided by HSPICE simulation (c) Neural network approximation to (b).

4.4.3. Modeling by Regression

Analytical expressions for the functional behaviour of the synapse and neuron circuits are not available. The synapse function can be modeled as a polynomial function with less than 5.0 per cent deviation, as follows:

$$\mu(x,w) = c_0 + (c_1x + c_2x^2 + c_3x^3 + c_4x^4)(c_5w + c_6w^2 + c_7w^3 + c_8w^4) \tag{4.12}$$

where $\mu(x,w)$ is the output current for the input pair x,w and the coefficients c_i are determined using a nonlinear regression program which may take nominal or Monte Carlo simulations as input. The opamp (I-V conversion) block can be represented by,

$$O(\mu) = \begin{cases} VDD & \text{if } -R\mu > VDD \\ -R\mu & \text{if } VDD > -R\mu > VSS \\ VSS & \text{if } -R\mu < VSS \end{cases} \tag{4.13}$$

where VDD and VSS are the positive and negative supply voltages, R is the resistance implemented (the value is $100k\Omega$), and μ is the total current supplied by the synapses. For the sigmoid block an exponential function with coefficients d_i is employed in regression:

$$\phi(x) = d_0 + \frac{d_1}{1 + exp(d_2x + d_3)}. \tag{4.14}$$

Regarding the quality of the approximations, several characteristics of the regression results are given in TABLE 5.4. The values for the coefficients c_i found by regression using the method of *least-squares fit* are given in TABLE 4.1. It is evident that the coefficients c_1 and c_5 for the linear terms dominate for small x,w values whereas the coefficients c_3 and c_7 for the cubic term come next in effect. Hence, it can be concluded that the nonlinearity of the multiplier is mainly in cubic terms. When all these models were compared in backpropagation training, it was observed that regression gave the best results for the neuron modeling and satisfactory results for synapse

modeling. Even though table look-up has performed better for synapse modeling, regression is used for modeling the synapses too, since the expressions obtained are easily differentiable so that the backpropagation algorithm can be programmed easily.

TABLE 4.1. Regression coefficients for the multiplier.

Coefficient	Value	Value normalized with respect to c_1
c_0	-2.480	$---$
c_1	-7.245	1
c_2	-0.047	0.0065
c_3	0.680	-0.093
c_4	0.0024	-0.0006
c_5	7.233	-0.9983
c_6	-0.039	0.0054
c_7	-0.384	0.0530
c_8	-0.011	0.0015

4.5. On-Software-Only Training

Since a perfect modeling is not possible, the models are based either on the results of HSPICE simulations or actual measurements which usually can not be expressed analytically. Small deviations from the actual physical behaviour in the models, may cause the analog neural network training to fail: The training is performed satisfactorily on the software; however, outputs of the actual circuitry may deviate heavily from the ideal training set [70]. As a remedy, we offered ANNSyS (An Analog Neural Network Synthesis System) through which the on-chip training of analog neural networks by software was enabled for backpropagation learning on multilayer perceptron structures. The main idea of ANNSyS can be summarized as follows: Models of the building blocks, namely the synapse (a multiplier), opamp, and the sigmoid block, are obtained from their HSPICE simulation outputs via regression. An initial backpropagation training based on the models is carried out using (4.9)-(4.11). Then, the actual circuitry is simulated in a special, fast DC simulator realizing the MADALINE Rule III.

The simulator, ANNSiS (An Artificial Neural Network Simulation System), has been developed by Bayraktaroğlu [119]. Hence, the weights obtained through approximate models are fine tuned by simulating the whole circuit using the most accurate models available (HSPICE models) which results in a heavy computational load. Details of this approach can be found in [120]. This method has been shown to be effective in the training of the analog neural network circuitry in software which will be shown by examples in the next section. However, it should be noted that the modeling only accounts for the nonidealities but not for mismatches found in analog hardware.

4.6. Numerical Experiments

ANNSyS has been applied to a number of neural network problems. Two rather small examples are the classical XOR problem and a sine function generator. Recognition of spoken phonemes is also included as a large size example.

4.6.1. Learning XOR Problem

For the XOR problem, a 2x3x1 structure is used. The weights are calculated via modified backpropagation using the models for the synapse and neuron circuits given in Section 4.4.3 and the rms error decreased below one per cent for the four training samples. At this time, the network is simulated by ANNSiS and the rms error was found to be 13 per cent for the XOR function. Finally, Madaline Rule III was applied for 50 epochs and the error obtained using the simulator decreased below one per cent again.

4.6.2. Learning Sine Function

A 1x4x1 structure was employed for the sine function ($0.3\sin(2\pi x)$) where the training set contains 20 pairs from the interval $(0, 1)$ (*desired* in FIGURE 4.2). Again, the neural network was trained via modified backpropagation until the error decreased below 1 per cent (*modified backprop* in FIGURE 4.2), and then the network was simulated by ANNSiS with the weights found. The error was around 30 per cent (*ANNSiS before Madaline* in FIGURE 4.3). Finally, Madaline Rule III was applied for 50 epochs and the error decreased below 3 per cent (*ANNSiS after Madaline* in FIGURE 4.3). The result of circuit simulation with weights obtained via the standard backpropagation training assuming ideal components, i.e., linear synapses and ideal sigmoid function, is also included (*backprop. with ideal components*) in FIGURE 4.2 to show that macromodeling is necessary for proper operation. It is also evident that training based on regression models is not adequate, and simulation based training employing Madaline Rule III can enable proper training of analog ANN.

4.6.3. Speech Phoneme Recognition

For /b, d, g, m, n, N/ speech phoneme recognition experiments, the database contains 5240 Japanese isolated words and phrases [7]. Two hundred samples for each class are taken from the even-numbered and odd-numbered words. 600 samples are used for training and 600 for testing. Phonemes are extracted from hand-labelled discrete word utterances and phrase utterances which have a sampling frequency of 12 KHz. Seven speech frames (each 10 ms) are used as input. For each 10 ms frame, 16 Mel-scaled Fast Fourier Transform coefficients are computed as feature values. This 112 dimensional input is then reduced to 21 using principal components analysis that explains 98.1 per cent of the variance. A 21x8x6 neural network structure is employed for the training. There were 600 samples and the modified backpropagation algorithm accomplished a correct classification rate of over 90 per cent by the software. However,

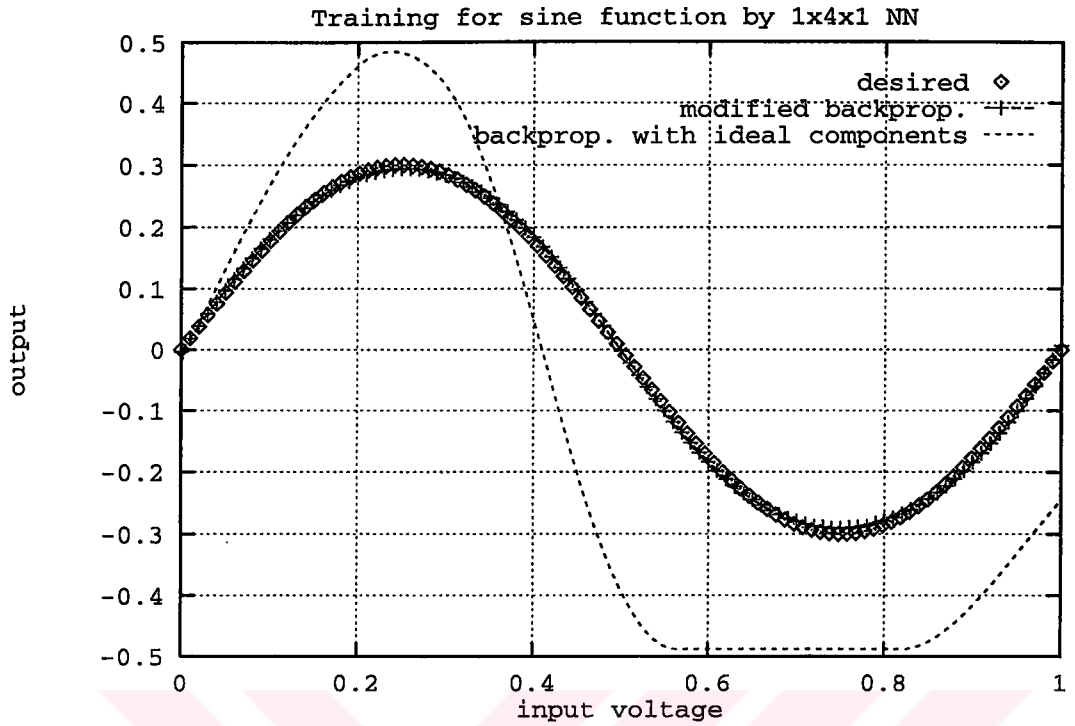

FIGURE 4.2. Training results for sine function (Part 1).

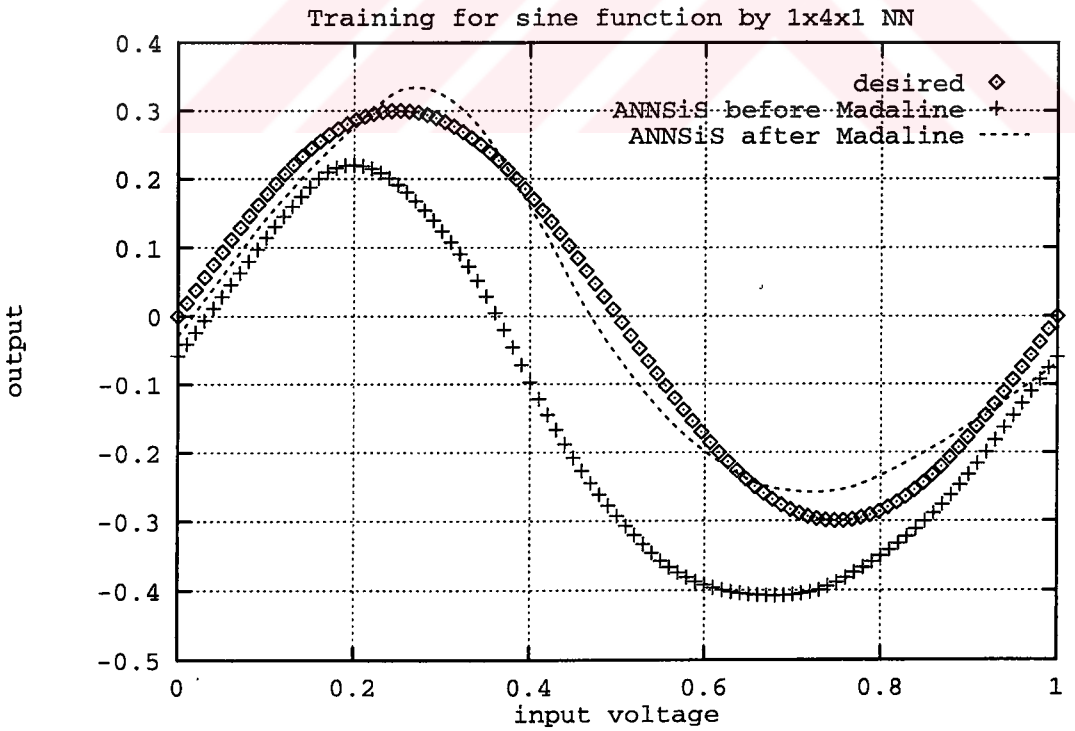

FIGURE 4.3. Training results for sine function (Part 2).

when the same neural network circuitry is simulated by ANNSiS, the correct classification rate dropped to 87 per cent and finally, the neural network is trained by the Madaline approach and the correct classification rate increased to 92 per cent. The results on the training and test samples are displayed in TABLE 4.2.

TABLE 4.2. Speech phoneme recognition.

Correct classification rate	Training set	Test set
Modified backpropagation	90%	72%
ANNSiS before Madaline	87%	65%
ANNSiS after Madaline	92%	78%

As can be deduced from the examples used, analog ANN can be modeled for OSO hardware training such that a satisfactory level of learning can be attained. However, the next chapter will introduce the problem of mismatches between identical blocks which detoriates the performance of the neural network heavily since the training methodology assumes that all identical blocks will exhibit the same characteristics.

5. SYSTEM LEVEL PROBLEMS IN HARDWARE TRAINING

Advances in VLSI technology has made it possible to realize the so called System-On-Chip (SOC) concept. The main idea for building a SOC is to integrate analog and digital functions on the same substrate. Hence, it can be stated that the need for implementing analog ANN in VLSI continues to be an important research area: Analog neural networks can be utilized in SOC applications where high performance classification, pattern recognition, function approximation, or control tasks have to be realized in real time. In this chapter, we will concentrate on multilayer perceptron neural networks implemented in MOS technology and trained by backpropagation algorithm, however, the assertions and results can be applied to other topologies and learning strategies as well. Despite their numerous advantages, analog implementations suffer from several shortcomings and nonidealities: The most important issue is the training of analog neural network circuitry, that is, determination of the weight set for optimal operation of the hardware which has been discussed in CHAPTER 2.

Training of the analog neural network hardware on software is another alternative as described in CHAPTER 4. An essential requirement for the SPICE simulations to have meaning is that the outputs of actual integrated circuits match the simulation results within some tolerance. Furthermore, the outputs of identical blocks in the same chip and on all different chips have to match in order to perform a meaningful design based on those components. Any deviation between the outputs of identical blocks would cause severe adverse effects in the overall system since they are not included in any part of the modeling. Unfortunately, statistical variations in the production process of integrated circuits result in the deviations mentioned above. As an example, the deviations in the prototype analog neural network produced in Mietec 2.4 micron technology are given in TABLE 5.1 for an input-weight pair of $1V - 1V$ for the neuron and for an input of $1V$ for the sigmoid block. There are 10 chips measured where the data are based on 200 identical synapses and 40 identical sigmoid blocks under

the same bias conditions. As it is clear from the data, such variations impose serious constraints on the training algorithm of the analog neural network: Each chip has to be trained individually since there are variations among "identical" chips. This makes it impossible to replace a trained chip by an off-the shelf component. The aim of our research is to overcome the problem of training with hardware nonidealities and variations so that, once a robust training of the system for a specific task is achieved, any chip of the same family can directly be utilized in the system.

TABLE 5.1. Sample data from measurements on the test chip.

Block	min.(V)	max.(V)	mean(V)	var.(V^2)
Synapse	1.9	4.15	2.97	0.38
Sigmoid	1.41	1.91	1.68	0.013

The variations occur due to several physical causes which will be explained in Section 5.1.1. Statistical process variations cause mismatch between parameters (threshold voltage V_T, current factor β) of identically designed transistors, which may accumulate to large deviations at the outputs of identical blocks. Such variations make it almost impossible to train the analog neural network circuitry without taking them into account. In this work, we will try to incorporate those variations into the training of analog neural networks. Closed form expressions of statistical variations from the nominal output are derived for the abovementioned circuits. These theoretical variations are compared to actual measurements obtained from chips and to Monte Carlo simulations. It seems evident from the comparison that those variations can be attributed to mismatches. In order to incorporate the variations at the output, the training algorithm (backpropagation) is modified: The building blocks are modeled according to their average outputs whereas the variations are considered to be noise with a certain normal distribution. Then, the training is carried out using "noisy" backpropagation where the outputs of the blocks are calculated in a probabilistic manner taking the noise into account (Section 5.2). In Section 5.3, sample simulations are conducted to verify that this method of training allows a higher degree of fault tolerance in the sense that noisy forward pass outputs exhibit better performance over outputs obtained from the network trained without including the variations. The noise (variations) can be estimated during the design of the circuitry using the data on transistor

geometries so that the usage of "noisy" backpropagation can be helpful for achieving a robust training for analog neural networks.

In general, those variations can also be considered as static hardware faults occurring in the forward operation since individual mismatches in transistor pairs result in variations at the outputs for the same set of inputs. The fault tolerance of the analog neural network hardware is severely degraded by such variations. There are several attempts to enhance the fault tolerance of ANN circuitry: Injecting synaptic noise during training has been shown to improve the fault tolerance performance of ANN considerably [72, 114, 118]. This could also be used as a means to overcome the problem associated with the training of the ANN since we can hope that the ANN trained by random noise injection would also be robust against statistical variations at neural network blocks. Section 5.3.3 constitutes a comparison between the two types of noise injection: based on modeling the transistor mismatch, and hence having a certain distribution related to the actual hardware, and based on noise injection with a random mechanism. It will be shown that modeling the variations due to mismatches will be more beneficial.

5.1. MOS Transistor Mismatch, Modeling and Verification

5.1.1. Modeling of MOS Transistor Mismatch

Mismatch between parameters of two identically designed MOS transistors, is the result of several random processes which occur during the fabrication phase. The essential parameters of interest are the zero-bias threshold voltage (V_{T0}), current factor (β) and the substrate factor coefficient (γ) which affect the current through the transistor. We will assume that all transistors are operating in the saturation region

described by the following formulae for the drain current I_D and threshold voltage V_T:

$$I_D = \frac{\beta}{2}(V_{GS} - V_T)^2 \tag{5.1}$$

$$V_T = V_{T0} + \gamma \left(\sqrt{\phi + V_{SB}} - \sqrt{\phi}\right) \tag{5.2}$$

where V_{GS} and V_{SB} are the gate-to-source and source-to-bulk potentials respectively, and ϕ is the surface potential. Any variations in these parameters of two matched transistors would cause a difference in the currents of equally biased transistors. The physical causes for variations in these parameters are fixed oxide charges, depletion charges, edge roughness, variations in substrate doping, oxide thickness and mobility values [121, 122]. Variations in any parameter may have systematic and random causes. Topography of the layout [123] and gradients in oxide thickness and wafer doping cause systematic variations in the parameters along a wafer and among different batches. On the other hand, random, local variations in physical properties of the wafer cause mismatch between closely placed transistors. Nonuniform distribution of dopants in the substrate and fixed oxide charges are responsible for local zero-bias threshold voltage mismatches whereas variations in substrate doping is the only cause for γ mismatch. The mismatch in current factor β is due to edge roughness and local mobility variations [121, 122, 124]. A mathematical analysis for the matching of MOS transistors was carried out by several researchers. The mismatch in a parameter is modeled by a normal distribution with zero mean. Moreover, the variance of the distribution for mismatches in the zero-bias threshold voltage, current factor and substrate factor coefficient can be expressed as [121, 122],

$$\sigma^2(\Delta V_{T0}) = \frac{A_{VT0}^2}{WL} + S_{VT0}^2 D_x^2 \tag{5.3}$$

$$\frac{\sigma^2(\Delta\beta)}{\beta^2} = \frac{A_\beta^2}{WL} + S_\beta^2 D_x^2 \tag{5.4}$$

$$\sigma^2(\Delta\gamma) = \frac{A_\gamma^2}{WL} + S_\gamma^2 D_x^2 \tag{5.5}$$

where A_{VT0}, A_β, A_γ, S_{VT0}, S_β, and S_γ are process related constants, W and L are the length and width of the transistors, and D_x is the spacing between the matched transistors. In case of two transistors with unequal width/length ratio, (W_1, L_1) and

(W_2, L_2), the term $\frac{1}{WL}$ will be replaced by $\frac{1}{2W_1L_1} + \frac{1}{2W_2L_2}$. Several other researchers worked on characterization of mismatch between two transistors using measurements obtained from test structures [125, 126, 127], and verified the mathematical expressions (5.3)-(5.5). It has been found in [126] that the threshold voltage mismatch for short-channel transistors ($L \leq 2\mu m$) is more complex:

$$\sigma^2(\Delta V_{T0}) = \frac{A^2_{1VT0}}{WL} + \frac{A^2_{2VT0}}{WL^2} - \frac{A^2_{3VT0}}{W^2L} \tag{5.6}$$

where three coefficients A_{1VT0}, A_{2VT0}, and A_{3VT0} are to be determined. It should also be noted that the distance dependent term can be eliminated from (5.3)-(5.5) for distances less than a few hundred micrometers since the production processes exhibit less gradients by the improvements in the last decade. This phenomenon is also justified by confidential technology documentations of several foundries involved in the Europractice program. In the remaining of this thesis, the original mathematical expressions will be utilized as the minimum channel length for the technology employed is $2.4\mu m$. However, computations verify that the distance dependent term can be neglected with respect to the area dependent term.

As can be seen in (5.3)-(5.5), the variance of the mismatch is inversely proportional to the area of the transistor and proportional to the square of the distance between the matched transistors. Hence, the variance can be minimized by placing matched transistors as close to each other as possible and by choosing large area transistors. However, the latter increases the total die area which should be avoided. A detailed analysis of the circuitry is necessary in order to determine transistor pairs which are most critical for mismatch behaviour so that they are designed accordingly. This requires that mismatches and their cumulative effects on the circuitry have to be modeled and simulated. A statistical MOS model has been developed in [128, 129], which allows the designer to determine circuit output variance due to mismatches in device parameters. However, the simulation methodology requires that a set of test structures have to be built and measured for each specific technology in order to gain knowledge on the model parameters in question. Then, the modeling has to be incor-

porated into a simulation environment. It is clear that this is not an easy task to be performed. There are also other attempts for modeling the mismatch based on the circuit structure [29, 130, 131]. Once the effects of mismatches can be predicted, changes in the transistor dimensions can be employed as an optimization aid. In this thesis, we primarily concentrate on mismatches between pairs of transistors and derive their effects on outputs analytically. Simulations and measurements will also accompany the mismatch analysis in Section 5.1.3.

The differential pair and current mirror are among the most frequently used structures in analog integrated circuits. This is also the case for our analog neural networks where the building blocks are the synapse (Gilbert type, 4-quadrant, current output multiplier), an opamp (used for adding synapse outputs and converting them to voltage) and the sigmoid block as shown in FIGURE 3.1, FIGURE 3.4, and FIGURE 3.6 respectively. The prototype chip contains four neurons, each having five synapses connected to it. The chip has been produced in Alcatel-Mietec 2.4μm technology. The transistors in the differential pairs are designed with identical W/L ratio for symmetrical operation. Similarly the current mirrors used as active loads in the circuitry have matched transistors. According to the experimental results of [122] and the data collected from the technology design kits, the process dependent constants for the process were estimated since they were not publicly available (TABLE 5.2). Using the data, calculation of the variance of the mismatch in matched transistor pairs of differential stages and current mirrors is possible. It should be noted that these parameters may not be exact; however, the general methodology for estimation of mismatch at outputs of neural network modules is not affected by that.

TABLE 5.2. Parameters for mismatch analysis.

	A_{VT0} $(mV\mu m)$	S_{VT0} $(\mu V/\mu m)$	A_β $(\%\mu m)$	S_β $(10^{-6}/\mu m)$	A_γ $(V^{0.5}\mu m)$	S_γ $(10^{-6}V^{0.5}/\mu m)$
NMOS	25	4	2.5	2	0.016	4
PMOS	30	4	3	2	0.012	4

The ultimate aim of mismatch analysis is to predict the mismatch at the outputs of the building blocks using the variances at individual pairs. In order to accomplish this task, variations at outputs of building blocks have to be expressed in terms of the variations in the matched pairs. The analysis is carried out for two separate cases: Either there are only variations in the threshold voltage (due to mismatches in zero-bias threshold voltage V_{T0} and substrate factor γ) or in current factor β of matched pairs. Mismatches in the threshold voltage and current factor are shown to be independent [121], and the common physical causes of mismatches for zero-bias threshold voltage and substrate factor let us expect a positive correlation between mismatches in these parameters. In fact, empirical formulae for correlation coefficients between the three parameters were obtained from measurements on test structures in [127]. We have computed the correlation coefficients for transistor pairs with different W/L ratio ranging from 40/40 to 1/1 spanning intermediate combinations. Our results indicate that there is a weak negative correlation (around -0.3 for NMOS and around -0.2 for PMOS transistors) between mismatches in the zero-bias threshold voltage and current factor. The correlation between mismatches in substrate factor and zero-bias threshold is weakly positive: around 0.4 for NMOS and around 0.3 for PMOS transistors. Finally, the correlation between mismatches in current factor and substrate factor depend heavily on the width W of the transistor pair: It ranges from 0 to -0.6. Hence, the variations in different parameters may be considered to be independent so as to simplify the computations. It should also be noted that the substrate factor only becomes effective in the differential pair input of the synapse circuit of FIGURE 3.1, so that the simplification does not cause a considerable error. Individual mismatches due to each parameter will be computed and the results will be combined later.

5.1.2. Mismatch Based Variations: Modeling for ANN Circuitry

In the following, the variance of output current for the synapse circuit of FIGURE 3.1 is derived. In the case of V_{T0} mismatches, the total mismatch in the output current Z can be attributed to mismatches in current mirrors $(1-5)$ and differen-

tial pairs. For the current mirror of FIGURE 5.1, the mismatch in current I_{D2} for a mismatch of ΔV_{T0} in the transistor M2 is derived as follows:

$$I_{D1} = \frac{\beta}{2}(V_{GS} - V_T)^2 \tag{5.7}$$

$$I_{D2} = \frac{\beta}{2}(V_{GS} - V_T - \Delta V_{T0})^2 . \tag{5.8}$$

Expanding (5.8), and neglecting the term ΔV_{T0}^2, I_{D2} becomes

$$I_{D2} = I_{D1}(1 - \frac{2\Delta V_{T0}}{V_{GS} - V_T}) . \tag{5.9}$$

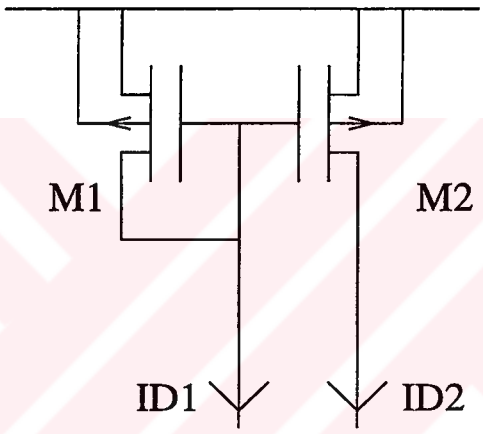

FIGURE 5.1. Current mirror.

For mismatches caused by the current mirrors, the analytical expression for output current for the synapse circuit is

$$Z = -7\,(\,(1 - M_3)(1 - M_4)U - (1 - M_5)T) \tag{5.10}$$

with

$$\begin{aligned} U &= \frac{1}{2} I_1 (1 - M_1) + \frac{1}{2} I_2 (1 - M_2) \\ &\quad -\Delta y \frac{\sqrt{\beta_y}}{2} \left(\sqrt{I_1(1 - M_1)} - \sqrt{I_2(1 - M_2)} \right) \end{aligned} \tag{5.11}$$

$$T = \frac{1}{2} I_1 (1 - M_1) + \frac{1}{2} I_2 (1 - M_2) + \Delta y \frac{\sqrt{\beta_y}}{2} \left(\sqrt{I_1(1 - M_1)} - \sqrt{I_2(1 - M_2)} \right) \quad (5.12)$$

where I_1 and I_2 are currents flowing through the transistors of the input (x) differential pair, $M_i = \frac{2\Delta V_{T,i}}{V_{GS,i} - V_{T0,i}}$ are the factors of mismatch in the current mirror i due to the mismatch $\Delta V_{T,i}$, β_y is the current factor of the transistors in the weight (y) differential pairs, and Δy is the weight $(y_2 - y_1)$. There are five such current mirrors, and the output current Z can be expressed as a function

$$Z = f(\Delta V_{T,1}, \Delta V_{T,2}, \Delta V_{T,3}, \Delta V_{T,4}, \Delta V_{T,5}) \quad (5.13)$$

for the mismatch analysis. Here, $\Delta V_{T,i}$ are considered to be random variables from a normal distribution with zero mean and variance as given by (5.3). Then, the variance in Z due to the zero-bias threshold mismatches in current mirrors would be [132],

$$\sigma(\text{mirror}, V_{T0})_Z^2 = \sum_{i=1}^{5} \left(\frac{\partial f}{\partial \Delta V_{T,i}} \right)^2 \sigma_{\Delta V_{T,i}}^2 \quad (5.14)$$

where the partial derivatives are computed with the zero mean value of the random variables $\Delta V_{T,i}$. Similarly, if the variation due to the threshold voltage mismatches in the differential pairs alone, is considered, the output current Z becomes,

$$Z = -7\sqrt{\frac{\beta_x \beta_y}{2}} \left(\Delta x + \Delta V_{T0,x} + (\gamma_x + \Delta\gamma_x) \left(\sqrt{\phi + V_{SB,x}} - \sqrt{\phi} \right) \right) (\Delta y + \Delta V_{T0,y}) \quad (5.15)$$

where $\Delta x = (x_2 - x_1)$ is the input to the multiplier, $\Delta\gamma_x$ is the mismatch of substrate factor for the input differential pair, $\Delta V_{T0,x}$ and $\Delta V_{T0,y}$ are the mismatches at the differential pairs of input and weight respectively. Then, the variance at Z would be,

$$\sigma(\text{diff-pair}, V_{T0})_Z^2 = \frac{49\beta_x \beta_y}{2} \left(\Delta y^2 \sigma_{\Delta V_{T0,x}}^2 + \Delta y^2 \left(\sqrt{\phi + V_{SB,x}} - \sqrt{\phi} \right)^2 \sigma_{\Delta\gamma_x}^2 + \left(\Delta x + \gamma_x \left(\sqrt{\phi + V_{SB,x}} - \sqrt{\phi} \right) \right)^2 \sigma_{\Delta V_{T0,y}}^2 \right). \quad (5.16)$$

Following the argumentation above, variations at output current due to the mismatches in the current factor β can be computed for the existence of mismatch at current mirrors and differential pairs. For the current mirror of FIGURE 5.1, a mismatch of $\Delta\beta$ in the transistor M2 would give the following mismatch in the current I_{D2}:

$$I_{D1} = \frac{\beta}{2}(V_{GS} - V_T)^2 \tag{5.17}$$

$$I_{D2} = \frac{\beta + \Delta\beta}{2}(V_{GS} - V_T)^2 . \tag{5.18}$$

Expanding (5.18), and rearranging the terms, I_{D2} becomes,

$$I_{D2} = I_{D1}(1 + \frac{\Delta\beta}{\beta}) . \tag{5.19}$$

If mismatches at current mirrors only are considered, Z becomes,

$$Z = -7((1 + Q_3)(1 + Q_4)U - (1 + Q_5)T) \tag{5.20}$$

where $Q_i = \frac{\Delta\beta_i}{\beta_i}$ are the factors of mismatch in the current mirror i due to the mismatch $\Delta\beta_i$, and all the terms $(1 - M_i)$ will be replaced by $(1 + Q_i)$ in (5.11)-(5.12). Hence, Z becomes a function of mismatches in β as,

$$Z = g(\Delta\beta_1, \Delta\beta_2, \Delta\beta_3, \Delta\beta_4, \Delta\beta_5) \tag{5.21}$$

and, the variance in Z due to β mismatches in current mirrors is,

$$\sigma(\text{mirror}, \beta)_Z^2 = \sum_{i=1}^{5} \left(\frac{\partial g}{\partial \Delta\beta_i}\right)^2 \sigma_{\Delta\beta_i}^2 . \tag{5.22}$$

Also the variance due to β mismatches in differential pairs can be derived after some manipulation as:

$$\sigma(\text{diff-pair}, \beta)_Z^2 = \frac{49\Delta x^2 \Delta y^2}{8} \left(\frac{\beta_y}{\beta_x}\sigma_{\Delta\beta,x}^2 + \frac{\beta_x}{\beta_y}\sigma_{\Delta\beta,y}^2\right) \tag{5.23}$$

where β_x and β_y are current factors of the input and weight differential pairs respec-

tively. Finally, the total variation at the output current can be approximated by,

$$\sigma_Z^2 = \sigma(\text{mirror}, V_{T0})_Z^2 + \sigma(\text{diff-pair}, V_{T0})_Z^2 + \sigma(\text{mirror}, \beta)_Z^2 + \sigma(\text{diff-pair}, \beta)_Z^2 \quad (5.24)$$

which assumes that the individual mismatches due to the parameters V_{T0} and β are independent. Dealing with the opamp and sigmoid blocks in the same way, analytical expressions for the variance at the outputs of those blocks can also be derived.

Considering the mismatches, the sigmoid output will be approximated by,

$$\begin{aligned} V_{out} &= (V_Q + \Delta V_Q) + \frac{1}{4}(K + \Delta K)(\beta_1 + \Delta\beta_1)(\beta_2 + \Delta\beta_2)(x + \Delta V_{T1})(R_L + \Delta R_L) \\ &\quad \sqrt{\frac{4I_{ss1}}{\beta_1 + \Delta\beta_1} - (x + \Delta V_{T1})^2}\sqrt{\frac{4I_{ss2}}{\beta_2 + \Delta\beta_2}} \end{aligned} \quad (5.25)$$

based on (3.13). Hence, the sigmoid output can be expressed as a function of mismatches, and the variance can be calculated as,

$$\begin{aligned} \sigma_{V_{out}}^2 &= \frac{\partial V_{out}}{\partial \Delta V_Q}^2 \sigma_{\Delta V_Q}^2 + \frac{\partial V_{out}}{\partial \Delta K}^2 \sigma_{\Delta K}^2 + \frac{\partial V_{out}}{\partial \Delta \beta_1}^2 \sigma_{\Delta\beta_1}^2 + \\ &\quad \frac{\partial V_{out}}{\partial \Delta \beta_2}^2 \sigma_{\Delta\beta_2}^2 + \frac{\partial V_{out}}{\partial \Delta V_{T1}}^2 \sigma_{\Delta V_{T1}}^2 + \frac{\partial V_{out}}{\partial \Delta R_L}^2 \sigma_{\Delta R_L}^2 . \end{aligned} \quad (5.26)$$

It should be noted that those values of variance depend on the inputs of the blocks, i.e., σ_Z^2 is a function of y and x (weight and input), and $\sigma_{V_{out}}^2$ is a function of sigmoid input.

5.1.3. Mismatch Based Variations: Verification in the Test Chip

As mentioned previously, a prototype analog neural network chip containing four neurons has been designed and manufactured through the Europractice program. The structure of each neuron is such that current outputs of five synapses are connected to a summing node where the current is then converted to voltage by means of an

opamp. Voltage output of the opamp is applied to the sigmoid block (FIGURE 3.13). We obtained 10 samples of the chip, hence there are 200 synapses, 40 neurons and 40 sigmoid blocks. During the design phase of the chip little effort has been spent for preventing mismatches. The mismatches of individual synapses could not be observed since they are current outputs, and they are not available as a pin on the chip. The chips have been tested using the LabView virtual instrumentation program employing a special data acquisition environment: At each step of the measurement for a neuron, four pairs of synapse inputs and weights are set to $0V$ while the fifth synapse input and weight values are swept over a range of $-5V$ to $+5V$ and $-2V$ to $+2V$, respectively. This procedure has been repeated for each of the 200 synapses so as to obtain characteristics of each synapse as seen on the neuron output (sigmoid input). Meanwhile, outputs of sigmoid block are also measured so that sigmoid blocks are characterized. The outputs are sampled by the data acquisition card and stored on a host computer for further analysis. In order to verify the measurements, 40 of the synapses and all sigmoids have also been tested manually by connecting all inputs externally and by measuring the output voltage values by a digital multimeter. Both measurements give identical results. Meanwhile, it has been observed that three neuron outputs are faulty, that is their outputs are stuck at the supply voltage. This may be the result of stuck-at faults either at one of the synapses or at the opamp.

In order to establish a solid framework for the mismatch analysis of the synapse circuit, HSPICE simulations of the synapse with nominal model parameters were performed. FIGURE 3.3 exhibits the results for the output current of the synapse for weight values of $w = \{-2, -1.5, ..., 2\}V$ where the input is swept over the range $[-2.5, 2.5]V$. The variance of mismatch for the parameters V_{T0}, β and γ, are computed using (5.3)-(5.5), and they are given in TABLE 5.3 for current mirrors and differential pairs of the synapse circuit. Then, 100 Monte Carlo DC sweep runs are simulated for a single synapse incorporating the mismatches in the above mentioned parameters. Each mismatch parameter has a zero mean and the variance as given in TABLE 5.3. That is, for each run of the Monte Carlo sweep, parameters of one transistor in each current mirror and differential pair are disturbed by a normal distribution in their model cards.

In this way, we obtained a family of characteristic values for each input-weight pair. The average of the Monte Carlo simulation is given in FIGURE 5.2. Moreover the variance of the sample of size 100 is computed for each input-weight pair, and the values are plotted in FIGURE 5.3 for nonpositive weight values. The variance of the synapse current Z exhibits a symmetrical behaviour as given by $\sigma_Z^2(x,w) = \sigma_Z^2(-x,-w)$, hence the curves for positive weight values are obtained by the reflection of the curves for negative weight values with respect to the y-axis. Finally, the mismatch in the synapse output current is also calculated using (5.10)-(5.24). This corresponds to the theoretical analysis of the mismatch based on the estimates of individual mismatches (TABLE 5.3), and DC simulations for nominal parameters. The results are again plotted for nonpositive weight values in FIGURE 5.4 in sake of clarity since the same symmetry also exists for the theoretical computations. It is evident from FIGURE 5.3 and FIGURE 5.4 that the mismatch in synapse current as computed by theoretical analysis and as simulated, exhibit a close relationship. Even though the values do not coincide exactly, the general shape of the curves are similar. The deviations may be the result of several factors: Our assumption that the mismatches in the parameters are not correlated may not hold exactly. Moreover the equations for the drain current and the threshold voltage employed in HSPICE are not as simple as (5.1)-(5.2). It should be noted that the variance is larger as input and/or weight magnitude increases. This has to be taken into consideration in the training issue, and we will also discuss this in the next section.

TABLE 5.3. Mismatch in the current mirrors and differential pairs of the synapse circuit.

	M1	M2	M3	M4	M5	diff,x	diff,y
$\sigma^2_{\Delta V_{T0}}(mV^2)$	78	78	72	11	26	27	15
$\sigma^2_{\Delta\beta}(\mu A/V)^2$	0.03	0.03	0.19	0.004	0.07	0.36	0.005
$\sigma^2_{\Delta\gamma}$	0	0	0	0	0	0.63e-3	0

In order to test measurement results on actual chips, a neuron circuit made up of five synapses is also simulated for 100 Monte Carlo runs where each current mirror and differential pair in all of five synapses and the opamp are disturbed by the appropriate amount of mismatch. The variance obtained is given in FIGURE 5.5 whereas

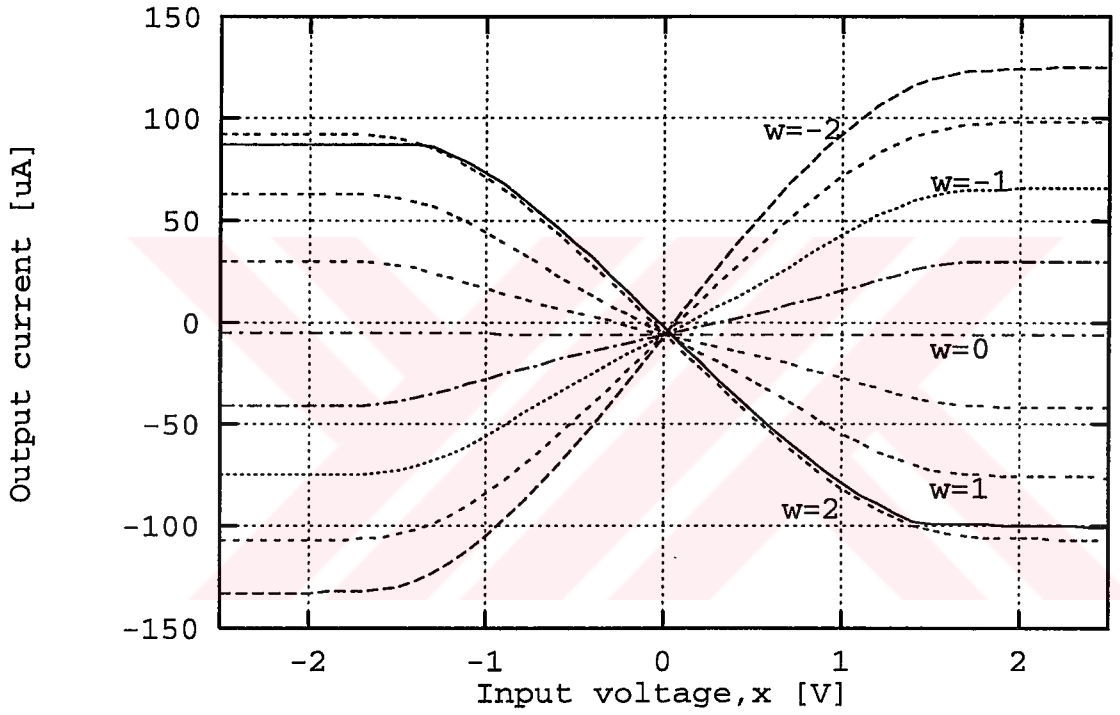

FIGURE 5.2. Average of 100 Monte Carlo runs for the synapse with mismatched model parameters.

FIGURE 5.6 exhibits the measurement results using the LabView environment. The close agreement between the two families of curves leads us to the conclusion that our approach of mismatch based deviations holds. Moreover, this also suggests that the process dependent parameters have been estimated close to the actual values.

A similar analysis has also been carried out for the sigmoid block. Three types of results are given in FIGURE 5.7 and FIGURE 5.8 for the sigmoid output and for the variance in sigmoid output, respectively:

- simulation results with nominal model parameters and variance based on theoretical formulae of (5.13)-(5.26);
- average (variance) of 100 runs of Monte Carlo simulations with mismatched model parameters;
- average (variance) of LabView measurements on actual chips.

Again the similarity of curves in FIGURE 5.7 and FIGURE 5.8 supports our claim of mismatch based variations. Although the models for mismatches are not perfect, the simulation and measurement results indicate that our approach forms a realistic way of predicting block level variations based on individual transistor level mismatches. Once those variations can be predicted during the design cycle, necessary modifications are possible on the actual transistor level design of the analog circuitry: The theoretical formulae give the sensitivity of variations at circuit outputs on individual mismatches in current mirrors and differential pairs. A detailed investigation on individual contributions from those pairs may label the more critical pairs, so that their sizes (W and L) can be increased. Starting from the highest level of the hierarchy (system level) the allowable variances can be defined for each block and sub-blocks. Consequently, the allowable level of variance for sub-blocks can be achieved by using the theoretical formulae and/or the Monte Carlo simulations.

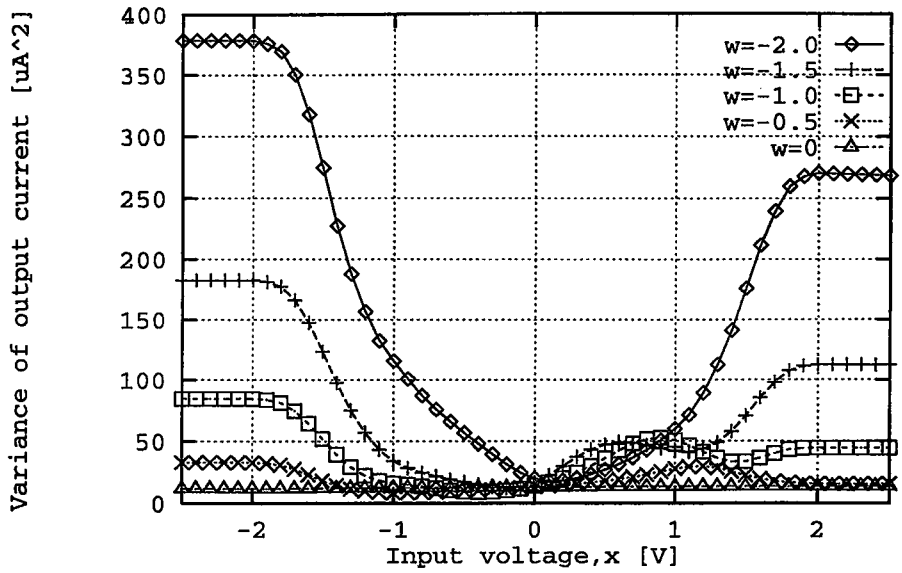

FIGURE 5.3. Variance in synapse output obtained from 100 Monte Carlo runs with mismatched model parameters.

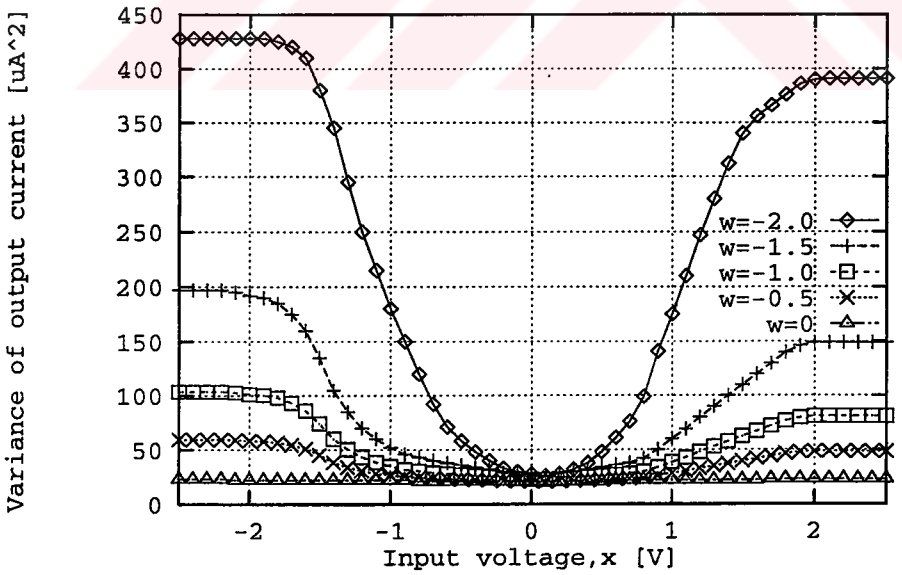

FIGURE 5.4. Variance in synapse output obtained using the formulae.

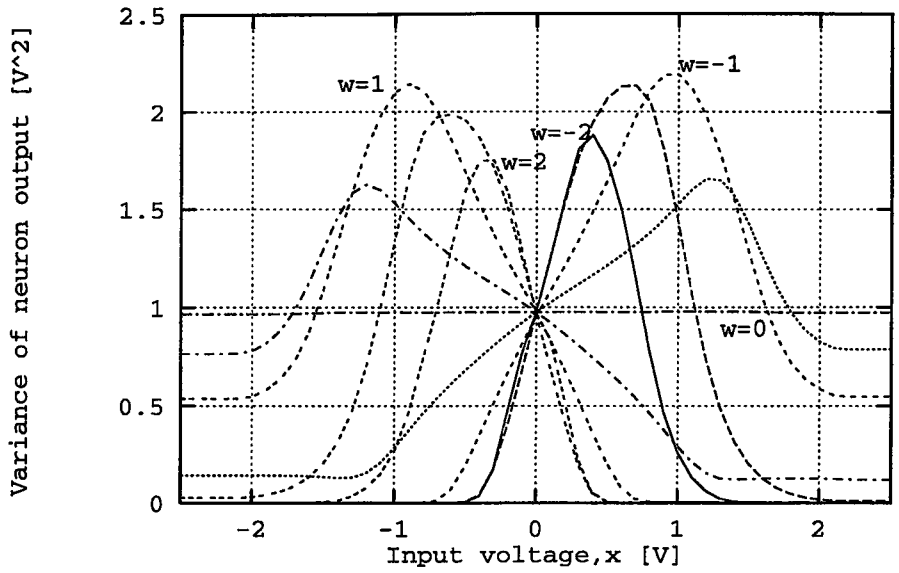

FIGURE 5.5. Variance in neuron output obtained from 100 Monte Carlo runs with mismatched model parameters.

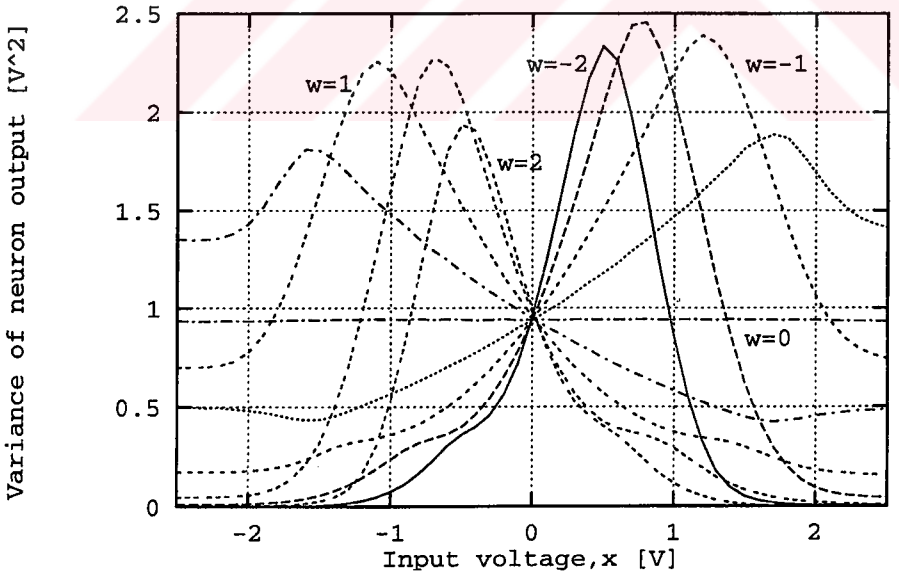

FIGURE 5.6. Variance in neuron output obtained from actual measurements on the chips.

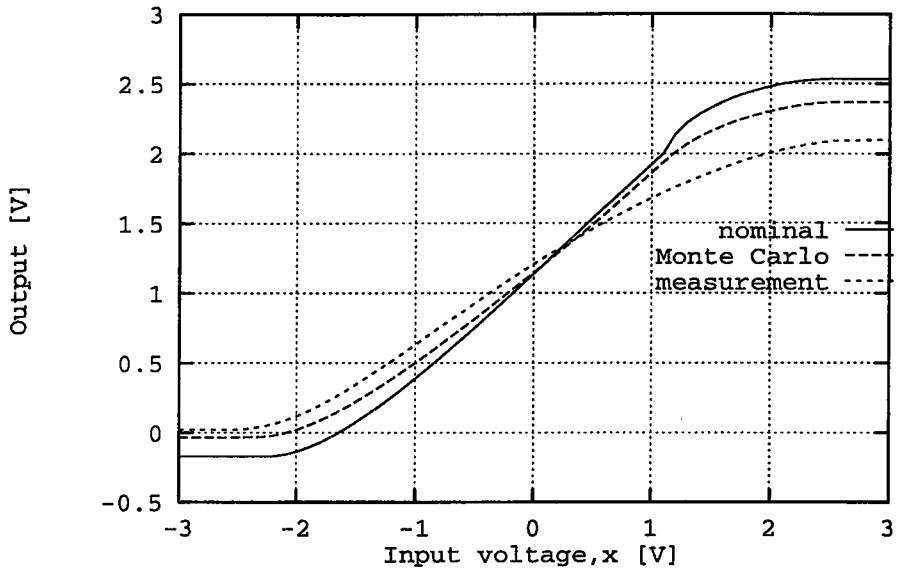

FIGURE 5.7. Characteristic of the sigmoid block.

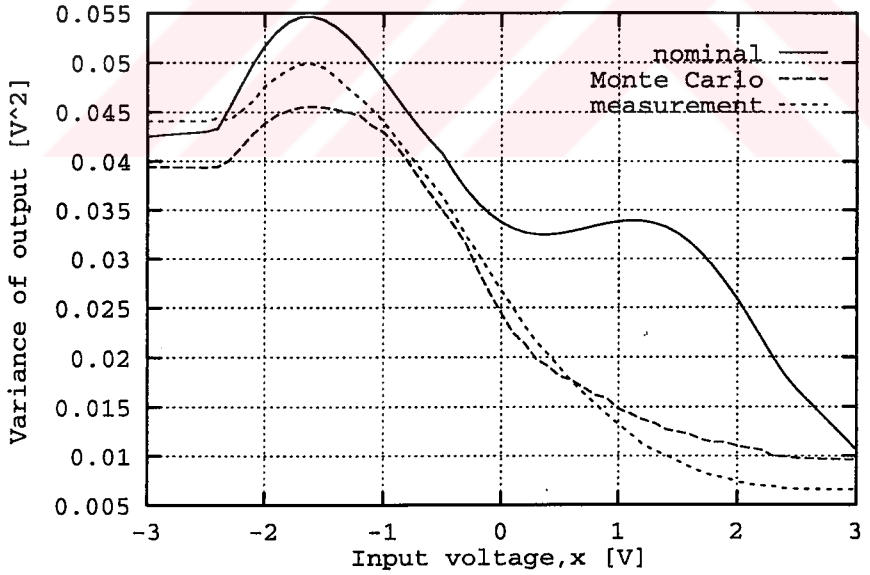

FIGURE 5.8. Variance in sigmoid output.

5.2. Incorporation of Mismatch into Backpropagation Training

Building blocks in an analog neural network are nonideal, that is, the synapse circuit does not perform the actual multiplication operation, instead, the output depends nonlinearly on both the weight and input. Similarly the summation of currents and I-V conversion at the neuron is performed by the opamp so that saturation takes place. The transfer characteristic of the opamp for our system is given in FIGURE 5.9 for the average of 100 Monte Carlo runs. Also the activation function which we denote by the term "sigmoid", does not represent the ideal sigmoid function as given by

$$\phi_{ideal}(x) = \frac{1}{1 + exp(-x)}. \tag{5.27}$$

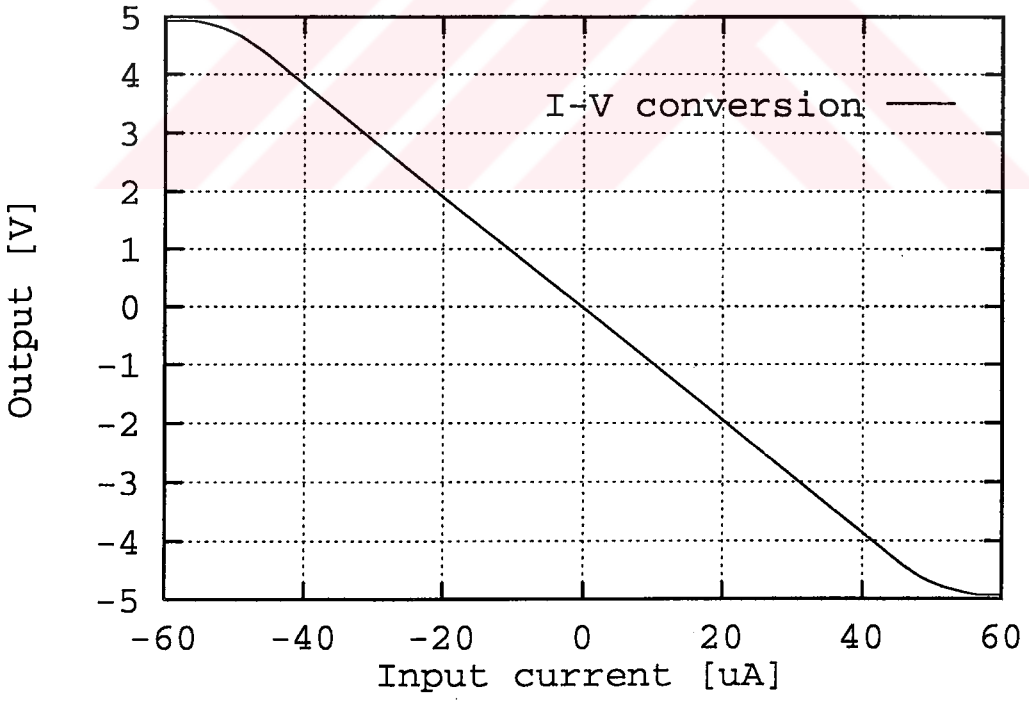

FIGURE 5.9. I-V conversion characteristics of the opamp.

It should be noted that the curves in FIGURE 5.2, FIGURE 5.7, and FIGURE 5.9 represent average behaviour for the analog neural network building blocks. There are also variations at outputs of identically designed blocks in the same (different) chip(s). Our primary concern is to incorporate the circuit nonidealities together with hardware variations into the backpropagation training. Several researchers have investigated effects of hardware nonidealities, and they concluded that chip-in-the-loop training has to be carried out where hardware models of ANN blocks are utilized [68, 112]. The requirement that each individual chip-set has to be trained separately in order to avoid the effects of variations among identical blocks degraded the applicability of this approach. However, they have shown that circuit nonidealities (nonlinearity of synapses etc.) do not form an obstacle against the learning ability of the networks. This has also been supported by the work in [72, 133] where variations among blocks (e.g. in synapse gain) are modeled by a probability distribution and applied to the circuits as static deviations. The conclusion was that ANN circuits could tolerate variations if they are known (and fixed) during the training stage. Random variations, on the other hand, could not be tolerated. Injecting noise into the inputs, weights or outputs during the backpropagation training has also been utilized for generating a fault tolerant neural network which, by definition, should also be tolerant to such variations. Theoretical derivations of [115, 134] for linear perceptrons which can be extended to nonlinear networks, revealed that injecting noise into inputs improves the generalization ability which is also justified by the work in [135] and [136]. Even though those theoretical derivations may give some hints on noise injection for better generalization, the robustness of training for the analog neural networks is not guaranteed by improved generalization since the problem for the analog hardware extends more into the randomness of the variations between identical blocks.

The concept of noise-enhanced learning was introduced by Murray [37]. According to his studies, injecting analog noise to weights during the training dramatically enhanced the generalization of the neural network. The effect of synaptic noise was to distribute the dependence of the output evenly across the weight set [118]. The simulations with ideal neural network elements revealed that injection of multiplicative

analog noise (whose value is reduced to a very small value as the network succeeds learning), during training would yield a more robust network against faults created by removal of synapses or by perturbation of final weight values [118]. Clearly, such an improvement would also be beneficial for analog neural network hardware where variations at outputs of building blocks are encountered. In [114], it is reported that an ANN hardware has been modeled in software for faults, and the enhanced fault tolerance of the network by means of weight noise is verified via simulations. However, they needed to measure the relative slope for each of the multiplier in the neurons for modeling, which is again a prohibitive matter. The limited dynamic range as a hardware restriction, prohibited the full utilization of the enhancement too: Large weight values obtained by the training could not be used.

Attempts to solve the problem of training in nonideal hardware building blocks in the presence of variations have not resulted in a satisfactory solution yet. In our approach, the ANN hardware blocks, as realized in VLSI, are modeled using analytical expressions for their input-output relations which has also been suggested in previous work [68, 72]. On the other hand, the actual dependence of variations at outputs of analog building blocks are investigated through theoretical derivations (Section 5.1.2), and extensive simulations and measurements (Section 5.1.3) so as to model the actual variations depending on the inputs of the analog circuitry. During backpropagation training, the variations are modeled according to the mismatch analysis of the circuitry, and they are considered as zero mean additive noise from a normal distribution with known variance depending on the input values of the blocks. For this aim, a new training scheme is developed (FIGURE 5.10).

The behaviour of the synapse, opamp, and the sigmoid have been modeled in order to be used in the backpropagation algorithm. Regarding the variances in synapse and sigmoid blocks, again regression is employed by fitting the following polynomial functions to the data obtained from theoretical computations, Monte Carlo simulations

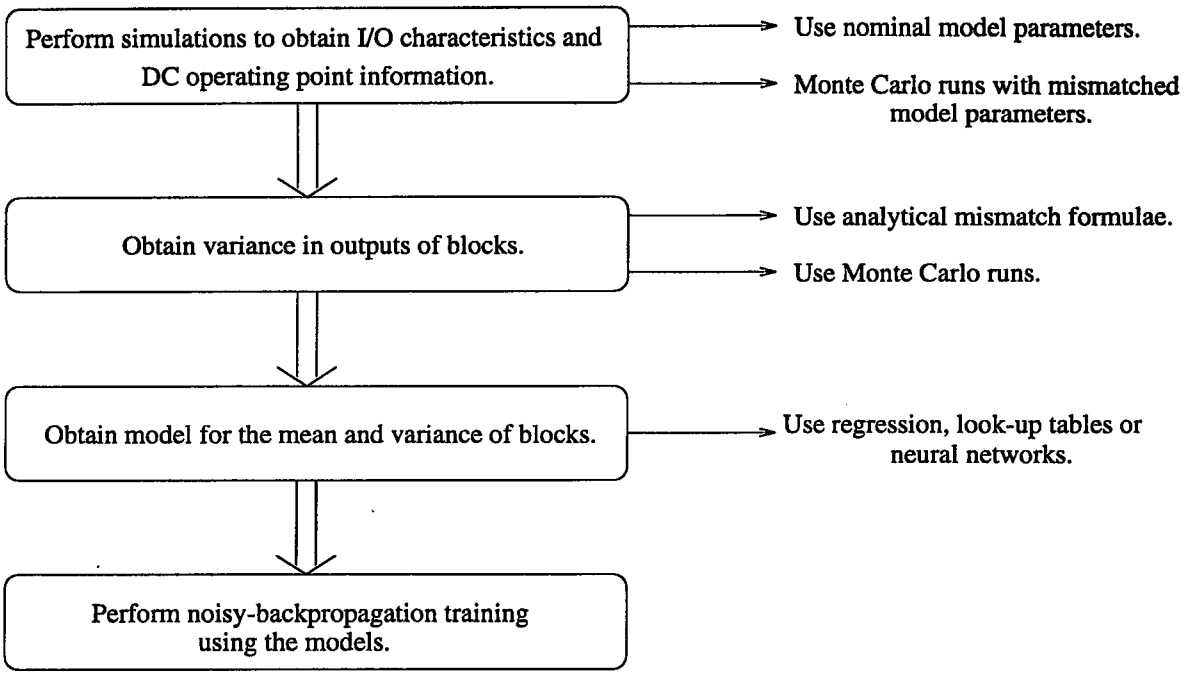

FIGURE 5.10. Flow for the noisy backpropagation.

and measurements separately.

$$\sigma_{\mu}^2(x,w) = c_0 + (c_1x + c_2x^2 + c_3x^3 + c_4x^4 + c_9x^6) (c_5w + c_6w^2 + c_7w^3 + c_8w^4) \quad (5.28)$$

$$\sigma_{\phi}^2(x) = d_0 + d_1x + d_2x^2 + d_3x^3 + d_4x^4 \quad (5.29)$$

The variance in the opamp will be dealt by the assumption that the resistor implemented will have a certain variation ΔR (approximately 5-10 per cent for standard processes) and that the output will be affected by the input off-set voltage ΔV_{offset} of the opamp only.

$$O(\mu) = -(R + \Delta R)\mu + \Delta V_{\text{offset}} \quad (5.30)$$

The resulting expression for the variance at opamp output becomes,

$$\sigma_O^2(\mu) = \mu^2\sigma_{\Delta R}^2 + \sigma_{\Delta V_{\text{offset}}}^2 . \quad (5.31)$$

Regarding the quality of the approximations, several characteristics of the regression results are given in TABLE 5.4.

TABLE 5.4. Characteristics of regression results.

		Standard error of estimate	Max. deviation for any observation	Proportion of variance explained
synapse	nominal	3.84	13.0	0.9952
	Monte Carlo	3.15	9.23	0.9979
synapse variance	theoretical	8.69	32.64	0.9924
	Monte Carlo	12.25	29.69	0.9737
sigmoid	nominal	0.039	0.076	0.9985
	Monte Carlo	0.028	0.055	0.9989
	measured	0.015	0.053	0.9996
sigmoid variance	Monte Carlo	0.89e-3	1.8e-3	0.9962
	measured	0.30e-3	1.1e-3	0.9997

Once the "noisy" behaviour of synapse and sigmoid circuits has been modeled based on analytical calculations (or Monte Carlo simulations), the backpropagation algorithm can be modified in order to incorporate those variations at the outputs. For this purpose, consider again the general multilayer perceptron structure depicted in FIGURE 1.2. The equations for backpropagation training as given in (4.1)-(4.4) are modified to become

$$net_o = \left(\sum_{i=1}^{h} \mu(T_i, H_i(x)) + \mu_n(T_i, H_i(x)) + \mu(T_0, 1) + \mu_n(T_0, 1) \right) \tag{5.32}$$

$$y(x) = \phi\left(O(net_o) + O_n(net_o)\right) + \phi_n\left(O(net_o) + O_n(net_o)\right) . \tag{5.33}$$

Note that the statistical variations at outputs of blocks are represented by additive noise terms $\mu_n(x, w)$, $O_n(\mu)$, and $\phi_n(x)$ for the synapse, opamp and sigmoid respectively. Output of each hidden unit is another similar expression, net_i being the net output of the synapses and O_h being the output of opamps at the hidden layer:

$$net_i = (\mu(W_i, x) + \mu_n(W_i, x) + \mu(W_{i0}, 1) + \mu_n(W_{i0}, 1)) \tag{5.34}$$

$$H_i(x) = \phi\left(O_h(net_i) + O_{hn}(net_i)\right) + \phi_n\left(O_h(net_i) + O_{hn}(net_i)\right) . \tag{5.35}$$

In the preceeding implementation of backpropagation, the noise terms are determined as follows: In each epoch of the training (forward pass) a random number is generated from a distribution of zero mean and variance as calculated by (5.28), (5.29), and

(5.31) for each synapse, opamp and sigmoid block. Those values are considered to be the statistical variations at the outputs. Since the noise terms are additive, the update equations of CHAPTER 4 are still valid.

5.3. Numerical Experiments

The efficiency of our noisy backpropagation approach employing transistor based mismatches as the source of noise, has been tested on several examples. Five different types of learning has been applied to each problem, where the core was a general purpose home-written software implementing gradient descent based backpropagation with several options like adaptive gain term, momentum term, weight decay and weight quantization. In all of the training experiments, on-line weight update scheme is employed. The following are the training types employing different types of models for the neural network blocks:

- *Nominal model*, based on simulations with nominal model parameters (no noise terms added),
- *Monte Carlo without noise*, based on average of simulations with mismatched model parameters (no noise terms added),
- *Monte Carlo with noise*, based on average of simulations with mismatched model parameters and noise terms added with variance obtained from Monte Carlo simulations,
- *Measurements and noise*, based on average of simulations with mismatched model parameters (for the synapse and opamp) and measurements (for the sigmoid), and noise terms added with variance obtained from theoretical formulae (for the synapse and opamp) and measurements (for the sigmoid),
- *Weight noise*, based on uniformly distributed random noise in percentage added to weights, and gradually reduced to zero as the network converges [114, 118].

The XOR problem is a simple and standard problem. We have performed training on the XOR problem and 3-bit parity check which is similar to the XOR problem. The training has been carried out 30 times for both problems, for each type of training mentioned above. The noisy forward passes using models from *"Monte Carlo with noise"* and *"measurements and noise"* have also been run for 20 times (i.e. 600 different training-forward pass pairs have been simulated). This allows us to derive statistically significant results for the problems. For the *"weight noise"* training, three noise levels have been used: 10 per cent, 20 per cent, and 40 per cent. Then, 10 forward runs are done for each noise level. Regarding the network sizes, a 2:3:1 network is used for the XOR problem, whereas a 3:6:1 network is employed for the 3-bit parity problem. Input levels are -2V for logic low, and +2V for logic high. However, output levels are -0.3V for logic low, and +2.6V for logic high as the desired outputs. Those values correspond to the minimum and maximum values of the sigmoid output. The averages of success rates for 600 (30 for nominal and Monte Carlo without noise) forward runs are given in TABLE 5.5. The numbers in parantheses are the results for the 3-bit parity problem.

The success rates are determined as follows: The allowable output range of $[-0.3, 2.6]$ has been divided into four equal subintervals, $[-0.3, 0.425]$, $[0.425, 1.15]$, $[1.15, 1.875]$, and $[1.875, 2.6]$. The set of outputs has been classified as being "settled" if the outputs are within the intervals $[-0.3, 0.425]$ or $[1.875, 2.6]$ representing logical "LOW" and "HIGH" respectively, for the four possible input combinations of the XOR problem. If the output for any input pair belongs to the other two intervals, the output set is said to be "rejected". An output set is classified as "successful" if the outputs for the four input pairs are "settled" and correct. On the other hand, if output for one or more input pairs is not "settled" or correct, the set is classified as "unsuccessful".

TABLE 5.5. Success rates in per cent for XOR (3-bit parity) problem for different training/forward pass types.

	TRAINING using models obtained from...				
Model used in forward pass	Nominal simulation	Monte Carlo		Measurements and noise	Weight noise
		w/out noise	with noise		
Nominal	100(100)	100(100)	100(100)	100(100)	100(100)
Monte Carlo	100(100)	100(100)	100(93)	100(100)	100(100)
Monte Carlo and noise	84(73)	60(59)	100(99)	100(100)	78(74)
Measurements and noise	65(58)	72(66)	94(90)	100(100)	82.5(85)

In order to test the effectiveness of the training, sample measurements have been carried out on the prototype chips. There are eight chips with no defective neurons. Since each chip contains four neurons, two chips are required to implement the XOR network employing two variable inputs, one bias input, three hidden neurons, one output neuron, and 13 weights. The topology of the test setup is given in FIGURE 5.11. For each one of the training methods, five different weight sets are used to test the XOR operation. For each weight set, 30 combinations of chips have been used in the test, that is, the chip for the hidden neurons and the chip for output neuron have been selected from the eight chips in 30 different ways so that the effects of statistical variations among different chips can be observed. In this way, for each training method, a total of 150 XOR networks have been costructed and tested. The weights have been applied as voltages directly using potentiometers as voltage dividers. The weights and inputs which are not used, have been connected to ground to imply zero input and zero

weight.

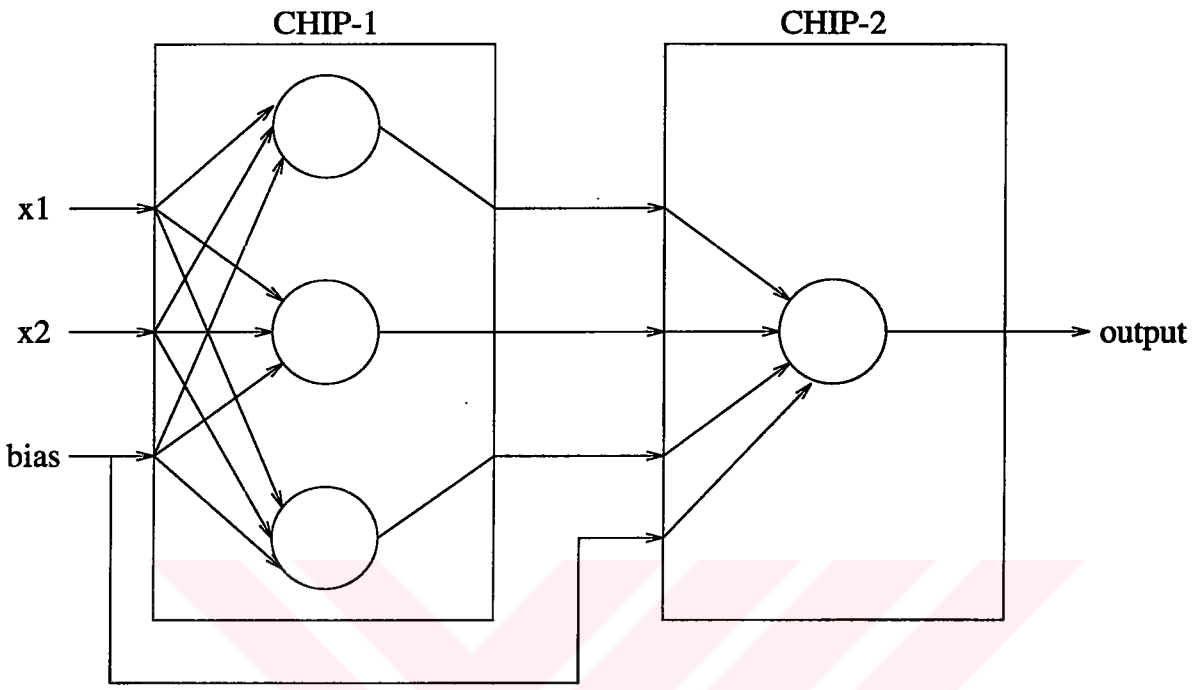

FIGURE 5.11. Test configuration for the XOR problem.

The results of the measurements on the chips are given in TABLE 5.6. Again, the allowable output range of $[-0.3, 2.6]$ has been divided into four equal subintervals, $[-0.3, 0.425]$, $[0.425, 1.15]$, $[1.15, 1.875]$, and $[1.875, 2.6]$. An output set is classified as "successful" if the outputs for the four input pairs are "settled" and correct. On the other hand, if output for one or more input pairs is not "settled", the set is classified as "reject". "Failure" denotes the case where the outputs are "settled", however, the output is not correct for one or more input pairs.

5.3.2. 2-Dimensional Classification Problem

As a "continuous input" classification problem, two sets of data points are generated from a normal distribution with the following properties: The one labeled by *class-1* has mean of -0.5 for x1 coordinate, mean of zero for x2 coordinate, where

TABLE 5.6. Success rates of measurements on the chips in percentage for XOR problem.

Training Method	Success	Failure	Reject
Nominal	49	29	22
Monte Carlo without noise	61	26	13
Monte Carlo with noise	85	6	9
Measurements and noise	85	10	5
Weight noise	81	14	5

the standard deviation for both coordinates is 0.3. *class-2*, on the other hand, has mean of 2.0 for x1 coordinate, mean of zero for x2 coordinate, where the standard deviation for both coordinates is 1.0. Of the 100 data points generated for each class, half of them are used as the training set, where the other half is used for test. FIGURE 5.12 displays the data points generated. A 2:14:2 structure is used for training where *class-1* and *class-2* outputs are designated by the pairs of (2.6V, -0.3V) and (-0.3V, 2.6V) as target values. All types of training are carried out until the rms training error dropped below one per cent. The training has been carried out 15 times, for each type of training mentioned above. The noisy forward passes using models from *"Monte Carlo with noise"* and *"measurements and noise"* have also been run for 10 times.

TABLE 5.7. Success rates in per cent for classification problem for different training/forward pass types.

	TRAINING using models obtained from...				
Model used in forward pass	Nominal simulation	Monte Carlo		Measurements and noise	Weight noise
		w/out noise	with noise		
Nominal	94(88)	96(86)	92(88)	90(86)	92(84)
Monte Carlo	46(44)	94(88)	96(92)	94(88)	96(88)
Monte Carlo and noise	32(38)	76(72)	88(92)	90(86)	82(79)
Measurements and noise	29(26)	68(64)	90(88)	95(92)	78(74)
The entries are rates for the training set and the test set in parantheses.					

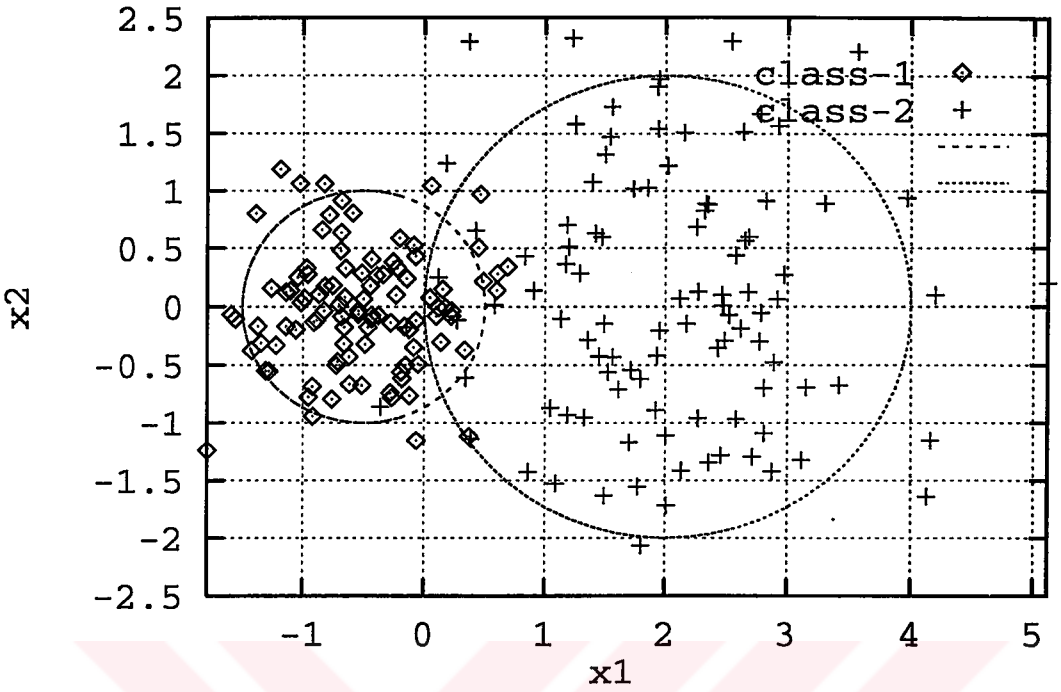

FIGURE 5.12. Data for the classification problem.

5.3.3. Comparison of Results

The purpose of our comparison is not to test the final performance of the training methodologies but rather the improvement brought about by the new approach. As can be seen from the results, incorporation of variations into the training enhances the capability of the network strongly: The comparison is performed with respect to the forward pass using the models obtained from measurements which resembles the actual electrical characteristics of the analog circuitry. For the XOR (3-bit parity) problem training without noise results in high level of error. The degradation in the classification problem is more severe: correct classification rates drop to 29 per cent and 68 per cent for *nominal* and *Monte Carlo* models. Inclusion of weight noise improves the fault tolerance of the network as expected. However, the performance of training with *weight noise* is worse in comparison to training with *Monte Carlo with noise* and

Measurements and noise. This implies that random injection of noise is not capable of compensating the effects of hardware variations fully.

Regarding the measurements on actual chips the followings can be stated:

- The measurement results do not coincide exactly with the simulation results of TABLE 5.5, however they are correlated, that is, injection of noise during the training really improves the performance of the analog neural network.
- Modeling the variations based on either Monte Carlo simulations or actual measurements on the chip do not differ in the robustness of the training, however, they perform better than random noise injection.
- Even though the success rate has been found to be 100 per cent in the simulation using measurement based variations, the actual success rate has been 85 per cent only. This may be due to several factors:
  - The precision of weights used during the measurements is not as high as the precision of computed weights. Moreover there has been electrical noise on the setup which may have effected the outputs slightly.
  - The modeling does not base on measurements solely: Variances in the synapse have been modeled using the theoretical formulae, hence the modeling may be not accurate enough to incorporate the deviations fully.
- The measurements indicate that a much more robust training has to be carried out in order to guarantee satisfactory operation in the presence of hardware nonidealities and variations. This can be achieved through either improvement of the analog circuitry so that they exhibit less variation, or through utilization of a simulation based training as offered in CHAPTER 4.

An investigation of the weight distributions also suggests some hints for the enhancement of the fault tolerance: The weights for the training types with noise exhibit a larger standard deviation in comparison to weights obtained without noise

terms (TABLE 5.8 for the classification problem). This confirms the results of [118] that the "information" is distributed evenly to the weights in training with noise injection.

TABLE 5.8. Distribution of weights for different training types in the classification problem.

	TRAINING using models obtained from...				
	Nominal simulation	Monte Carlo		Measurements and noise	Weight noise
		without noise	with noise		
Mean	-0.035	-0.033	-0.015	-0.05	0.001
Standard Deviation	0.14	0.24	0.73	0.54	0.64

6. CONCLUSION

In this thesis, a new approach for design and training of analog neural networks has been proposed. The approach aims to make possible the integration of analog neural networks into larger systems-on-chip by proposing solutions to problems of hardware training on software which would eliminate the need for chip-in-the-loop training and on-chip training. The main problems of analog hardware in terms of training is that the circuitry exhibits nonideal characteristics and mismatches between identically designed components. The nonidealities of the hardware are incorporated into the training by modeling the characteristics of building blocks via regression, and by using the models during the backpropagation training. In order to incorporate the mismatches into training, building blocks of an analog neural network, namely the synapse, opamp and sigmoid circuitry, are analyzed for their mismatch characterization analytically. Mismatches in the threshold voltages (V_T) and current factors (β) are considered to be the causes of variations on matched MOS transistor pairs which result in deviations at outputs of identically designed blocks. Closed form expressions of statistical variations from the nominal output are derived for the abovementioned circuits. These theoretical variations are compared to actual measurements obtained from chips. It seems to be evident from the comparison that those variations can be attributed to mismatches. In order to incorporate the variations at the outputs, the training algorithm (backpropagation) is modified: The building blocks are modeled according to their average outputs whereas the variations are considered to be noise with certain normal distribution. Then, the training is carried out using "noisy" backpropagation where the outputs of blocks are calculated in a probabilistic manner taking the noise into account. Sample simulations are conducted to verify that this method of training allows a higher degree of fault tolerance in the sense that noisy forward pass outputs exhibit better performance over outputs after a training without including the variations. The comparison of the modified backpropagation algorithm for different types of modeling also indicates that incorporating variations based on mismatch model obtained through Monte Carlo simulations and actual measurements also coincide. This verifies that noise (variations) can be estimated during the design stage of the circuitry

using the data on transistor geometries so that the usage of "noisy" backpropagation can be helpful for achieving a robust training for analog neural networks. The ultimate aim of this research is to enable training of analog neural networks on software eliminating the need for chip-in-the-loop or on-chip training. The author believes that the incorporation of variations based on mismatches will be an important step towards that goal.

Moreover, estimation of mismatch either through analytical derivations or through Monte Carlo simulations can be utilized as a means for circuit optimization. Sensitivity of the overall performance of the analog neural network with respect to circuit elements (sub-blocks such as current pairs or differential pairs) can be estimated during the design. Improvements can be done in the sizing of transistors for critical pairs so as to obtain reasonable levels of variations which could be tolerated during the training.

During the course of the thesis the following original contributions are done:

- Design, characterization, and modeling of analog neural network building blocks suitable for hardware implementation.
- Development of a silicon assembler for generating the layout of analog neural networks.
- Derivation of mismatch based statistical variations on outputs of analog building blocks analytically, and the verification on the prototype neural network chip.
- Incorporation of MOS transistor mismatches into training of analog neural networks for enhancing fault tolerance.

It is the opinion of the author that the future of analog neural network realizations has to consider the effects of statistical variations during the design phase (through analytical modeling and Monte Carlo simulations,) so that a robust training will be enabled for the analog hardware.

APPENDIX

CHARACTERISTICS OF THE ALCATEL-MIETEC 2.4μm PROCESS

Minimum feature size : 2.4μm

Number of metal and poly layers : 2

Type of substrate and well : n-well on p-type substrate

Resistance of n-well diffusion : 2$k\Omega$/square

Model parameters for NMOS transistor:

.model N nmos level=3 vto=0.86 nsub=1.39e15 tox=40.29n gamma=0.26 ld=0.22u

+ kp=51.71u phi=0.62 uo=611.37 delta=0.85 cj=6.9e-5 mj=0.5 pb=0.65

+ mjsw=0.27 rsh=33.42 vmax=158e3 cjsw=3.43e-10 cgbo=5.57e-10

+ nfs=1.35e11 ucrit=1e4 eta=0.07 kappa=1.4 xj=0.3u js=0.001 theta=0.05

+ capop=4 xw=0 xl=0 lmlt=1 wmlt=1 acm=2hdif=3u wd=-0.028u del=-0.16u

Model parameters for PMOS transistor

.model P pmos level=3 vto=-0.85 nsub=9.1e15 tox=42.46n gamma=0.69 ld=0.35u

+ kp=19.14u phi=0.67 uo=233.84 delta=0.96 cj=3.1e-4 mj=0.5 pb=0.76

+ mjsw=0.38 rsh=35.01 vmax=223.67e3 cjsw=3.67e-10 cgbo=5.57e-10

+ nfs=3.92e11 ucrit=1e4 eta=0.06 kappa=9.23 xj=0.5u js=0.001 theta=0.12

+ capop=4 xw=0 xl=0 lmlt=1 wmlt=1 acm=2 hdif=3u wd=0.034u del=-0.01u

REFERENCES

[1] Sheu, B. J. and J. Choi, *Neural Information Processing and VLSI*, Kluwer Academic Publishers, Norwell, 1995.

[2] Zornetzer, S. F., J. L. Davis, C. Lau, and T. McKenna (editors), *An Introduction to Neural and Electronic Networks*, Academic Press, San Diego, 1995.

[3] Lippmann, R. P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4–22, April 1987.

[4] Fausett, L., *Fundamentals of Neural Networks*, Prentice Hall, Englewood Cliffs, 1994.

[5] Hassoun, M. H., *Fundamentals of Artificial Neural Networks*, The MIT Press, Cambridge, 1995.

[6] Leong, P. H. W. and M. A. Jabri, "A Low-Power VLSI Arrhythmia Classifier," *IEEE Transactions on Neural Networks*, Vol. 6, pp. 1435–1445, December 1995.

[7] Alpaydın, E. and F. Gürgen, "Comparison of Statistical and Neural Classifiers and Their Applications to Optical Character Recognition and Speech Classification," in C. T. Leondes (Editor), *Neural Networks Systems Techniques and Applications*, pp. 61–88, Academic Press, 1998.

[8] Special Issue on Everyday Applications of Neural Networks, *IEEE Transactions on Neural Networks*, Vol. 8, no:4, pp. 825–975, July 1997.

[9] Manevendra, M., "Parallel Environments for Implementing Neural Networks," *Neural Computing Surveys*, Vol. 1, pp. 48–60, 1997.

[10] Lindsey, C. S., B. Denby, H. Haggerty, and K. Johns, "Real Time Track Finding in a Drift Chamber with a VLSI Neural Network," *Nuclear Instruments and Methods in Physics Research-A*, no. 317, pp. 346–356, 1992.

[11] Mundie, D. B. and L. W. Massengill, "Weight Decay and Resolution Effects in Feedforward Artificial Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 2, pp. 168–170, January 1991.

[12] Schönauer, T., A. Jahnke, U. Roth, and H. Klar, "Digital Neurohardware: Principles and Perspectives," in *Proceedings of MicroNeuro'98*, 1998.

[13] Lindsey, C. S. and T. Lindblad, "Review of Hardware Neural Networks: A User's Perspective," in *Proceedings of Third Workshop on Neural Networks: From Biology to High Energy Physics*, September 1994.

[14] Fisher, A., R. J. Fujimoto, and R. C. Smithson, "A Programmable Analog Neural Network Processor," *IEEE Transactions on Neural Networks*, Vol. 2, pp. 222–229, March 1991.

[15] Mauduit, N., M. Duranton, J. Gobert, and J.-A. Sirat, "Lneuro 1.0: A Piece of Hardware LEGO for Building Neural Network Systems," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 414–422, May 1992.

[16] Watanabe, T., K. Kimura, M. Auki, T. Sakata, and K. Ito, "A Single 1.5-V Digital Chip for a 10^6 Synapse Neural Network," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 387–493, May 1993.

[17] Denby, B., P. Gole, A. Sartori, and G. Tecchiolli, "Real Time Data Acquisition Techniques in Meteorological Applications," *IEEE Transactions on Nuclear Science*, Vol. 45, pp. 1840–1844, August 1998.

[18] Harrer, H., J. A. Nossek, and R. Stelzl, "An Analog Implementation of Discrete-Time Cellular Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 466–476, May 1992.

[19] Cardarilli, G. C. and F. Sargeni, "Very Efficient VLSI Implementation of CNN with Discrete Templates," *Electronics Letters*, Vol. 29, pp. 1286–1287, July 1993.

[20] Special Issue on Bio-Inspired Processors and Cellular Neural Networks for Vision, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 46, no:2, pp. 226–334, February 1999.

[21] Çilingiroğlu, U., "A Charge-Based Neural Hamming Classifier," *IEEE Journal of Solid State Circuits*, Vol. 28, pp. 59–67, January 1993.

[22] Robinson, M. E., H. Yoneda, and E. Sanchez-Sinencio, "A Modular CMOS Design of a Hamming Network," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 444–456, May 1992.

[23] He, Y. and U. Çilingiroğlu, "A Charge-Based On-Chip Adaptation Kohonen Neural Network," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 462–469, May 1993.

[24] Macq, D., M. Verleysen, P. Jespers, and J.-D. Legat, "Analog Implementation of a Kohonen Map with On-Chip Learning," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 456–461, May 1993.

[25] Melton, M. S., T. Phan, D. S. Reeves, and D. V. den Bout, "The TINMANN VLSI Chip," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 375–384, May 1992.

[26] Fang, W.-J., B. J. Sheu, O. T.-C. Chen, and J. Choi, "A VLSI Neural Processor for Image Data Compression Using Self-Organization Networks," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 506–518, May 1992.

[27] Linares-Barranco, B., E. Sanchez-Sinencio, A. Rodriguez-Vazquez, and J. L. Huertas, "A CMOS Analog Adaptive BAM with On-Chip Learning and Weight Refreshing," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 445–455, May 1993.

[28] Starzyk, J. A. and X. Fang, "CMOS Current Mode Winner-Take-All Circuit with Both Excitatory and Inhibitory Feedback," *Electronics Letters*, Vol. 29, pp. 908–910, May 1993.

[29] Serrano-Gotarredona, T. and B. Linares-Barranco, "An ART1 Microchip and its Uses in Multi-ART1 Systems," *IEEE Transactions on Neural Networks*, Vol. 8, pp. 1184–1194, September 1997.

[30] Clarkson, T. G., C. K. Ng, and Y. Guan, "The pRAM: An Adaptive VLSI Chip," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 408–412, May 1993.

[31] Massengill, L. W. and D. B. Mundie, "An Analog Neural Hardware Implementation Using Charge-Injection Multipliers and Neuron-Specific Gain Control," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 354–362, May 1992.

[32] Satyanarayana, S., Y. P. Tsividis, and H. P. Graf, "A Reconfigurable VLSI Neural Network," *IEEE Journal of Solid-State Circuits*, Vol. 27, pp. 67–81, January 1992.

[33] Yuh, J.-D. and R. W. Newcomb, "A Multilevel Neural Network for A/D Conversion," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 470–483, May 1993.

[34] Montalvo, A. J., R. S. Gyurcsik, and J. J. Paulos, "An Analog VLSI Neural Network with On-Chip Perturbation," *IEEE Journal of Solid-State Circuits*, Vol. 32, pp. 535–543, April 1997.

[35] Montalvo, A. J., R. S. Gyurcsik, and J. J. Paulos, "Toward a General-Purpose Analog VLSI Neural Network with On-Chip Perturbation," *IEEE Transactions on Neural Networks*, Vol. 8, pp. 413–423, March 1997.

[36] Chou, E. Y., B. J. Sheu, and M. Yibing, "A Compact Neural Network for VLSI PRML Detectors: Scalable Architecture," *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, Vol. 45, pp. 709–719, April 1998.

[37] Murray, A. F., "Analogue Noise-Enhanced Learning in Neural Network Circuits," *Electronics Letters*, Vol. 27, pp. 1546–1548, August 1991.

[38] Hamilton, A., A. F. Murray, D. Baxter, S. Churcher, H. Reekie, and L. Tarassenko, "Integrated Pulse-Stream Neural Networks: Results, Issues, and Pointers," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 385–393, May 1992.

[39] Moon, G., M. E. Zaghloul, and R. W. Newcomb, "VLSI Implementation of Synaptic Weighting and Summing in Pulse Coded Neural-Type Cells," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 394–403, May 1992.

[40] Bor, J.-C. and C.-Y. Wu, "Realization of the CMOS Pulsewidth-Modulation (PWM) Neural Network with On-Chip Learning," *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, Vol. 45, pp. 96–107, January 1998.

[41] Lehmann, T., "Classical Conditioning with Pulsed Integrated Neural Networks: Circuits and Systems," *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, Vol. 45, pp. 720–728, June 1998.

[42] Special Issue on Pulse Coupled Neural Networks, *IEEE Transactions on Neural Networks*, Vol. 10, no:3, pp. 461–625, May 1999.

[43] Cauwenberghs, G., C. F. Neugebauer, and A. Yariv, "Analysis and Verification of an Analog VLSI Incremental Outer-Product Learning System," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 488–497, May 1992.

[44] Krishnamoorthy, A. V., G. Yayla, and S. C. Esener, "A Scalable Optoelectronic Neural System Using Free-Space Optical Interconnects," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 404–413, May 1992.

[45] Kane, J. S., "A Smart Pixel-Based Feedforward Neural Network," *IEEE Transactions on Neural Networks*, Vol. 9, pp. 159–164, January 1998.

[46] Säckinger, E., B. E. Boser, J. Bromley, Y. LeCun, and L. D. Jackel, "Application of the ANNA Neural Network Chip to High-Speed Character Recognition," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 498–505, May 1992.

[47] Vai, M. M., S. Wu, B. Li, and S. Prasad, "Reverse Modeling of Microwave Circuits with Bidirectional Neural Network Models," *IEEE Transactions Microwave Theory and Techniques*, Vol. 46, pp. 1492–1494, October 1998.

[48] Intel Corporation, 80170NX ETANN Data Sheets, 1991.

[49] Lindsey, C. S., B. Denby, and H. Haggerty, "Drift Chamber Tracking with Neural Networks," *IEEE Transactions on Nuclear Science*, Vol. 40, pp. 607–614, August 1993.

[50] Danielsson, M., A. Go, K. Jon-And, T. Lindblad, E. Machado, and M. Timm, "A Hardware Neural Network for On-Line Particle Identification," *Nuclear Instruments and Methods in Physics Research-A*, no. 350, pp. 322–326, 1994.

[51] Lindsey. C. S., T. Lindblad, J. R. Vollaro, G. Szekely, and J. Molnar, "Performance of a Cascadable Neural Network VME-Module with Intel 80170NX Chips," *Nuclear Instruments and Methods in Physics Research-A*, no. 351, pp. 466–471, 1994.

[52] Widrow, B. and M. A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proceedings of the IEEE*, Vol. 78, pp. 1415–1442, September 1990.

[53] Mumford, M. L., D. K. Andes, and L. Kern, "The Mod 2 Neurocomputer System Design," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 423–433, May 1992.

[54] Wawrzynek, J., K. Asanovic, and N. Morgan, "The Design of a Neuro-Microprocessor," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 394–399, May 1993.

[55] Lehmann, C., M. Viredaz, and F. Blayo, "A Generic Systolic Array Building Block for Neural Networks with On-Chip Learning," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 400–407, May 1993.

[56] Beiu, V., *Digital Integrated Circuit Implementations,* Prentice Hall, Englewood Cliffs, 1997.

[57] Annema, A.-J., *Feed-Forward Neural Networks, Vector Decomposition Analysis, Modelling and Analog Implementation*, Kluwer Academic Publishers, Boston, 1995.

[58] Binici, H. , G. Dündar, and S. Balkır, "A New Multiplier Architecture Based on Radix-2 Conversion Scheme," in *Proceedings of European Conference on Circuit Theory and Design*, 1995.

[59] Treleaven, P. , M. Pacheco, and M. Vellasco, "VLSI Architectures for Neural Networks," *IEEE Micro Magazine*, pp. 8–27, September 1989.

[60] Rossetto, O., C. Jutten, J. Herault, and I. Kreuzer, "Analog VLSI Synaptic Matrices as Building Blocks for Neural Networks," *IEEE Micro Magazine*, pp. 56–63, December 1989.

[61] Dündar, G. and K. Rose, "Analog Neural Network Circuits for ASIC Fabrication," in *Proceedings of the Fifth IEEE ASIC Conference*, 1992.

[62] Mead, C. and M. Ismail (editors), *Analog VLSI Implementation of Neural Systems*, Kluwer Academic Publishers, Norwell, 1989.

[63] Borgstrom, T. H., M. Ismail, and S. B. Bibyk, "Programmable Current-Mode Neural Network Implementation in Analogue MOS VLSI," *IEE Proceedings - Circuits Devices Systems*, Vol. 137, pp. 175–184, April 1990.

[64] Andreou, A. G., K. A. Boahen, P. O. Pouliquen, A. Pavasovic, R. E. Jenkins, and K. Strohbehn, "Current-Mode Subthreshold MOS Circuits for Analog VLSI Neural Systems," *IEEE Transactions on Neural Networks*, Vol. 2, pp. 205–213, March 1991.

[65] Cohen, M. H. and A. G. Andreou, "Current Mode Subthreshold MOS Implementation of the Herault-Jutten Autoadaptive Network," *IEEE Journal of Solid-State Circuits*, Vol. 27, pp. 714–727, May 1992.

[66] Kub, F. J., K. K. Moon, I. A. Mack, and F. M. Long, "Programmable Analog Vector-Matrix Multipliers," *IEEE Journal of Solid-State Circuits*, Vol. 25, pp. 207–214, February 1990.

[67] Lee, B. W., B. Sheu, and H. Yang, "Analog Floating Gate Synapses for General Purpose VLSI Neural Networks," *IEEE Transactions on Circuits and Systems*, Vol. 38, pp. 654–658, June 1991.

[68] Lont, J. B. and W. Guggenbühl, "Analog CMOS Implementation of a Multilayer Perceptron with Nonlinear Synapses," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 457–465, May 1992.

[69] Dündar, G., F.-C. Hsu, and K. Rose, "Effects of Nonlinear Synapses on the Performance of Multilayer Neural Networks," *Neural Computation*, Vol. 8, pp. 939–949, July 1996.

[70] Şimşek, A., M. Civelek, and G. Dündar, "Study of the Effects of Nonidealities in Multilayer Neural Networks with Circuit Level Simulation," in *Proceedings of Mediterranean Electronics Conference MELECON96*, 1996.

[71] Dündar, G. and K. Rose, "The Effects of Quantization on Multilayer Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 6, pp. 1446–1451, November 1995.

[72] Dolenko, B. K. and H. C. Card, "Tolerance to Analog Hardware of On-Chip Learning in Backpropagation Networks," *IEEE Transactions on Neural Networks*, Vol. 6, pp. 1045–1052, September 1995.

[73] Piche, S. W., "The Selection of Weight Accuracies for Madalines," *IEEE Transactions on Neural Networks*, Vol. 6, pp. 432–445, 1995.

[74] Hollis, P. W. and J. J. Paulos, "Artificial Neural Networks Using MOS Analog Multipliers," *IEEE Journal of Solid State Circuits*, Vol. 25, pp. 849–855, 1990.

[75] Edwards, P. J. and A. F. Murray, *Analogue Imprecision in MLP Training*, World Scientific Publishing, 1996.

[76] Gilbert, B., "A Precise Four-Quadrant Multiplier with Subnanosecond Responce," *IEEE Journal of Solid State Circuits*, Vol. 3, pp. 365–373, 1968.

[77] Qin, S.-C. and R. L. Geiger, "A +/-5-V CMOS Analog Multiplier," *IEEE Journal of Solid-State Circuits*, Vol. 22, pp. 1143–1146, December 1987.

[78] Ngolediage, J. E., S. S. Dlay, and R. N. Gorgui-Naguib, "CMOS Phase Detector and Four-Quadrant Multiplier for Implementation in Analogue Neural Networks," *Electronics Letters*, Vol. 28, pp. 1142–1143, June 1992.

[79] Bo, G. M., D. D. Caviglia, H. Chible, and M. Valle, "Design of an Analog CMOS Self-Learning MLP Chip," in *Proceedings of International Symposium on Circuits and Systems ISCAS98*, 1998.

[80] Coue, D. and G. Wilson, "A Four-Quadrant Subthreshold Mode Multiplier for Analog Neural Network Applications," *IEEE Transactions on Neural Networks*, Vol. 7, pp. 1212–1219, September 1996.

[81] Seng, Y. K. and S. S. Rofail, "Design and Analysis of a +/-1V CMOS Four-Quadrant Analogue Multiplier," *IEE Proceedings - Circuits Devices Systems*, Vol. 145, pp. 148–161, June 1998.

[82] Hasler, P., C. Diorio, and B. A. Minch, "A Four-Quadrant Floating-Gate Synapse," in *Proceedings of International Symposium on Circuits and Systems ISCAS98*, 1998.

[83] Hollis, P. W. and J. J. Paulos, "A Neural Network Training Algorithm Tailored for VLSI Implementation," *IEEE Transactions on Neural Networks*, Vol. 5, pp. 784–791, 1994.

[84] Salam, F. A. and Y. Wang, "A Real-Time Experiment Using a 50-Neuron CMOS Analog Silicon Chip with On-Chip Learning," *IEEE Transactions on Neural Networks*, Vol. 2, pp. 461–464, July 1991.

[85] Hasler, P. and L. Akers, "Circuit Implementation of Trainable Neural Networks Employing Both Supervised and Unsupervised Techniques," in *Proceedings of the Joint Conference on Neural Networks*, 1992.

[86] Lansner, J. A. and T. Lehmann, "An Analog CMOS Chip Set for Neural Networks with Arbitrary Topologies," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 441–444, May 1993.

[87] Lau, K.T., S. Lee, and V. Ong, "4-Quadrant Analog CMOS Multiplier Cell for VLSI Signal and Information Processing," *IEE Proceedings-Circuits Devices Systems*, Vol. 145, pp. 132–134, February 1998.

[88] Bult, K. and H. Wallinga, "A Class of Analog CMOS Circuits Based on the Square-Law Characteristics of an MOS Transistor in Saturation," *IEEE Journal of Solid-State Circuits*, Vol. 22, pp. 357–365, June 1987.

[89] Sakurai, S. and M. Ismail, "High Frequency Wide Range CMOS Analogue Multiplier," *Electronics Letters*, Vol. 28, pp. 2228–2229, November 1992.

[90] Massengill, L. W., "A Dynamic CMOS Multiplier for Analog VLSI Based on Exponential Pulse-Decay Modulation," *IEEE Journal of Solid-State Circuits*, Vol. 26, pp. 268–276, March 1991.

[91] Shima, T., T. Kimura, Y. Kamatani, T. Itakura, Y. Fujita, and T. Iida, "Neuro Chips with On-Chip Back-Propagation and/or Hebbian Learning," *IEEE Journal of Solid-State Circuits*, Vol. 27, pp. 1868–1876, December 1992.

[92] Coue, D. and G. Wilson, "CMOS Subthreshold-Mode I/V Converter for Analog Neural Network Applications," *Electronics Letters*, Vol. 32, pp. 990–991, May 1996.

[93] Gray, P. R. and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, John Wiley and Sons, New York, 1991.

[94] Ismail, M. and T. Fiez, *Analog VLSI Signal and Information Processing*, McGraw-Hill, New York, 1994.

[95] Maeda, Y. and Y. Kanata, "An Analog Neural Network Circuit with Simultaneous Perturbation Learning Rule," in *Proceedings of International Symposium on Circuits and Systems ISCAS98*, 1998.

[96] Bogason, G., "Generation of a Neuron Transfer Function and its Derivatives," *Electronics Letters*, Vol. 29, pp. 1867–1869, October 1993.

[97] Annema, A.-J., "Hardware Realization of a Neuron Transfer Function and its Derivatives," *Electronics Letters*, Vol. 30, pp. 576–577, March 1994.

[98] Al-Ruwaihi, K. M., "CMOS Analogue Neuron Circuit with Programmable Activation Functions Utilising MOS Transistors with Optimised Process/Device

Parameters," *IEE Proceedings-Circuits Devices Systems*, Vol. 144, pp. 318–322, December 1997.

[99] Amin, H., K. M. Curtis, and B. R. Hayes-Gill, "Piecewise Linear Approximation Applied to Nonlinear Function of a Neural Network," *IEE Proceedings-Circuits Devices Systems*, Vol. 144, pp. 313–317, December 1997.

[100] Ayres, R. F., *VLSI Silicon Compilation and the Art of Automatic Microchip Design*, Prentice Hall, Englewood Cliffs, 1983.

[101] Conway, L. and C. Mead, *Introduction to VLSI Systems*, Addison-Wesley, Massachusets, 1980.

[102] Erten, G., A. S. Öğrenci, and G. Dündar, "A Compaction Algorithm for SAFANN," in *Proceedings of the Sixth Turkish Artificial Intelligence and Neural Networks Symposium*, 1997.

[103] Masa, P., K. Hoen, and H. Wallinga, "A High Speed Analog Neural Processor," *IEEE Micro Magazine*, pp. 40–50, June 1994.

[104] Gowda, S. M., B. J. Sheu, J. Choi, C.-G. Hwang, and J. S. Cable, "Design and Characterization of Analog VLSI Neural Network Modules," *IEEE Journal of Solid-State Circuits*, Vol. 28, pp. 301–313, March 1993.

[105] Donald, J. and L. Akers, "An Adaptive Neural Processing Node," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 413–426, May 1993.

[106] Choi, J., S. H. Bang, and B. J. Sheu, "A Programmable Analog VLSI Neural Network Processor for Communication Receivers," *IEEE Transactions on Neural Networks*, Vol. 4, pp. 484–495, May 1993.

[107] Hamilton, A., S. Churcher, P. J. Edwards, G. B. Jackson, A. F. Murray, and H. M. Reekie, "Pulse Stream VLSI Circuits and Systems: The EPSILON Neural Network Chipset," *International Journal of Neural Systems*, Vol. 4, pp. 395–405, December 1993.

[108] Swenson, R. M., J. F. Barbieri, D. Andes, D. Witcher, and R. Licklider, MADALINE III Neural Networks: Analog Hardware Considerations, Internal Report, Naval Weapons Center, China Lake, 1990.

[109] Jabri, M. and B. Flower, "Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 154–157, January 1992.

[110] Yang, J. and W. Miller, "Neuron Modelling for in-the-Loop Training," *Electronics Letters*, Vol. 34, pp. 998–999, May 1998.

[111] Hertz, J., A. Krogh, B. Lautrup, and T. Lehmann, "Non-Linear Back-Propagation: Doing Back-Propagation without Derivatives of the Activation Function," *IEEE Transactions on Neural Networks*, Vol. 8, pp. 1321–1327, November 1997.

[112] Frye, R. C., E. A. Rietman, and C. Wong, "Back-Propagation Learning and Nonidealities in Analog Neural Network Hardware," *IEEE Transactions on Neural Networks*, Vol. 2, pp. 110–117, January 1991.

[113] Vidal-Verdu, F., R. Navas, and A. Rodrigeuez-Vazquez, "Mixed Signal CMOS High Precision Circuits for On-Chip Learning," in *Proceedings of International Symposium on Circuits and Systems ISCAS98*, 1998.

[114] Edwards, P.J. and A. F. Murray, "Fault Tolerance via Weight Noise in Analog VLSI Implementations of MLP's–A Case Study with EPSILON," *IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing*, Vol. 45, pp. 1255–1262, September 1998.

[115] Krogh, A., "Generalization in a Linear Perceptron in the Presence of Noise," *Journal of Physics A–Mathematical and General*, Vol. 25, no. 5, pp. 1135–1147, 1992.

[116] Krogh, A. and J. A. Hertz, "A Simple Weight Decay Can Improve Generalization," in J. E. Moody, S. J. Hanson, and R. P. Lippmann (Eds.), *Advances in*

Neural Information Processing, Vol. 4, pp. 950–957, Morgan Kauffmann Publishers, 1995.

[117] Bishop, C., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[118] Murray, A. F. and P. J. Edwards, "Enhanced MLP Performance and Fault Tolerance Resulting from Synaptic Weight Noise During Training," *IEEE Transactions on Neural Networks*, Vol. 5, pp. 792–802, September 1994.

[119] Bayraktaroğlu, İ., "Circuit Level Simulation Based Training Algorithms for Analog Neural Networks," MS. Thesis, Boğaziçi University, 1996.

[120] Bayraktaroğlu, İ., A. S. Öğrenci, G. Dündar, S. Balkır, and E. Alpaydın, "ANNSyS: An Analog Neural Network Synthesis System," *Neural Networks*, Vol. 12, pp. 325–338, February 1999.

[121] Lakshmikumar, K. R., R. A. Hadaway, and M. A. Copeland, "Characterization and Modeling of Mismatch in MOS Transistors for Precision Analog Design," *IEEE Journal of Solid-State Circuits*, Vol. 21, pp. 1057–1066, December 1986.

[122] Pelgrom, M. J. M., A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching Properties of MOS Transistors," *IEEE Journal of Solid-State Circuits*, Vol. 24, pp. 1433–1440, October 1989.

[123] Gregor, R. W., "On the Relationship Between Topography and Transistor Matching in an Analog CMOS Technology," *IEEE Transactions on Electron Devices*, Vol. 39, pp. 275–282, February 1992.

[124] Lan, M.-F. and R. Geiger, "Matching Performance of Current Mirrors with Arbitrary Parameter Gradients Through the Active Devices," in *Proceedings of International Symposium on Circuits and Systems ISCAS98*, 1998.

[125] Forti, F. and M. E. Wright, "Measurement of MOS Current Mismatch in the Weak Inversion Region," *IEEE Journal of Solid-State Circuits*, Vol. 29, pp. 138–142, February 1994.

[126] Steyaert, M., J. Bastos, R. Roovers, P. Kinget, W. Sansen, B. Graindourze, A. Pergoot, and E. Janssens, "Threshold Voltage Mismatch in Short-Channel MOS Transistors," *Electronics Letters*, Vol. 30, pp. 1546–1547, September 1994.

[127] Serrano-Gotarredona, T. and B. Linares-Barranco, "Cheap and Easy Systematic CMOS Transistor Mismatch Characterization," in *Proceedings of International Symposium on Circuits and Systems ISCAS98*, 1998.

[128] Michael, C. and M. Ismail, "Statistical Modeling of Device Mismatch for Analog MOS Integrated Circuits," *IEEE Journal of Solid-State Circuits*, Vol. 27, pp. 154–165, February 1992.

[129] Michael, C., C. Abel, and M. Ismail, "Statistical Modeling and Simulation," in M. Ismail and T. Fiez (Eds.), *Analog VLSI: Signal and Information Processing*, ch. 14, pp. 615–655, McGraw-Hill, 1994.

[130] Wang, Z., "Design Methodology of CMOS Algorithmic Current A/D Converters in View of Transistor Mismatch," *IEEE Transactions on Circuits and Systems*, Vol. 38, pp. 660–667, June 1991.

[131] Yufera, A. and A. Rueda, "Studying the Effects of Mismatching and Clock-Feedthrough in Switched-Current Filters Using Behavioral Simulation," *IEEE Transactions on Circuits and Systems–II:Analog and Digital Signal Processing*, Vol. 44, pp. 1058–1067, December 1997.

[132] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, Inc., 3 ed., 1991.

[133] Dolenko, B. K. and H. C. Card, "Neural Learning in Analogue Hardware: Effects of Component Variation from Fabrication and from Noise," *Electronics Letters*, Vol. 29, pp. 693–694, April 1993.

[134] Krogh, A., "Learning with Noise in a Linear Perceptron," *Journal of Physics A–Mathematical and General*, Vol. 25, no. 5, pp. 1119–1133, 1992.

[135] Matsuoka, K.; "Noise Injection into Inputs in Back-Propagation Learning," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, pp. 436–440, May/June 1992.

[136] Holmström, L. and P. Koistinen, "Using Additive Noise in Back-Propagation Training," *IEEE Transactions on Neural Networks*, Vol. 3, pp. 24–38, January 1992.