DESIGN OF BUILDING BLOCKS FOR TRIT ALGORITHM

By

BALAJI PARTHASARATHY

Bachelor of Engineering

College of Engineering

Guindy, India

1990

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1993

OKLAHOMA STATE UNIVERSITY

DESIGN OF BUILDING BLOCKS FOR TRIT ALGORITHM

Thesis Approved:

[illegible]
Thesis Advisor

[illegible]

[illegible]

Thomas C. Collins
Dean of the Graduate College

PREFACE

This thesis attempts to design the building blocks for TRIT algorithm. PSPICE was used for simulation. The building blocks were laidout in Magic.

I would like to express my sincere gratitude to Dr. Chriswell Hutchens for his support, guidance and encouragement. I appreciate the time and effort that he spent on this project. I would also thank the Naval Ocean Systems Center for the facilities and funding of this project. I am thankful to Dr. Johnson and Dr. Baker for serving in my committee. I thank my research colleagues and friends in Stillwater.

I am always indebted to my good friend Gunna, and would like to thank Viswanathan, Sunil Mathews and Raja.

This work is dedicated to my parents Ranganayaki and Parthasarathy, my grandmother Rukmani, uncles Dr.S.V.Kannan and Dr.S.V.Vijayaraghavan, Mr.A.V.S.Raja and their families, my brothers and their families, and my sister Padmini Vasudevan and her family.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

NOMENCLATURE

$(W/L)_{Mx}$	Width to length ratio of subscripted MOSFET x
β_x	Transconductance of MOSFET x
λ_x	Channel length modulation parameter of MOSFET x
A_{Vx}	Voltage gain of subscripted operational amplifier x
C_{GD}	Gate to drain capacitance
C_{GS}	Gate to source capacitance
C_{Gx}	Gate capacitance of subscripted MOSFET
C_{OX}'	Oxide capacitance/unit area
g_{ds}	Small signal drain to source conductance
g_{mx}	Small signal channel transconductance of MOSFET x
I_{Dx}	Drain current of MOSFET x
M_x	Subscripted MOSFET
V_{DD}	Positive supply voltage
V_{DS}	Drain to source voltage of MOSFET x
V_{GSx}	Gate to source voltage of MOSFET x
V_{SS}	Negative supply voltage
V_{Tx}	Threshold voltage of MOSFET x
W	Weight matrix
W^T	Transpose of the weight matrix

CHAPTER I

INTRODUCTION AND LITERATURE SURVEY

This thesis addresses the design and simulation of the building blocks for Trinary Backpropagation (TRIT) algorithm. TRIT is a modified form of the Backpropagation (BP) [1,2] algorithm for Artificial Neural Networks (ANN) and is also referred to as Trinary Backpropagation algorithm. Architecture for parallel on-board learning based on the TRIT algorithm is the aim of this work. TRIT quantizes the BP algorithm, by updating the weights of a Multi-Layer Perceptron (MLP) network, in parallel. The weight updates are not unique for each and every element of the weight matrix but can be only one of the three values : an increment, decrement of the same magnitude, or zero. This results in large saving in silicon, because of the reduced complexity of the weight updates. Also the same weight matrix is used both in the forward propagation and back propagation mode resulting in area reduction by two. Larger layer sizes can be implemented in lesser area because of this modification to BP algorithm. On-board learning is the goal of this research, which will widen the scope of the

ANN applications.

1.1 Introduction

The resurgence of interest in Artificial Neural Networks (ANN) that started in the late eighties has led to a host of new potential applications for these ANN models [3]. These Neural Network models offer great potential in the areas of speech processing, image recognition and pattern classification due to their high fault tolerance and parallel computation capability [4].

The complexity of these neural networks does not stem from the complexity of the individual components but from the multitude of ways in which a large collection of the components can interact. These network models reflect highly parallel, regular, and modular architectures that make them attractive for Very Large Scale Integrated (VLSI) systems [5]. The implementation of such models in hybrid VLSI analog/digital circuitry is one of the active current research areas [6].

The technologies used in special purpose ANN implementations are broadly classified as analog [7,8], digital or mixed analog/digital (hybrid) IC's [9], optical and electro-optical [10].

1.2 Comparison of analog and digital ANN's

Analog VLSI neural networks perform better than digital circuits in specific applications. Current studies [11] indicate that 10^9 to 10^{11} interconnections can be achieved with analog circuits, a rate much higher than digital circuits [11]. Analog VLSI ANN's make use of very simple building block which are reconfigurable and versatile. The simple building block approach simplifies the design time, making efficient use of the Computer Aided Design (CAD) tools. The design of simple well-defined analog cells that can be interconnected to achieve different linear and/or nonlinear functions is the key to the success of the analog ANN approach. This approach will bring neural nets VLSI design a step closer towards automation. Also, some of the traditional analog design requirements such as accurate absolute component values, device matching, precise time constants, etc are often of a lesser concern in Neural Nets applications because computational precision of individual neurons is not of paramount importance [12].

Schnieder and Card [13] have discussed the effect of the low-accuracy components on the design of ANN chips. They argue that ANN's with in-situ learning i.e. networks in which the synapses contain circuitry which performs local computation of weight updates, can adapt the weights to compensate for component to component variations present in analog networks. In fact, this thesis attempts to incorporate many of the transistor imperfections in the simulations for testing the validity of our algorithm. The results are discussed in Chap II of this thesis. Frye et. al. [14] have proved that the average error becomes less than 4% in an adaptive ANN, which uses hardware components with 30% variation.

Analog rather than digital VLSI has been identified as a major technology for future ANN applications. A large effort is being devoted to the ANN implementation in analog Metal Oxide Semiconductor (MOS) VLSI [5]. Efficient tools for the synthesis at both circuit and layout levels, simulation and testing of large scale analog IC's are being developed [12].

1.3 Comparison of Current and Voltage mode approaches

Many of the neural network functions involve current rather than voltage. The summing of many signals is readily achieved when those signals are currents. The dynamic range of the signals are greatly increased when MOS transistors are operated over the range from weak to strong inversion. This dynamic range is very critical for the scaled VLSI technologies which are expected to see a reduction in supply voltages. The frequency of operation is also potentially increased due to lower impedance internal nodes, and reduced full scale swing [12]. Reduced power consumption and increased speed of operation are the other inherent advantages of the analogue current-mode approach [15].

For the reasons outlined above, we make full use of current-mode processing in analog VLSI for the implementation of the TRIT algorithm.

1.4 Learning

ANN models include : Hopfield, Hamming, Single layer perceptrons, Multilayer perceptrons (MLP), Grossberg and Carpenter, Boltzmann machines, Kohonen Self-organizing maps, Bidirectional associative memories, and Neocognitrons. The discussion of all these types of networks is beyond the scope of this thesis but will include a review of BP and MLP's.

Backpropagation (BP) is a supervised learning scheme used in multi-layer perceptrons (feed-forward networks). The backpropagation networks are very attractive in applications such as pattern and speech recognition, waveform classification, etc.

It has been shown that the multi-layer perceptrons (MLP), can approximate any function of interest to any degree of accuracy. The multilayer perceptron network is shown in Figure 1. MLP are an important and popular subclass of ANN's. They have simple dynamics because of the

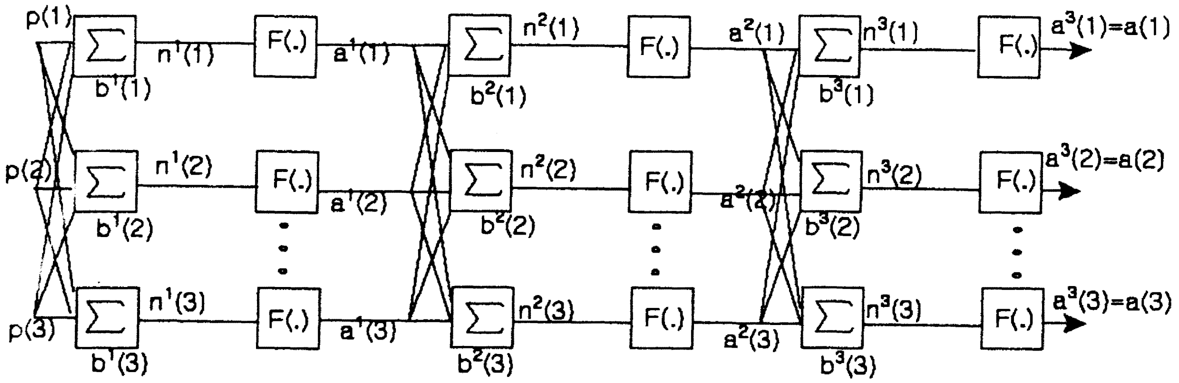


Figure 1. Multi Layer Perceptrons (MLP)

absence of feedback paths. The simple dynamics ensures the stability of the multi-layer perceptrons. Also the existence of powerful learning and adaptation algorithms for these networks make them very attractive from the engineering perspective.

The learning capability of ANN's is one of the most intriguing and challenging areas in theoretical neuroscience. Some researchers [16] have used fixed interconnection weights between processing units to implement learning in ANN's using the various algorithms discussed before. This however limits the application of the network. There has been several attempts [17-22] to address the problem of modifiable weight circuitry. Learning historically requires connectionist elements to have a considerable amount of circuitry, and, hence, a large amount of silicon area in addition to high inaccuracy [11]. On-chip learning procedures has been reported by several authors [17,18]. Furman et.al [18] used a dynamic memory cell and circuitry for weight modification and storage to implement the BP algorithm. They attempted analog storage in digital fashion by storing graded charges on a capacitor. The value of the charge represents the weight value which complicates the whole circuitry. Alspector [17] implemented an digital/analog weight stochastic learning network. In Alspector's work, the weights are subjected to fixed increment or decrement at each step of the learning process.

1.5 Review of Standard Backpropagation [27,28]

Standard BP can be represented as

$$\begin{aligned} O_i &= I_i && (\textit{Unit } i \textit{ an input unit}) \\ &= F(\sum_i W_{ij} O_j + b_i) && (\textit{otherwise}) \end{aligned}$$

(1)

where I_i is the external input to the unit i, W_{ij} is the weight associated with the interconnection from the j th processing unit in the network to the i th unit, b_i is the bias or the offset term, and F is the sigmoidal activation function for the hidden and output units. Propagation in the forward direction can be represented by the following equations

$$\begin{aligned} \overline{a}^0 &= p \\ \overline{a}^{k+1} &= \overline{F}^{k+1}(W^{k+1}\overline{a}^k + \overline{b}^{k+1}) \quad k=1,2,\ldots M-1 \\ \overline{a} &= \overline{a}^M \end{aligned}$$

(2)

while the backpropagation of the error can be represented as

$$\begin{aligned} \delta^M &= -F^{M'}(\overline{n}^M)(\overline{t} - \overline{a}) \\ \delta^M &= -F^{k'}(\overline{n}^k)\, W^{k+1^T} \delta^{k+1} \quad k=M-1, M-2, \ldots, 1 \end{aligned}$$

(3)

Weights and offsets are changed according to

$$\Delta W^k = -\alpha \bar{\delta}^k \bar{a}^{k^T} \qquad k = 1,2,\ldots M$$
$$\Delta \bar{b}^k = -\alpha \bar{\delta}^k \qquad k = 1,2,\ldots M \tag{4}$$

The BP algorithm tends to converge very slowly. Also the incremental changes in delta (error propagating) and output vectors near convergence are extremely small. The weight and bias changes are proportional to the error. As the error becomes too small, the weight change becomes too small as the circuit approaches a stable weight configuration. Based on the noise figure of the analog process, the incremental changes can be too small to be implemented practically in analog hardware. The lower bound on the convergence error is set by the limitation of the analog hardware. The signal to noise ratio (S/N) expected for our circuit is 60dB. So if the network fails to converge in this range of signal changes, the noise in the circuit takes over. The final circuit state then becomes dependent on the noise present in the circuit.

The potential difficulties associated with computing and imposing graded weight updates in parallel in analog hardware have led researchers to investigate better and easier methods of parallel learning procedures in which weight changes are coarsely quantized [6]. Small modifications of learning procedures can considerably enhance the computational power of neural networks and can make practical implementation of such networks easier [23]. Peterson & Hartman [24] examined the effect of update quantization into two states (increment or decrement) on the performance of a mean field theory learning algorithm. Alspector et. al. [17] implemented a hybrid digital/analog circuit in which weights are subjected to fixed increments or decrements per step of the learning process. M.Marchesi studied the effect of restricting the weights in multi-layer perceptrons to powers-of-two or sums of powers-of-two. A learning procedure based on backpropagation was used for a neural network with these discretized weights [25].

Shoemaker [6] proposed a modified Sgn-Sgn or trinary learning algorithm, which forces the same weight update increment for all elements in the network for efficient implementation of backpropagation in electronic perceptrons which will henceforth be referred as TRIT algorithm. Several electronic neural network solutions have been offered to date, but none with parallel onboard TRIT learning. The importance of on-board learning has been demonstrated by Frye, Wong et. al. [14] They argue that performance of the hardware when it learns by simulation is much poorer than that of obtained by learning on the network itself.

In our architecture, charge is stored on an electrically isolated floating gate of a MOS device [6], which would represent the weight value. However, precise control of increments of charge, and hence change in weights, is difficult in existing technology because the charge tunneling in the gate of a floating MOS device is difficult to control [6] due to extreme nonlinearities. If a unique and small weight change has to be accomplished on each element of the weight matrix, it will require a very large and complex circuitry even for small layers. The routing complexities related to high voltage problem and individual program control also necessitate a large area of silicon. Therefore it is advantageous from an implementation perspective to increment or decrement all the elements of the weight matrix in parallel across an entire network [6] or at the very least a complete row or column simultaneously.

Trinary Backpropagation (TRIT) is a simple variant of the classical BP algorithm which makes the practical implementation of on-board chip learning feasible. In this algorithm, the weight changes are assumed to be only one of the three values: an increment, a decrement of the same magnitude, or zero.

The TRIT algorithm allows a parallel implementation of learning rules with coarsely quantized parameter changes in analog integrated circuitry [6]. Conceptually, the implementation of the trinary algorithm is relatively easy and eliminates the need for complicated circuitry for weight updates [6]. However, the dynamic range of the weight updating will be limited practically by the lack of local control at the elemental weight cell.

Trinary Backpropagation uses the same forward and backward equations. The change of algorithm from the regular BP is in the weight and bias modification after the values of deltas are calculated as:

$$\begin{aligned}\Delta W_{ij} &= \eta \delta_i O_j && (|O_j| \geq \epsilon_1, |\delta_i| \geq \epsilon_2)\\ &= 0 && (|O_j| < \epsilon_1, |\delta_i| < \epsilon_2)\\ \Delta b_i &= \eta\, sgn(\delta_i) && (|\delta_i \geq \epsilon_2) \qquad (5)\\ &= 0 && (|\delta_j| < \epsilon_2)\end{aligned}$$

where η, ϵ_1 ,ϵ_2 are positive constants. For constant η, the learning process correspond to motion on a lattice in a weight/bias space, which is in a direction of decreasing sum- square error for each training pattern pair, although not generally in the direction of steepest descent. ϵ_1 ,ϵ_2 are current or voltage programmable constants.

The derivative $F(s_j)$ is implemented in a piecewise fashion as follows:

$$\begin{aligned}F(s_j) &= R_L && for \quad F(s_j) < V_f\\ &= R_H && for \quad F(s_j) > V_f \qquad (6)\end{aligned}$$

The programming circuitry will conceptually consist of a three position switch:

1) One position allows application of a programming current or voltage pulse to the weight circuit which would increment the stored charge by a discrete amount. This is equivalent to a fixed positive increment in the weight value.

2) A second position allows decrementing the stored charge by tunneling off charge. This is equivalent to a fixed discrete decrement of the weight value.

3) The third and final position leaves the circuit open and prevents any program modification of the stored charge. This is equivalent to zero change or no change in weight matrix.

1.6 Advantages

The primary advantage is the on-board chip learning implementation. It makes it practically feasible to train any network by building application specific hardware. It conserves area in case of VLSI implementation.

The network is inherently faster than the standard BP. So it is possible to build real time systems with this algorithm, which can be used in Natural Language Processing, Vision control etc.

This algorithm has been proved to be faster in convergence than the standard BP and even the BP with adaptive weight modification. Therefore it is our intention to realize faster convergence through hardware learning.

From the simulations, it can be stated that the component-to-component variation has a negligible effect on the convergence property on hardware implementation. Thus hardwired TRIT Neural Networks appear to be robust and error tolerant of the imperfections in poorly matched devices. Because of the enormity of the processing nodes involved, damage to a few nodes or links does not significantly impair their functionality.

Massively parallel Analog hardware networks, which are very fast and operate in parallel, can be developed based on these simulation which prove that the Algorithm is convergent for small benchmark problems.

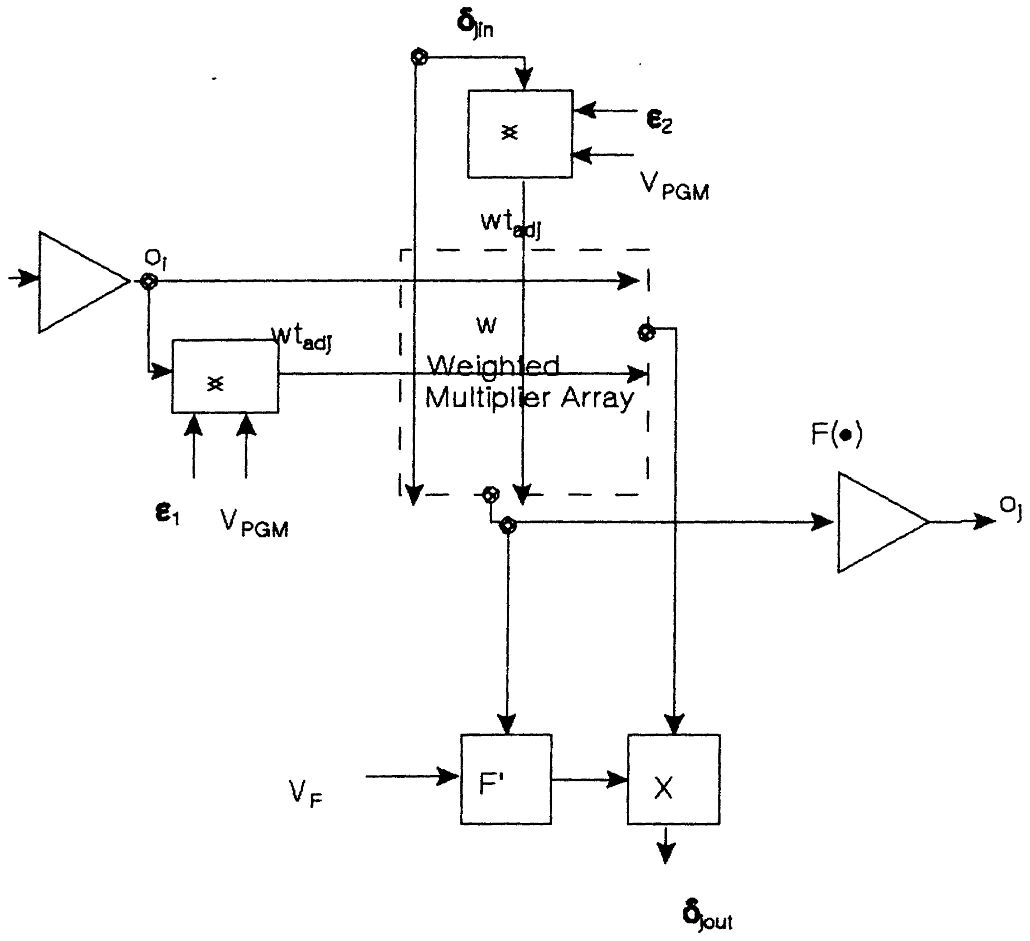
Such hardware, which can be used to test the validity of the practical implementation of ANN's and collective systems, can be designed for various specific applications.

The software learning accomplished on Von-Neumann computers does not exploit the inherent parallelism of the Neural Networks [14]. By using Hardware rather than Software learning, the Neural Networks inherent parallelism is fully utilized.

TRIT implementation can be easily understood from the single layer system diagram (Figure 2) which consists of an input function, and output circuit function (F(.)), a delta input function, a delta output function and the weight matrix array.

1.7 Literature Survey

Several CMOS analog implementations of ANN's have been reported in the literature. Some are inspired by biological



$$\Delta w_{ji} = \eta \, SGN(\delta_j) SGN(o_i)$$

$$\text{for } |\delta_j| > \varepsilon_1 \ \& \ |o_i| > \varepsilon_2 \text{ ELSE } 0$$

Figure 2. IC Signal Flow Floor Plan

models [23] while some are derived from artificial models. They are dedicated to signal processing, image processing or pattern recognition without or with in-situ learning. They include digital or continuous valued analog signals. These networks use different learning algorithms like Hopfield, Grossman, Backpropagation etc. Also some of these works involve the building of basic cells on a chip with their test results. Since it is very difficult to discuss all the building block approaches of all the types of learning algorithms, we restrict our discussion to VLSI implementation of backpropagation learning of ANN's which have been fabricated.

Boser et.al [4] implemented an optical digit recognizer on a neural network chip which is trained by backpropagation. It recognizes handwritten digits from a 20x20 pixel image with 2.9% miss-classifications compared to a typical value of 2.5% for human beings. The network consists of 133000 connections of 3500 neurons arranged in 5 layers. The throughput of the chip is 130MC/s and the operating frequency is 20MHz.

Nijhuis et.al [26] have fabricated a collision avoidance neural network in a 2 micron double metal CMOS technology. They had used fully digital network which was laid out using standard cell library. It has an operating frequency of 20MHz and 10M interconnects per second. The chip consists of 12 neurons and 144 synapses and 134 I/O pads.

1.8 Proposal for Hardware Implementation of TRIT

We propose the hardware implementation of potentially useful TRIT model in 2 micron Thin-Film Silicon-on-Sapphire (TSOS) process.

The most significant reasons for preferring TSOS over bulk process is the reduced V_T variation due to reduction of bulk threshold parameter (γ) and that Thin oxide film of 250^0A facilitates electron tunneling. The TSOS Process has both depletion and enhancement mode devices.

TSOS devices are made with epitaxial silicon islands on a sapphire substrate. There is no body (substrate) contact on the device. The threshold voltage V_T for a n-channel transistor is given by

$$V_T = V_{T0} + \gamma\,[\sqrt{2|\Phi|_F + V_{SB}} - \sqrt{2|\Phi|_F}] \qquad (7)$$

where

V_{T0}	=	V_T at zero source to body potential
γ	=	bulk threshold parameter
Φ	=	strong inversion potential
V_S	=	source to body potential

For TSOS devices, $V_T \approx V_{T0}$. The threshold shifts are minimized to a great extent. Also source-to-body and drain-to-body capacitances are negligible reducing parasitics and increasing Bandwidth (BW).

The electronic implementation of TRIT BP involves building sub-components or blocks. The final architecture can be realized be interconnecting such blocks on a single substrate. Being a parallel processing structure, Backpropagation networks are both iterative and highly structured. The building blocks take advantage of that fact simplifying design and testing at both cell and the system levels. This thesis addresses the design, simulation, and layout, of these basic building blocks. System level integration is beyond the scope of this thesis.

The proposed IC is universal in the sense that a single IC implements each layer of a multilayer perceptron. There is one to one relationship between the weight matrix and each IC. The building block is at least theoretically extensible in the horizontal fashion to any number of layers. However the maximum number of neurons is fixed vertically by fabrication and limited by the pin count. The power supply rails also limit the magnitude of the backpropagated term which would limit their horizontal extension. The vertical extension is the number of neurons per layer of the chip. The horizontal extension is the number of layers used in an application. Normally three-layer networks with an hidden layer is sufficient to approximate any function to a reasonable degree [28].

As opposed to traditional voltage mode analog signal processing, in which inherently current signals are converted to the voltage domain before any analog signal processing takes place, a recently reintroduced, current mode analog signal processing approach is taken. The current mode approach takes advantage of 1) the convenience of summing inner product and backpropagtion currents 2) the bidirectionality of triode Floating gate CMOS based weight multipliers. The use of current rather than voltage as an active parameter can result in higher gain, accuracy, and wider bandwidth due to the reduced voltage excursion at dynamic nodes [15].

Simulations were performed using SPICE. Layouts are accomplished by using CAD layout tool MAGIC. All circuits are fabricated using NRaD's fabrication facilities.

Chapter II will discuss the software implementation of this model. Chapter III will focus on the design, simulation, and performance testing of all building blocks. Chapter IV will offer conclusions based on the results and suggestions for the future work connected with investigation of this proposed hardware implementation.

CHAPTER II

TRIT MODEL SIMULATIONS

This chapter discusses the software program developed to test and validate the behavioral aspects of the TRIT algorithm. The purpose of the simulations is to test the convergence properties of the TRIT algorithm with variation in learning rate and ϵ_2 under non-ideal conditions. The component non-idealities are incorporated in the TRIT simulations. The TRIT algorithm will be compared to the Standard Backpropagation in the speed of convergence and its sensitivity to device imperfections. A character mapping and a pattern fitting problem will be used to establish the base line performances. The programs were developed in Matlab.

In VLSI circuits, effects including random offsets and mismatch, system distortion, frequency response, and temperature variations perturb the system outputs [7]. The effects that dominate the error in the system depends on the system implementation. In our simulations, we have attempted to include most of the device imperfection effects that play a major role in our implementation. The variation in transconductance, the V_T mismatch, and the channel length modulation parameter (λ) effect are the major concerns in analog circuit inaccuracies. The errors due to the above--mentioned inaccuracies in current conveyors, weight matrix, and output function (F(.)) will be discussed in this chapter.

2.1 Trit Program Description

A program which calculates the initial weight matrix, biases and does the feed-forward calculation is run first and is enclosed in Appendix C.

The program sets all the values of the weight and bias elements to 0.5 so that all the circuit points are started at the same initial condition. A random number corresponding to 5% variation of the weight elements is added to all the elements of the network. This number simulates the error present in the multiplier circuit. The reason is that the multiplier and the weight update circuits are non-ideal and a detailed analysis of the errors is presented in Appendix D. The current conveyors are driven by an opamp which is non-ideal. The opamp error is due to the mismatch of the input transistors as shown in Figure 13. This mismatch of the transistors introduces both V_T and β errors. Also the current mirroring transistors in current conveyor (see Figure 8) is not ideal because of the channel length modulation parameter (λ) effect. Dynamic cascoding is utilized to reduce the λ effect. The resulting effective λ is then negligible compared to the V_T and β errors. The β error is reduced by laying out the transistors M1 and M2 in a common centroid geometry and maintaing moderate geometries. These errors are represented by adding a 5% random number to the following elements of the network.

In the main program, shown in Appendix A, the values of ϵ_1 , ϵ_2 and η are set. The value of the learning rate determines the number of iterations needed by the network to converge. Then the iteration is started as shown by a flowchart in Figure 3 & 4 . If the steady state error (SSE) is less than 0.1, the program is terminated.

If SSE is greater than 0.1, the following procedure is started :

The values of δ_1 and δ_2 are calculated. The values of δ's determine whether the bias elements are to be adjusted. The values of δ's and outputs at the previous node corresponding to the weight matrix determines whether the weight matrix has to be updated. The region where the weights are to be updated are clearly shown in Figure 5. As seen from the Figure, if the values of δ_1 and O_j are greater than the threshold values, then and only then, are the corresponding weight elements W_{ij} updated. Otherwise the corresponding value of that weight element W_{ij} remains unchanged. Similarly if and only if, the value of δ_1 is

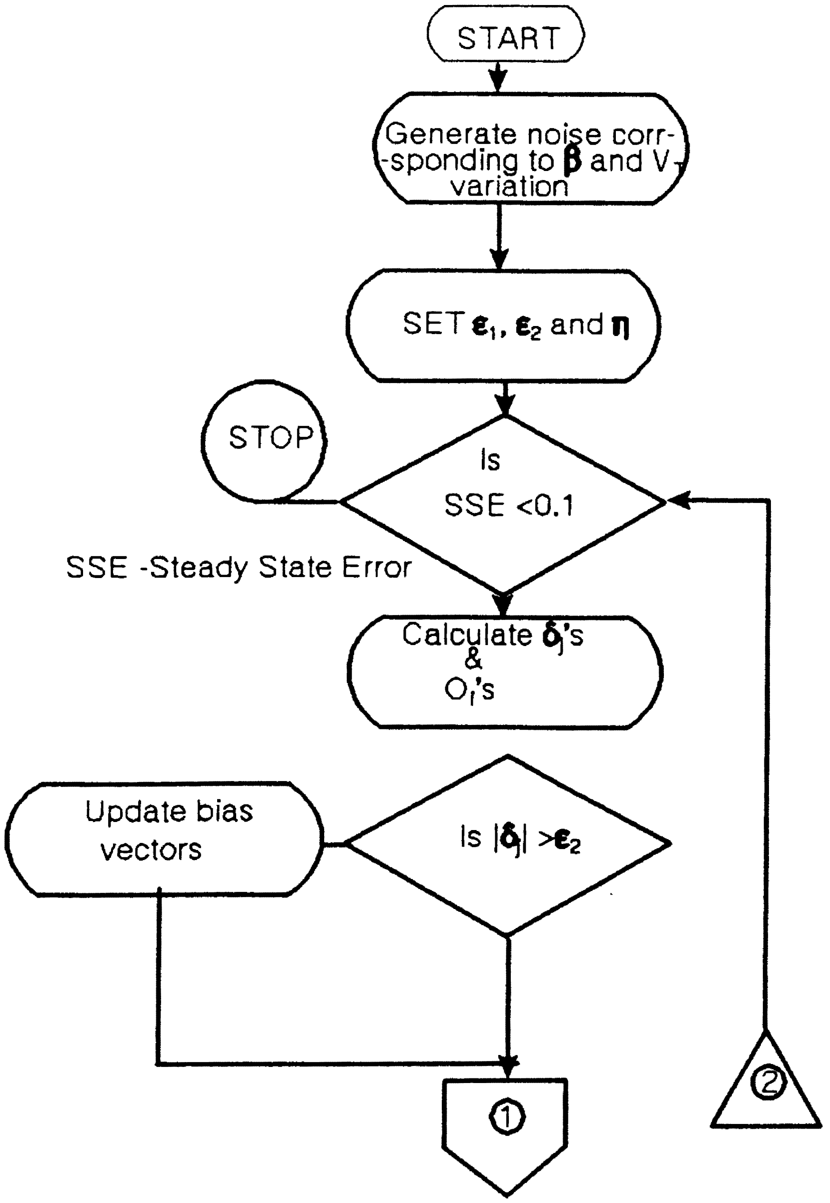


Figure 3. Flowchart of the TRIT program

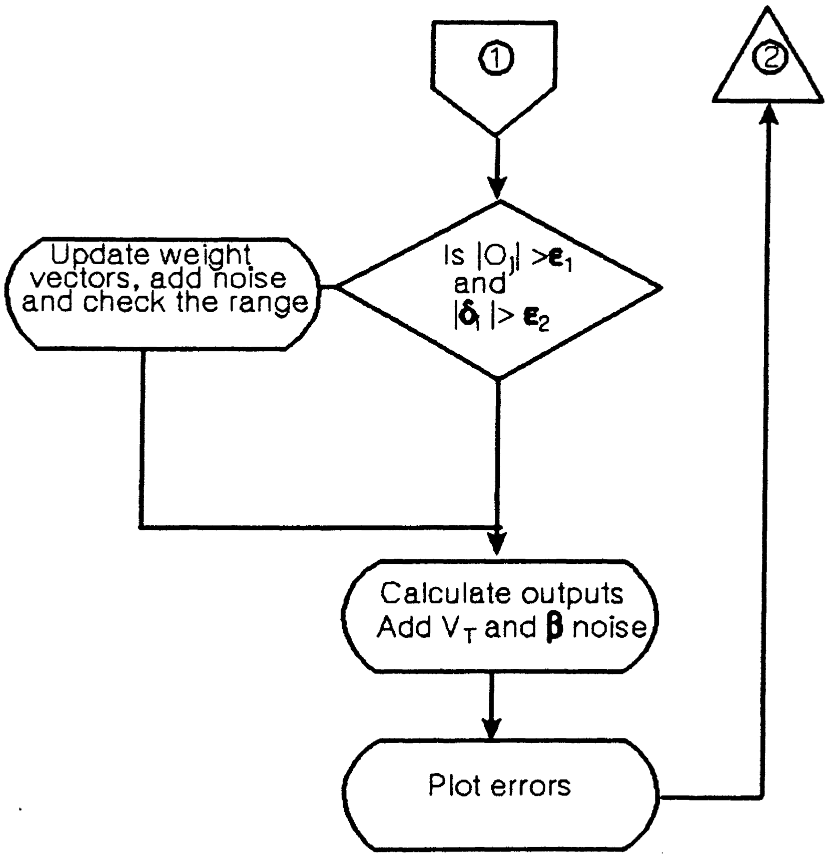

1
2
Is $|O_j| > \varepsilon_1$ and $|\delta_j| > \varepsilon_2$
Update weight vectors, add noise and check the range
Calculate outputs Add V_T and β noise
Plot errors

Figure 4. Flowchart of the TRIT program

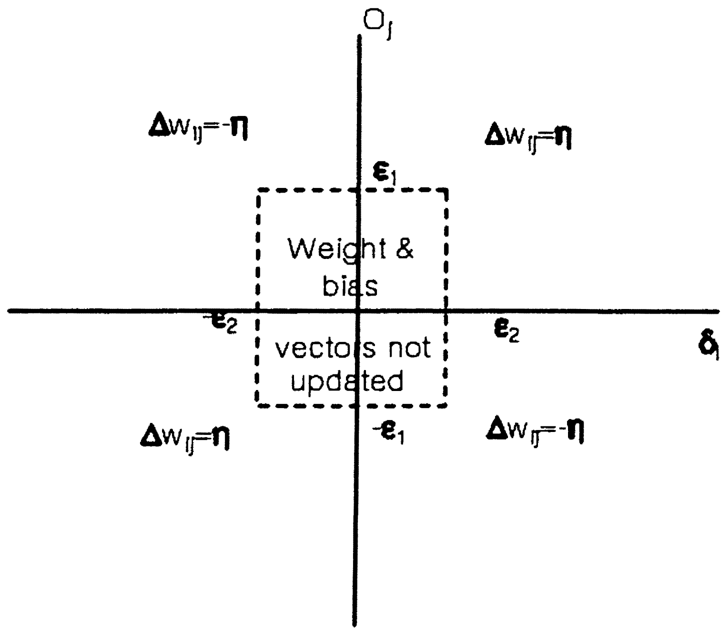


Figure 5. Regions of updation of weight vectors

greater than that of ϵ_2, the bias elements are updated.

After the modification of weights a noise term of 0.01 (σ) is added to each element of the weight and bias matrix. The value of .01 stems from the fact that the values of the weight matrix and the bias elements vary from 0.5 to 1.5 and the noise floor is assumed to be atleast 60dB down and centered around the mean value of 1.0. The input vector is multiplied by the multiplier circuit which is explained in Chapter III. The analysis of the multiplier circuit indicate a maximum error of 1% due to the β and threshold mismatch. Appendix D analyses the errors in the multiplier circuit. This error effect is introduced in the simulations by adding a value of 0.01.

The forward computation is now completed and the output error has been determined. The output error is plotted with respect to the number of epochs to observe the behavior of the network. Since there is always a finite range of weight values, which depends on the dynamic range of the weight multiplier circuit, weight variation is bounded. The upper limit is set at ±3 and the lower limit at ±0.2V. Also the squashing current conveyor or each layer outputs is also not ideal due to V_T and β effects. This effect is not symmetrical. A detailed analysis of the effect is shown in Appendix D. This effect is taken care of by adding the noise values to the output values at each layer.

2.2 Standard BP Program Description

A copy of the program is enclosed in Appendix B. The same initial weight matrix and the feedforward values are used to start the program.

First the value of η is set. Then the iteration is started. The SSE is checked for a value less than 0.1. If it is less than 0.1, the program is terminated. Else the following procedure is continued.

The weight matrix and the bias vector elements are modified according to the Standard BP formulas. The forward computation is completed and the network output is calculated. The network output is subtracted from the desired output to get the output steady state error. The error is plotted with respect to the number of epochs to observe the behavior of the network. Further the next loop is started by checking the Steady State Error (SSE) and calculating the values of δ's.

2.3 Adaptive BP Program Description

The adaptive modification is enclosed in Appendix C. The backpropagation networks typically employ adaptive modification to eliminate local minima or to speed up the convergence of the network. The speed of convergence is increased by increasing the learning rate if the error vectors tend toward minima and vice versa.

If the value of steady state error decreased in successive iterations, the value of learning rate η, is multiplied by a factor of 1.07. If the steady state error increased in successive iterations, the learning rate is decreased by a factor of 1.02.

If the steady state error is constant, the value of ϵ_2 is decreased. Care is taken to test that the value of ϵ_2 is not decreased by more than a factor of 1000. ϵ_2 was started initially with 0.02. If the steady state error was constant in successive iterations, then the values of ϵ_2 was halved. If the value of error was constant even though the value of ϵ_2 is reduced by a factor of 1000, the network was considered as nonconvergent. The argument proposed is that the variation of this parameter ϵ_2 in an actual network is limited by the noise floor of the network. We assume a SNR of 60dB or noise floor of -60dB. Then if we reduce the value of ϵ_2 below 60dB, the noise in the circuit takes over, and it becomes practically impossible to control the circuit.

2.4 Testing Procedures

The initial simulations were performed to provide evidence for comparing the convergence properties of back propagation with this three-state or trinary quantization of weights and bias updates with those of standard back propagation when applied to the same problems. During each iteration of the learning trial, the pairs of the input/desired output patterns in the training set were presented in a fixed sequence to the network and weights updated for each, and then the network output was tested over all pairs of the training set. This was continued till the convergence was obtained.

Convergence was defined such that all errors between desired and actual network outputs were required in magnitude to be smaller in magnitude than 0.1. As mentioned earlier the values of W_{ij} and b_i's are varied by adding a random constant at every iteration.

A character-mapping and a pattern-matching problem was selected. The number of hidden units and the learning rate were varied for standard BP, whereas for the trinary scheme, the value of ϵ_2 is also varied. The value of ϵ_1 was set at 0.33. The initial weights were set to be 0.5 + a random number (σ=0.2).

2.5 Results

Table 1 depicts the result of the simulations in which the number of hidden units and the learning rate were varied. The striking result is the difference in the convergence time of the two algorithms. Standard BP takes a very long time to converge. After a range of correcting the weights, the network goes through a prolonged phase in which the improvement is very low. In fact the learning rate has to be large (>.2) for the standard BP to converge. As the number of hidden units is increased, the number of iterations goes down significantly. For learning rates less than 0.1, it performs very poorly compared to the trinary algorithm.

In the case of trinary BP as shown in Table 1, for learning rates less than 0.05, it is 5 to 10 times faster than standard BP. But for a learning rate in the range of 0.3, it is even 15 times faster than BP. But it is doubtful whether trinary algorithm can have such a large learning rate because the RMS weight correction may be very large per iteration. However this simulation gives a feel for the convergence of the trinary algorithm and its relative speed of convergence compared to the standard BP. The reason for rapid convergence of the trinary algorithm may be due to the scaling imposed upon the weight and bias vectors updates by quantization and its investigation is beyond the scope of this thesis.

Failure to converge within the iteration limit occurred for the TRIT problem whenever the value of ϵ_2 was set to be greater than 0.01 for the various numbers of hidden units and the learning parameters. This occurs because the SSE limit set was 0.1 and unless the delta terms become smaller

TABLE I

CONVERGENCE COMPARISION OF STANDARD BP AND TRIT

Algor i-- thm	N_h = 10	N_h = 20	N_h = 40	N_h = 10	N_h = 20	N_h = 40	N_h = 10	N_h = 20	N_h = 40
BP	440 (0.3)	260 (.3)	145 (.3)	>500 (.1)	490 (.1)	290 (.15)	>500 (.05)	>500 (.05)	>500 (.05)
TRIT BP	9 (.3)	9 (.3)	9 (.3)	45 (.1)	33 (.1)	27 (.1)	158 (.02)	53 (.05)	54 (.05)

than 0.01, convergence is not possible. if the delta terms fall below ϵ_2 the weight corrections ceases due to quantization and the network doesn't converge.

2.6 Comparison of Adaptive BP and TRIT

It is clear from the simulations that BP with adaptive weight modification is slower than the TRIT implementation in speed of convergence. See the results of simulation shown in Table 2.

The TRIT BP is 2 times faster than the Adaptive weight modified BP. The number of iterations remains constant for the variation of the number of hidden units in Standard BP with adaptive weight modification while it varies very little in TRIT BP. So based on our limited simulation results, even the adaptive weight modification of BP will not result in comparatively faster in convergence than the TRIT.

2.7 Summary

These simulations give a feel for the potential success of a TRIT based algorithm. The TRIT algorithm is found to be faster in convergence compared to the BP algorithm. The component-to-component variation appears to have an negligible effect on the convergence property of TRIT, which

TABLE II

CONVERGENCE COMPARISION OF ADAPTIVE BP AND TRIT

Algo ri-- thm	N_h = 20	N_h = 40	N_h = 60
BP	74	74	74
TRIT BP	41	42	47

is encouraging because the analog hardware components are inherently low-accuracy components. The various building blocks that went into this development of TRIT hardware and their simultion results are discusssed in Chapter III.

CHAPTER III

SYSTEM BUILDING BLOCKS

This chapter presents the design and simulation results for the basic building blocks of the TRIT algorithm. The proposed architecture takes advantage of the bidirectionality of the current conveyors and the EEPROM CMOS based weight multipliers. The architecture of the network is shown in Figure 2. The EEPROM based weight multipliers are arranged in a matrix form as shown in the Figure 2. The input circuit consists of two current conveyors driving the weight matrix. The current conveyor functions as a processing element and as well as a bi-directional voltage/current buffers (BiVI) [52]. The current conveyor based weight matrix drivers are shown in Figure 6. The current conveyors on the input side (CC_1 and CC_2) function as voltage buffers driving a single weight matrix column. The current conveyor CC_3 is used as current controlled current source. The current conveyor CC_4 functions as an output buffer, driving a non-linear mapping (squashing) function (F(.)). The derivative circuit, which is used to find the delta vectors, is also shown in the

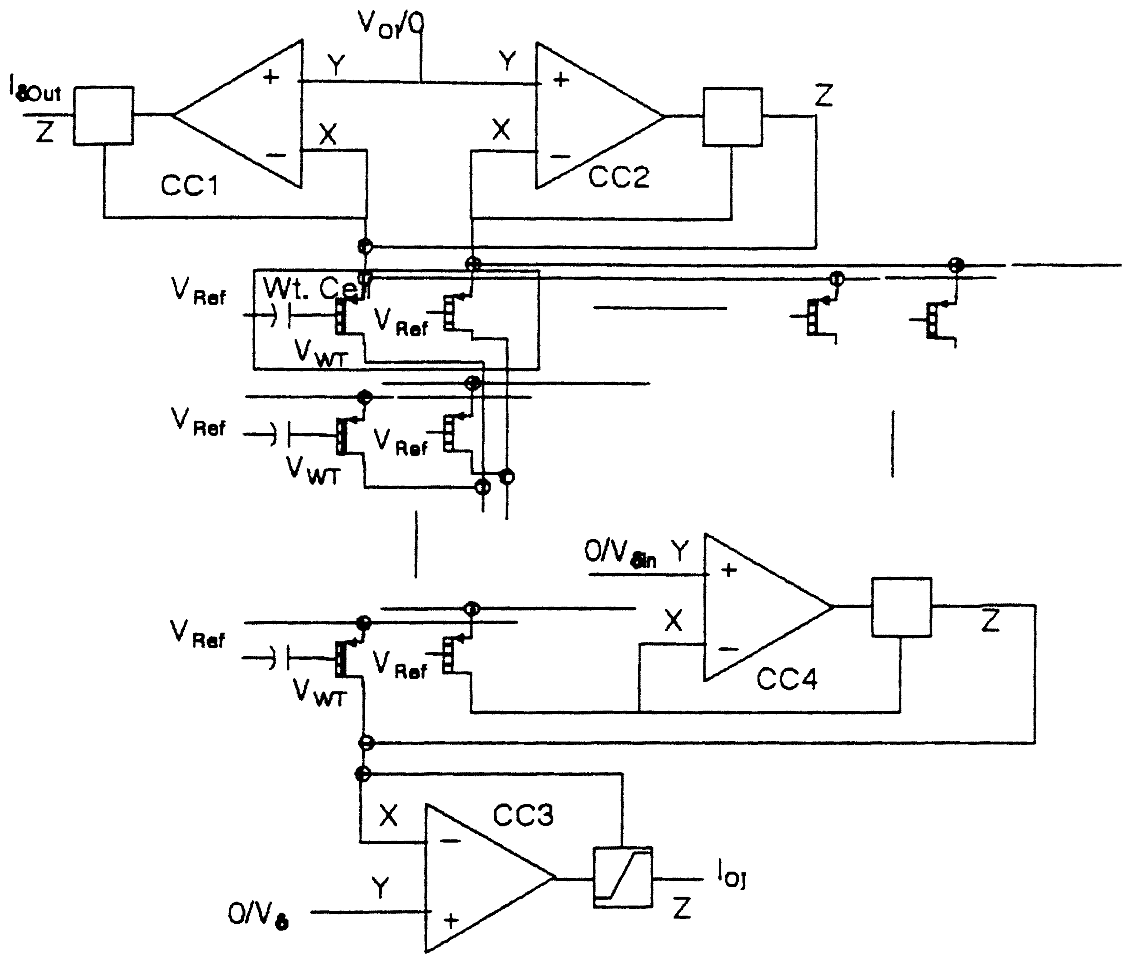


Figure 6. CC Based Weight Matrix Drivers

system architecture. A weight adjustment logic, based on the TRIT algorithm, is needed for the weight update during network learning.

From this discussion, it is clear that the current conveyors form the most basic element of this architectural approach and care must be taken to specify their performance. This is necessary for the input/output circuits to interface properly, because both the input and the output circuits are made up of current conveyors. The following section is a brief summary of the interface specifications.

3.1 TRIT Interface Specifications

Linear Limiting Squashing CCII+

I_{in}(Full Scale): 50uA @X = ±0.5V

I_{out} (Sat):±3uA

I_{out} (sat) = $\beta_{wt} V_{wtFS} V_{FS}/4$

R_{out}>10MEG at I_{out} <=3uA

$R_L=4/(\beta_{wt} * V_{wtFS})$

Input & Output Biasing CCII+

X:±2.5V

Y:I_{out}>=±1500uA @ Z= ±2V

Z:Scale 1:1 at Follower and 1:1 @mirror

R_{out}>=1MEG @50uA (Cascoded)

Current Error<=1%

Delta CCII+

X:±2.5V

Y:I_{out}>=±50uA @ Z= ±0.5V

Z:Scale 1:1 at follower and 1:1 @ mirror

R_{out}>=1Meg @50uA (Cascoded)

Current Error<=1%

3.2 Current Conveyors

3.2.1 Introduction

A current conveyor is a four terminal device which performs many useful analog signal processing functions when used in arrangment with other electronic elements. Current conveyors are functionally flexible and versatile. They can form an integral part of all I/O circuits [49-51]. Current conveyors offer several advantages over conventional operational amplifiers. They provide the highest gain bandwidth product of the process, which depends only on the opamp used [48]. They are used within this thesis as a practical building block for the implementation of the TRIT algorithm.

The block diagram of a current conveyor (CC) is shown in Figure 7. Class-I (CCI±) and class-II (CCII±) conveyors have well defined properties [52]. A CCII± can be expressed

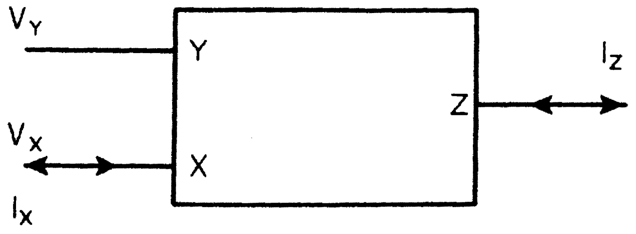


Figure 7. Block Diagram of the CC-II+-

as:

$$\begin{bmatrix} I_Y \\ V_X \\ I_Z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & \pm 1 & 0 \end{bmatrix} \begin{bmatrix} V_Y \\ I_X \\ V_Z \end{bmatrix} \tag{8}$$

From the above equation it is clear that no current flows into terminal Y. The voltage applied to terminal Y will cause an equal voltage to appear on terminal X. Terminal Y exhibits an infinite input impedance and terminal X exhibits a zero input impedance. An input current I_X on terminal X causes an equal current to flow into or out of the high impedance output terminal Z. The positive sign indicates that at any instant both, I_X and I_Z are in the same direction (CCII+) while the minus sign denotes the opposite directions of the currents signifying CCII-.

The current CCII configuration allows convenient switching between the current conveyor mode and the Voltage controlled Voltage Source (VCVS) mode. This choice supports the generation of matrix inverse function which is essential for implementing backpropagation.

"The CCII may be viewed as an ideal transistor" [45]. The ideal behavior of the NMOS (M_{FN} in Figure 8) transistor can be achieved by using it in the negative feedback loop of the operational amplifier. The current can only flow away from the X terminal. If a PMOS (M_{FP}) transistor is used in the feedback loop, current will be restricted to flow into the X terminal. Bi-directional current flow can be achieved

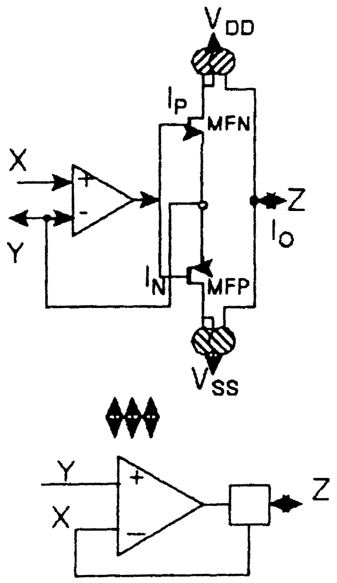


Figure 8. Current Conveyor Symbol

by using a complementary pair of MOS transistors (M_{FN} and M_{FP}) in the opamp feedback loop. This drain current of M_{FN} and M_{FP} can be mirrored to the output node Z. Thus the input current I_x is conveyed to the output current I_z. This is a CCII+ realization, since both I_x and I_z simultaneously flow in the same direction.

The bi-directional voltage/current (BiVI) buffers which are based on the current conveyor concept, are shown in Figure 6. These buffers provide the dual function of voltage drivers and current sources/sinks to isolate the W matrix in the forward/reciprocal mode. During the feed forward cycle, the buffers on the input side are configured as voltage controlled voltage sources, and the buffers on the output side are configured as current controlled current sources. The output side consists of a current conveyor driving the non-linear squashing function F(.) which develops the output voltage to the next layer. A nearly identical structure is duplicated to achieve Backpropagation. Each current conveyor accepts current in the input mode or supplies the drive voltage (and current) to the weight row matrix (math column).

During the forward propagation cycle the Y point of CC_1 and CC_2 are tied to O_i while the Y point of CC_3 and CC_4 are grounded. The voltage controlled voltage source configured CC ensures $V_{X1}=V_{Y1}$. This results in voltage O_i being applied across the drain to source of both the floating gate and reference transistors in the i^{th} row. The current flowing in the reference transistor is mirrored and applied to the floating gate column. Due to the presence of the weight charge on the floating gate, these currents will differ by the inner product of the applied voltage (O_i) and the stored weights. The difference or signal current O_j is then measured by CC_3. Current squashing is accomplished by the modified CC_3. The current controlled current source configured CC ensures $I_{Z3}=I_{X3}$. This current is the input to a linear saturating function which limits the current to 3uA. Similarly, if all the column currents are summed in the X_4 terminal then the resulting current indicates the multiplied value of the input weight vectors and the weight matrix.

During back propagation a voltage $V_{\delta in}$ is applied to the Y inputs of both CC_3 and CC_4 with the Y inputs of CC_1 and CC_2 grounded resulting in role reversal of CC's with the $I_{\delta' out}$ current available at the Z output of CC_1. $I_{\delta' out}$ must be further processed (multiplied by a derivative) to compute δ_{jout}. The whole approach critically depends on the high accuracy of the CC's. The following section presents the design of CC_1 through CC_4.

3.2.2 High Power Conveyor

3.2.2.1 Design

The high power current conveyor CC_2 must be able to supply the total weight current in one column of the weight matrix. The signal swing is also determined by the two current conveying transistors M_{FN} and M_{FP} as shown in Figure 9. The two transistors have to be saturated for satisfactory performance. The total weight current in one column is

$$I_{HPCC} = 100*3*5=1500uA$$

To source/sink this current, the source/sink transistors, M_{IPA}, M_{FN}, M_{INA}, and M_{FP} must be sized appropriately. For a weight current of 1500uA and an output swing of ±3.5V (Section 3.1).

$$(W/L)_{MFN} =2*I/((\Delta V)^2 * K_n) = 2*1500/((3.5-V_{TN})^2*48) = 10$$

$$(W/L)_{MFP} =2*I/((\Delta V)^2 * K_{p)} = 2*1500/((2.5-V_{TN})^2*21) = 20$$

We expect a output swing (at Z terminal) of at least ±2V. This is because, the weight transistor drain to source voltage swing is determined by this Z terminal. The dynamic range of the weight multipliers depend on the Z terminal swing. The length of the cascoding transistors (M9 and M10) was fixed to be 2um, while the length of the mirroring transistor was 6um (MINA and MINB). The long mirroring transistor was selected to reduce the channel length

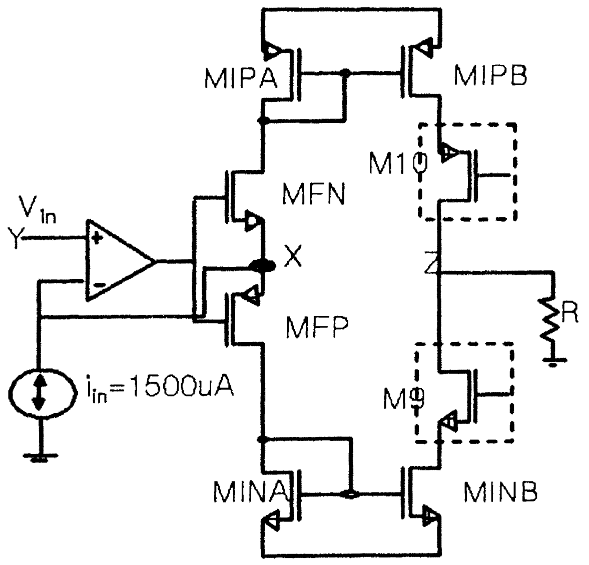


Figure 9. Current Conveyor circuit

modulation parameter (λ) effect and to reduce local geometric and doping mismatches.

So $(\Delta V)_{M10} = (\Delta V)_{MIPB}/\sqrt{(6/2)}$ (9)

$(\Delta V)_{M9} = (\Delta V)_{MINB}/\sqrt{(6/2)}$ (10)

$(\Delta V)_{MIPB}/\sqrt{(6/2)} + (\Delta V)_{MIPB} = 3+V_{TP}$ (11)

$(\Delta V)_{MINB}/\sqrt{(6/2)} + (\Delta V)_{MINB} = 3-V_{TN}$ (12)

where $(\Delta V)_{MINB} = \sqrt{(2*I_Z/\beta_{MINB})}$ (13)

and $(\Delta V)_{MIPB} = \sqrt{(2*I_Z/\beta_{MIPB})}$ (14)

$I_Z = 1500uA,\ k_n=48,\ k_p=21$

Solving these equations, we get

$(W/L)_{MIPB} \approx 75$

and $(W/L)_{MINB} = 37.5$.

For a unity ratio current mirror, the dimensions of the transistors MIPA and MINA are same as that of MIPB and MINB respectively.

$(W/L)_{MINA} = (W/L)_{MINB} = 37.5$

$(W/L)_{MIPA} = (W/L)_{MIPB} = 75$.

Since we have fixed the length of the transistors at 6um,

$W_{MINA} = W_{MINB} = 6*37.5 = 225um$

$W_{MIPA} = W_{MIPB} = 6*75 = 450um$

$W_{MINA} = W_{MINB} = 2*116 = 232um$

$W_{MIPA} = W_{MIPB} = 2*240 = 480um$

The result is that the widths are very large for the mirror transistors. This is due to the design objective of maintaining at least a 2V swing at the output (Z terminal), while sinking such a large current in the mA range. This large current will translate into large ΔV drop across the mirroring and the cascoding transistors (MIPB and M10, MINB and M9). For the required Z output swing, we need to have only a 2V (ΔV) drop across the two large current carrying transistors. This causes the widths of these transistors to be large to reduce the ΔV value across the transistors.

The design requirement for the Z terminal output swing of 2V would necessitate P-transistor widths of 900um. Therefore for pragmatic reasons, the Z terminal swing was reduced to 1V for final fabrication.

The revised values of (W/L)'s for the mirrors considering a value of 1V output swing in Z terminal are

$$W_{MIPA} = W_{MIPB} = 180um$$

$$W_{MINA} = W_{MINB} = 90um$$

$$W_{M9} = 90um$$

$$W_{M10} = 180um$$

3.2.2.2 Simulations

The D.C., and A.C. characteristics of the high power current conveyor are shown in Figures 10 & 11 respectively. The D.C. curve shows that the output current tracks the input current in the range of 1.5mA. The MIPA and MINA current indicate the mirroring currents, while MIPB and MINB indicate the mirrored currents. The simulated error between

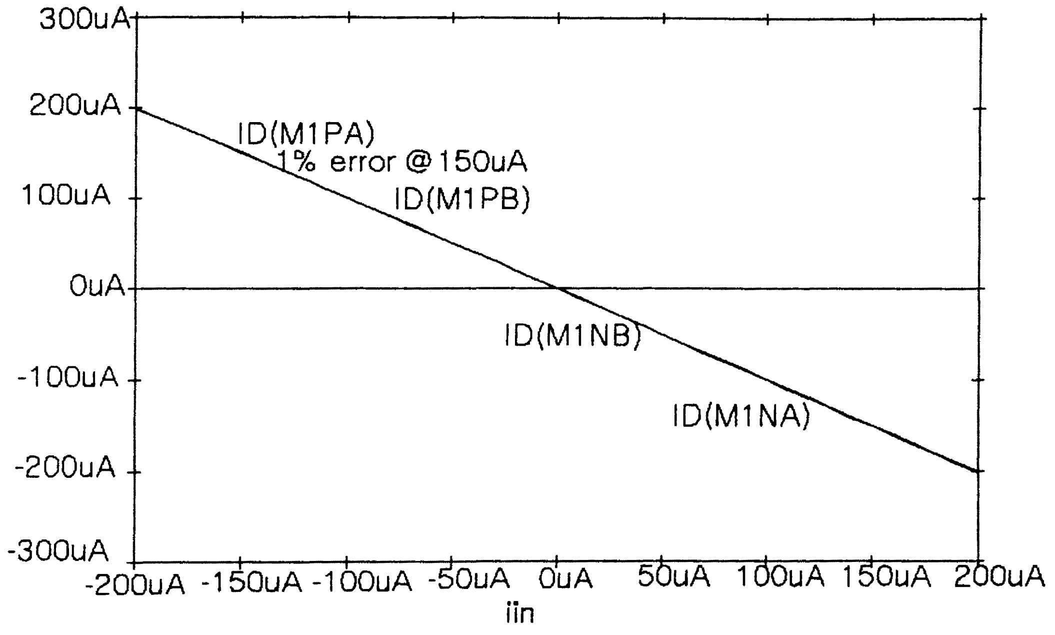


Figure 12. D.C. Characteristics of Low Power CC

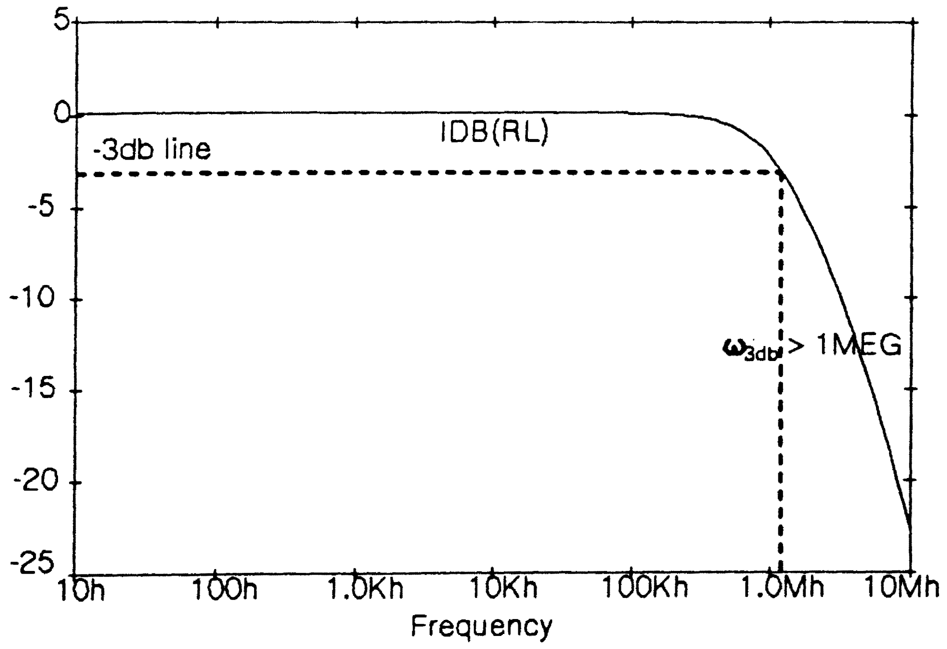


Figure 11. A.C. Response of High Power CC

these two currents is found to be less than 1% (@ 1.5mA), which indicates that our objective of accurate current conveying is achieved at the required output swing. The output swing is determined by forcing the Z terminal to remain at 1V while sweeping the input current. The ω_{3db} point can be determined from the A.C. characteristics. The A.C. characteristic of Figure 11 was obtained by biasing the circuit at an d.c. input current of 1uA. For specific applications, the full power bandwidth is also an important criterion which is computed from the slew rate.

3.2.3 Low Power Conveyor

The output current requirement is 150uA, which is 1/10 of the high power conveyors's current. We want the same swing as the high power conveyor in Z, but at reduced current value. So the dimensions of all the transistors are the scaled values (by 10) of that of the high power current conveyor as follows

$$(W/L)_{FN} = 1$$

$$(W/L)_{MP} = 2$$

$$W_{MIPA} = W_{MIPB} = 18\text{um}$$

$$W_{MINA} = W_{MINB} = 9\text{um}$$

$$W_{M10} = 18\text{um}$$

$$W_{M9} = 9\text{um}$$

3.2.3.1 Simulations

The D.C. characteristics of the low power current conveyor is shown in Figure 12. The D.C. output current follows the input current in the range of 150uA and the current error due to λ effect (excluding $\Delta\beta$ and ΔV_T) is again found to be less than 1%.

3.2.4 Opamp

A simple self-compensating opamp is chosen for the required opamp function for the CCII due to its simplicity and the moderate offset demands of the CCII structure. This circuit results in a stable, self-compensated, minimum area opamp structure. The bandwidth of the opamp can be increased by increasing the bias currents at the expense of the power dissipation. More complex opamps will produce a better performance than this opamp but with increased area. The weakness of the opamp is the offset. However, since offset is adjusted as a part of backpropagation learning, offset requirements will not be stringent. Also practically, with an open loop gain of at least 60dB offset performance will be limited more by the threshold matching of the input differential pair. Future refinement beyond the scope of this thesis will focus on Bandwidth and output swing improvement. The opamp determines the performance and

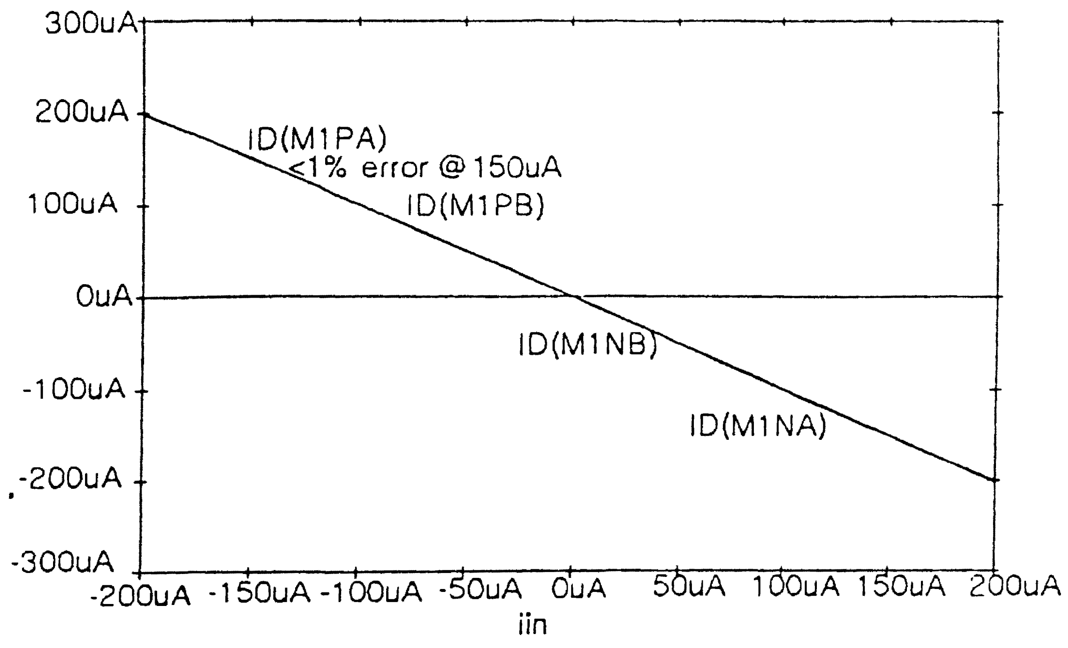


Figure 12. D.C. Characteristics of Low Power CC

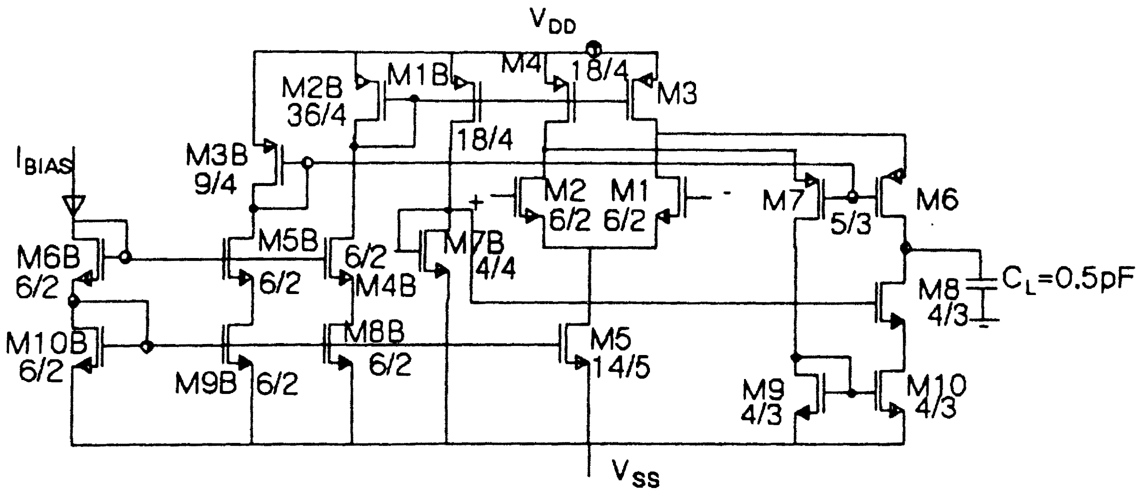


Figure 13. opamp circuit diagram

the dynamic range of the CC circuit. The circuit diagram of the opamp is shown in Figure 13.

3.2.4.1 Analysis of Opamp

In all the subsequent discussions, the symmetry of the opamp is exploited to reduce the redundancy of the equations i.e. M1 and M2, M3 and M4 and M6 and M7 are assumed to be symmetric and matched transistors.

Output Swing

Positive : It is obvious that to maintain M6 in saturation, the output voltage V_o is limited to

$$V_O \le V_{GM6} - V_{TP} \tag{15}$$

Negative : To maintain M8 in saturation, the negative output voltage swing is limited to

$$V_O \ge V_{GM8} - V_{TN} \tag{16}$$

Input Common Mode Range

The input common mode range V_{CMR-} is given by

$$V_{CMR-} \ge V_{GM5} - V_{TNM5} + V_{TNM1} \tag{17}$$

Because $V_{GM5} - V_{TNM5}$ is the minimum voltage to maintain M5 in saturation and the gate voltage of M1 should be at least a V_T above its source voltage. V_{CMR-} will be less than V_{GM5} because $V_{TNM1} >= V_{TNM5}$. and there is a threshold shift associated with M1 and not with M5. The threshold shift is due to the fact that the source of M1 is at a different potential than its substrate.

The input common mode range V_{CMR+} is given by

$$V_{CMR-} \geq V_{GM5} - V_{TNM5} + V_{TNM1} \tag{18}$$

The voltage $V_{GM3} - V_{TPM3}$ is the voltage to maintain M3 in saturation and the gate voltage of M1 should be a V_T above its drain voltage to maintain it in saturation.

Output Resistance

The output resistance of the opamp is

$$r_o = \mu_{M6} * (r_{dsM1} \| r_{dsM3}) \ \| \ \mu_{M8} * r_{dsM10} \tag{19}$$

where

$$\mu_{M6} = 1/(I_{DM6} * \lambda_{M6}) * g_{mM6}$$

$$r_{dsM10} = 1/(I_{DM6} * \lambda M6) \tag{20}$$

$$g_{mM6} = \sqrt{(2\beta I_{DM6})}$$

So in order to maintain high output impedance, the λ's of the transistors M8, M6 and M10 should be high. Since the channel length modulation parameter is inversely proportional to the length, the lengths of M6, M8 and M10 are made reasonably high to ensure a high output impedance.

Gain

The gain of the opamp is given by

$$A_v = g_{mM1} * r_o \tag{21}$$

The value of r_o is given by Equation 18.

Gain-Bandwidth Product (GBW)

The GBW is given by

$$\begin{aligned} & GBW = g_{mM1}/C_{o1} \\ \textit{where} \quad & C_{o1} = C_L + C_o \\ & C_o = C_{dbM8} + C_{dbM6} \\ \textit{Since} \quad & Co < C_L \\ & C_{o1} = C_o = C_L \end{aligned} \tag{22}$$

Slew Rate

The Slew rate (SR) is given by

$$SR = I_{SS}/C_o \tag{23}$$

where I_{SS} is current through M5. So it is advantageous to increase the value of the width of M5 to increase input CMR and SR.

Cut-off Frequency

The cut-off frequency ω_{3db} is given by

$$\omega_{3db} = 1/r_o C_o \tag{24}$$

where r_o and C_o are given previously.

<u>3.2.4.2 Biasing Circuit of the opamp</u>

The bias circuit that forms a part of the opamp circuit is shown in Figure 13. The transistor M3B sets up the gate voltage for M6. By varying the gate voltage of M6, the required output positive output swing can be achieved. The transistors M6B and M10B mirror the input current to the transistors M5B and M9B. The reason for using two stack mirroring transistors is to achieve the required matching with the transistors M6, M8 and M10 in addition to cascoding. M2B controls the gate voltage of M3 and M4. The gate voltage of M3 and M4 and the widths effectively control the output current. The transistor M7B controls the gate voltage of M8. By controlling the gate voltage of M8, the required output swing in the negative direction can be achieved. The lack of symmetry in the biasing voltage of M8 is apparent. This can be corrected in future versions by including a transistor to match that of M10. The lack of symmetry produces an offset voltage as evident from the D.C. characteristics in Figure 14. The transistors M8B, M4B and M1B are used for mirroring and matching purposes. The gate voltage of M5 is set by dimension of M10B and the current through it.

The design criteria from Section 3.1 results in the following biasing constraints.

Bias current : 50uA

Output swing : ±3.5V

Since we desire an output swing of +3.5V the gate voltage of M6 should be at least $3.5+V_{TP}$ to maintain M6 in saturation i.e. the gate to source drop on the transistor voltage of M6 should be at least $3.5+V_{TP}$ to maintain M6 in saturation i.e. the gate to source drop on the transistor M3B should be $(3.5+V_{TP})V$.

So $V_{GM3B} = V_{GM7,M6} = (3.5 + V_{TP})V = 2.63V$

$$\sqrt{\frac{2*I_{BIAS}}{\mathrm{B}_{M3B}}} + V_{TP} = 2.63V \qquad (24)$$

Substituting the values of

$I_{BIAS} = 50uA$

$V_{TP} = 0.92V$

$\beta_{M3B} = 2*I_{BIAS}/(2.5-0.92)^2 = 39$

$(W/L)_{M3B} = 39/21 = 2$

Similarly the negative output swing is -3.5V. So the gate voltage of M7B should be $-3.5+V_{TN}$ to maintain M8 in saturation. The gate to source drop on M7B should be $(-3.5+V_{TN})$ V

So $V_{GSM7B} = V_{GSM8}=2.6V$

$$\sqrt{\frac{2*I_{BIAS}}{\mathrm{B}_{M7B}}} + V_{TN} = 2.6V \qquad (25)$$

Solving for M7B by substituting the values

$\beta_{M7B} = 2*I_{BIAS}/(2.6-0.92)^2 = 41$

$(W/L)_{M7B} = 1$

The gate to source voltage on M5, M3 and M4 should be at least $1.5V_T$ to reduce the effect of V_T mismatch.

$V_{GSM3,M4} = V_{GSM2B} = \sqrt{}(2*I_{BIAS}/ \beta_{M2B}) + V_{TP}$

$= \sqrt{}(2*57/21*9) + 0.92 = 1.7V > 1.5V_T$

and a load capacitance of 0.5pF with a resulting slew rate of 50/0.5=100V/usec. However after layout and simulation 50uA current was not sufficient. Resimulations resulted in a current of 57uA. The 100V/usec requirement should result in a peak full power frequency of

$$f_p = SR/2\Pi Vpp = 2*100/2\Pi(4) = 8MHz$$

3.2.4.3 Power Circuit Design

The transistors M1 and M2 were laid out in a common-centroid geometry to minimize DC offset. To facilitate the common centroid design in Magic, each cell has a (W/L) ratio of 6/2. The only consideration in the design of transistors M6, M8, and M10 is that their lengths should be sufficiently large to achieve output impedance of the opamp. High output impedance translates into high gain of the opamp. The increase in capacitance with their increase in length of the transistors is negligible compared to the output capacitance of 0.5pF. The design of M5 influences the slew Rate and the negative Power Supply Rejection Ratio (PSRR). So increasing the width of M5 will provide an increased CMRR, increased input Common Mode Range (CMR) and increased Slew Rate (SR), which have no significant impact on our design objectives.

3.2.4.4 Simulations

The D.C., A.C., and transient characteristics of the opamp are shown in Figure 14, 15 & 16. The D.C. characteristics show an offset output voltage of 2.26V. The A.C. characteristics indicate the the gain-bandwidth product is 20MHz and a gain of 72dB. This shows that gain achieved is higher than the design objectives. The transient characteristics indicate a negative slew rate of 16V/usec and a positive slew rate of 4V/usec .

3.2.5 Squashing function

3.2.5.1 Design

Figure 17 shows the linear limiting version of the squashing CC.

Squashing is achieved by addition of two transistors's at each conveyor half. This generates a linear limiter function. One of the two transistors (MS2N) results in additional current branch to the supply rail. The upper transistor in the traditional mirror branch takes on the classical follower role (MFN), while the lower transistor (MS1N) serves to limit the current that can be mirrored to the Z output. The saturation level is a function of the gate bias and geometry of the lower Transistor (MS1N) of

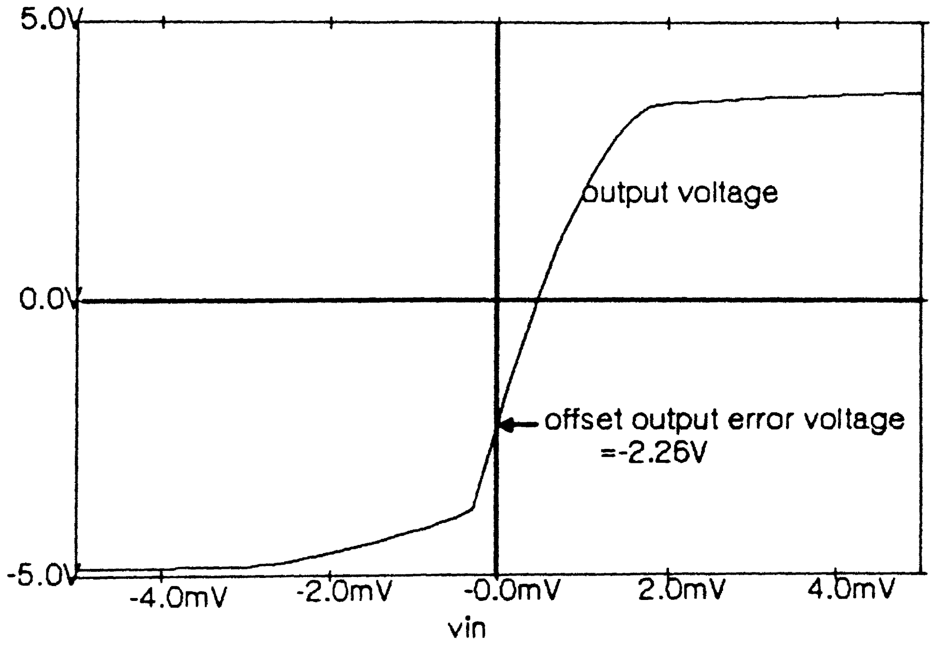


Fig 14. D.C. characteristics of op-amp

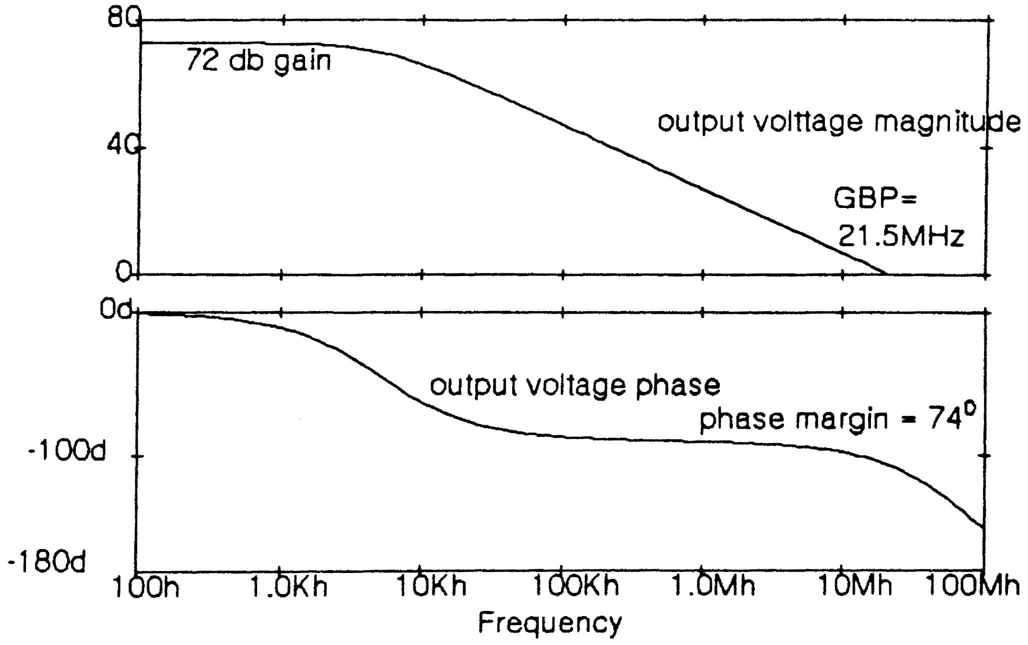


Fig 15. A.C. characteristics of op-amp

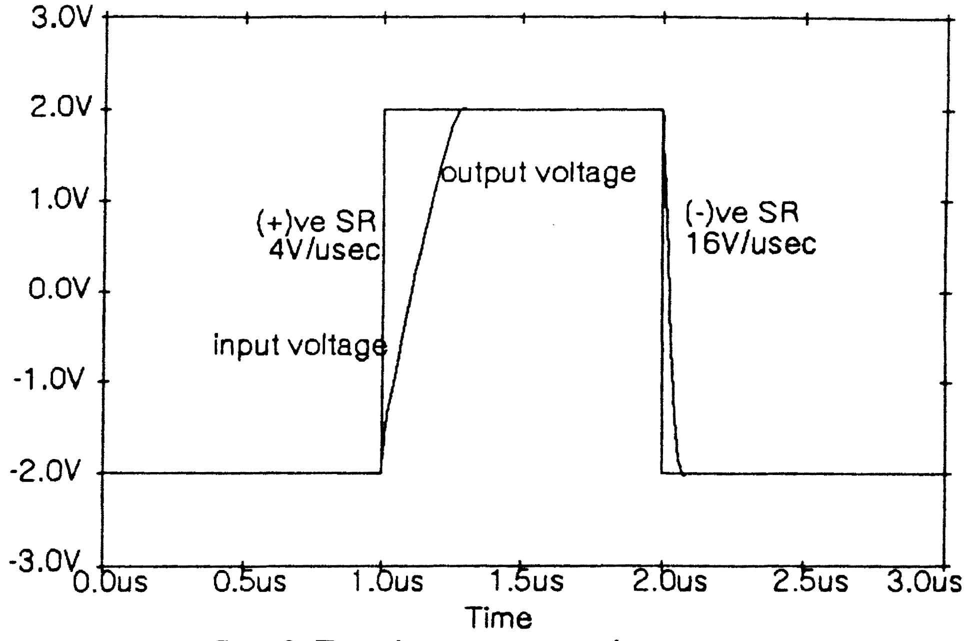


Fig 16. Transient response of op-amp

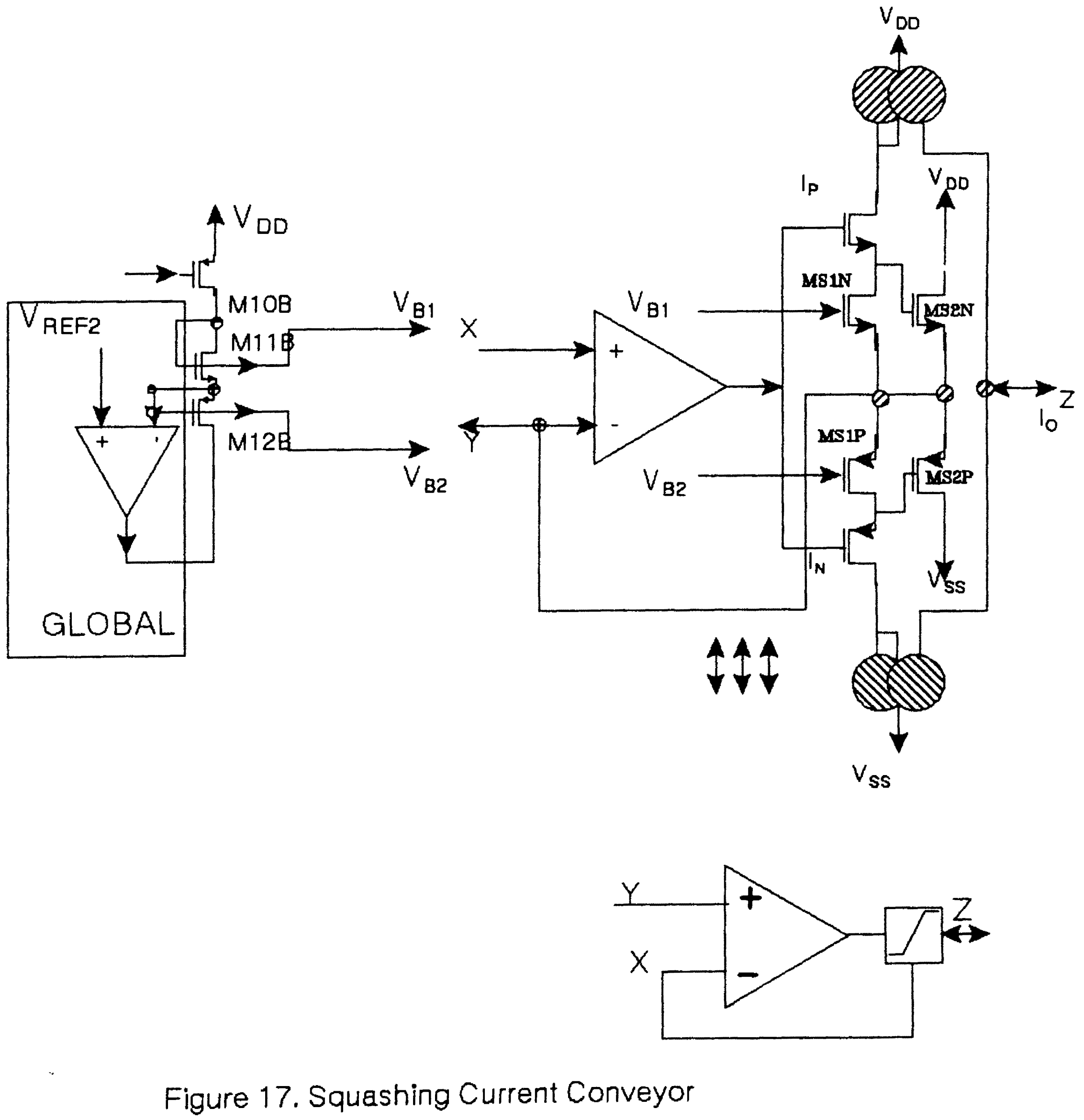


Figure 17. Squashing Current Conveyor

branch one. Once the current in the mirrored path saturates, all additional current is routed to the supply rails through the secondary path (MS2N). This results in a linear limited function with a globally programmable saturation limit.

The transistor MS2N should be large enough to supply the current for the entire column of the weight matrix once the transistor M1 saturates. The saturation current was fixed at 1500uA.

The opamp output swing is ±3.5V. So the source of MFN should be 3.5V + V_{TP} to maintain MFN in conduction. The source of MS2N, the X terminal, is assumed to be at 2V during the forward propagation. So the gate to source voltage of MS2N is $3.5V+V_{TP}-V_{TN}$.

The current requirement is

$32*V_{WTMAX}*\beta_{WT}*V_{pp}$ for a 32 column matrix

$100*V_{WTMAX}*\beta_{WT}*V_{pp}$ for a 100 column matrix

$(W/L)_{MS2N}= 2*(I)/((3.5-V_{TN}-V_{TN})*k_n)$

$= 14$ for a 32x32 weight matrix

$= 43$ for a 100x100 weight matrix

$(W/L)_{MS2P} = 2*(I)/((3.5-V_{TN}-V_{TN})*k_n)$

$= 27$ for a 32x32 weight matrix

$= 85$ for a 100x100 weight matrix

3.2.5.2 Simulations

Figure 18 shows the D.C. characteristics of the squashing current conveyor. The saturation current is fixed to be ±3uA.

3.2.5.3 Approximate Sigmoid function

There are many approaches for the saturation function in ANN's : Linear saturation function, S-shaped sigmoid function, Hyperbolic tangent etc. The sigmoid function generation circuit is discussed in this section.

A MOS device has a nonlinear output I-V characteristic. It can be utilized as a sigmoid function. The derivative has to be generated as a piecewise linear function. The gain can be obtained from the slope of the output-input curve of the MOS transistor.

In Figure 19 , V_{in} , V_C, and I_o are the input voltage, the gain control voltage, and output current respectively. The design of all the MOS devices ensure that they operate in the saturation region for the entire range of input voltage. Applying KVL around the loop shown in Figure 19,

$$\begin{aligned} V_i &= V_{GSM5}+V_{GSM8}+V_{DSM9}-V_{GSM7}-V_{GSM6} \\ &= \sqrt{\frac{2I_{D5}}{\beta_5}}+V_{T5}+\sqrt{\frac{2I_{D8}}{\beta_8}}+V_{T8}+V_{DS9}-\sqrt{\frac{2I_{D7}}{\beta_7}}-V_{T7}-\sqrt{\frac{2I_{D6}}{\beta_6}}-V_{T6} \\ &= V_{DSM9} \end{aligned} \qquad (26)$$

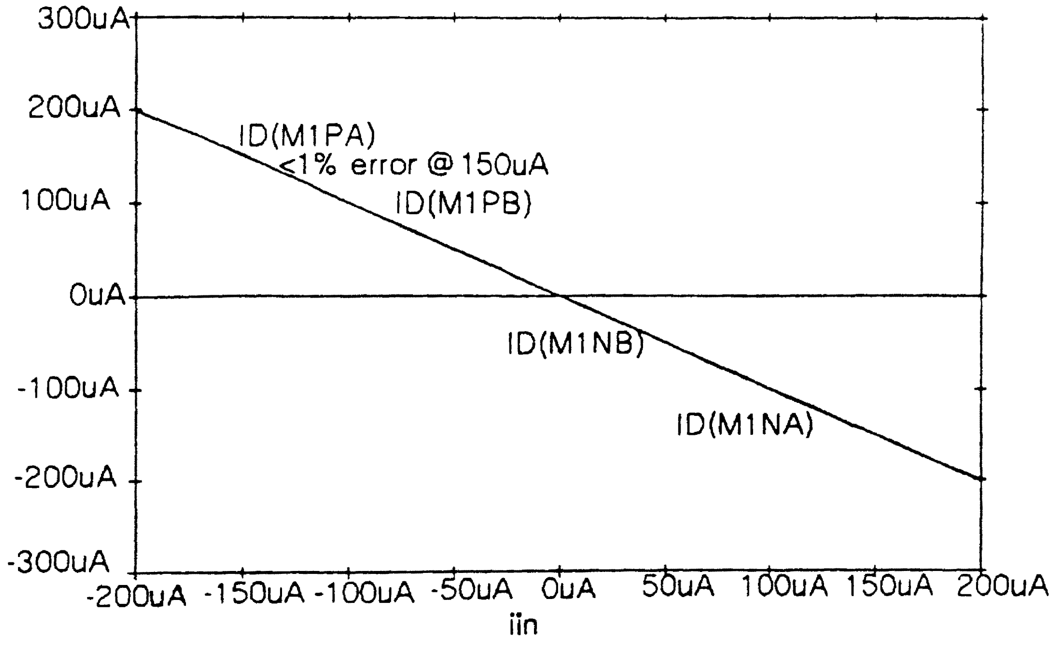


Figure 12. D.C. Characteristics of Low Power CC

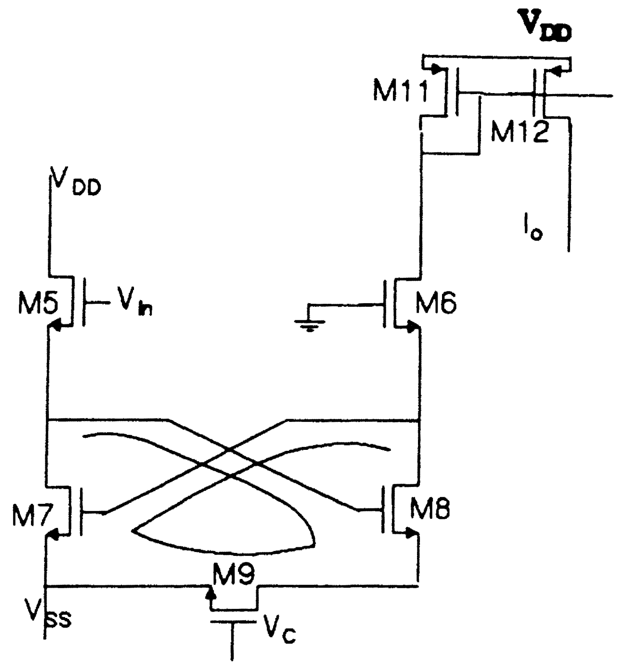


Figure 19. Sigmoidal Function (Single Quadrant)

The equation above assumes that the transistors M5, M6, M7 and M8 are matched. The TSOS process ensures almost exact cancellation of V_T's. Low γ in TSOS process results in $V_T \approx V_{TO.}$ In regular orbit process, V_T of transistors M5, M6 and M8 would be of different values and the circuit may fail.

Two quadrant operation can be achieved by adding PMOS transistors to the circuit in Figure 20. Depending on the polarity of the input voltage, the N and P transistors would conduct. Symmetry in two quadrants is maintained by the proper selection of device geometries in respective parts.

Transistors M11-14 act as mirroring transistors, transferring the drain current of M9 and M10 to the output load. The linear and saturation regions of M9 and M10 almost approximates the Sigmoid function.

A family of curves can be generated by adjusting the value of the control voltage V_C, varying the gain of the circuit. The transitors M16-M22 reduces the steady state power dissipation.

<u>3.2.5.3.1 Simulations</u> The D.C. and A.C. characteristics of the sigmoid function are shown in Figure 21 & 22 respectively. The symmetry of the D.C. curve is maintained by matching the n-half and p-half of the squashing circuit.

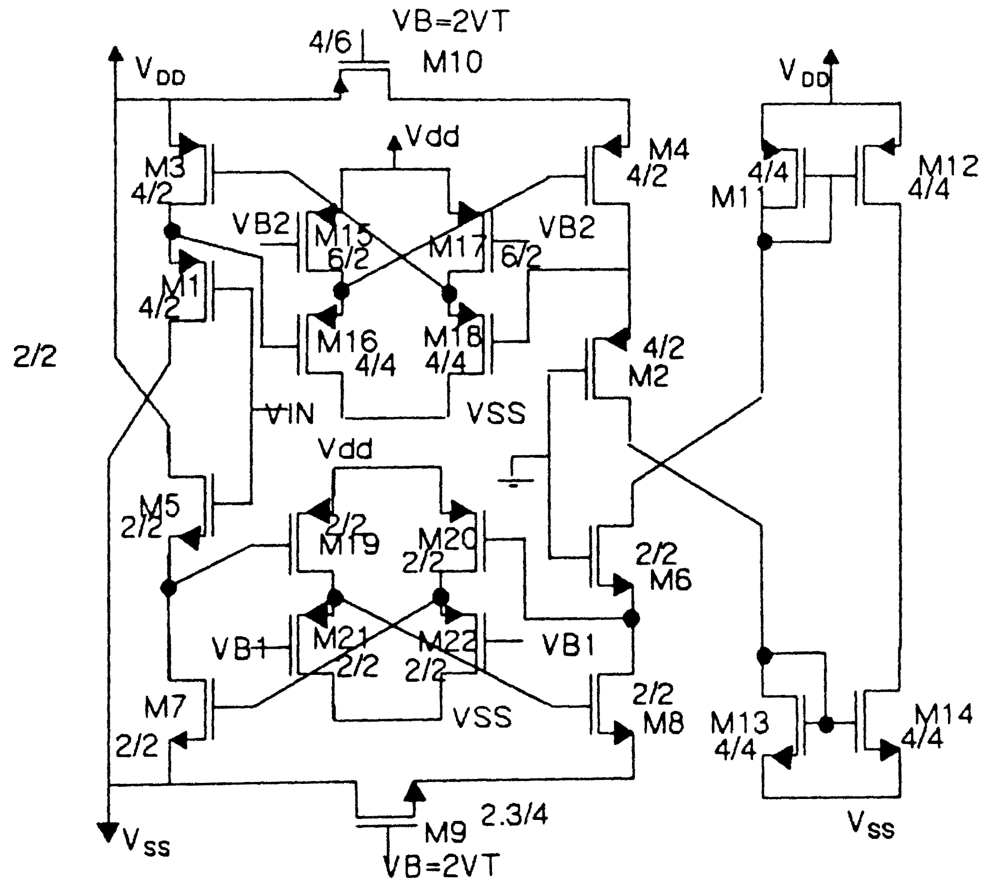


Figure 20. Sigmoidal Nonlinearity (Two Quadrant)

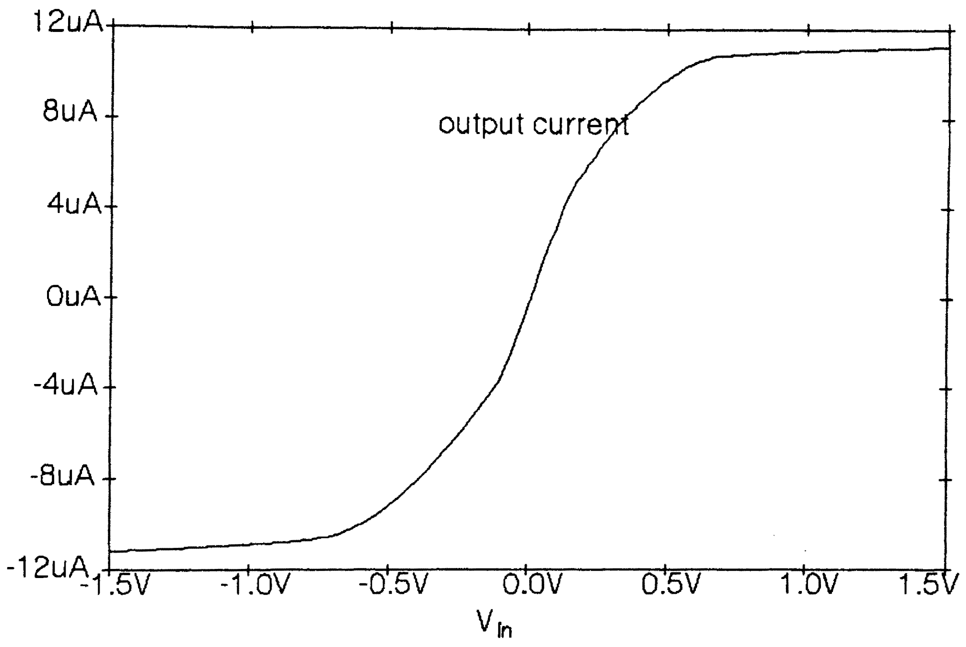


Figure 21. D.C. characteristics of sigmoidal circuit

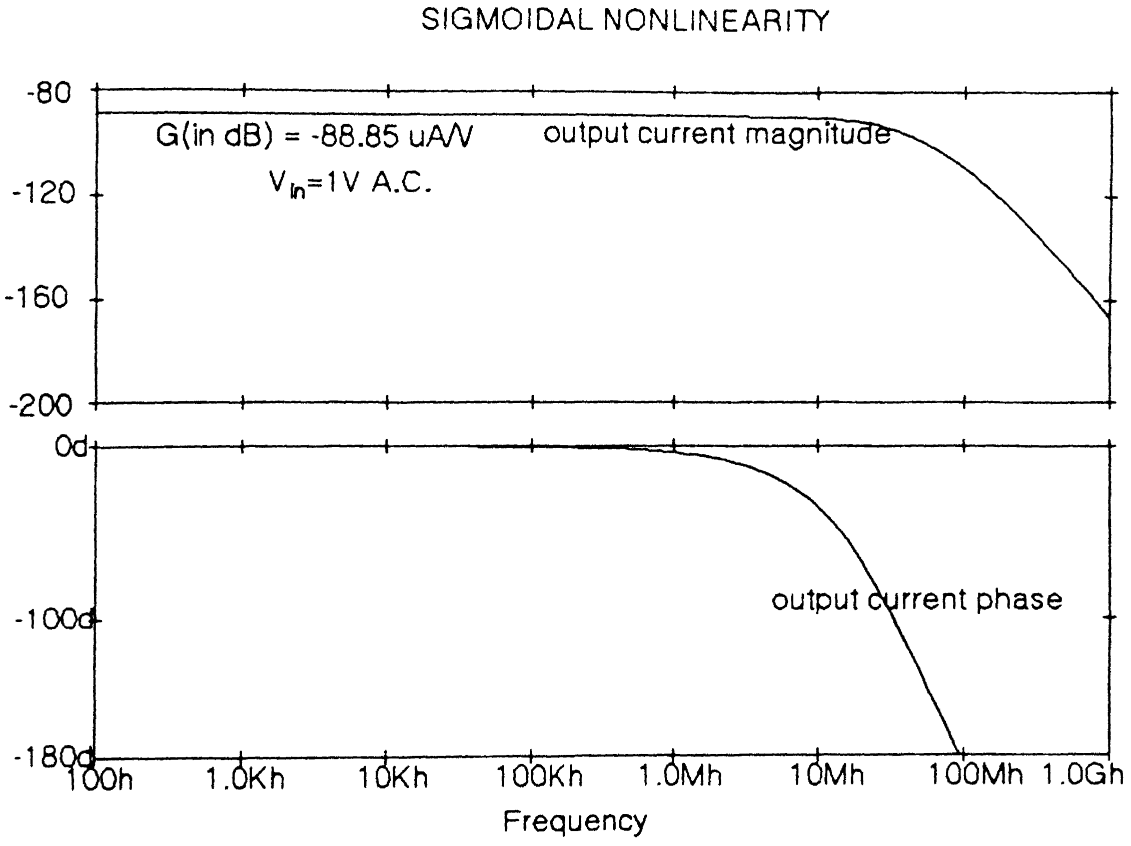


Figure 22 A.C. characteristics of sigmoidal circuit

3.2.6 Dynamic cascode biasing

The requirements for the current mirrors to be used with all the current conveyors are : linear current gain, high output impedance, wide output voltage swing, small input bias voltage, and a good high frequency response. The ability to satisfy the requirements depends on the type of current mirror chosen [45]. There are basically five types of current mirrors in CMOS technology [45]: simple current mirror, cascode or stacked current mirror,Wilson current mirror, Improved Wilson current mirror and cascode current mirror with improved biasing. The tradeoff can be a high output impedance and good current conveying capacity for a reduced output swing. Simple current mirrors have poor mirroring accuracy and low output impedance but have large output voltage swing. The stacked or cascode configurations suffer from reduced output swing but have high output impedance and good accuracy.

The current conveyor is at the heart of this building block approach. A simple current mirror will not achieve the required current conveying accuracy. Therefore cascode mirrors were used in all the circuits. The cascode mirrors have the disadvantage of reduced output swing. But since our crucial requirement is an accurate current transfer ratio, we selected cascode mirrors with dynamic biasing circuit as shown in Figure 23.

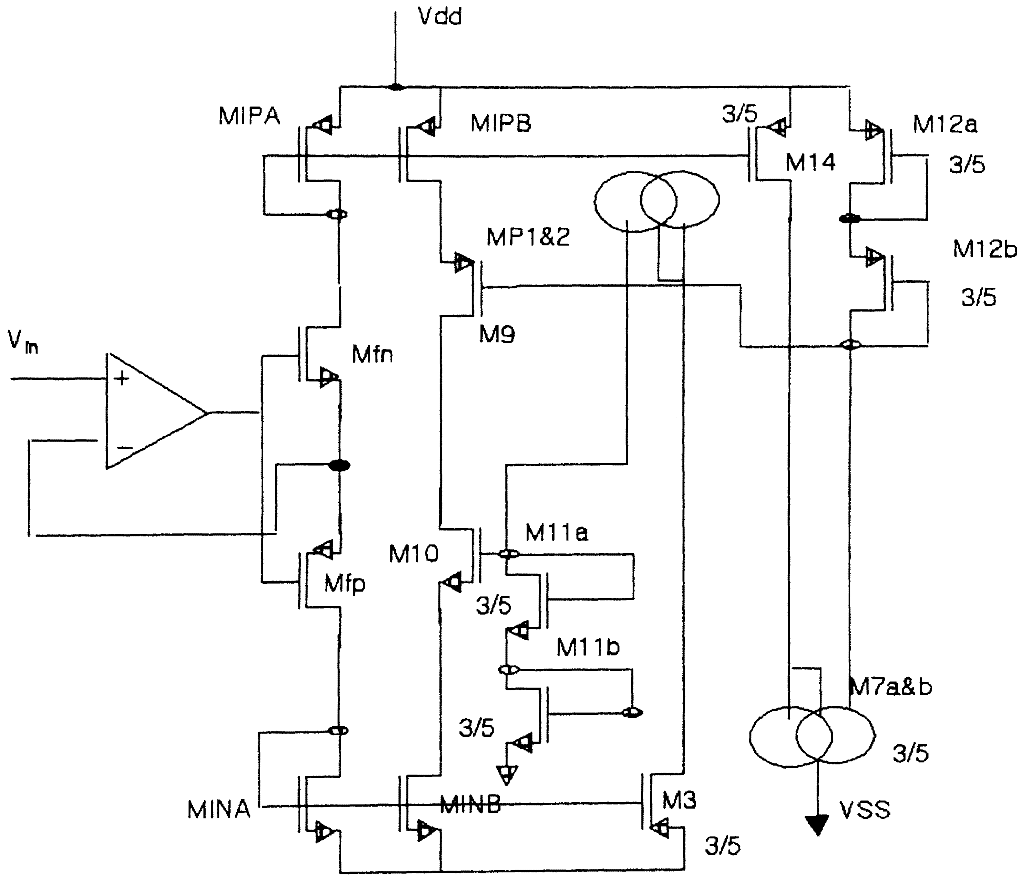


Figure 23. Dynamic cascode biasing circuit

The transistor M3 is used to mirror a portion of current that flows through the cascoding transistor M1NB. It is mirrored by the P-current mirrors. The λ effect is not completely eliminated because the mirroring (P-mirrors) are not 100% accurate, and there is a λ effect at junction of MP1 and M3. The λ effect on the mirrors and M3 node can be reduced by making the lengths of all the cascode biasing transistors large. Further improvement is possible by cascoding MP1 and MP2 as well as M3.

3.3 Derivative Circuits

The derivative circuit of the linear saturating curve is shown in Figure 24. The backpropagtion vector consists of the derivative of the output vectors of the previous stage. So the derivative circuit must accomplish multiplying the output vector with the function value. The approximate sigmoid circuit derivative circuit acts as a transconductance transferring the input voltage to an output current. The derivative of that output current can be achieved as a peicewise linear derivative function. It needs another complex circuitry to multiply the derivative current and the output voltage.

The linear derivative circuit has two impedance states - low and high. The low impedance state indicates slow learning, while the higher impedance state indicates high

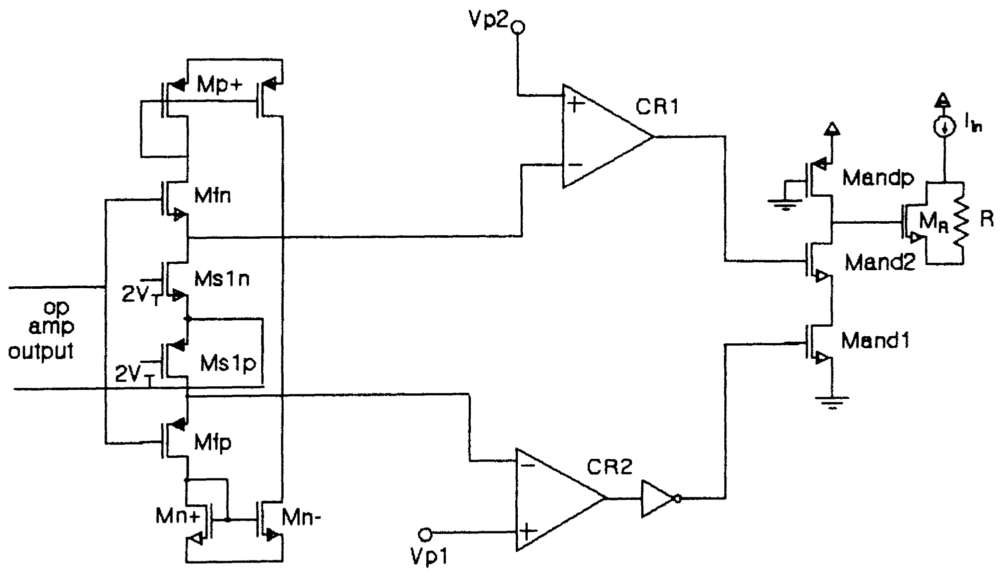
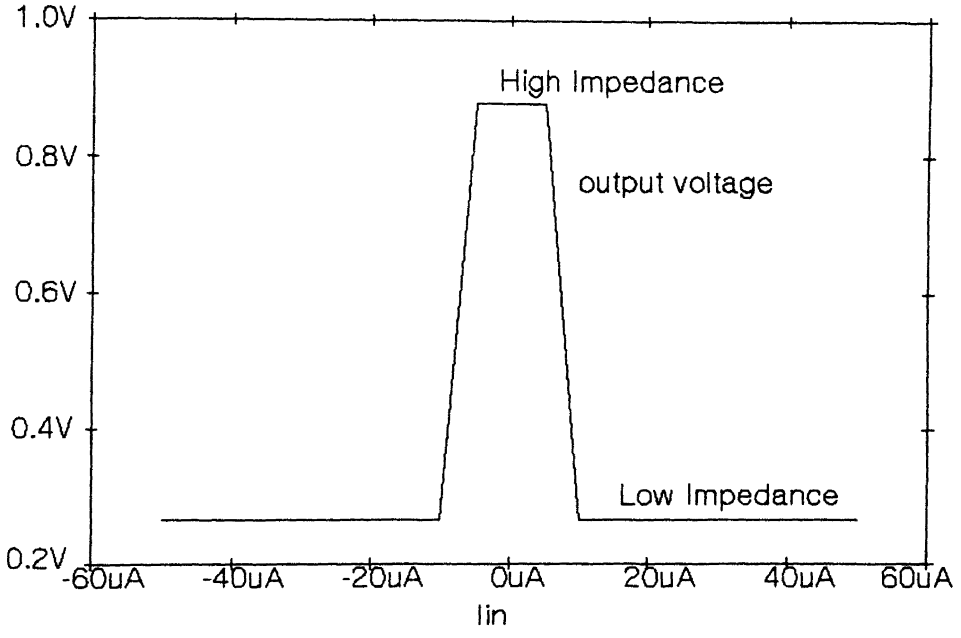


Figure 24. Derivative Circuit



D.C. characteristics of Derivative circuit learning. The derivative circuit is shown in Figure 24. As soon as the drain voltage of the transistor MS1N becomes V_T, it saturates. This point corresponds to the knee of the linear saturating curve, when the transistor MS2N takes over conduction from MS1N. The comparator CR1 switches from low to high state, which drives the output of the nor gate to high output state. Since the gate of load transistor ML is connected to the output of the "nor" gate, it starts conducting and the current I_{in} gets a low resistance or high conductance path. The output voltage goes from high impedance state to low impedance state. Similar actions occur in the p-half as the transistor MS1P saturates, and the output goes from the high impedance state to the low impedance state. This low state will drive the corresponding delta vectors to low values. The reduced value of delta vectors will prevent learning, because the output vectors in that layer exceeds a certain value, or is saturated. Two types of comparators (CR1 and CR2) are used to trip at two different saturation points of MS1N and MS2N [56].

3.4 Weight Matrix

Neural networks "learn" by modifying weights (synapses). The weights must be alterable and should take a wide range of positive and negative values. The incremental weight changes should be small [33]. If continuous weight values are used, then there is a need to store these values. This storage requirement imposes a quantization effect either because of digital storage and A/D converters, or by using analog storage and countering the effect of noise. The effective noise in the system determines the dynamic range of analog values that can be stored and retrieved. i.e. the resolution. An even more stringent requirement is the development of high density storage medium which is readily accessible in IC form. Digital storage with A/D and D/A's will not have the required chip density. Therefore analog storage is the solution [34].

There are variety of a methods of producing analog storage. Capacitors and integrators will allow the stored charge to degrade too fast. In pure analog storage there is no noise margin, and, hence no possibility of signal restoration. An analog signal can only be maintained, with memory decay, and the design objective is to maintain the signal as long as necessary.

An analog memory element can be characterized by: (1)location, on-chip or off-chip (2) volatility, volatile or nonvolatile (3) programming/erasing method, electrical or non-electrical, and (4) the precision in bits.

Storage of analog weights necessitates, 1) truly non-volatility, for long term retention of the stored knowledge,

2) on-chip and rapid programmability, to expedite the network learning by minimizing read and write times, and 3) application specific yet simple, for ease of fabrication, analog memories. Discrete programming of true analog memories results in finite resolution, usually specified in bits.

Several analog memory designs have been presented in the literature [36]. Furman and Abidi [18] presented a feed forward network with back error propagation. The weights are stored as charges on capacitors on the nodes at cryogenic temperatures. Card and Schniedel have used capacitors with positive or negative charges with periodic refreshing using training data. Bibyk et. al. [37] used floating gate MOS transistors to store charges. Hubbard, Schwartz and Howard [29] introduced a circuit utilizing dynamic charge storage on MOS capacitors. Hoecht et. al. [38] presented a method in which a finite number of charge levels can be stored on a MOS capacitor. These charge levels are preserved by a sense circuitry and regular refreshes. Additional analog designs [39-42] are also present in the literature.

The favorable learning feature of the TRIT model is that the weights are varied in parallel and across the whole network. This eliminates the need for a complex circuitry to locate the weights. As the magnitude of the weight changes are predetermined, the weight modification is further simplified. The floating-gate analog semiconductor memories has been proposed by a number of researchers [43] as a suitable analog medium for the long-term storage of the weights. Y.Tsividis and S.Satyanarayana [44] had suggested storing analog voltages at the gate capacitance of the MOS transistor itself. The inherent non-linearity of a transistor can be cancelled by using complementary input voltages through the matched weighing transistor, or by passing the same voltages through the complementary weighing transistors: the n-channel and the p-channel. Learning takes place by addressing the proper capacitors and charging them according to a specified learning algorithm. Once the MOS weights have settled (RC time constant), the capacitors are periodically accessed for reading, charging and refreshing. This scheme suffers from a relatively short retentivity resulting in decreased accuracy. As a result, the network becomes "absent minded", forgetting information shortly after learning.

3.4.1 Floating gate analog (FGA) memories

Floating gate analog memories are alterable and non-volatile. They provide local on-chip weight storage on the floating gate of a transistor. It is small, consumes less power, has slow memory decay and is compatible with standard fabrication processes. The extra gate layers are used to store trapped charges on a floating gate. Once trapped these charges produce a shift in the gate to source voltage which varies the current through the transistor. This type of memory element exhibits long term retention because no discharge path is available since the gate is surrounded by the dielectric material SiO_2. This memory transistor is operated in the triode region where non-linearity of the transistor is fairly low. Usually depletion devices are used to eliminate the floating bias.

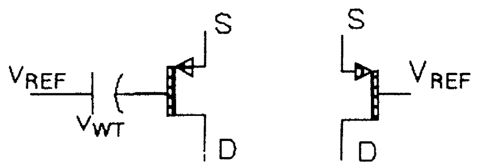
The charge on the floating gate of a transistor represent the value of the weight. As the network learns, the strength of the synapse increases. That is the electrical equivalent of dumping more charge on the floating gate, i.e., programming and thereby modulating the electrical conductivity of the synapse (PMOS). Thus during programming, the electrical conductivity of the synapse is expected to increase. The P-sense transistor was specifically chosen to achieve this desired operation. While programming, the floating gate acquires electrons which develop a negative potential on the floating gate of the P-MOS sense transistor. The floating gate voltage tends to become more negative as programming proceeds. Therefore, the drain current through the device increases, i.e., conductivity increases.

Until recently, the memories discussed above required a special fabrication process such as ultrathin window, nitrite trap oxide, or a conventional textured polysilicon. Usually, these special processes are expensive, immature. and simply not available in many design environments, especially universities. In order to fulfill the need of an analog neural network designers for programmable memories, existing standard CMOS process without modifications had to be improvised to provide a solution to realize floating gate memories. Recently several such implementations have been reported [45-46]. The interested reader is referred to the earlier work in this field by S.Patil [47].

A number of these floating gate analog memories can be interconnected suitably to form a weight matrix structure. The same weight matrix can be used in both forward and back propagation increasing both density and yield.

Current summing is used for the common analog computation of the inner product of the weight vector and the input vector. Current summing offers more dynamic range, which is of importance in signal processing applications. The linearity is due to summing of the non-linear elements currents of the transconductor into a virtual ground. The common mode nonlinear terms are eliminated, while the difference currents develop an inner product computation with wide dynamic range. The compact method of multiplying for inner product uses a single Transistor per cell.

From the Figure



$$I_{D1} = \beta[(V_{GS} + V_{WT} - V_T)V_{DS} - V_{DS}^2/2] \qquad I_{D2} = \beta[(V_{GS} - V_T)V_{DS} - V_{DS}^2/2]$$

$$I_O = I_{D1} - I_{D2}$$

$$= \beta V_{wt} V_{ds}$$

Figure 26. Principle of Weight multiplier

$$I_{D1} = \beta((V_{GS}-V_T+V_{WT})V_{DS}-V_{DS}^2/2) \quad (27)$$

$$I_{D2} = \beta((V_{GS}-V_T)V_{DS}-V_{DS}^2/2) \quad (28)$$

The output current as shown in Figure 26 is

$$I_O=I_{D1}-I_{D2} = 2*\beta*V_{WT}*V_{DS} \quad (29)$$

The TRIT backpropagation IC consists of this analog EEPROM weight matrix surrounded by current conveyors.

3.4.2 Weight Adjustment Circuit

The weight adjustment circuitry is shown in Figure 27 & 28. The circuit implements the TRIT algorithm based on the values of the delta and output vectors. The comparator CH1 switches from high to low state, if the input delta value exceeds ϵ_2. CH2 switches from high to low, if the value of delta vector becomes less than ϵ_2. The switching of either comparators results in a high state latched in latch (N9 and N10). The complemented output of the latches is fed to the transmission gate, which is clocked. The latch states are "OR"ed, clocked and fed to a NOR gate. The other input of the NOR gate is the strobe (STR) control signal. A high state at any of the latches is translated to an INC signal

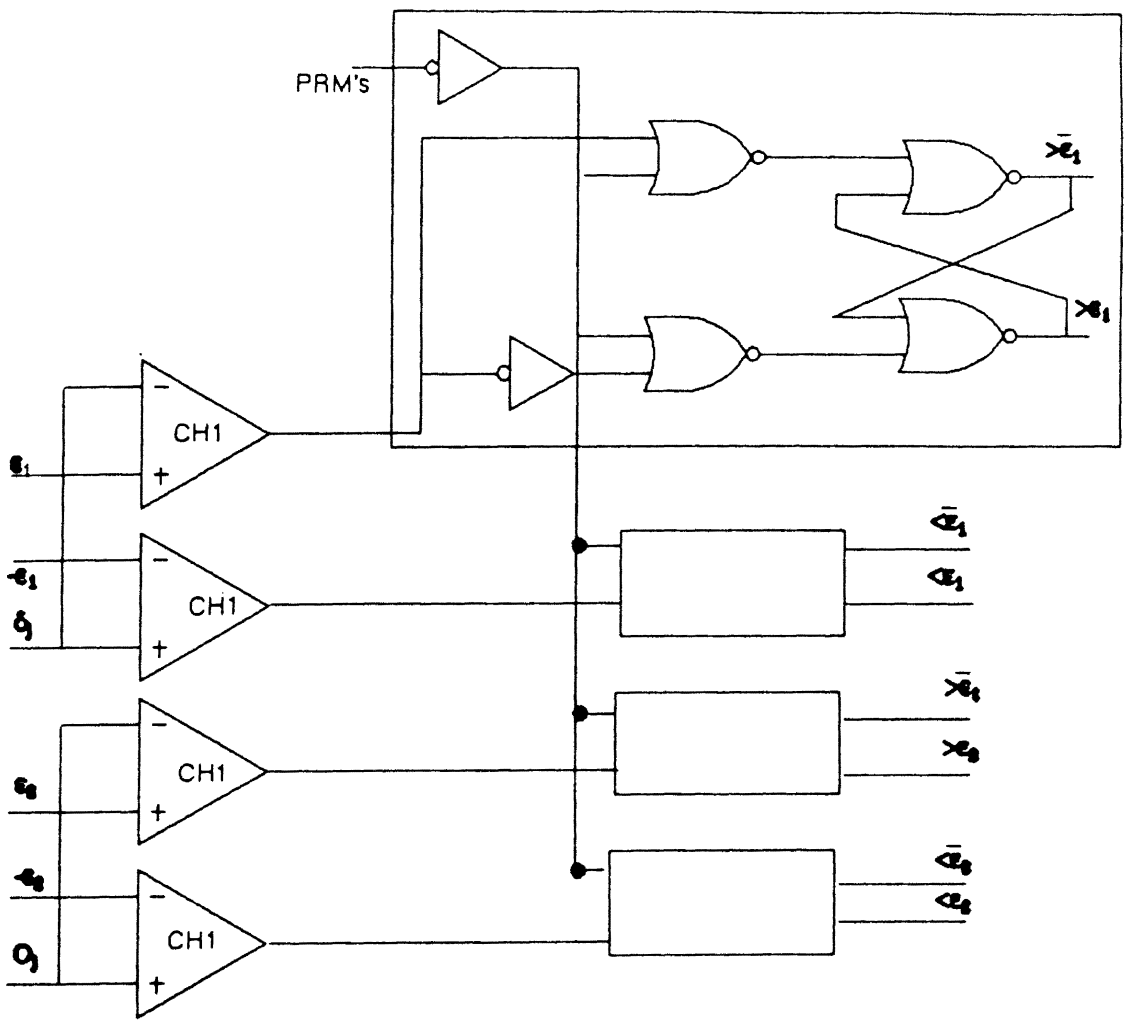


Figure 27. Weight Adjustment Circuitry-1

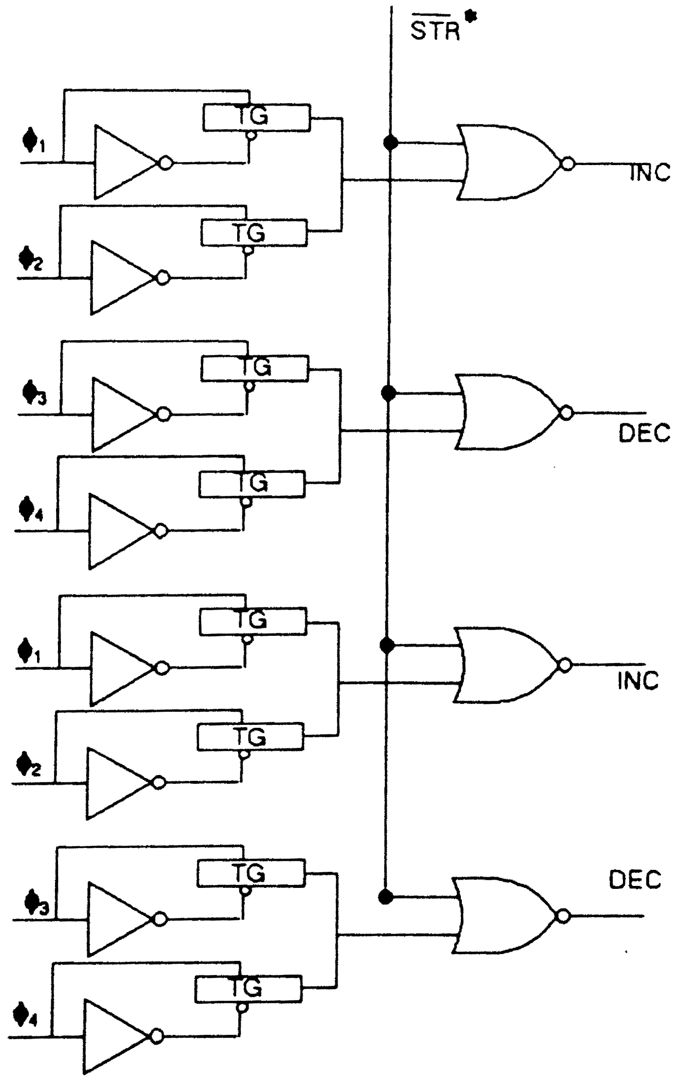


Figure 28. Weight Adjustment Circuitry-2

being high.

Similar actions take place for the comparator CH2's output, which results in DEC signal being high, if the delta vector is less than ϵ_2. Similar logic can be implemented for the output (O) vectors.

3.5 Sample and Hold (S/H) circuit

3.5.1 Introduction

The dynamic current copier (current self-calibrating circuit, and dynamic current mirror, etc.) is used. The gate capacitor of the MOS device is used to store the information for a short period of time since the gate of a MOS device practically has infinite input impedance. Figure 29 shows the basic N-copier cell. To sample the input current, switches S_1 and S_2 are closed. The gate capacitor C_{GS} of M_1 will charge to voltage V_{GS} required by the transistor to achieve the drain current I_O. If M_1 is in saturation, the gate voltage is given by:

$$V_{GS}=\sqrt{\frac{2I_O}{\beta_1}}+V_{T1} \tag{30}$$

The switch S_1 and S_2 are opened successively. The circuit goes into hold phase and stores the current information as the capacitor voltage in the gate of M1.

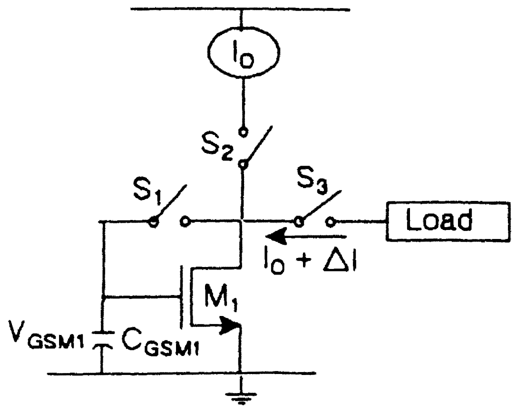


Figure 29. Basic Sample and Hold Circuit

Since the gate voltage of M1 is coupled to transistor M2 and M3, an equivalent current can sink through M2 at the hold phase. The P-copier cell can be achieved by replacing the NMOS with a PMOS transistor, and by reversing the direction of currents. In such a case, the cell will source I_0 when connected to the load. The copier cells need not be accurately matched. An error current (ΔI) is present due to: (1) charge sharing between the gate capacitor C_{GS} and switch capacitor C_{GSSW}. (2) channel length modulation parameter (3) junction leakage associated with S_1, causing a steady discharge of the storage capacitor,

The minimum dimension switches, M_{SP} and M_{SN} are used to reset the gate voltage or hold capacitors. A dummy switch can be added in series with the switching transistor to further minimize the effect of charge sharing. The channel length modulation error is reduced by cascoding the current sampling and holding transistors. Dynamic biasing of these cascode transistors gives improved cascoding and improved current transfer ratio. M_{CN}, M_{IPA}, M_{IPB}, M_{RN} are used for dynamic biasing circuitry .

3.5.2 Errors in S/H circuit

3.5.2.1 Charge Injection Error

The switching transistor is made conductive by mobile carriers that are attracted into the channel by the gate voltage during its closing. For charge equilibrium, the total charge of the mobile carriers in the channel must be equal to the total charge stored on the gate. The charge is stored on the gate in strong inversion. In N-copier when switch S_1 opens, a fraction Δq of q is dumped on the capacitor C_{GS1}, which causes an error in the stored voltage. This voltage error (ΔV) in turn creates a relative error in the output current of the copier. ΔV can be decreased by making the switch gate oxide capacitance a small percentage of the C_{GSN} where one limit is given by the area of the C_{GSN}. It can also be decreased by reducing the total charge q in the channel which in turn reduces the fraction Δq that flows onto C_{GSN}. This can be achieved by minimizing the gate area WxL and/or by controlling the gate voltages of the switch or increasing V_{GSN}. Similar treatment applies to the P-copier for determination of the error due to the charge injection. The factor Δq determines the amount of charge that is dumped on the source.

3.5.2.2 Switch Feedthrough Error

This contribution is due to the clock voltage that is coupled to the gate via C_{GD}. The clock voltages is partially transferred to the gate via the capacitive network as,

$$\Delta V = \frac{C_{GD}}{C_{GD}+C_{GSN}} V_{\Phi_2} \qquad (31)$$

where $V_{\Phi 2}$ is the clock voltage and C_{GD} is gate to drain capacitance of the transmission gate. The change in the gate voltage multiplied by the transconductance reflects an error in the drain current. This error is reduced by connecting a dummy transistor in series with the switching transistor [47].

3.5.2.3 Cascode Configurations

The contribution due to the channel length modulation produces change in the drain current as the drain to source voltage changes. The λ effect is reduced by cascoding the transistors using a regulated cascode structure as shown in Figure 30.

3.5.3 Simulations

The transient simulation of the sample and hold circuit is shown in Figure 31. The output current is out of phase with the input current. The output current is the sampled and stored value of the input current.

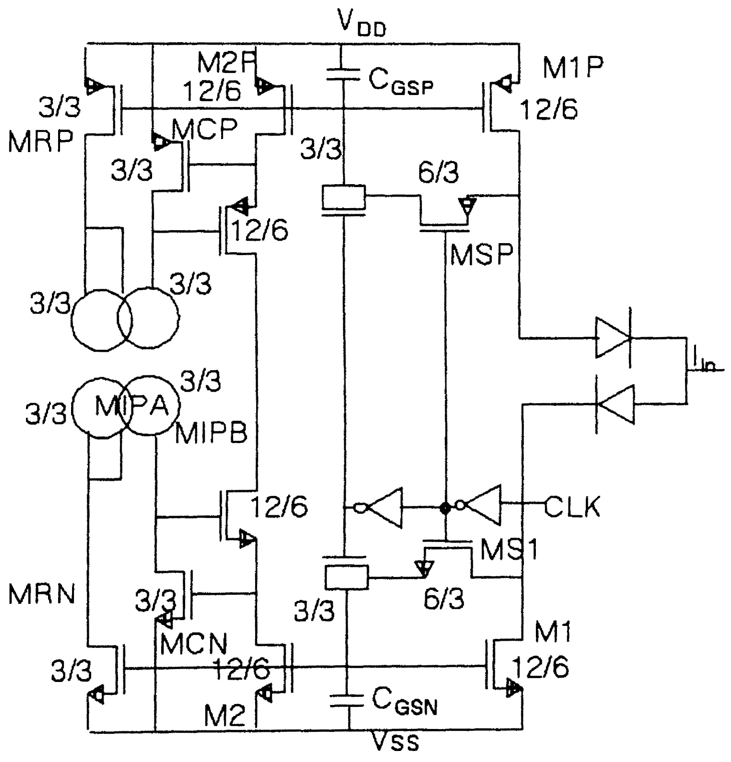


Figure 28. Sample and Hold Circuit

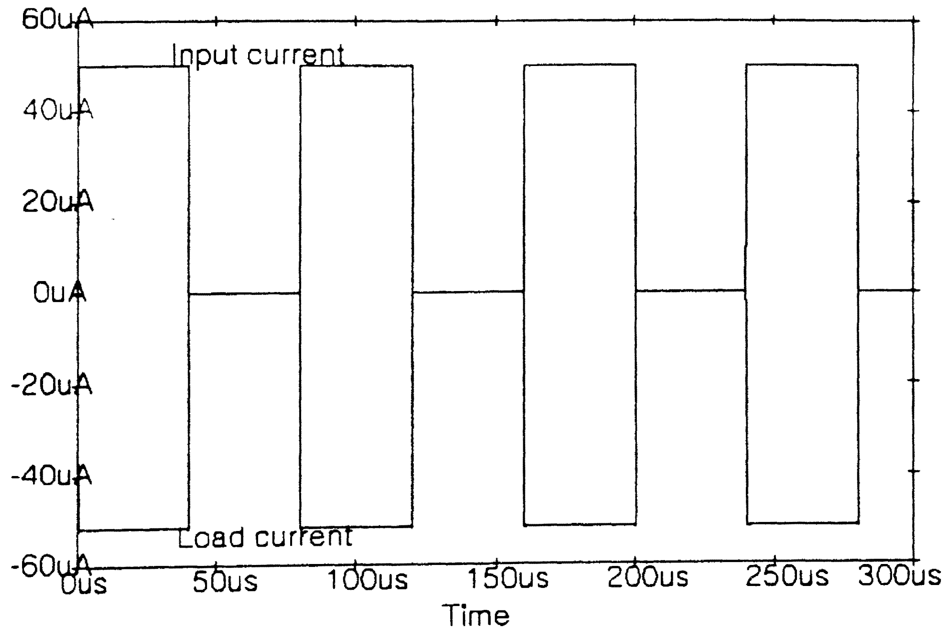


Figure 31. Transient Characteristics of S/H circuit

CHAPTER IV

CONCLUSIONS AND FUTURE PROSPECTS

The design of the basic building blocks for the TRIT algorithm in (TSOS) process is completed. A single weight matrix is used in both the forward and backpropagation mode resulting in reduction of area by two. A Matlab program which partially simulates the transistor mismatches in this architecure was also developed. The TRIT program with the transistor imperfections demonstrate faster convergence than BP and insensitivity to MOS parameter variation.

The system level integration of the TRIT model will require the exact specification of all the system parameters. The optimal values of learning rate, ϵ_1, and ϵ_2 should be investigated. The forward propagation parameters like I_{SAT}, I_{WT}, V_O and the backpropagtion parameters like δ_i's, R_L and R_H should be specified at the system level.

The fabricated blocks have to be tested thoroughly to test their effectiveness and further refined. The high power current conveying transistors are very large. The derivative circuit can be further improved.

Floating gate memories provide the best answer to electrically programmable/erasable non-volatile semiconductor memories. Reduction in cell size, improvement in performance, and circuit density will be the products of the floating gate memories research. So future developments in FGA memories have to be followed closely to be adopted for our design.

For effective learning, local or on-chip storage and modification of the weight is the preferred solution. The task of weight updates is complex since it involves issues related to high voltage, learning algorithm, and weight storage. Precise control of the weight needs extensive experimentation to mathematical model and understand the programming and erasing behaviors of Floating gate memories.

The on-chip generation of high voltage poses an additional challenge. However, the tunneling physics and high voltage pulse generation are two separate issues and initially should by handled separately for conceptual testing and understanding, and then should be combined together. Other issues relating to the weight matrix are cell layout, placement, and signal routing. Cell layout will have a direct impact on both the silicon area as well as on the cell performance. Significant expertise is required to arrive at the optimal design. A suitable signal routing scheme is required since the weight matrix is expected to be dominated by routing wires. In this regard, high voltage concerns such as field threshold, reverse breakdown etc. need special attention.

The process maturity will play an very important role in TRIT design. Also better analog simulation tools which represents the transistor more excatly has to be used to further improve the design efficiency.

BIBLIOGRAPHY

1. Rumelhart, D.E.Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by Error propagation, Parallel Distributed Processing, Cambridge, MA, MIT Press, pp. 318-362

2. Richard P. Lippman, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April 1987, pp. 4-22

3. Derek B.I.Feltham, and Wojciech Maly, "Physically Realistic Fault Models for Analog CMOS Neural Networks," IEEE Transactions on Neural Networks, 1991, pp. 1223-1230

4. Bernhard E. Boser, Eduard Sackinger, Yann Le Cun, Lawrence D. Jackel, "An Analog Neural Network Processor with Programmable Topology," IEEE Journal on Solid State Circuits, December 1991, pp. 2017-2024

5. Karl Goser, Ulrich Hilleringmann, Ulrich Ruekert, and Klaus Schumacher (1989). VLSI Technologies for Artificial Neural Networks, Dec 1989, pp. 28-43

6. Shoemaker, P.A., Shimabukoro, R , and Michael J.

Carlin (1991), "Back Propagation Learning with Trinary Quantization of Weight Updates," Neural Networks Vol. 4, pp. 231-241

7. K.A.Boahen, R.E.Jenkins et.al,"A Heteroassociative Memory Using Current-Mode MOS Analog VLSI Circuits," IEEE Transactions on Circuits and Systems, 1989, vol.36, pp. 747-755

8. S.W.Tsay and R.W.Newcomb, "VLSI Implementation of ART1 Memories," IEEE Transactions on Neural Networks," vol.2, 1991, pp. 214-221

9. A.F.Murray, D.Del Corso, and L.Tarassenko, "Pulse-Stream VLSI Neural Networks Mixing Analog and Digital Techniques," IEEE Transactions on Circuits and Systems, vol.36, 1989, pp. 193-204

10. B. Linares, E. Sanchez, A.Rodriguez, and J.L.Huertas, "Modular Analog Continuous-Time VLSI Neural Networks with on Chip Hebbian Learning and Analog Storage," IEEE Transactions on Neural Networks, 1992, pp. 1533-1536

11. Simon Y. Foo, Lisa R. Anderson, Yoshigasu Takefuji, "Analog Components for the VLSI of Neural Networks," Circuits and Systems, 1990, pp. 18-25

12. C. Toumazou, F. J. Lidgey, and D. G. Haigh, Analogue IC Design: The Current-Mode Approach, Eds., Peregrinus, London, 1990

13. Christian Schneider and Howard Card, "CMOS

Implementation of Analog Hebbian Synaptic Learning Circuits," IEEE Transactions on Neural Networks, 1991, pp. I437-I442

14. Robert C. Frye, Edward A. Rietnam, and Chee C. Wong, "Back-Propagation Learning and Nonidealities in Analog Neural Network hardware," IEEE Transactions on Neural Networks, 1991, pp. 110-117

15. S. Espejo, A. Rodriguez et. al, "Switched-Current Techniques for Image Processing cellular neural networks in MOS VLSI," IEEE Transactions on Neural Networks, 1992, pp. 1537-1540

16. Jackel, L.D, Graf, H.P., and Howard, R.E.,"Electronic Neural Network Chips," Applied Optics, 1987, pp. 5077-5080

17. Alspector, J., Allen, R.B.Hu,V., & Satyanarayana, S. (1988), "Stochastic learning networks and their implementation," Proceedings of the IEEE conference on the Neural Information Processing Systems. pp. 9-21

18. Furman B., and Abidi. A, "CMOS Analog IC implementing the back propagation algorithm, "First Annual Meeting, (Abstract) Neural Networks, 1988, pp. 38

19. Shoemaker, P.A., and Shimabukoro, R (1988), "A modifiable weight circuit for use in adaptive neuromorphic networks," Neural Networks, 1, Sup. 1,409

20. Hu V. Kramer A., and Ko. P. K., "EEPROM's as analog storage devices for neural nets," Neural Networks,1988, pp. 385

21. Shimabkuro R.L., Shoemaker, P.A, and Astewart M., "Circuitry for artificial neural networks with non-volatile analog memories," Proceedings of the IEEE Symposium on Circuits and systems, 1989, pp. 1217-1220

22. Holler,M.,Tam, S.Castro,H., and Benson, R., "An electrically trainable artificial Neural network (ETANN) with 10240 "floating gate" synapses," Proceedings of the IEEE Joint Conference on Neural Networks, 1989, pp. 177-182

23. Alan F. Murray, and Anthony Smith, "Asynchronous VLSI Neural Networks using Pulse-Stream Arithmetic," IEEE Transactions on Neural Networks, 1988, pp. 688-697

24. Peterson, C., & Hartman, E. (1989), "Exploration of the mean field theory learning algorithm," Neural Networks, 2, pp. 475-494

25. M.Marchesi, G.Orlandi et.al.,"Multi-layer Perceptrons with Discrete Weights", Proc. IEEE ISCAS 1990, New Orleans, pp. 623-629

26. Bernd Hofflinger, Stefan Neuber et. al, "VLSI Implementation of a Neural Car Collision Avoidance Controller," IEEE Transactions on Neural Networks,

191, pp. I493-I499

27. C.A. Mead, Analog VLSI and Neural Systems, Addison Wesley Publishing Co. Inc., 1989

28. Martin Hagan (1992), Back Propagation Class Notes on Neural Networks (ECEN 5050.3), 1991

29. Kurosh Madani, Patrick Gadra, Eric Belhaire, and Francis Devos, "Two analog Counters for Neural Network Implementation," IEEE Transactions on Neural Networks, 1991, pp. 966-973

30. Paul Hasler and Lex Akers, "Circuit Implementation of Trainable Neural Networks Employing both Supervised and Unsupervised Techniques," IEEE Transactions on Neural Networks, 1992, pp. 1565-1568

31. W.R.Smith, "Trinary Back-Propagation Simulation with Component Nonidealities," NOSC Newsletter

32. Randy L. Shimabukuro, Pat Shoemaket et. al., "Effects of Circuit Parameters on Convergence of Trinary Update Back-Propagation"

33. Daniel B. Schwartz, Richard E. Howard, and Wayne E. Hubbard, "A programmable Analog Neural Network Chip," IEEE Transactions on Neural Networks, 1989, pp. 313-319

34. J.Raffel, J.Mann et. al., "A generic architecture for wafer-scale neuromorphic systems," IEEE Conference of Neural Networks, Vol. 3, 1987, p.501

35. P. A. Shoemaker, C. G. Hutchens and, S. B. Patil, "A Hierarchical Clustering Network Based on a Model of Olfactory Processing," Submitted, 1992

36. Aria Nostrinia, M. Ahmadi, M. Sridhar, G.A. Julien, "A hybrid Architecture for multi-layer Neural Networks, IEEE Transactions on Neural Networks, 1992, pp. 1541-1544

37. T.H.Borgstorm, M.Ismail, and S.B. Bibyk, "Programmable current mode network for implementation in analogue VLSI," IEEE Proceedings on Neural Networks, 1990, pp. 75-84

38. B.Hochet, V.Peiris, S.Abdo et. al., "Implementation of a learning Kohonen neuron based on a new multilevel storage technique," IEEE Journal on Solid State circuits, 1989, pp. 262-267

39. P. Mueller et. al, "A general purpose analog neural computer," IEEE Second International Conference on Neural Networks, 1988, pp. 177-182

40. H.P.Graf and L.D.Jackel, "Analog Electronic Neural Network Circuits," IEEE Circuits and Devices Magazine,July 1989, pp. 44-49

41. E.A.Vittoz, "Analog VLSI Implementation of Neural Network," Proceedings of International Symposium on Circuits and Systems, 1990, pp. 2524-2527

42. F.M.A. Choi et. al, "An All-MOS analog Feedforward Neural Circuit with Learning," Proceedings of

International Symposium on Circuits and Systems, 1990, pp. 2508-2511

43. R. L. Shimabukuro, and P. A. Shoemaker, "Circuitry for Artificial Neural Networks with Nonvolatile Analog Memories," Proceedings, IEEE International Symposium on Circuits and Systems, pp. 1217-1220, 1989

44. Y. Tsividis, and S. Satyanarayana, "Analog Circuits for Variable-Synapse Electronic Neural Networks," Electronic Letters, Vol. 23, No. 24, pp. 1313-1314, November 1987

45. L. R. Carley, "Trimming Analog Circuits Using Floating-Gate Analog MOS Memory" Circuits, Vol. 24, No. 6, pp. 1569-1575, December 1989

46 B. W. Lee, B. J. Sheu, and H. Yang, "Analog Floating-Gate Synapses for General-Purpose VLSI Neural Network Computation," IEEE Transaction on Circuits and Systems, Vol. 38, No. 6, June 1991

47. S.B.Patil, "VLSI Design of olfactory Network," Master's Thesis, Oklahoma State University, 1992

48 C. Toumazou, J. Lidgey, and D. Haigh "Introduction," Ch. 1 in Analogue IC Design: The Current-Mode Approach, C. Toumazou, F. J. Lidgey, and D.G. Haigh, Eds., Peregrinus, London, 1990

49. K.C.Smith and A.S.Sedra, "The current conveyor - a new

circuit building block," Proceedings of IEEE, vol 56, pp. 1368-1369, Aug 1968

50. A.S.Sedra and K.C.Smith, "A second-generation current conveyor and its applications," IEEE Transactions on Circuit Theory, Vol CT-17, pp. 132-134, Feb 1970

51. A.S.Sedra, "A new approach to active network synthesis," Ph.D Thesis, University of Toronto,1969

52. A. S. Sedra, and G. W. Roberts, "Current Conveyor Theory and Practice," Ch. 3 in Analogue IC Design: The Current-Mode Approach, C. Toumazou, F. J. Lidgey, and D. G. Haigh, Eds., Peregrinus, London, 1990

53. Paul R.Gray and Robert G.Meyer, analysis and design of Integrated Circuits,(2nd Edition), Wiley, NY 1984

54. S. B. Patil, and C. G. Hutchens, "A Novel Squashing Function for Electronic Implementation of Neural Networks," 5th Oklahoma Symposium of Artificial Intelligence, 1991

55. Y. Tsividis, Operation and Modeling of MOS Transistor,

56. P. E. Allen, and D. R. Holberg, "Two Stage Comparators," Ch.7, in CMOS Analog Circuit Design, HRW Inc., 1987

APPENDIX A

STARTUP PROGRAM

```
% program to do initial calculations

     for i=1:63
          p=[p sin(8*pi/63*i)] ;
     end;

     p=p';
     t=p;

     hidden_units=64;

     for m=1:hidden_units
          for n=1:64
               for j=1:100
                    vari=rand-0.5;
                    vari=vari/25;
                    if abs(vari)<=0.02,
                         break;
                    end;
               end;
               w1(m,n)=0.5+vari;
          end;
     end;


     for m=1:hidden_units
          for n=1:64
               for j=1:100
                    vari=rand-0.5;
                    vari=vari/25;
                    if abs(vari)<=0.02,
                         break;
                    end;
               end;
               w2(n,m)=vari+0.5;
          end;
     end;

     for m=1:hidden_units
          for n=1:1
```

```
                for j=1:100
                     vari=rand-0.5;
                     vari=vari/25;
                     if abs(vari)<=0.02,
                          break;
                     end;
                end;
                b1(m,n)=vari+0.5;
           end;
      end;

      for m=1:64
           for n=1:1
                for j=1:100
                     vari=rand-0.5;
                     vari=vari/25;
                     if abs(vari)<=0.02,
                          break;
                     end;
                end;
                b2(m,n)=vari+0.5;
           end;
      end;

      for n=1:hidden_units
           for m=1:1
                for k=1:64
                     n1(n,m)=w1(n,k)*p(k,m)+b1(n,m);
                     a1(n,m)=n1(n,m);
                     if n1(n,m)>1
                          a1(n,m)=1.0;
                     end;
                     if n1(n,m)<-1
                          a1(n,m)=-1.0;
                     end;
                end;
           end;
      end;

      for m=1:1
           for n=1:64
                for k=1:hidden_units
                     n2(n,m)=w2(n,k)*a1(k,m)+b2(n,m);
                     a2(n,m)=n2(n,m);
                     if n2(n,m)>1
                          a2(n,m)=1;
                     end;
                     if n2(n,m)<-1
                          a2(n,m)=-1.0;
                     end;
                end;
           end;
      end;
```

```
e=t-a2;

sse=0;

for i=1:64
     for n=1:1
          sse=e(i,n)^2+sse;
     end;
end;
```

APPENDIX B

TRIT PROGRAM

```
    epsilon2=input('Enter the value of epsilon2:');

    learning_rate=input('Enter the value of learning
    rate:')

    epsilon1=0.33;

    errors=[sse];

    hidden_units=64;

    examp=0;

% random # of 0.02 is generated

    for n=1:64
         for m=1:hidden_units
              for j=1:100
                   vari=rand-0.5;
                   vari=vari/25;
                   if abs(vari)<=0.02,
                        break;
                   end;
              end;
              noisew1(m,n)=vari;
         end;
    end;

    for n=1:hidden_units
         for m=1:64
              for j=1:100
                   vari=rand-0.5;
                   vari=vari/25;
                   if abs(vari)<=0.02,
                        break;
                   end;
              end;
              noisew2(m,n)=vari;
         end;
    end;
```

```
% beta variation

     for n=1:1
          for m=1:hidden_units
               for j=1:100
                    vari=rand-0.5;
                    vari=vari/25;
                    if abs(vari)<=0.02,
                         break;
                    end;
               end;
               delbeta1(m,n)=vari;
          end;
     end;

     for n=1:1
          for m=1:64
               for j=1:100
                    vari=rand-0.5;
                    vari=vari/25;
                    if abs(vari)<=0.02,
                         break;
                    end;
               end;
               delbeta2(m,n)=vari;
          end;
     end;


% VT variation

     for n=1:1
          for m=1:hidden_units
               for j=1:100
                    vari=rand-0.5;
                    vari=vari/25;
                    if abs(vari)<=0.2,
                         break;
                    end;
               end;
               delvta1(m,n)=vari;
          end;
     end;

     for n=1:1
          for m=1:64
               for j=1:100
                    vari=rand-0.5;
                    vari=vari/25;
                    if abs(vari)<=0.2,
                         break;
                    end;
               end;
               delvta2(m,n)=vari;
          end;
     end;

% iteration starts

     for i=1:250
          check=sse;
          if sse<=0.1,
               i=i-1,
               break;
          end;

% calculation of deltas

          for m=1:64
               for n=1:1
                    d2(m,n)=t(m,n)-a2(m,n);
                    if abs(n2(m,n))>=1.0,
                         d2(n,m)=0;
                    end;
               end;
          end;

          d2=t-a2;
          d1=w2'*d2;

% weight & offset correction

          for m=1:hidden_units
               for n=1:1
                    if abs(n1(m,n))>=1,
                         d1(m,n)=0;
                    end;
               end;
          end;

          for n=1:1
               for m=1:hidden_units
                    if d1(m,n)>=epsilon2,
                         b1(m,n)=b1(m,n)+learning_rate;
                    end;
                    if d1(m,n)<=-epsilon2,
                         b1(m,n)=b1(m,n)-learning_rate;
                    end;
               end;
          end;

          for n=1:1
               for m=1:64
                    if d2(m,n)>=epsilon2,
                         b2(m,n)=b2(m,n)+learning_rate;
                    end;
                    if d2(m,n)<=-epsilon2,
                         b2(m,n)=b2(m,n)-learning_rate;
          end;
     end;
end;

for m=1:64
     for n=1:hidden_units
          if abs(d2(m))>=epsilon2,
               if abs(a1(n))>=epsilon1,
                    if d2(m)*a1(n)>=0
                         w2(m,n)=w2(m,n)+
                         learning_rate +
                           noisew2(m,n);
                    end;
                    if d2(m)*a1(n)<=0
                         w2(m,n)=w2(m,n)-
                           learning_rate +
                             noisew2(m,n);
                    end;
               end;
          end;
          if abs(w2(m,n))<=0.2,
               if w2(m,n)<0,
                    w2(m,n)=-0.2;
               end;
               if w2(m,n)>0,
                    w2(m,n)=0.2;
               end;
          end;
          if abs(w2(m,n))>3,
               if w2(m,n)<0,
                    w2(m,n)=-3;
               end;
               if w2(m,n)>0,
                    w2(m,n)=3;
               end;
          end;
     end;
end;

for m=1:64
     for n=1:hidden_units
          if abs(d1(n))>=epsilon2,
               if abs(p(m))>=epsilon1,
                    if d1(n)*p(m)>=0
                         w1(n,m)=w1(n,m)+
                           learning_rate +
                             noisew1(n,m);
                    end;
                    if d1(n)*p(m)<=0
                         w1(n,m)=w1(n,m)-
                           learning_rate +
                             noisew1(n,m);
                    end;
               end;
```

```
                    end;
                    if abs(w1(n,m))<=0.2,
                         if w1(n,m)<0,
                              w1(n,m)=-0.2;
                         end;
                         if w1(n,m)>0,
                              w1(n,m)=0.2;
                         end;
                    end;
                    if abs(w1(n,m))>=3,
                         if w1(n,m)<0,
                              w1(n,m)=-3;
                         end;
                         if w1(n,m)>0,
                              w1(n,m)=3;
                         end;
                    end;
               end;
          end;

% calculation of outputs

          for n=1:64
               for m=1:1
                    for k=1:64
                         n1(n,m)=w1(n,k)*p(k,m)+b1(n,m);
                         a1(n,m)=n1(n,m);
                         if n1(n,m)>1
                              a1(n,m)=1.0;
                         end;
                         if n1(n,m)<-1
                              a1(n,m)=-1.0;
                         end;
                    end;
               end;
          end;

          for m=1:1
               for n=1:64
                    for k=1:hidden_units
                         n2(n,m)=w2(n,k)*a1(k,m)+b2(n,m);
                         a2(n,m)=n2(n,m);
                         if n2(n,m)>1
                              a2(n,m)=1;
                         end;
                         if n2(n,m)<-1
                              a2(n,m)=-1.0;
                         end;
                    end;
               end;
          end;

% addition of noise
```

```
        for m=1:64
             for n=1:1
                  if a2(m,n)>-0.5,
                       a2(m,n)=(1+delbeta2(m,n))*a2(m,n);
                  end;
                  if a2(m,n)<=0.5,
                       a2(m,n)=(1+delvta2(m,n))*a2(m,n);
                  end;
             end;
        end;

        for m=1:hidden_units
             for n=1:1
                  if a1(m,n)>-0.5,
                       a1(m,n)=a1(m,n)*(1+delbeta1(m,n));
                  end;
                  if a1(m,n)<=-0.5,
                       a1(m,n)=a1(m,n)*(1+delvta1(m,n));
                  end;
             end;
        end;

        e=t-a2;

        sse=0;

% plotting of errors

        for m=1:64
             for n=1:1
                  sse=sse+e(m,n)^2;
             end;
        end;

        errors=[errors sse];

        ploterr(errors);

        hold on;

   end;
```

APPENDIX C

STANDARD BP PROGRAM

```
errors=[sse];


% iteration starts

for i=1:250

     if sse<=0.1,
          i=i-1,
          break;
     end;

% calculation of deltas

     for m=1:5
          for n=1:3
               d2(m,n)=t(m,n)-a2(m,n);
               if abs(n2(m,n))>=1.0,
                    d2(n,m)=0;
               end;
          end;
     end;

     d1=w2'*d2;

% weight & offset correction

     for m=1:hidden_units
          for n=1:3
               if abs(n1(m,n))>=1,
                    d1(m,n)=0;
               end;
          end;
     end;

     for n=1:3
          for m=1:hidden_units
               b1(m,n)=b1(m,n)+learning_rate*d1(m,n);
          end;
     end;
     for n=1:3
          for m=1:5
               b2(m,n)=b2(m,n)+d2(m,n)*learning_rate;
          end;
     end;

     for m=1:5
          for n=1:hidden_units
               w2(m,n)=w2(m,n)+d2(m)*a1(n)*learning_rate;
          end;
     end;

     for m=1:5
          for n=1:hidden_units
               w1(n,m)=w1(n,m)+p(m)*d1(n)*learning_rate;
          end;
     end;

% calculation of outputs

     for n=1:hidden_units
          for m=1:3
               for k=1:5
                    n1(n,m)=w1(n,k)*p(k,m)+b1(n,m);
                    a1(n,m)=n1(n,m);
                    if n1(n,m)>1
                         a1(n,m)=1.0;
                    end;
                    if n1(n,m)<-1
                         a1(n,m)=-1.0;
                    end;
               end;
          end;
     end;

     for m=1:3
          for n=1:5
               for k=1:hidden_units
                    n2(n,m)=w2(n,k)*a1(k,m)+b2(n,m);
                    a2(n,m)=n2(n,m);
                    if n2(n,m)>1
                         a2(n,m)=1;
                    end;
                    if n2(n,m)<-1
                          a2(n,m)=-1.0;
                    end;
               end;
          end;
```

```
end;

e=t-a2;
sse=0;

for m=1:5
     for n=1:3
          sse=sse+e(m,n)^2;
     end;
end;

%if i==1,
     check=sse*2;
end;

if check==sse,
     epsilon2=0.5*epsilon2,
end;

%if check>sse,
     learning_rate=1.07*learning_rate;
end;

%if check<sse,
     learning_rate=learning_rate/1.02;
end;

errors=[errors sse];

%plot(errors);

%hold on;

end;
```

APPENDIX D

ERRORS IN MULTIPLIER CIRCUIT

D.1 Error in multiplier

$I_{DN}=(\beta_N+\Delta\beta)[(V_1-(V_{TN}\pm\Delta V_T)-V_2^2/2]$

$I_{DP}=(\beta_P+\Delta\beta)[(V_1-(V_{TP}\pm\Delta V_T)-V_2^2/2]$

$I_O=I_{DN}-I_{DP}$

Assuming $\beta_1=\beta_2$

$I_O=2\beta V_1V_2 + 2\Delta\beta V_1V_2 \pm 2\beta\Delta V_TV_2 \pm 2\Delta V_TV_2\Delta\beta$

= Ideal term + ERROR

ERROR= $2\Delta\beta V_1V_2 \pm 2\beta\Delta V_TV_2 \pm 2\Delta V_TV_2\Delta\beta$

where the last term can be neglected

ERROR= $2\beta V_1V_2(\Delta\beta/\beta \pm \Delta V_T/V_1)$

So $I_o= 2\beta V_1V_2(1+ \Delta\beta/\beta \pm \Delta V_T/V_1)$

$\Delta\beta/\beta$=1-2%

Assuming V1≥0.5-1V

and $\Delta V_T\approx$5-10mV

ΔV_T error is 1%

D.2 Error in output [F(.)] function

Due to ΔV_T and $\Delta\beta$ errors, the output is modified. The output current is proportional to the square of the gate to source voltage. So error in transconductance due to $\Delta\beta$ and ΔV_T can be derived as follows:

$\Delta I = \beta(V_C - V_T)^2 - (\beta \pm \Delta\beta)[(V_C - V_T \pm \Delta V_T)]^2$

$\Delta I = \beta(\Delta V_C)^2 - (\beta \pm \Delta\beta)[(\Delta V_C \pm \Delta V_T)]^2$

Simplifying

$\Delta I = \beta\Delta V_C(\Delta\beta/\beta \pm 2\Delta V_T/V_C)$

IF $\Delta V_C = V_T$, V_T error is 1-2%

The β error can be considered as a slope error or addition of noise.

VITA

Parthasarathy Balaji

Candidate for the Degree of

Master of Science

Thesis: DESIGN OF BUILDING BLOCKS FOR THE TRIT ALGORITHM

Major Field: Electrical and Computer Engineering

Biographical:

Personal Data: Born in Pondicherry, India, March 20, 1968, the son of K.N. Parthasarathy and K.P. Ranganayaki.

Education: Graduated from Madras Christian College School, India, in May 1986; received Bachelor of Engineering degree in Electrical and Electronics Engineering from College of Engineering in May 1990; completed requirements for the Master of Science degree at Oklahoma State University in May, 1993.

Professional Experience: Research Assistant, Department of Electrical Engineering, Oklahoma State University, January, 1992, to December, 1992.