Analog VLSI Supervised Learning System

Thesis by Ronald Gary Benson

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California
1994
(Defended August 4, 1993)

Acknowledgments

I thank my adviser, Professor John Hopfield, for providing an excellent environment for academic pursuit. I thank Professor Carver Mead for introducing me to the power of Analog VLSI as a means for performing highly parallel, low-power computations.

I thank Kwabena "Buster" Boahen for showing me the elegance and beauty in current-mode circuit designs. I thank Douglas Kerns for asking the pertinent techy questions.

I thank my mom and family for giving me support and encouragement to pursue a doctorate, and I thank my dad for his inquisitiveness, which helped me to keep on track.

I thank my friends, who helped to make my stay at Caltech enjoyable.

I thank everyone in the Hopfield group, in the Mead laboratory, and in the Computation and Neural Systems program, for many spirited and engaging discussions over the past 6 years.

Specifically, I thank Thomas Annau, Wyeth Bair, Öjvind Bernander, Rodney Brittner, Carlos Brody, Tobias Delbrück, Kurt Fleischer, Bhusan Gupta, John Harris, Paul Hasler, Andreas V. M. Herz, Robin Horrell, Vance Howard, David Kewley, John Lazarro, Maurice Lee, John Lemoncheck, Michael Lewicki, Shih Chii Liu, David MacKay, Jeffrey McGonigle, Bradley Minch, Marcus Mitchel, John Montgomery, Andrew Moore, Josée Morissette, Sylvie Ryckebusch, Rahul Sarpeshkar, Gregory Sullivan, Humbert Suarez, Massimo Sivilotti, and Lloyd Watts. They listened, discussed, advised, and helped.

I thank the administrative staff of the Hopfield and Mead laboratories: Jim Campbell, Debbie Chester, Helen Derevan, Chris Favata, Donna Fox, and Laura Rodriguez. They guided me through the bureaucratic avenues.

I thank the following sources for financial support: The National Science Foundation Fellowship, which supported me for three years; the PEW Charitable Trust Foundation; the Parsons Foundation; the California Institute of Technology; the Office of Naval Research; the State of California Department of Commerce; the Office of Competitive Technology; the Beckman Hearing Laboratory; and the MOSIS chip-fabrication service.

For those who dare to imagine.

Abstract

I built an analog very large-scale integration (VLSI) chip that learns in real-time. I have designed and tested this network in a 2 μm complementary metal-oxide-silicon (CMOS) process. The chip was fabricated through the Metal-Oxide-Silicon Implementation Service (MOSIS). This fabricated chip contains 12 neurons, each fully connected to the other 11 neurons, but with no self-feedback connection.

The goal of this research is to build a supervised-learning neural-network VLSI chip. This neural chip could reside at a remote site unattended by a microcontroller, be battery operated, and be able to adapt autonomously to a changing environment. The neural chip has connection weights (or synapses) which are analog nonvolatile memories programmed in the presence of ultra-violet light. The chip consumes ultra-low power (less than one nW per synapse). Several other features of the chip distinguish it from previous work: the feedforward nonlinear mapping proceeds concurrently with the training process in real time; the weight modifications are performed in parallel and are calculated collectively as part of the network.

I have successfully trained this chip to perform various mappings. The test mappings performed by the chip have two inputs and one output with four hidden units recruited by the network. The chip is presented with inputs and target outputs and proceeds to learn the mapping from input space to output space. Additional memory on the chip allows any or all of the input units to be enabled; similarly, the neurons can be configured as output units or hidden units. The weights, teaching signals, neuron outputs, error units, inputs,

and target outputs can all be displayed on a multisync monitor to aid in debugging while the chip is being trained or run in feedforward mode. I show data of the chip learning.

Contents

List of Figures

Chapter 1

Analog VLSI Supervised Learning System

The medium is the message.

Marshall McLuhan
Understanding Media, 1964

Learning in artificial neural networks has received increased attention over the past several years [27, 28, 30, 37, 53, 54]. The promise of neural networks to solve nonlinear mappings by inference from a set of examples is largely responsible for this interest. Much work has been done in software, but relatively less attention has been given to the actual implementation of these learning networks in hardware [29, 45, 19]. Many researchers consider implementation a straightforward translation of the algorithms onto the medium. In fact, disregard for the fundamental constraints imposed by the medium is the reason that most neural architectures never leave the digital world where they are conceived or tested. I believe that algorithms

touted for their parallel and analog nature need to be implemented and tested in a medium suitable for the task being solved. By the same token, the medium itself needs to drive the formulation of new algorithms. As championed by Marshall McLuhan [41], "The medium is the message."

1.1 Neural-Network Hardware Solutions: A Brief History

The implementation medium that I have used for my research is silicon; the implementation uses **complementary metal-oxide-silicon** (CMOS) transistors run in the subthreshold region. The low-power consumption in the subthreshold mode makes these devices ideal for use in large-scale neural-network systems. In the past 10 years, many large-scale neural systems have been built in silicon. These systems include peripheral sensing devices such as cochleas [35, 38, 69] and retinae [14, 6, 47, 43, 39, 24], which mimic certain aspects of low-level biological sensory-processing systems. A few of these chips are adaptive in the sense that offsets are adapted away or nonchanging sensory inputs are selectively filtered or inhibited [4, 15]. These low-level sensory chips can be characterized by their homogeneous connection strengths; for example, each pixel of a silicon retina has the same connection strength to a resistive processing layer as do all other pixels.

More general types of neural chips allow the connection strengths to be heterogeneous. In addition, the inputs are typically less structured than are those received by the low-level sensory chips. This class of general-purpose neural chips allows for extremely general nonlinear mappings. Early types of these general-purpose neural-network chips had fixed connection strengths [60, 61, 64]. Subsequently, a whole class of chips was introduced that allowed the connection strengths to be modified by an external controller, typically a personal computer [26, 56]. These chips came in many varieties: Some had nonvolatile analog storage [26], others had volatile analog storage that required capacitor-refresh circuitry [18, 48], and still others had digital weights stored in **dynamic random access memory** (DRAM) cells [2].

Few neural chips have the learning paradigm imbedded in the network. An **imbedded algorithm** implies that the learning algorithm is an intrinsic part of the network; the two—network and learning algorithm—are integral and are inseparable. The idea of an imbedded

algorithm is different from a microcontroller running the learning algorithm on the same chip. Imbedded carries the additional connotation that weight updates are calculated in a similar manner as the feedforward mapping, and that these updates are parallel in nature. The power of a neural network comes from the network's ability to process information with relatively slow and inaccurate elements, but to perform the operation in a massively parallel fashion [27]. In algorithms with imbedded learning paradigms, this same notion of massively parallel computation is broadened to encompass the learning paradigm as well [53].

Recently, a small class of networks has been built that actually imbeds the learning paradigm in the network. Alspector and colleagues [2] build neurochips that use a Boltzmann type algorithm. In their implementation, the weights are stored digitally. The algorithm alternates between feedforward and training phases. Other implementations of imbedded learning chips store the weights on capacitors. In one implementation, the weights are refreshed periodically [10]; in other applications, the weights are trained continuously [68, 12].

Imbedded learning chips are necessary for the success of very large-scale neural systems. Consider a modestly large system of 10^6 synapses, each a nonvolatile analog memory. We assume that each weight can be altered with a minimum pulse time of $50\,\mu$sec. This pulse time is the time required for Intel's **electrically trainable analog neural network** (ETANN) [26], which uses a digital **electrically programmable read-only memory** (EPROM) flash technology optimized for fast writing. Faster pulse times are certainly possible, but they carry a risk of permanently damaging the oxide to the floating node. With such pulsing times, a sequential-weight-change architecture would require 50 sec for a single weight increment or decrement of every synapse of the entire synaptic array. In a scheme that has completely parallel updates of the weights, the time to increment or decrement all the weights would be only $50\,\mu$sec. Of course, there is a tradeoff here between silicon area and time, but if we want a large neural system to learn in real time, we require a silicon area expenditure.

1.2 Goal of Current Research

The goal of my research is to build a supervised-learning neural-network **very large-scale integration** (VLSI) chip. Such a neural chip could reside at a remote site unattended by a microcontroller, be battery operated, and be able to adapt autonomously to a changing environment. With these goals in mind, the following features are a reasonable set of characteristics that the chip should embody:

- The feedforward nonlinear mapping should proceed concurrently with the training process; there exist many applications where down time in the feedforward processing to allow for training could be deleterious to the system being controlled.
- The training process should proceed in real time; the real-time constraint is imposed to avoid the complexities associated with batch or offline training, such as additional memory to store weight updates for later training; also, it may be necessary to adapt quickly to novel stimuli.
- The weight updates should be done in parallel; as the size of these networks grows, parallel updates become increasingly necessary to keep training times down.
- The computation of weight updates should be calculated as part of a network; this collective computation ensures that the same benefits associated with collective, emergent properties achieved by neural networks can be applied to the calculation of weight updates.
- The weights should be nonvolatile; if a remote system experiences a power failure, information will not be lost if nonvolatile memories are used.
- The elemental components of the neural chip should consume very little power; if the system is battery powered or if the system becomes very large, it is necessary to have very low-power consumption by the elemental parts of the network.

1.3 Developments Presented Here

The chip presented here addresses the goals outlined in Section 1.2. The chip has the learning paradigm imbedded in the network. In contrast to the work reviewed in Section 1.1,

the connection strengths are nonvolatile; that is, the weights are preserved even through long periods of power shutdown. In addition, feedforward processing and training occur continuously and coincidentally. There are no explicit training and feedforward periods.

1.3.1 Learning Algorithm

The learning algorithm that I implemented was proposed by Pineda [53]; mathematically, it has two layers: one layer is used for calculating the nonlinear mapping (**feedforward layer**); the other is used for calculating the gradient of the weights to minimize the errors of the first layer (**error layer or feedback layer**). The processing of the feedforward layer is not strictly "feedforward," since this layer is fully connected. Nevertheless, I use the term "feedforward" to describe the processing of this fully connected layer as an analogue to traditional strictly feedforward networks, because the static nonlinear mappings of traditional feedforward networks are the only type of mappings being exploited by the fully connected layer of the Pineda algorithm.

The dynamical system proposed by Pineda [53] to perform forward propagation and backward propagation consists of three dynamical equations. The equation governing the forward dynamics is

$$\tau_x \dot{x}_i = -x_i + \sum_{j \neq i} w_{ij} f(x_j) + I_i, \tag{1.1}$$

where the $f(x_j)$ are saturating, monotonically increasing functions representing the input–output relation of the summing nodes; the w_{ij} are the weights of the connections between these nodes that are to be learned; the I_i represents **external input** supplied to the network; the xs are the inputs to the sigmoid; and τ_x is the time constant of relaxation for this layer. This equation defines the time-evolution of the state x_i at unit i; the value of x_i is fed through the sigmoid, $f(x_i)$ to generate the node's output signal. The equation governing the backward dynamics is

$$\tau_y \dot{y}_i = -y_i + f'(x_i)(\sum_{j \neq i} w_{ji} y_j + J_i), \tag{1.2}$$

where

$$J_i = Target_i - f(x_i). \tag{1.3}$$

These J_i are referred to as the **external error inputs.** The schematic structure of the connectivity described by Equations 1.1 and 1.2 is shown in Figure 1.1, which has three planes. The arrows show how the inputs flow in this network. The top plane contains three fully connected X units (excepting self-connections). These units collect inputs connected to them by the thin lines and generate outputs that propagate to units connected to them via the thick lines. The inputs are composed of the sum of outputs from the other X units, with each output modulated by a weight before it contributes to the sum, and an input supplied externally. The X unit computes a squashing function, or compressive nonlinearity, of this input.

The bottom plane consists of Y units. As in the X plane, thin lines are input lines and thick ones are output lines. The same summation occurs on the Yunit input lines as for the X units, but the Y unit does not perform a squashing function on its input. Rather, it modulates its input by the derivative of the associated X unit. The external inputs received by the Y units are error signals, which are the differences between targets and X unit outputs.

The weights are shown in the middle plane. These weights are shared by the X and Y processing layers. Notice, however, that the weights are transposed as used by the X and Y layers (you can see the transpose by noting that the subscripts are switched between the weights of the X-layer dynamics and the Y-layer dynamics in Equations 1.1 and 1.2). The weight updates are now calculated from quantities that are local to the weights themselves. The equation governing the weight dynamics is

$$\tau_w \dot{w}_{ij} = y_i \cdot f(x_j). \tag{1.4}$$

As stated mathematically in Equation 1.4 and shown schematically in Figure 1.2, the weight update is simply calculated as the output of the X layer multiplied by the output of the Y layer. Using these dynamics, the weights follow a trajectory down the gradient of the error surface (J_i^2 in Equation 1.3) in weight space. The weights are guaranteed to follow this trajectory only if $\tau_x \ll \tau_y \ll \tau_w$. For multiple patterns presented to the network sequentially, the dynamics approximate gradient descent, if $\tau_x \ll \tau_y \ll \tau_p \ll \tau_w$, where each input is presented for the duration τ_p; the error function minimized is now $\sum_{\text{patterns}} J_i^2$. I describe

an implementation of a modified version of this network in this dissertation.

1.3.2 Backward Error-Propagation Algorithm

The main reason the Pineda algorithm was used instead of the more widely used **backward error-propagation algorithm** (back prop) [54] is the flexibility in network architecture allowed by Pineda's algorithm. In Pineda's algorithm, input units, output units, and hidden units are specified at run time rather than design time. With back prop, the network architecture is fixed at design time. Another advantage of the Pineda algorithm is the continuous dynamics used by the entire system. These continuous dynamics allow for a natural implementation in an unclocked analog VLSI system. On the other hand, as formulated by Rumelhart and colleagues [54], back prop would require clocking if implemented. Also, in back prop, processing proceeds in two distinct phases—a feedforward phase followed by a training phase—whereas, in Pineda's algorithm, both phases occur concurrently. For these reasons, I chose the Pineda algorithm to achieve the goals set out in Section 1.2.

1.3.3 Network Implementation in Analog VLSI

The algorithm implemented in silicon contains two layers, as in the Pineda network. One layer calculates a nonlinear mapping from input to output space. This mapping is programmable and can be learned by the network from examples of input–output pairs. This layer is referred to as the **feedforward-processing layer.** A second layer coexists with the first and shares the weights with the first layer. The second layer is designed to compute the updates for each weight in a collective and parallel fashion. It receives as input an error signal that is the difference between the first layer's outputs and the externally supplied targets. This layer is referred to as the **feedback, or error-processing, layer.** The two layers process coincidentally and together form the entire network; the nonlinear mapping and learning paradigm are imbedded in the same network.

The neural chip designed uses transistors operating in the subthreshold region. Thus, the chip consumes very little power; for example, a network of 1000 neurons and 10^6 synapses[1] would consume 5 mW of power with a power supply rail of 5V and a current of 1 nA per

[1] This architecture is completely connected. This network is much larger than the system that I designed.

synapse.[2]

The network is designed to be self-scaling. The nonlinear processing elements (referred to as **sigmoids**) and the connection strengths (referred to as **synapses**) are designed such that the sigmoid automatically scales the input by the number of synapses. If total input level changes, but the normalized differential input remains constant, the output will not change. The total input may change because more synapses are added, the output levels of the other sigmoids change, different scales of external inputs are applied, or the chip heats up (causing larger current levels on the chip). Such input changes are rejected; only changes in the differential signal affect the output and only in proportion to the fraction of the total signal. This invariance achieved by the sigmoid is a result of the form of circuits used known as **normalized differential current-mode circuits.** This type of circuit form allows for an elegant, compact, and powerful CMOS subthreshold VLSI circuit solution.

I designed and tested the learning network in a 2-μm CMOS process. The chip was fabricated through the **Metal-Oxide-Silicon Implementation Service** (MOSIS) [58]. This test chip contains 12 neurons, each fully connected to the other 11 neurons, but with no self-feedback connection. Thus, there are 132 synapses. Each synapse contains a weight, which is embodied by the differential voltage between two floating nodes; a structure used for weight modification; a four-quadrant multiplier for the feedforward network; another four-quadrant multiplier for the error network; circuitry to calculate the training signal; and two differential pairs used for scanning out the weights and the training signal. The synapse is approximately 200 μm square. The 12 fully connected neurons each contain a sigmoidal unit (X unit) and an error-processing unit (Y unit). The sigmoidal unit is used for the nonlinear mapping; the error unit is used to calculate the weight updates. The sigmoidal unit and error unit, with two differential pairs for scanning out the state information of each unit, occupy approximately a 200 μm square.

Inputs and targets are scanned in on a single line onto sample and hold circuits. Each input and each target has an associated 1 bit of memory called the **enable bit.** If an input is enabled, the associated sigmoid receives an external input. If a target is enabled, the associated error unit receives an error signal calculated from the target and the output

[2] This amount of current per synapse is more than adequate and could be made smaller if needed.

of the associated sigmoid. These targets also specify the output units of the feedforward network. Any unit that receives neither an external input nor a target is a **hidden unit.** In this fashion, units can be configured at any time as input, hidden, or output units. The weights, teaching signals, neuron outputs, error units, inputs, and targets can all be displayed on a multisync monitor to aid in debugging while the chip is being trained or run in feedforward mode. I successfully trained the neural chip to learn a mapping from two inputs to one output for one, two and four distinct patterns in a training set.

1.4 Organization of this Work

In this dissertation, I discuss the design, implementation, and characterization of this learning chip. The text is organized as follows:

- **Chapter 2:** I discuss the method of nonvolatile analog weight storage. The weights are charge stored on floating gates. **Ultra-violet** (UV) light is used to activate a conductance between the isolated floating gate and the programming control node. I present experimental results characterizing the UV-photoinjection devices. These experimental results show that the conductance of the UV-photoinjection device varies nearly linearly with intensity of UV exposure, and nearly linearly, as well, with exposed edge length. I present results of UV attenuation with distance under metal shielding.

- **Chapter 3:** I discuss the basic design philosophy of the circuits used. The circuits are designed in the normalized differential current mode. The basic computational entity is differential currents, which are normalized by the total current. I introduce the notation used to describe both voltages and currents through a **field-effect transistor** (FET) run in the subthreshold mode. I analyze one of the basic building blocks, the differential pair, from the normalized differential current-mode view-point. The Delbrück current correlator [13], another very useful building block, is next analyzed. I introduce several other interesting current-mode circuits, including a four-quadrant sine-approximation circuit and an inverted differential-pair circuit. The methods used to test inputs and outputs of these current-mode circuits are discussed also.

- **Chapter 4:** I introduce the synapse circuit. I present the weight representation, the dynamics of the weights, the synaptic four-quadrant multiplier characteristics, and

the learning nature of the synapse. The synapse circuit is an example use of the UV-activated structures and device physics described in Chapter 2. I designed this synapse to be an element in the larger learning system that requires continuously adjustable, nonvolatile synaptic weights, continuous weight decays with a slow time course, and an uninterrupted feedforward multiplication operation, while the weight is being trained or allowed to decay. In this chapter, I analyze each of these functions and show data from a fully functional synapse circuit.

- **Chapter 5:** I discuss the processing units in the feedforward layer and the error layer. I show how the sigmoid is self-scaling relative to the input, how the sigmoidal gain is varied, and how the sigmoidal output can be scaled. In addition, I show how the processing unit of the error layer computes an approximation to the derivative of the sigmoid. Finally, the dynamics of these circuits are considered.

- **Chapter 6:** I discuss the entire network. I introduce the new learning algorithm that the network implementation performs, and show a variant of the gradient-descent algorithm devised by Pineda [53]. Also, I discuss the stability of the network and convergence of the new learning algorithm. I discuss various examples of successful and unsuccessful learning tasks.

- **Chapter 7:** I conclude by showing how I have met the goals of this research, by pointing to several drawbacks of the present design, and by outlining future directions of research.

- **Appendix A:** I discuss in detail novel current-mode circuits which I invented, but were not part of the learning chip. First, I present a subthreshold four-quadrant sine-approximation circuit. This circuit uses the Delbrück current correlator, and performs a sine computation to within 2 percent near minimax error. The second circuit I discuss is an inverted differential pair. This circuit is similar in function to a differential pair, but uses a different approach to perform the function. This circuit could also be used as a gain-control device.

- **Appendix B:** I discuss the stability of the X layer. The stability criteria for this layer are complicated by the fact that I use diodes, rather than resistors, coupled with

capacitors for this dynamical system. I show that the sigmoidal gain and the sum of the weights to which a neuron connects are the relevant quantities that I need to control to ensure global asymptotic stability of the X layer.

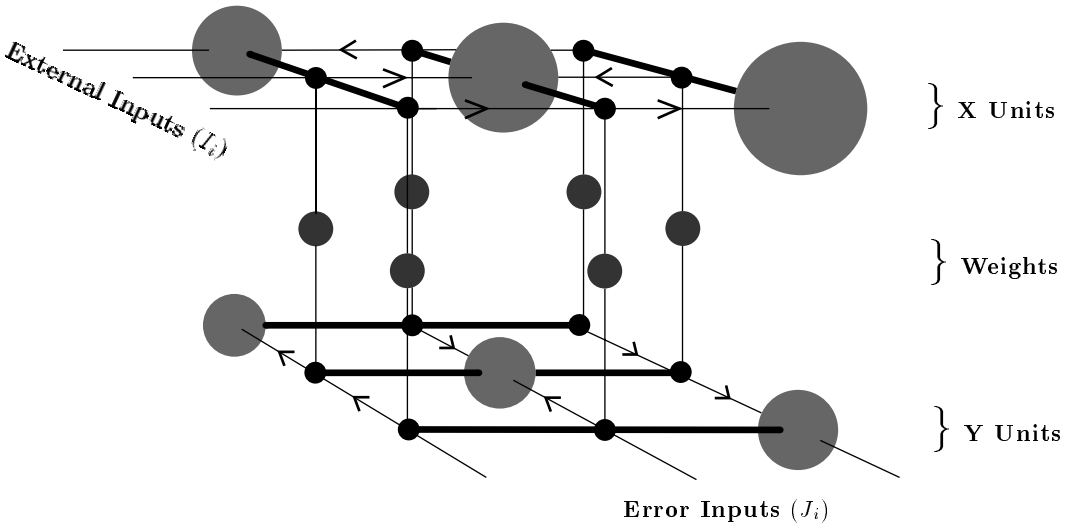


Figure 1.1 Network embodiment of Pineda's recurrent learning algorithm. The network consists of two processing layers with equal number of units: three feedforward-processing units or X units shown as large balls in the top plane and three error-processing units or Y units shown as large balls in the bottom plane. The thick lines represent outputs from each of the units. The weights in the middle plane modulate the connections between one unit's output and another unit's input. The feedforward layer receives external inputs; the output of the X units is the output of the network. The error-processing layer receives error inputs—errors calculated as the difference between top layer's outputs and externally provided target outputs. This error layer calculates gradients of the weights to minimize the error of the forward layer.

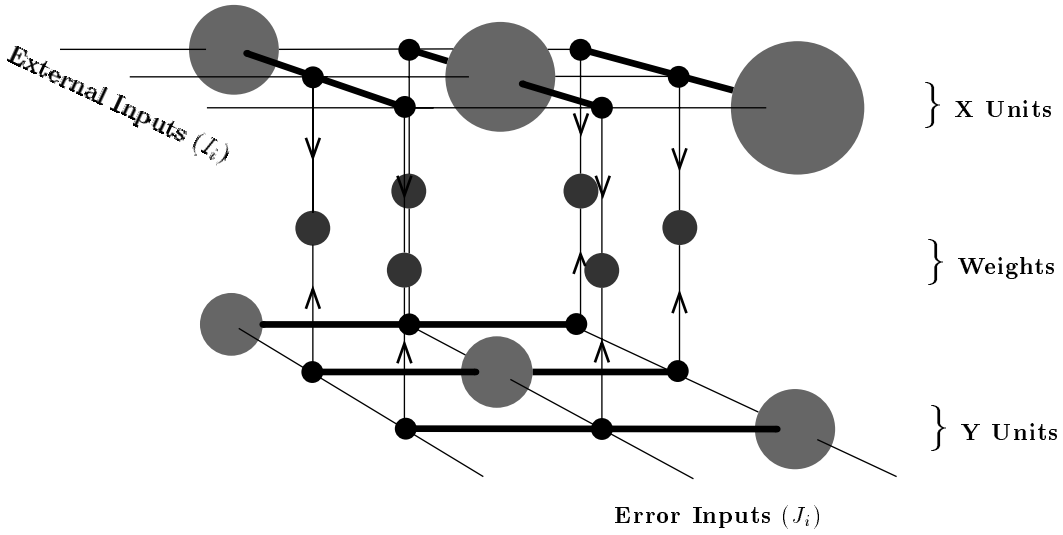


Figure 1.2 **Computation of weight dynamics in the Pineda network. The weights (middle plane) are adjusted in proportion to the product of the outputs of the X and Y units at each weight.**

Chapter 2

UV-Light Basics

There is a wicked inclination in most people to suppose an old man decayed in his intellects. If a young or middle-aged man, when leaving a company, does not recollect where he laid his hat, it is nothing; but if the same inattention is discovered in an old man, people will shrug up their shoulders, and say, "His memory is going."

Samuel Johnson

Boswell *Life*, V. 4, 1783

CMOS technology allows the simple construction of long-term memory structures in the form of floating-circuit nodes that can store electrical charge for periods measured in years.[1]

[1] This chapter contains much of the same text and figures that appear in an article written in collaboration with D. A. Kerns [5].

This capability is attractive for building adaptive, or learning, machines. In the past, CMOS long-term-storage techniques have been primarily digital in nature, and commercial products such as EPROM and **electrically eraseable programmable read-only memory** (EEPROM) digital memories have long been available. Recently, the same techniques have been refined and extended to analog data storage (more than 1 bit of information per node) in several applications [9, 11, 22, 26, 34, 43, 55, 65, 66],

The two main methods for programming floating nodes are Fowler–Nordheim tunneling and hot-electron injection. Fowler–Nordheim tunneling requires high electric fields, which are achieved in practice typically by either very thin oxides or high voltages. Hot-electron injection also requires high fields which are achieved in practice typically by using heavily doped junctions or high voltages. Several investigators, [22, 34, 43, 65, 66], have recently demonstrated the use of a **UV-light activated** (UV-activated) mechanism to allow analog programming of floating circuit nodes. Kerns [33] presents interesting applications of such devices, including a UV-light dosimeter. The advantage of the UV-activated mechanism is that it allows programming of floating nodes in a circuit without either high programming voltages or a special CMOS fabrication process. In addition, the intrinsic physics of the UV-activated mechanism provides time scales that are conveniently matched to the requirements of a learning system implemented in silicon. At room temperature, standard analog silicon circuits can have time constants ranging from nanoseconds to seconds, whereas the UV-activated mechanism spans time scales from seconds to megaseconds. Thus, more than 10 orders of magnitude in time scale are available in the same medium.

I present new experimental results characterizing the UV-photoinjection devices. The new experimental results show that the conductance of the UV-photoinjection device varies nearly linearly with intensity of UV-light exposure and nearly linearly with exposed edge length, as well. Results of UV-light attenuation with distance under metal shielding are presented.

2.1 The UV-Activated Mechanism

This section introduces the physical basis for the UV-activated mechanism, discusses the first-order circuit description and shows the layout of the structure. A physical expla-

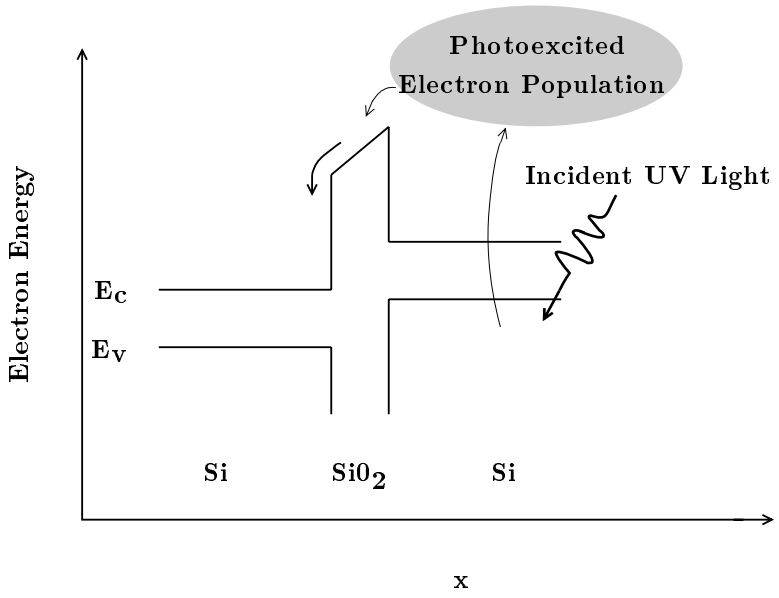


Figure 2.1 UV-activated conductance energyband model. A simple band diagram of the Si-SiO_2-Si UV-sensitive structure: incident UV photons excite a population of electrons from the valence band of Si to energies above the conduction band of SiO_2. Some of the excited electrons enter the oxide layer and are swept across by the voltage gradient.

nation for the operation of the UV-activated structure comes from semiconductor band theory. The first-order circuit description of the UV-activated devices consists of a capacitor in series with a switched conductance. The layout of the device consists of two overlapping layers of polysilicon shielded everywhere by a layer of metal except where the UV-activated conductance is desired.

2.1.1 Physical Model

The simplest physical explanation for the operation of the UV-activated mechanism comes from semiconductor-energyband theory. The UV-activated structures are Si-SiO_2-Si layered structures in which the SiO_2 presents a 4.2- eV energy barrier to electrons from the valence band in Si to the conduction band in SiO_2. When the structure is exposed to shortwave UV light, the UV photons impart sufficient energy to the electrons in the Si valence band

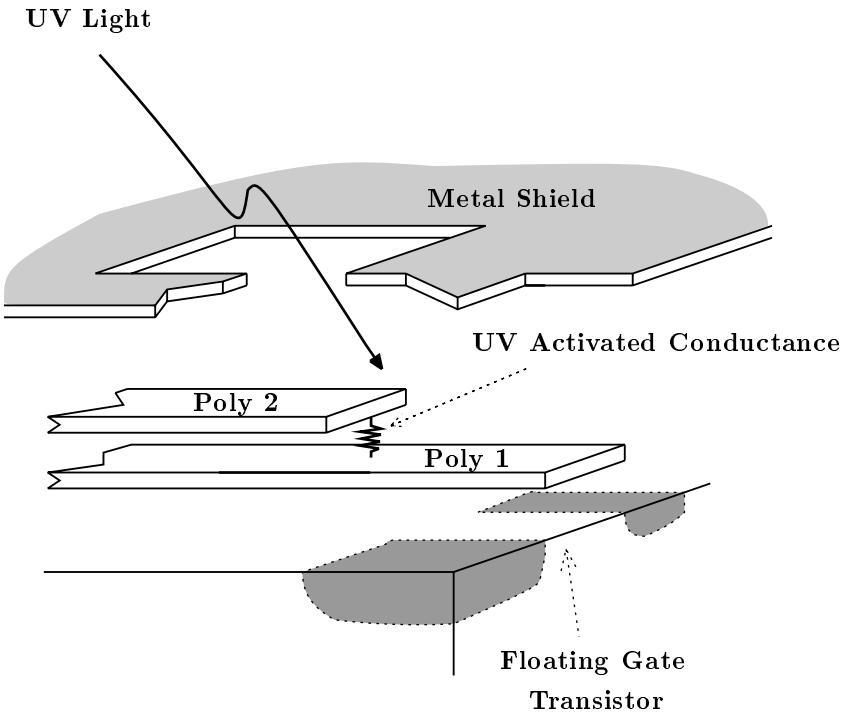


Figure 2.2 Cutaway view of the UV-activated structure. The UV-activated conductance is shown as a resistor between the poly1 and poly2 plates. Either the poly1 or poly2 layer can be used as the floating gate.

to enter the SiO_2 conduction band. These excited electrons support an electrical current through the oxide (see Kerns [34] for a more detailed explanation of this model). Figure 2.1 diagrams this simple physical model; a cutaway view of a typical structure is shown in Figure 2.2. When a field is applied across the SiO_2, the barrier to electrons presented by the SiO_2 is lowered, which allows more electrons to overcome the barrier. This barrier lowering is discussed in detail in [49].

2.1.2 Circuit Model

The simple physical model, coupled with the experimental measurements shown in Figure 2.4, leads us to a simple first-order circuit model for the UV-activated structures, as shown in the inset of Figure 2.4 The capacitance of the structure is due to the parallel plates of Si separated by the SiO_2 dielectric, and the switched conductance models the UV-activated electron current through the oxide.

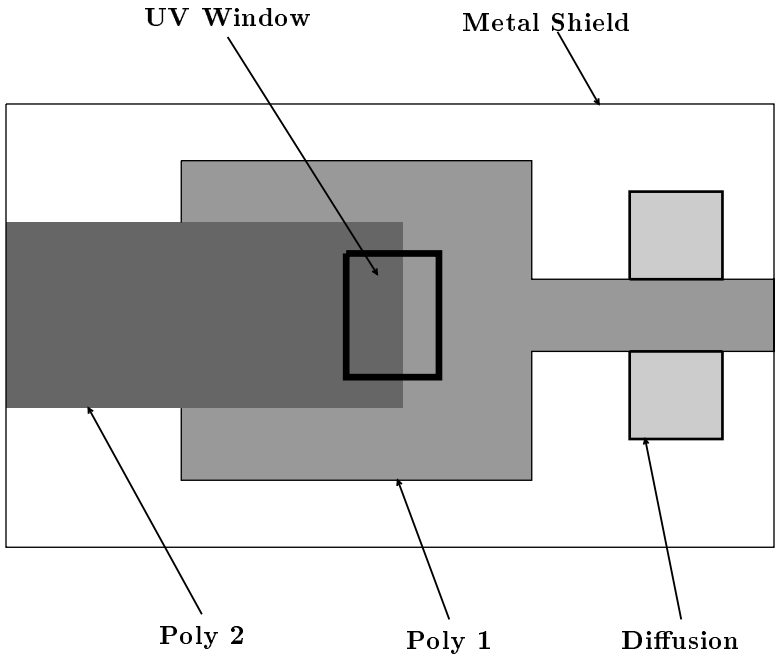


Figure 2.3 Layout of the UV-activated structure. A layer of metal is used to insulate sensitive circuitry from the effects of UV light. A hole in the metal layer is used to target the UV light to particular structures.

A closer examination of the measured I–V curve shows that the simple first-order circuit model is not perfect. There is a significant and repeatable lower conductance near the origin, within the first 100 mV (corresponding to electric fields of less than 2.5×10^4 V per centimeter across the oxide layer). The inset plot in Figure 2.4 shows the measured I–V characteristics of a UV-activated structure at low voltages; this picture suggests that the conductance in our circuit model should be somewhat nonlinear. The flattening of the I–V curve near the origin may be due to some sort of trapping mechanism in the SiO_2 layer [17, 50]. For many designs, however, this nonlinearity is not detrimental.

2.1.3 Layout Geometry

The layout of a UV-activated device is simple; as implied by the circuit model, the device is simply a capacitor that is exposed to UV radiation. Usually, one needs to consider surrounding circuit structures, as well as the UV-activated structure. It is often necessary

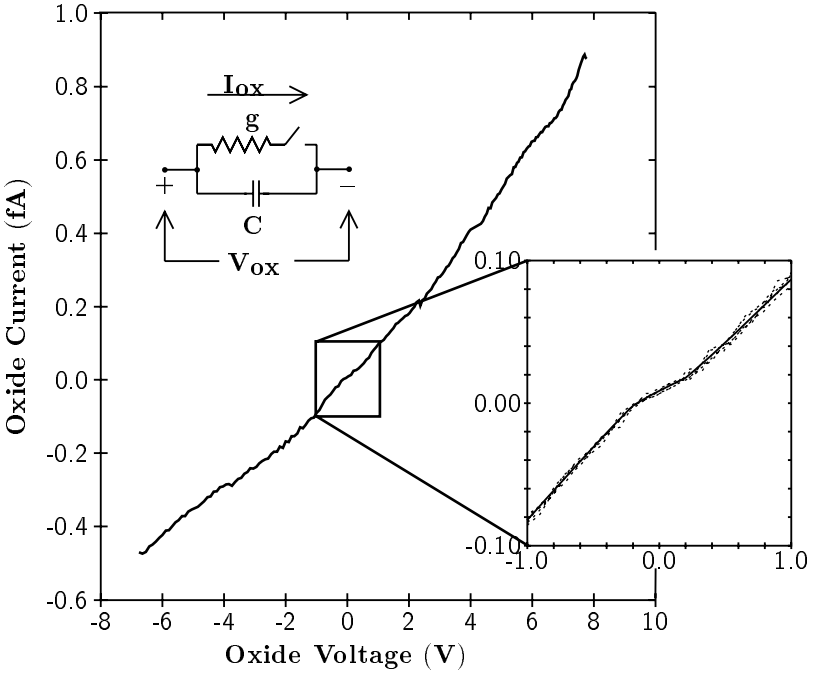


Figure 2.4 **Measured current–voltage characteristics of the Si-SiO_2-Si structure. These characteristics indicate that a simple capacitance–conductance model is a good first-order fit. The solid line in the larger figure represents averaged data. The nonlinear behavior for small oxide fields is shown in the blowup near the origin. Dots represent measured data, and the solid curve is a nonlinear fit.**

to shield surrounding circuitry from incident light to prevent undesired photocurrents at p–n junctions. A standard layout technique is that of covering the entire chip with the second metal layer, with windows cut in the metal shield to expose UV-activated devices and possibly other photosensitive devices.

An important issue in the layout of UV-activated devices in a circuit is how far the UV light can propagate under the metal shield layer to induce parasitic conductances. This issue is discussed in Section 2.2.

2.2 UV-Activated Conductance

To design UV-activated structures effectively, we must understand how the UV-activated conductance varies with electrical, optical, and geometric parameters.

2.2.1 Electrical Behavior

As mentioned in Section 2.1, the conductance of the UV-activated device is nonlinear, particularly at low voltages. Although the physical mechanism for this nonlinearity is not clear, the experimental data are well fit near the origin by a simple functional form:

$$I = I_{\text{leak}} + GV - V_0 (G - g) \tanh\left(\frac{V}{V_0}\right).$$

This fit curve is plotted in the inset of Figure 2.4 for the values $G = 1.0 \times 10^{-16}$S, $g = 3.0 \times 10^{-17}$S, $I_{\text{leak}} = 9 \times 10^{-18}$ A. The leakage-current term accounts for unwanted conductances elsewhere in the circuit, perhaps due to light leakage.

2.2.2 Optical Variations

We might expect a variation in the observed UV-activated conductance with changes in UV-light illumination. This variation is in fact experimentally observable, as shown in Figure 2.5. The conductance varies nearly linearly with the energy density of the UV light on the device. The reason for the departure from linearity is not clear.

2.2.3 Geometric Considerations

In designing a circuit that incorporates UV-activated structures, the layout of the circuit is an important consideration for several reasons. The exposed perimeter of the upper capacitor plate in the UV-activated structure dominates the conductive behavior of the device. Indirect UV-light exposure can cause undesired UV-activated oxide currents and can interfere with the operation of certain circuits.

2.2.3.1 Edge Length

The UV-activated conductance is dominated by the length of exposed edge and is essentially unaffected by the area of the structure. The oxide thickness between silicon layers is typically

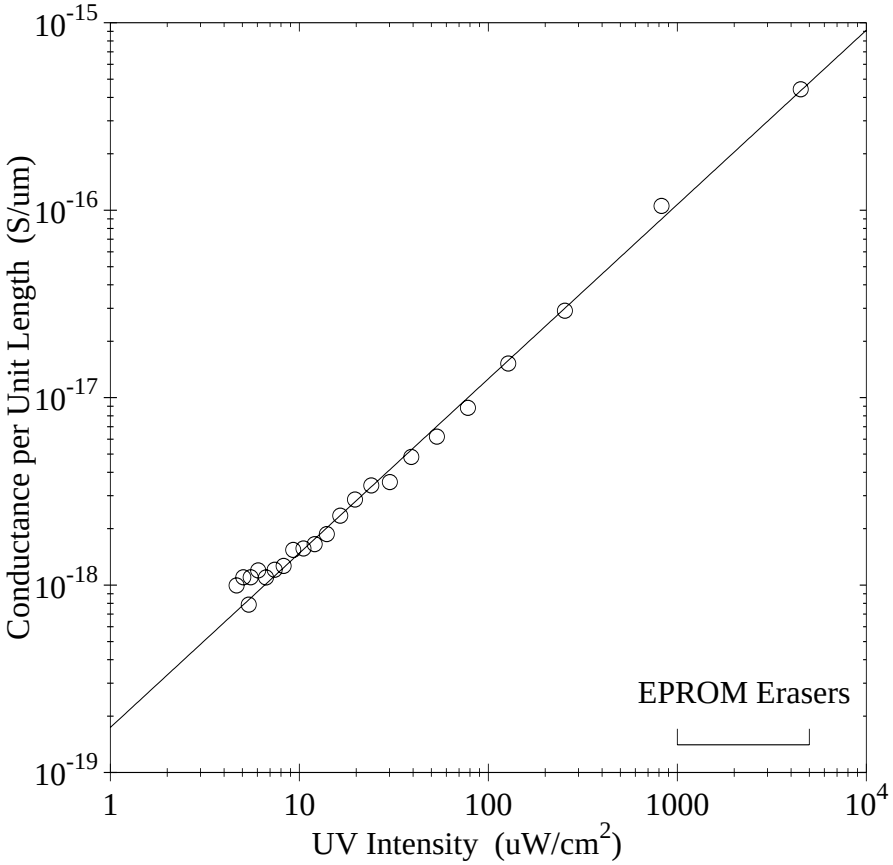


Figure 2.5 Measured device conductance as a function of intensity. The variation is slightly less than linear, following a power law of 0.93. All data are for 254 nm UV light (4.8 eV); typical intensity ranges for EPROM erasers are noted.

less than 50nm–far less than UV wavelengths–so UV light does not propagate between the silicon layers. The polysilicon capacitor plates are thick (several hundred nanometers) and so effectively attenuate the UV light, casting shadows on lower circuit structures. Because of this shadowing effect and the inability of the polysilicon layers to act as a wave guide, the only parts of the UV-activated structure where both silicon layers are exposed to incident UV light are the edges of the upper polysilicon layer. I have verified the edge dependence experimentally. Figure 2.6 shows results for structures that have constant area, but have varying perimeter. The conductance increases linearly with perimeter, nearly intersecting the origin on extrapolation to zero edge length. Based on these results, the area contribution

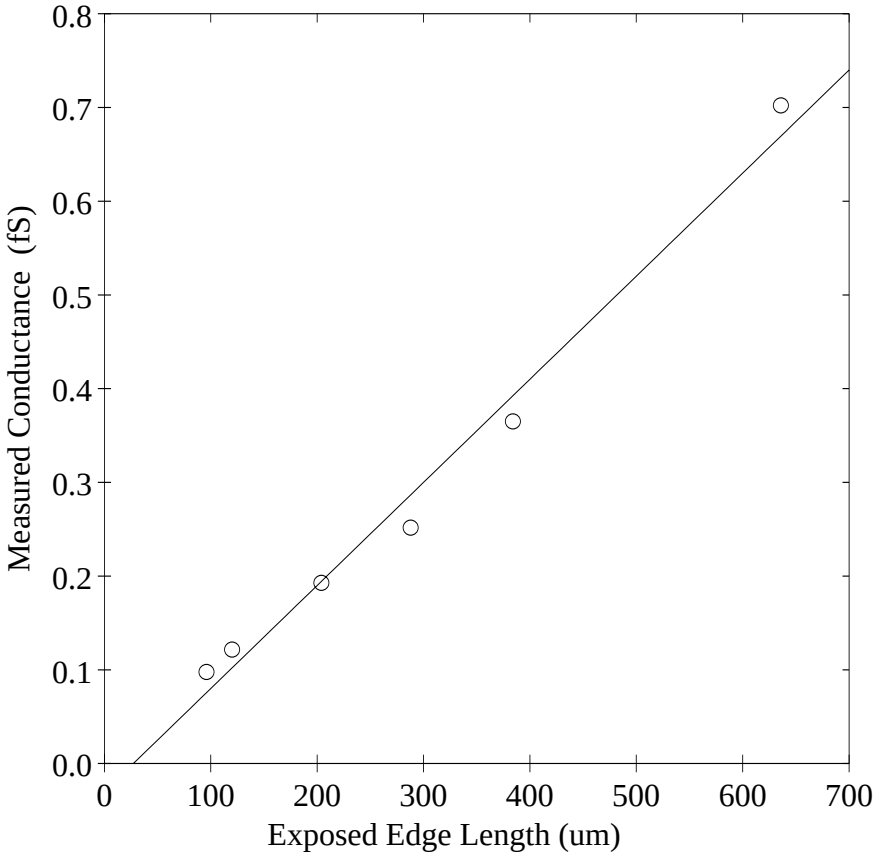


Figure 2.6 Measured device conductance as a function of edge length: constant-area devices with different edge lengths are compared. UV-activated conductance is dominated by the total length of exposed poly edge in the structure.

(per square micron) is less than 5 percent of the edge contribution (per linear micron).

For maximum conductance, a UV-activated structure should be built with as much edge as possible: for a given area of silicon, long, narrow structures (rather than square ones) maximize the UV-activated conductance. By controlling the perimeter and area, the designer has independent control over both capacitive and conductive coupling strengths for UV-activated structures.

2.2.3.2 UV-Light Shielding

One of the most elementary problems that can arise in these and other optical-input circuits is the optically induced leakage currents across p–n junctions in the circuit. The p–n junctions can have leakage currents that vary from picoamperes in the dark to microamperes in bright light, either UV or visible. Such drastic variations in junction leakage may not be tolerable in many circuits. The designer should take precautions to shield the sensitive circuits from undesired exposure. An effective technique uses the uppermost metal layer in a multiple-metal fabrication process (for example, the metal2 layer in a MOSIS double-metal process) to cover completely an area of sensitive circuitry. Windows in this metal-shield layer allow UV light to enter only where desired.

Undesired oxide–leakage currents can also occur under UV-light exposure. Large floating nodes have a correspondingly large area of exposure to UV radiation unless some sort of shielding is provided. Again, metal shielding works well.

2.2.3.3 Indirect UV-Light Exposure

Even under a metal shielding layer, parts of a circuit can be exposed to UV light by reflections from the metal shield and the substrate. The energy density of the UV light drops off with distance under the shield, as it is attenuated by spreading and absorption, but the light-leakage region is of a substantial extent. I characterized this UV-light leakage using an array of UV-activated test structures that were covered by a metal shield with windows at various distances from the structures. Figure 2.7 plots the measured conductance as a function of distance x from the edge of the window. If we assume a rectangular window admitting UV light to the circuit, with a UV-activated device at some distance x away from the edge of the window, we can model the leakage of UV light to the device by the following simple model. First, only some fraction of the incident UV light is scattered in the proper direction to propagate under the shield away from the window; hence, an "insertion loss" factor is added at the edge of the window. Next, the UV-energy density lessens with distance from the window due to spreading (same energy flux over a larger area) and absorption. For a rectangular geometry, the preceding simple model of the UV-energy distribution in

the test structures leads to the theoretical solid curve seen in Figure 2.7, given by

$$g = g_0 \int_{\text{edges}} E(s)ds,$$

where $E(s)$ is the UV-energy density at position s, and g_0 is a scale factor. The integral for g is taken on only the edges of the UV-activated device; the area of the device has little effect, as noted previously. Near the window, the measured conductance is approximately the same as in open regions, whereas far from the window, conductance drops off asymptotically as $e^{-x/\lambda}$. The λ found for my devices was 70μm. This parameter is likely to be dependent on the fabrication process.

2.2.4 Time Constants

Often, it is desirable to build systems that adapt over various time scales. For example, in a learning system, it is desirable to have the weights change relatively quickly for an input from a training signal, and to decay at a much slower rate.

From the previous discussion, two different methods for controlling such time constants are readily available. The first method is to keep the UV-activated structures directly under the UV-light window and to vary the perimeter of the top layer of polysilicon that is exposed to UV light while keeping the area of the structure constant. In this way, structures with different time constants can be constructed. The second method for constructing adaptation structures with different time constants is to place the structures varying distances from the UV-light window. By varying the distance from the window, we can create structures of the exact same configuration that have time constants that are considerably different.

Figure 2.7 UV-activated test structure and measured conductance as a function of distance from the window edge. (a) Diagram of UV-activated test structures used to measure attenuation of UV light under a metal shield. The windows were placed at various distances from the otherwise identical UV-activated devices. (b) Measured UV-activated device conductance as a function of distance from window edge. The discontinuity at the window edge (distance = 0) is due to an insertion loss factor included in the model for the structure.

Chapter 3

Current-Mode Circuit Basics

O! this learning, what a thing it is.

Shakespeare
The Taming of the Shrew

This chapter discusses the design methodology and several fundamental circuits that I use to design the learning network. The technique that I use to design and analyze the learning circuits is called normalized differential current mode. Unidirectional currents are the basic computational entities; differences between two currents are the computational quantities which may be positive or negative. Differential currents normalized by the total current yield a dimensionless quantity that represents mathematical variables in the equations implemented by the circuits. These circuits are called **normalized differential current-mode circuits.** For a comprehensive description of current-mode circuits and the translinear principle, see Andreou [3], Gilbert [20], Seevinck [57], or Vittoz [67]. This chapter introduces these basic computational constructs; however, it assumes a basic understanding of the translinear principle, current-mode circuits, and MOS device behavior.

The concept of normalization in this circuit form is powerful. The output of the circuit is invariant to the level of the input. The input level can change over many orders of magnitude and the output remains invariant; the circuit computes the same function with respect to the normalized differential input for all absolute input levels. This normalization feature allows us to build circuits that yield scalable networks or architectures (that is, we can add more input units without modifying the output circuitry or the output bias levels for the same static output functionality). The circuits are relatively insensitive to temperature variations, which can cause the input current levels to change. In this chapter, I show how this invariance is achieved; in subsequent chapters, I show how I use this same invariance to design a larger current-mode system.

The basic building blocks of the learning chip use normalized differential currents for both inputs and outputs. The circuits can be run in weak-inversion, moderate-inversion, or strong-inversion regimes of the transistor, but the function computed might change. For a description of these regimes, see any of a number of books on semiconductor device physics, such as Sze [63]. Vittoz [67] deals extensively with the moderate-inversion regime. Most of the analysis contained herein deals with the subthreshold regime and this regime is used as the default.

First, I introduce the notation that I use to describe voltages and currents through a FET run in the subthreshold mode. Then, I analyze one of the basic building blocks, the differential-pair circuit, as a normalized differential current-mode circuit. The current-correlator circuit is analyzed next. I discuss how the translinear principle is modified for FETs run in the subthreshold regime as compared with the **bipolar junction transistor** (BJT), for which the principle was originally conceived. I briefly introduce several other interesting current-mode circuits, including a current-correlator follower, a four-quadrant sine-approximation circuit and an inverted-differential-pair circuit and analyze them in detail in Appendix A. Finally, I show several techniques that I use to test the current-mode circuits.

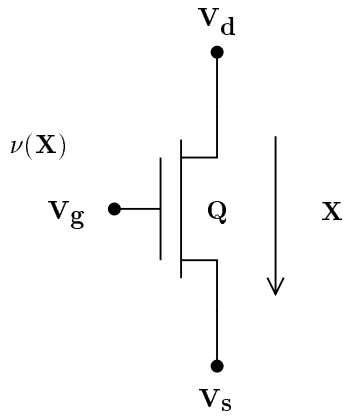


Figure 3.1 FET with relevant current and voltages. A voltage $\nu(X)$ applied from gate to source of the FET to yields a current X through the exponential relation in Equation 3.1. Unless otherwise specified, transistors are used in saturation (that is, $V_d - V_s$ is large enough to guarantee this condition).

3.1 Conventions Used for Normalized Differential Current-Mode Circuits

When a FET is biased in the subthreshold mode, a stylized equation describing the I–V relationship helps make analysis of current-mode circuits quite easy.

3.1.1 Subthreshold Regime

The dimensionless drain–source current through transistor Q of Figure 3.1 in saturation can be described as

$$X \equiv \frac{I_x}{I_0} = e^{\nu(X)}, \tag{3.1}$$

where X is the dimensionless drain–source current (that is, the current through Q in units of I_0, the zero-bias current, of the device), and where $\nu(X)$ is the dimensionless gate to source voltage; that is,

$$\nu(X) = V_g - V_s \tag{3.2}$$

in units of V_t, the thermal voltage. $\nu(X)$ specifies the gate and source voltage needed to get a dimensionless current X flowing through transistor Q given the transistor is in saturation. I use this stylized equation to take advantage of the translinear principle in order to get a basic feel for the operation of a circuit. To get a more precise description of a circuit's operation, we often must include the ineffectiveness of the gate voltage in determining the

barrier potential at the substrate surface, where now the voltage obtained for a current X is given by

$$\nu(X) = \kappa V_g - V_s, \tag{3.3}$$

where X is the dimensionless drain–source current, and V_g and V_s are the dimensionless gate voltage and source voltage, respectively, in units of V_t. The factor κ is a fraction (usually in the range of 0.5 to 0.8) used to parameterize the effectiveness of the gate in changing the barrier potential at the channel surface [42]. I show the effect that κ has on the translinear principle when I analyze the diode stack in Section 3.4.

Note that I have assumed that the device described by Equations 3.2 and 3.3 is in saturation. When the device is not saturated, the contribution of the reverse current needs to be considered as discussed by Boahen [7]; the current-correlator circuit is an example circuit where the reverse current is significant as shown in Section 3.3.

3.1.2 Normalized Differential Currents

In the circuits that I use in this dissertation, the computational entities are differences of currents. Typically, both inputs to and outputs from these circuits are dual rail and carry two unidirectional currents. The difference between these currents is normalized by the total current; for example, the normalization of $X^+ - X^-$ is

$$x \equiv \frac{X^+ - X^-}{X^+ + X^-}. \tag{3.4}$$

Normalized quantities are represented by lower-case letters and are dimensionless. Notice that the normalized quantity, x, can range **only** from -1 to $+1$. I show how normalized differential currents are used by analyzing the operation of a differential pair.

3.2 Differential-Pair Circuit

The current-mode differential-pair circuit shown in Figure 3.2 takes two input currents, X^+ and X^-, and produces two output currents, Z^+ and Z^-. The arrowheads on each line specify the direction of current flow. I place the constraint that the total input $X_t = X^+ + X^-$ is constant. In addition, Q_{bias} is operated in saturation; that is, V_{ref} is large

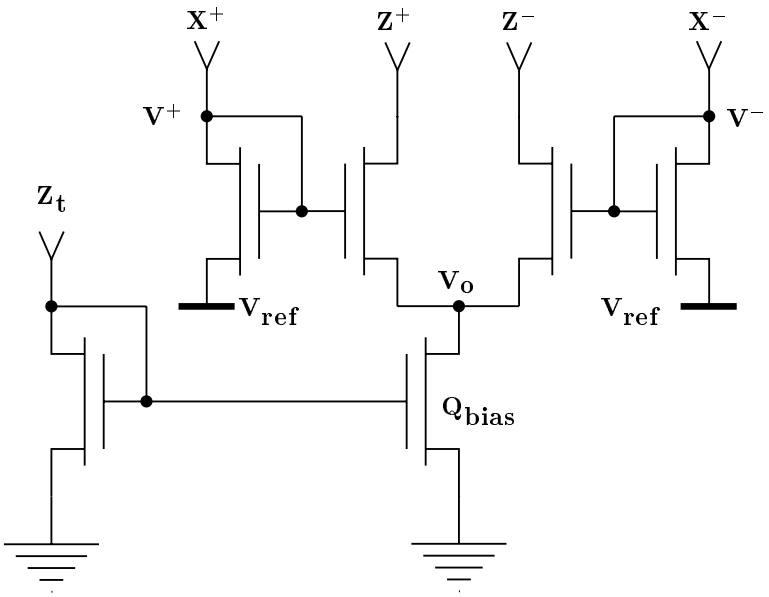


Figure 3.2 The differential-pair circuit. The differential pair is a basic current-mode building block, which computes $z = x$. Inputs are X^+ and X^-. Outputs are Z^+ and Z^-. Current directions are shown as arrowheads on each line.

enough to guarantee that Q_{bias} is saturated, and therefore, $Z_t = Z^+ + Z^-$ is a constant set by the bias to Q_{bias}.

It is straightforward to solve for the normalized differential output current as a function of the normalized differential input current. We know that

$$\begin{aligned} X^+ &= e^{V^+ - V_{\text{ref}}}, \\ X^- &= e^{V^- - V_{\text{ref}}}, \\ Z^+ &= e^{V^+ - V_0}, \quad \text{and} \\ Z^- &= e^{V^- - V_0}. \end{aligned}$$

Solving these equations for X and Z, we get

$$X^+ Z^- = X^- Z^+. \tag{3.5}$$

We can also easily obtain this result using the well known translinear principle (see Gilbert [20, 21] or Seevinck [57]).

We would like to solve for the normalized differential current output versus the normalized differential current input. The normalized quantities we need are defined as

$$x \equiv \frac{X^+ - X^-}{X_t} \quad \text{and} \quad z \equiv \frac{Z^+ - Z^-}{Z_t}, \tag{3.6}$$

or, equivalently,

$$X^+ = \frac{X_t}{2}(1+x) \quad \text{and} \quad X^- = \frac{X_t}{2}(1-x); \tag{3.7}$$

and similarly for Z^+ and Z^-. Note that x and z can range from -1 to $+1$.

Substituting X^+, X^-, Z^+, and Z^- from Equation 3.7 into Equation 3.5, we obtain

$$z = x. \tag{3.8}$$

The differential pair is the identity function when operated in the normalized differential current mode, whereby the bias current, (that is, Z_t), scales the total output current. Also notice that the output is unaffected by the scale of the input current. This circuit converts from the scale set by the input, X_t, to the scale set by the output, Z_t. This invariance of the output to the input scale is a powerful feature of the normalized differential current-mode circuits, and is exploited in the design and operation of the learning network.

In the differential-pair circuit, the κ of Equation 3.3 does not affect the operation of the circuit.

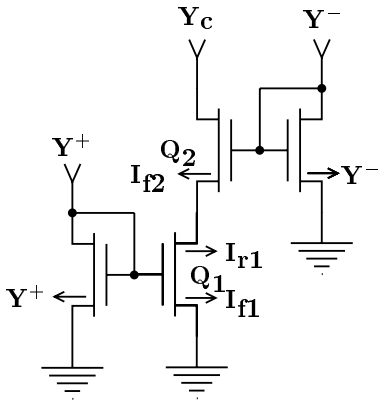


Figure 3.3 **The current-correlator circuit. The inputs are Y^+ and Y^- and the output is Y_c. The correlator output produces a bump of the input.**

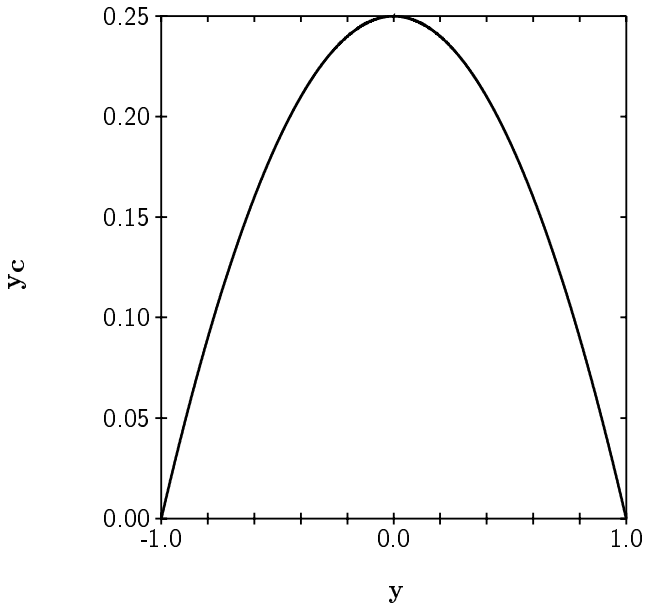


Figure 3.4 Ideal behavior of the current correlator. The normalized input is y, and y_c is the normalized output.

3.3 Current-Correlator Circuit

The current correlator of Delbrück [13], shown in Figure 3.3, computes a quadratic function or **bump** of the input (hence, the name bump circuit) and is useful in calculating the derivative of a simoid-like function. The inputs to the bump circuit are Y^+ and Y^-, and the output is Y_c. The normalized output current $\frac{Y_c}{Y_t}$ is

$$y_c = \frac{1}{4}(1 - y^2), \tag{3.9}$$

where

$$y \equiv \frac{Y^+ - Y^-}{Y^+ + Y^-} \quad \text{and} \quad y_c \equiv \frac{Y_c}{Y^+ + Y^-}.$$

Notice in Equation 3.9 that the output of the current correlator is not a differential signal, but rather is simply single ended. Therefore the current correlator is not compatible with the normalized differential current-mode scheme. However, this unidirectional current may be used to set the scale for the the output of a normalized differential current-mode circuit. The value of y_c can range from 0 to 1/4.

To solve the circuit equations using the translinear principle for the current correlator, I use a technique conceived by Boahen [7], in which both forward and reverse currents for a

transistor are defined. Figure 3.3 shows the current correlator and all the associated forward and reverse currents that I use in the analysis. Notice that transistor Q_1 has two currents associated with it, I_{f1} and I_{r1}, whereas transistor Q_2 has only the forward current, I_{f2}, shown; Q_1 is not necessarily in saturation; Q_2 is in saturation, and its reverse current is therefore zero. Using **Kirchoff's voltage law** (KVL) and the translinear principle,

$$\begin{aligned} I_{f1} &= Y^+ \\ Y^- I_{r1} &= I_{f2} Y^+. \end{aligned} \tag{3.10}$$

The reason that the currents multiply in Equation 3.10 is that the associated node voltages add as specified by KVL; to convert to current, we exponentiate the voltages, giving us currents that are multiplied.

Using **Kirchoff's current law** (KCL) and the definition of forward and reverse currents,

$$\begin{aligned} I_{f1} - I_{r1} &= Y_c, \\ I_{f2} &= Y_c. \end{aligned}$$

I_{f1} and I_{r1} are the forward and reverse currents of Q_1, respectively. They are exponential functions of the difference between the gate voltage and a source–drain voltage as determined by the current. In the case of I_{f1}, we treat ground as the source. To obtain the reverse current, I_{r1}, we treat the drain of Q_1 as the source (see Boahen [7]).

Solving for Y_c, we get

$$Y_c = \frac{Y^+ Y^-}{Y^+ + Y^-}. \tag{3.11}$$

Substituting into Equation 3.11 for $Y^+$and Y^-defined as

$$Y^+ \equiv \frac{Y_t}{2}(1+y) \quad \text{and} \quad Y^- \equiv \frac{Y_t}{2}(1-y) \tag{3.12}$$

yields Equation 3.9.

Figure 3.4 shows the normalized behavior of an idealized current correlator—actual current-correlator circuits deviate due to the κ effect and transistor mismatch.

3.4 The Translinear Principle

The translinear principle is a powerful technique for analyzing quickly circuits whose basic elements exhibit an exponential relationship between current and voltage. Originally conceived by Gilbert [20] using BJTs, the technique is equally valid for subthreshold FETs.

3.4.1 The κ Effect

One important concern when using FETs is the inability of the gate to determine directly the barrier height at the channel as is the case for BJTs. If each FET is placed in its own well with its source tied to the well, the translinear principle can be applied to these circuits. For all other FET circuits run in the subthreshold regime of the FET, the presence of a nonunity κ necessitates a modification in the application of the translinear principle. The modification is elucidated in an example in Section 3.4.2.

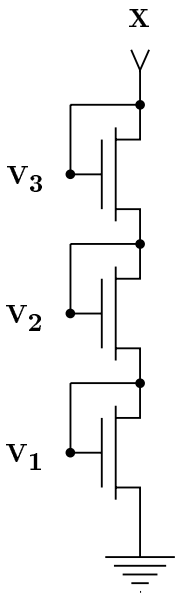


Figure 3.5 The diode-stack circuit. The diode-stack circuit receives a current input x and the resulting voltages V_1 through V_3 are analyzed.

3.4.2 The Diode-Stack Circuit

Figure 3.5 shows a circuit that I analyze using a modified translinear principle. When analyzing one of the diode stacks, we get the following relationships:

$$\begin{aligned} \ln(X^{1/\kappa}) &= V_1, \\ \ln(X^{1/\kappa+1/\kappa^2}) &= V_2, \\ \ln(X^{1/\kappa+1/\kappa^2+1/\kappa^3}) &= V_3. \end{aligned} \tag{3.13}$$

We introduce a function $\sigma(n,\kappa)$, where n is the number of transistors in the diode stack:

$$\sigma(n,\kappa) = \frac{1}{\kappa} + \frac{1}{\kappa^2} + \cdots + \frac{1}{\kappa^n} = \frac{1-\kappa^n}{(1-\kappa)\kappa^n}. \tag{3.14}$$

Now, from Equation 3.13, using the function $\sigma(n,\kappa)$, we see that $X^{\sigma(n,\kappa)}$ describes the relationship between the voltage and the current at each node—for example, $\ln(X^{\sigma(1,\kappa)}) = V_1$. Thus, the application of the translinear principle is the same as in Section 3.3, but now we raise the current variable to the function $\sigma(n,\kappa)$, rather than squaring or cubing the current variable.

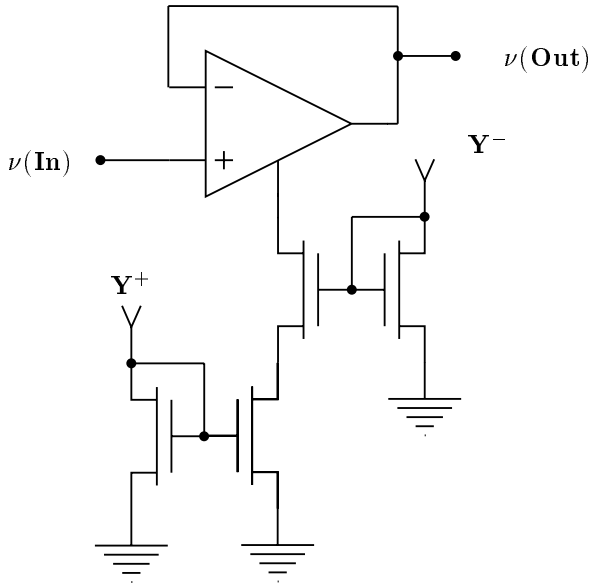


Figure 3.6 The current-correlator follower. The current-correlator follower has an input voltage, $\nu(In)$, which is followed by the output $\nu(Out)$. The bias current of the follower is the correlation of Y^+ and Y^-.

3.5 Other Current-Mode Circuits

In this section, I briefly introduce three new current-mode circuits. The current-correlator transconductance amplifier (used here as a follower), and the four-quadrant sine-approximation circuit use the current-correlator circuit discussed in Section 3.3. The inverted-differential-pair circuit is an alternative to the differential-pair circuit discussed in Section 3.2.

3.5.1 The Current-Correlator Follower

In Section 4.3, I describe using a follower that is biased by a current correlator, as shown in Figure 3.6. This follower, called the **current-correlator follower**, follows $\nu(In)$ when the difference in the control input, $Y^+ - Y^-$, is small. When this difference becomes larger, the current-correlator follower has less current drive to follow. The amount of drive current available to this follower is given in Equation 3.9, and is shown in Figure 3.4.

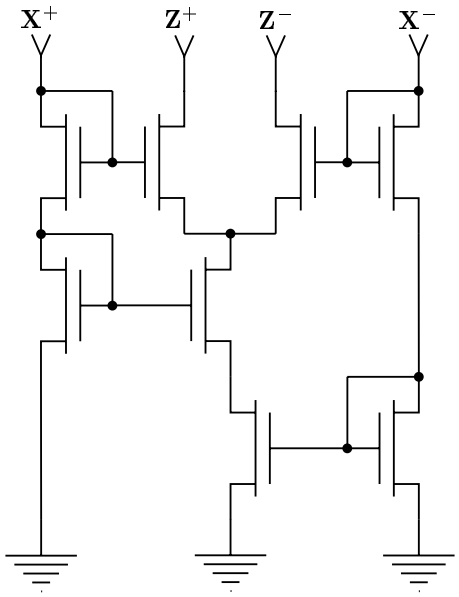


Figure 3.7 The four-quadrant sine-approximation circuit. This circuit receives inputs through X^+ and X^- and provides outputs through Z^+ and Z^-. The output, z, differs from an ideal sine by approximately 2 percent.

3.5.2 The Four-Quadrant Sine-Approximation Circuit

The four-quadrant sine-approximation circuit is discussed in detail in Appendix A; I simply introduce it here. Using the current correlator discussed in Section 3.3 as the bias to a

differential pair, we can obtain an approximation to the sine function. Figure 3.7 shows the sine circuit. The maximum deviation from an ideal sine is about ±2 percent. Inputs to the sine are X^+ and X^- and the outputs are Z^+ and Z^-.

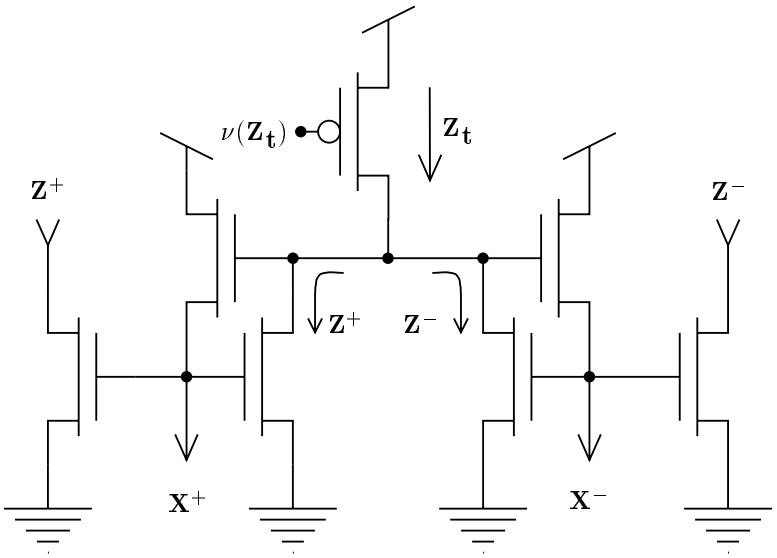


Figure 3.8 The inverted-differential-pair circuit. The inputs are X^+ and X^- and the outputs are Z^+ and Z^-. This circuit is nice because the output-current direction is compatible with the input; the circuit can be cascaded easily without using current mirrors.

3.5.3 The Inverted-Differential-Pair Circuit

The inverted-differential-pair circuit is discussed in more detail in Appendix A; I simply introduce it here. Figure 3.8 shows the inverted-differential-pair circuit. The inverted-differential-pair circuit takes its inputs as X^+ and X^-; its outputs are Z^+ and Z^-. The circuit computes

$$z = -x. \tag{3.15}$$

Note that in the normalized differential current mode we can easily perform or negate this operation by simply swapping outputs. This circuit is nice because the output-current direction is compatible with the input so it can be cascaded without using a current mirror—as is required with the standard differential-pair circuit.

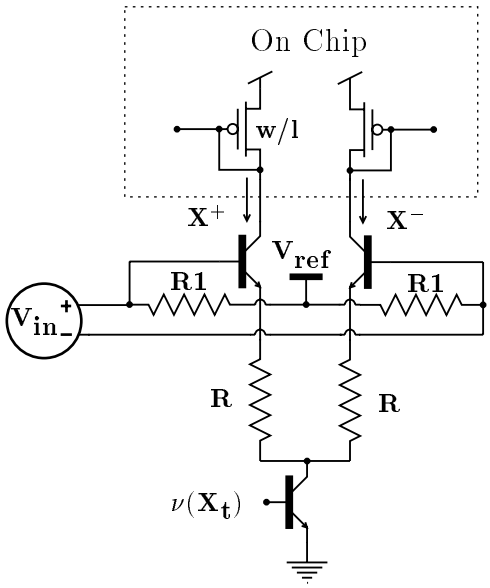


Figure 3.9 Current-input amplifier circuit. This circuit is used to apply input currents. Differential currents, X^+ and X^-, vary nearly linearly with applied voltage, V_{in}. V_{ref} is used to provide a common-mode voltage about which V_{in} swings.

3.6 Current-Mode Test Methods

Various methods are used to apply inputs to and to read outputs from the current-mode circuits. These test methods are discussed here.

3.6.1 Application of Inputs

When testing the behavior of a normalized current-mode circuit, we must provide an input, which is a combination of two differential current signals. These current signals should sum to a constant current, X_t, and the differential current should swing between $\pm X_t$. Figure 3.9 shows the amplifier that satisfies these constraints and provides voltages to be used on the chip. See Horowitz [31] for a complete description of such amplifiers. In addition, this circuit converts a voltage, V_{in}, linearly to the differential input current, $X^+ - X^-$. V_{ref} is used to provide a common-mode voltage about which V_{in} can swing. Figure 3.10 shows data for the relationship between V_{in} and X^+ and X^-. The current mirrors that I put on-chip, as shown in Figure 3.9, have a much larger **width to length** (w:l) ratio than do the transistors that receive the current-mirror voltages, so the off-chip currents can be

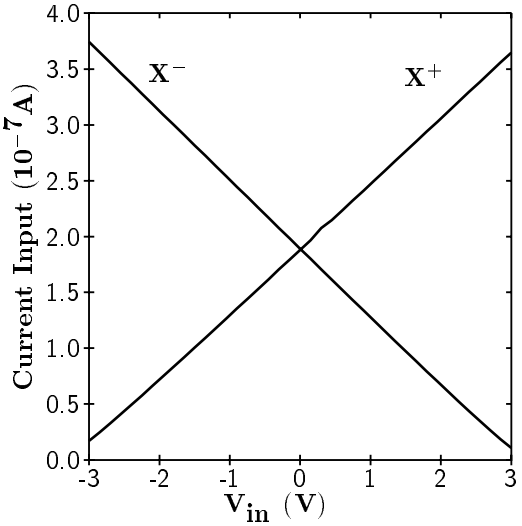


Figure 3.10 **Data of the linear relationship between $V_{\rm in}$ and X^+ and X^-. This data is taken from the current-input amplifier circuit of Figure 3.9.**

larger than the currents used on the chip. I use a w:l ratio of approximately 80 for these mirror devices. These mirrors should be designed to match closely.

3.6.2 Measurement of Outputs

It is necessary to measure current outputs. The most straightforward approach is to measure directly the two branch differential currents, and to compute the normalized current from these two currents. Often, it is necessary to measure dynamics of extremely small currents. In these instances, a simple current amplifier is used to guarantee that the dynamics of the measured currents are a result of the pertinent dynamics, and not of dynamics introduced by the sensing circuitry. Figure 3.11 shows the output current amplifier that I use. At each stage of the amplifier, the current is mirrored and is scaled up. This output amplifier has a calculated gain of 9600 (with $V_{\rm gain} = V_{\rm dd}$). By moving $V_{\rm gain}$ down from $V_{\rm dd}$, we increase the gain as $e^{(V_{\rm dd}-V_{\rm gain})/V_t}$. This result applies only when all the transistors remain in subthreshold, which is true for modest increases in gain as a result of changing $V_{\rm gain}$.

3.6.3 Helpful Hints

Here are several tips for building these input and output test structures.

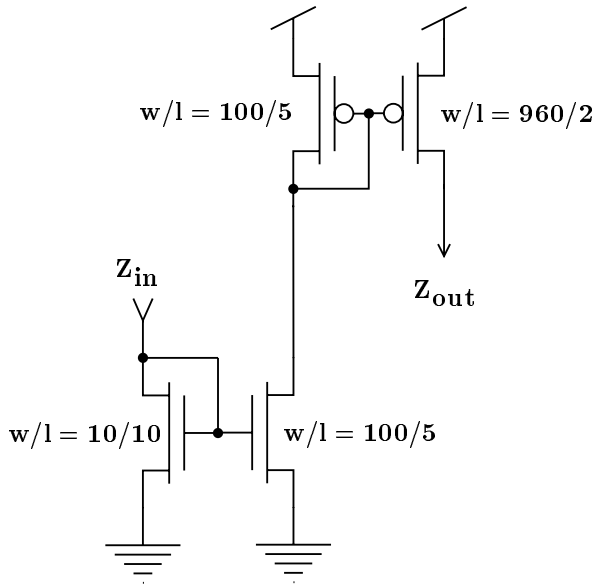


Figure 3.11 **The current-output amplifier circuit. The current-output amplifier is used to measure the dynamics of small currents. The gain of the amplifier shown here (with $V_{\mathrm{gain}} = V_{\mathrm{dd}}$) is 9600. V_{gain} is tied to the source of the p-well transistor with a w:l ratio of 100:5 (shown here as connected to V_{dd}).**

3.6.3.1 Transistor Matching

It is important that the transistors in the input and output test structures match. For good matching, use long and wide transistors. For example, I did not have good matching in my output amplifier in Figure 3.11, because the output transistor (w:l=960:2) is only 2 μm long. A longer transistor would have provided much better matching of my output data. Also, physically locate test amplifiers close together. These are basic tips for transistor matching. Other, more sophisticated techniques, can be used if even better matching is required [51, 52, 68, 33].

3.6.3.2 Current-Gain Measurement

It may be important to know the gain of the test amplifiers. If it is, provide a means for measuring gain explicitly. I did not provide such a means, and needed to infer the gain from the layout of the chip—an inaccurate method of figuring the gain.

Chapter 4

Adaptive Synaptic Unit

Gie me ae spark o' Nature's fire,
That's a' the learning I desire.

Robert Burns
Epistle to John Lapraik, 1786

This chapter introduces a synapse circuit, shown in Figure 4.2, that operates in the subthreshold regime of the MOSFET devices. This circuit is used as a programmable weight in the artificial neural-network system described in Chapter 6. It is an example use of the UV-activated structures and device physics described in Chapter 2. This synapse is part of a larger system that requires continuously adjustable nonvolatile synaptic weights, continuous weight decays with a slow-time course, and an uninterrupted multiplication operation, while the weight is trained or is allowed to decay.

The synapse circuit performs several tasks. It stores weights differentially (W^+ and W^-) on floating nodes. It performs a multiplication of the stored weight with an externally

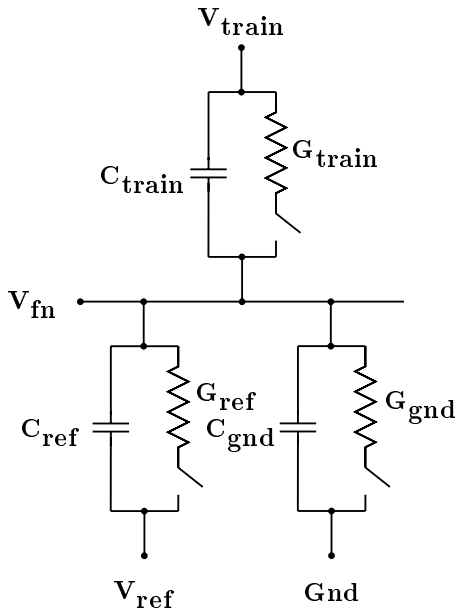


Figure 4.1 **Conductance and capacitance model of the synapse. Conductive and capacitive coupling as seen by the floating nodes, W^+ and W^-. $V_{\rm fn}$ is the voltage at the floating node. $V_{\rm train}$ is the training signal. $V_{\rm ref}$ is an explicit reference, which is the voltage applied to a well directly below the floating nodes. The floating nodes are completely contained within the boundaries of this well; $G_{\rm ref}$ is the conductance to this well. The conductances and switches shown are UV-activated conductances. The switches are closed when UV is present and open otherwise.**

supplied input (ΔX). In addition, the synapse provides the training circuitry to automatically alter its weight through UV-activated conductances. I consider each of these functions in this chapter.

4.1 Weight Storage

The two nodes, W^+ and W^- in Figure 4.2, represent floating nodes, whose difference, $\Delta W \equiv W^+ - W^-$, is used to define a weight. These two nodes are referred to as the **weight nodes.** Each weight node is coupled capacitively and conductively to three nodes, $V_{\rm train}$, $V_{\rm ref}$, and Gnd; see Figure 4.1. The values for the capacitances are shown in Table 4.1. The synapse is designed to have a relatively large UV-activated conductance from $V_{\rm train}$ to $W^+$and a much smaller UV-activated conductance from $V_{\rm train}$ to W^-; in the presence of UV light, a positive $V_{\rm train}$ (with respect to W^+) causes W^+ to program toward $V_{\rm train}$ more

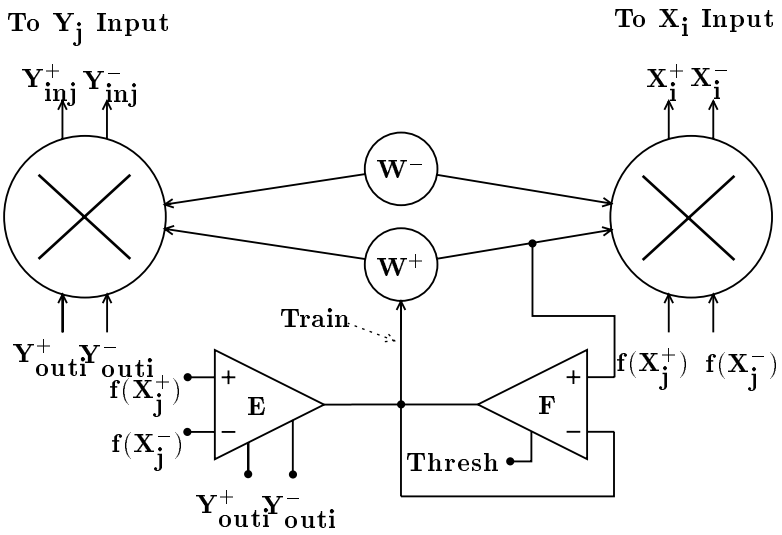


Figure 4.2 **The entire adaptive learning synapse. The weights are stored on floating nodes labelled W^+ and W^-. The outputs of the feedforward units, $f(X^+) - f(X^-)$, and the error units, $Y^+ - Y^-$, are multiplied by the weight. The adaptation circuitry provides an increment or decrement training signal, Train, to the positive weight node, labelled W^+, or a hold signal, F, which freezes the weight.**

quickly than W^- approaches V_{train}. These differing conductances provide a means for increasing or decreasing ΔW. The V_{train} to W^+ conductance was made large by placing the overlap between the floating node (poly1 layer) and training node (poly2 layer) directly under a window in the metal2 layer.[1] The V_{train} to W^- conductance was made smaller by hiding the overlap approximately 10 μm from a window in the metal2 layer. The synapse is also designed for equivalent capacitances from V_{train} to both W^+ and W^- to avoid changes in ΔW for large swings in the training signal.

[1] Layer names are those used by MOSIS [59].

4.1.1 Weight Dynamics

The positive weight, W^+, trains as

$$W^+(t) = V_{\text{train}} + (W_0 - V_{\text{train}}) \exp(-t/\tau_{train}),$$

where W_0 is the initial weight and $\tau_{train} = C_{total}/G_{train}$ is the time constant of the training synapse. The weight decay dynamics, which govern both weights, are

$$W(t) = W_0 \exp(-t/\tau_{ref}) + V_{ref},$$

where τ_{ref} is determined by the UV-activated conductance, G_{ref}, and V_{ref} is a reference voltage to which each weight will eventually decay. In this synapse, the weight decay was explicitly added to cause the weights to decay toward a reference when not being trained.

The floating nodes used as the weights are implemented in the poly1 layer. The floating nodes sit above a well which is driven to V_{ref}. If the synapse is created with an n-well process, each of the weights will decay to V_{ref} (assuming all parasitic conductances are negligible). On the other hand, if the process is p-well, the built-in barrier between the degenerately doped n-polysilicon and p-well is about 1.1 V and the floating polysilicon nodes will decay to a voltage offset of about 1.1 V. The voltage offset can be different from 1.1 V due to charge trapped in the oxide. Figure 4.3 shows the dynamics of a weight while being trained. The UV light is on for the entire trace. The trace shows the weight first training down and then up. For this trace, W^+ and W^- were initially set to 2.0 V and 1.95 V, respectively. During the relatively steady output of the weight, the follower F in the synapse circuit dominates the training amplifier and only the explicit decay causes the ΔW to decay. The time constant for programming, τ_{train}, is approximately 6400 seconds. The time constants for weight decay, τ_{ref}, was greater than 50 000 seconds.

The ratio of training time constant to decay time constant is easily adjusted during the design phase by the two methods described in Chapter 2. In the design of this synapse, I use the method of hiding the decay conductances under the metal shield and making the lengths of these structures shorter than the training structures.

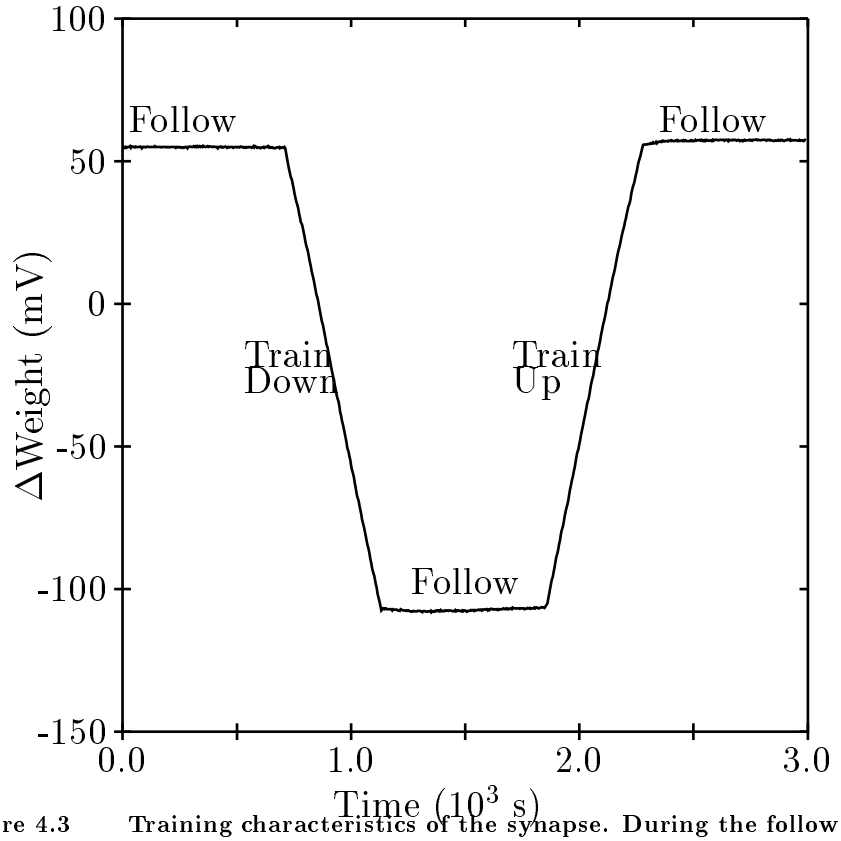


Figure 4.3 Training characteristics of the synapse. During the follow periods, the weights decay to a reference voltage with a time constant $\tau_{ref} \approx 50\,000$ sec. During the train periods, the positive weight node programs up or down with a time constant $\tau_{train} \approx 6400$ sec.

4.1.2 Measurement of Capacitive Coupling

The floating structure is coupled both capacitively and conductively to various places. Figure 4.1 shows the floating node, $V_{\rm fn}$, coupled to three nodes, $V_{\rm train}$, $V_{\rm ref}$, and Gnd. With the switches to the conductances in Figure 4.1 open (that is, the UV source is off), $V_{\rm fn}$ can be computed as

$$V_{\rm fn} = \frac{C_{\rm train}V_{\rm train} + C_{\rm ref}V_{\rm ref} + C_{\rm gnd}Gnd + Q_{fn}}{C_t}, \tag{4.1}$$

where C_t is the total capacitance at the floating node and Q_{fn} is the charge stored on the floating node. Equation 4.1 can be used to measure the capacitance to the various coupling nodes. For example, to determine $C_{\rm train}$, hold $V_{\rm ref}$ and Gnd constant and measure $V_{\rm fn}$ while varying $V_{\rm train}$. The slope of the $V_{\rm fn}$ versus $V_{\rm train}$ curve is $\frac{C_{\rm train}}{C_t}$. Table 4.1 shows the capacitance between the floating weight nodes and $V_{\rm train}$, $V_{\rm ref}$, and Gnd found by this method. The capacitance to Gnd is calculated by measuring all other capacitances and

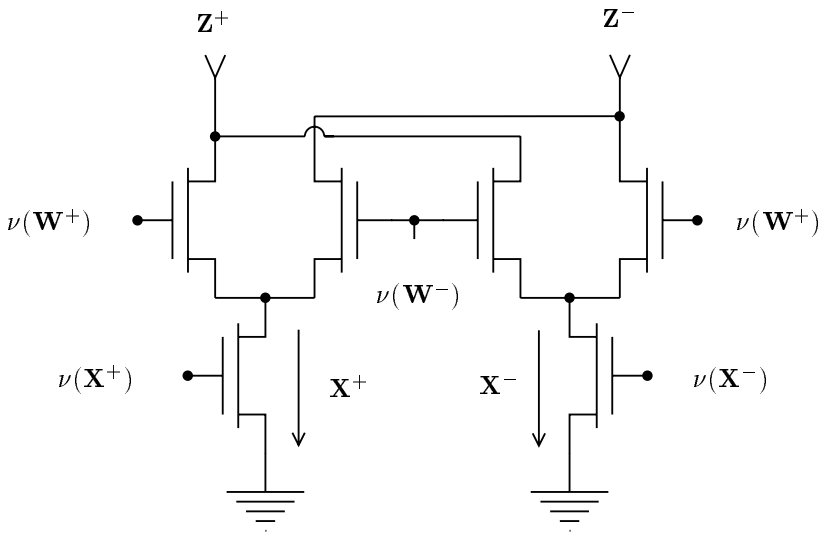


Figure 4.4 Multiplier circuit used in the learning synapse. This circuit is shown schematically as the large circles in Figure 4.2. The weights are stored on nodes W^+ and W^-. Inputs enter through X^+ and X^-; Z^+ and Z^- are the outputs. In the normalized form, the circuit performs $z = wx$.

Node	W^+ (C/C_t)	W^- (C/C_t)
V_{train}	0.109	0.114
V_{ref}	0.156	0.172
Gnd	0.735	0.714

Table 4.1 Capacitive coupling ratios from the floating node (W^+ or W^-) to the various nodes of the synapse (V_{train}, V_{ref}, and Gnd). The capacitive coupling ratios are measured values.

assuming the remaining capacitance is to Gnd.

4.1.3 Zero Weights

With this synapse, it is very difficult to achieve zero weights (that is, zero difference between W^+ and W^- in Figure 4.4). Several methods are possible to zero the weights; by turning the follower bias on hard in Figure 4.2, the training signal follows W^+. By shining UV on the chip and allowing W^+ and W^- to reach equilibrium, the weight, $W^+ - W^-$, should approach zero. The problem with this approach is that any small conductance difference between W^+ and W^- to $V_{\rm ref}$ and Gnd causes the weight nodes to reach equilibrium at very different values. Another method is to tie $V_{\rm dd}$ and $V_{\rm ref}$ to Gnd and allow each weight node to relax. They both relax to Gnd minus any difference in flatband voltage between the poly floating node and the well[2] and minus any charge trapped in the oxide. Then $V_{\rm dd}$ and $V_{\rm ref}$ are brought back to their respective operating levels. The problem with this method is any difference in capacitive coupling from these nodes to the weight nodes causes the weight nodes to be capacitively moved to slightly different values. The inability to zero the weight effectively and conveniently remains the single largest problem with the design of this synapse and certainly requires attention.

4.2 The Multiplying Differential-Pair Synapse

The basic operation of the circuit (see Figure 4.4) is to multiply a differential voltage, $\Delta W = W^+ - W^-$, by a differential current, $X^+ - X^-$, using a Gilbert multiplier [44]. In the normalized differential current mode it computes

$$z = wx, \tag{4.2}$$

where z is the normalized output, x the normalized input, and w the normalized weight.

Figure 4.5 plots the measured transfer characteristics of the multiplier for various ΔW. The crowding of the most extreme curves indicates saturation of the weight parameter

[2] This difference is small for an n-well process and much larger for a p-well process given that the poly silicon is n-type degenerately doped silicon.

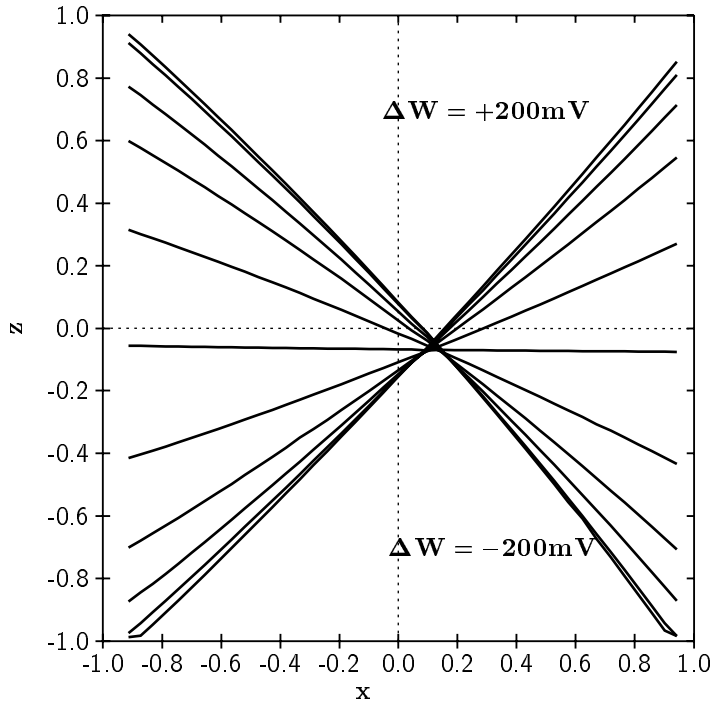


Figure 4.5 **Data of the multiply characteristic of the synapse.**

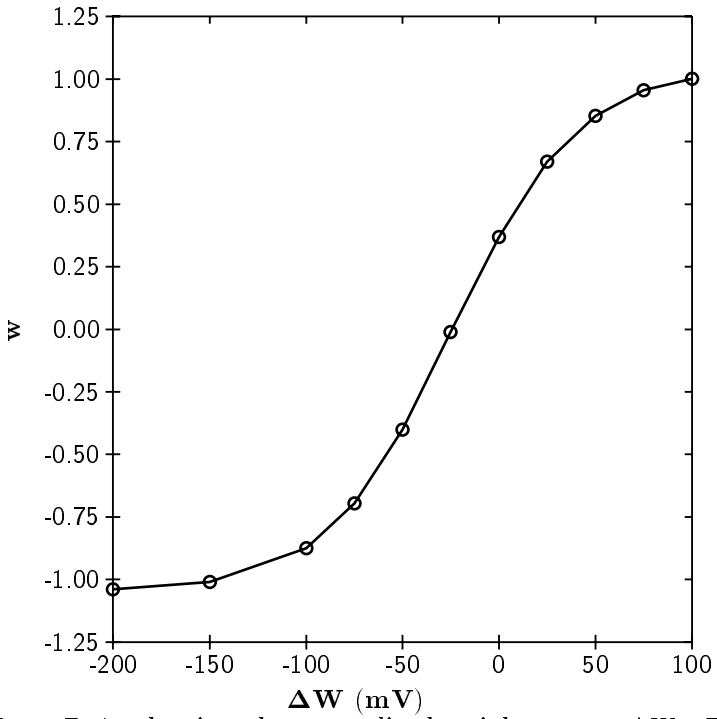


Figure 4.6 **Data showing the normalized weight versus ΔW. This curve clearly shows the saturation of the weight.**

ΔW, which ranges from -200mV to +200mV in this figure.[3] The shift of the characteristics from the origin is a result of the test-input-mirror offsets and the test-output-mirror offsets. The offsets due to the multiplier are small (less than 5 percent). Offsets are reduced from previous designs by using larger than minimum-size transistors in the multiplier [52, 68] (for example, the FETs used in this multiplier are 10x10 λ). Lower offsets are also achieved because a differential current is output instead of a single, bipolar current, where the two currents of the Gilbert multiplier are locally subtracted via a current mirror, which introduces additional offsets.

Figure 4.6 shows the relationship between the applied ΔW and the resulting normalized weight of the synapse multiplier. Notice the weight begins to saturate at about $\pm 100mV$.

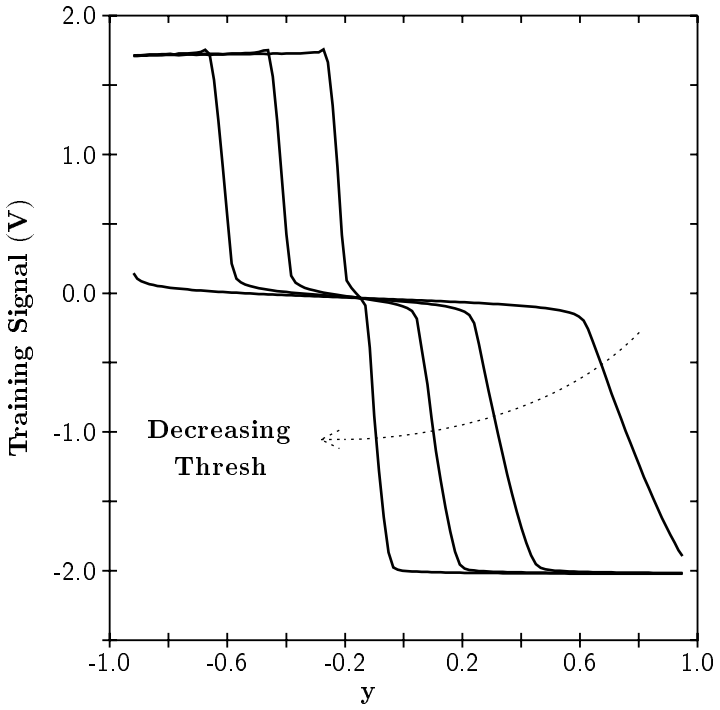


Figure 4.7 Data showing the training signal as a function of y. The training signal is shown as $V_{\text{train}} - W^+$. $f(x)$ is held at a negative constant for these traces. This family of curves is obtained by changing follow threshold.

[3] A detailed description of the offsets in the multiplier characteristics is given by Mead [44].

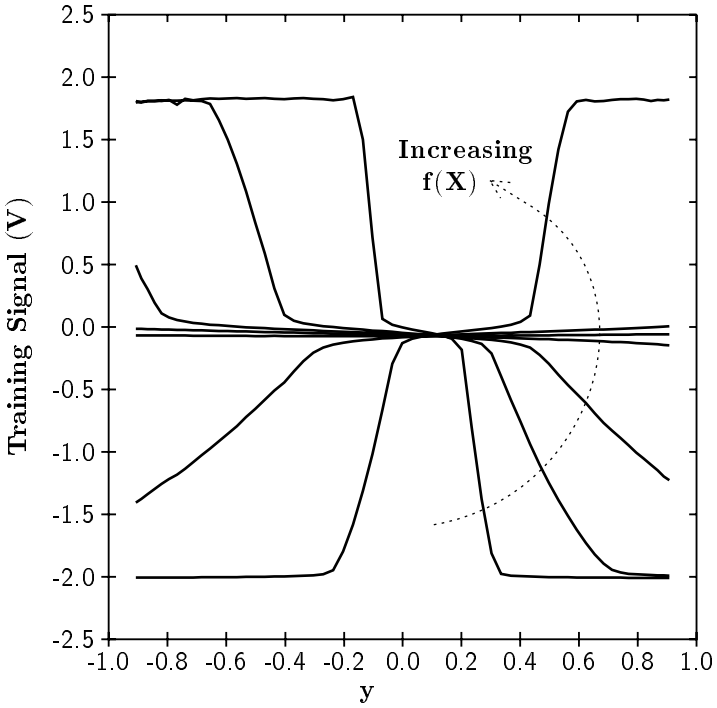


Figure 4.8 Data showing training signal as a function of y. The training signal is shown as $V_{\text{train}} - W^{+}$. Follow threshold is held constant. This family of curves is obtained by changing $f(x)$.

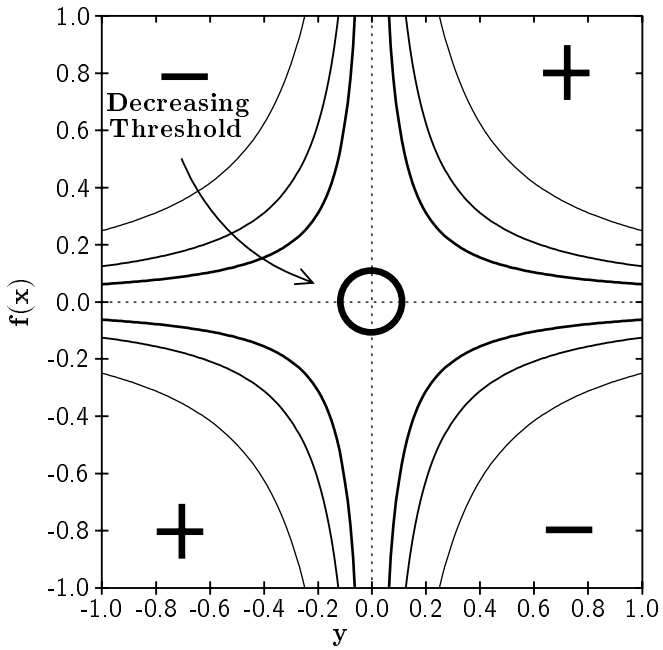


Figure 4.9 Contour plot showing the training signal. The contour plot shows $f(x)$ versus y. Contour shows ideally $|f(x) \cdot y| = \theta$ for $\theta = 1/4, 1/8, 1/16$ as shown. Positive weight changes occur in the area labelled plus, negative weight changes shown with a minus and no weight changes shown with a zero.

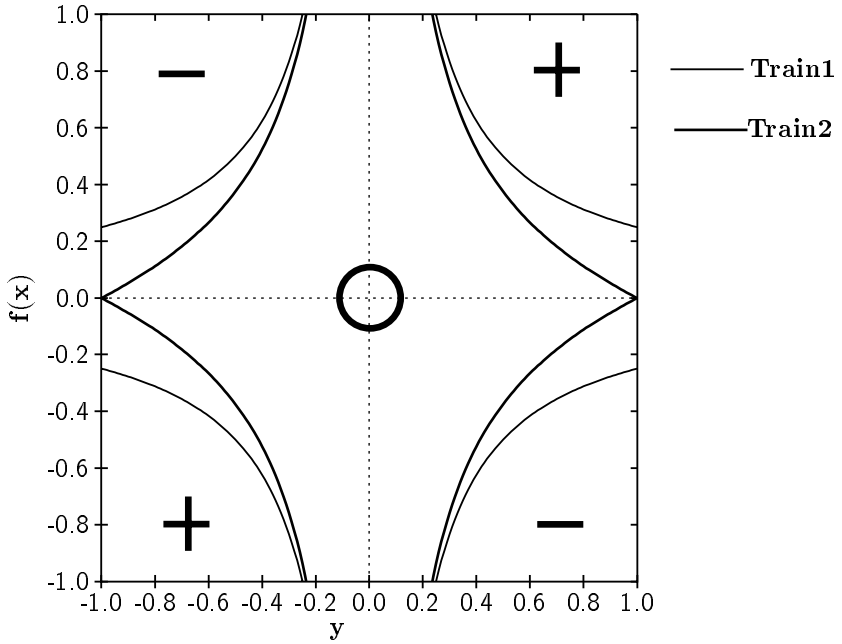


Figure 4.10 **Contour plot showing the follow-with-error training signal. The contour plot shows $f(x)$ versus y. Contour shows ideally $f(x) \cdot y = \theta$ for $\theta = 1/4$, for the curve labelled Train1 and $\theta = 1/4(1 - y^2)$ for Train2. Positive weight changes occur in the area labelled plus, negative weight changes shown with a minus and no weight changes with a zero.**

4.3 Training Circuitry

Referring back to Figure 4.2, training is achieved by making the training node larger (smaller) in voltage than W^+ for a positive (negative) weight change. Two subcircuits control the training node. These subcircuits are denoted by E and F in Figure 4.2. The output of E is the multiplication of two signals, $f(x)$ and y. In the learning network architecture (see Chapter 6), $f(x) \cdot y$ represents the negative gradient of an error signal in weight space. The output of E is opposed, though, by the output of F, which attempts to follow W^+. Thus if the gradient is large enough (in magnitude), it overpowers the follower output and the weight changes in the direction of gradient-descent. Otherwise, if the gradient signal is too small (in magnitude), the weights are not changed in direction of gradient-descent, but rather decay slowly to the reference. Figure 4.7 shows the training signal for various follow thresholds as a function of y with $f(x)$ held at a negative constant. As the threshold is turned up, larger gradient signals are needed to 'overpower' the follower. Including the follower guarantees that offsets in the gradient signal calculation do not cause the weight to be trained in the wrong direction. Figure 4.8 shows the behavior of the training signal as a function of y for various $f(x)$ with the follower threshold held constant. The idealized function which models the training signal is labelled here tsgn for thresholded signum, which obeys,

$$\text{tsgn}(z,\theta) \equiv \begin{cases} -1 & \text{if } z < -\theta \\ 0 & \text{if } -\theta \leq z \leq \theta \\ 1 & \text{if } z > \theta. \end{cases} \tag{4.3}$$

In the circuit analyzed here, the training signal is

$$Train1 = \text{tsgn}(f(x) \cdot y, \text{follow threshold}). \tag{4.4}$$

Figure 4.9 is an ideal two-dimensional contour plot of this learning signal. Shown are three different curves for different values of threshold. The values used are $\theta = 1/4$, $1/8$, and $1/16$. The plus signs show areas of positive weight changes, the negative signs shows areas of negative weight changes, and the area labelled by zero shows an area of no weight change.

4.3.1 Bang-Bang Learning

When the training amplifier 'overpowers' the follower, depending on the sign of $f(x) \cdot y$, V_{train} either rails to V_{dd} or Gnd. V_{train} is only lightly capacitively coupled to both nodes W^+ and W^- (see Table 4.1), and therefore a relatively large voltage drop occurs across C_{train} in the training mode. This allows W^+ to train towards V_{train} through G_{train}, whereas W^- has no such conductance and remains fixed in training mode. In this way, the sign of $f(x) \cdot y$ determines whether the weight is increased or decreased. The amplitude of the weight change is determined by the length of time $f(x) \cdot y$ is asserted. During the training process, both nodes W^+ and W^- are changed capacitively by $\Delta V_{\text{train}}\ C_{\text{train}}/C_{total}$, where $C_{total} = C_{\text{train}} + C_{\text{ref}} + C_{\text{gnd}}$. The ratio $C_{\text{train}}/C_{total}$ for my layout is 0.109. Thus, for a ΔV_{train} of 2.5 V, the common-mode change on the weight input is approximately 272 mV.

4.3.2 Follower Mode

The adaptation of this synapse can be shut off electrically and more permanently by shutting off the UV source. The ability to shut off the training electrically is important to many applications. When this synapse is not being trained, the follower (see Figure 4.2) 'overpowers' the training amplifier and the weight remains unchanged. V_{train} then follows W^+ and ideally no voltage drop occurs across the conductance G_{train}. In actuality, there will be a small voltage difference ($< 20mV$) across G_{train} due to offsets in the follower amplifier, generating a current which causes W^+ to drift slowly. Two properties limit this drift. First, there is a decay conductance, G_{ref}, to a reference, V_{ref}. This conductance limits W^+ to $V_{offset} \cdot \frac{G_{\text{train}}}{G_{\text{ref}}} + V_{\text{ref}}$, where V_{offset} is the offset in the follower. Second, such small voltage drops across UV conductances have a lower conductance than larger drops (see the nonlinearity in Figure 2.4), causing the leakage current to be smaller. This lower conductance can actually decrease the offset by a factor of two to three.

4.3.2.1 How UV affects the follower

With the UV source on, leakage due to UV-generated minority carriers causes the follower to follow as if it had a bias of 0.34 V (that is, the leakage becomes dominant in the follower for a bias less than 0.34 V). This leakage underscores the need to consider the effects of minority-carrier generation on sensitive circuits. In the case of the follower circuit, this

effect simply means the follow mechanism is on at a higher minimum value; this leakage causes no problems for the circuit discussed here. For other circuits, minority carriers may be troublesome indeed necessitating a metal shield over sensitive areas.

4.3.3 Follow with Error

An alternative algorithm for control of the training signal is also used. Instead of using a follower to shut off training when the gradient information is too small, a current-correlator follower is used; see Section 3.5.1 for a description of the current-correlator follower. The function E and F now compute is

$$Train2 = \text{tsgn}(f(x) \cdot y, 1/4(1 - y^2)), \tag{4.5}$$

shown as a contour plot in Figure 4.10, where this function is compared with Equation 4.4 for a constant threshold of $\theta = 1/4$. Now the threshold is a function of the squared error. For errors near zero, the threshold is maximum and relatively large gradient signals are necessary to drive training. For large errors, only relatively small gradient signals are needed. In this way, the threshold for training is not fixed but is a function of the error.

4.3.4 Common-Mode Rejection

When the synapse is performing its multiplication function, a quantity of concern is how much the multiplication is affected by the training node swinging from Gnd to V_{dd}. From Table 4.1, we notice there is a slightly different capacitive coupling of the training node to the plus and minus floating weight nodes (a -0.6 percent difference). This difference in coupling accounts for the -30 mV change we see when changing V_{train} from Gnd to V_{dd}. What is of interest, though, is how much this affects the normalized weight. From Figure 4.6, we expect a constant change in ΔW mV to result in the largest normalized weight change around a weight of zero and this change drops off for larger weights (in magnitude). For two weights measured, the normalized weight change around zero weight was 22.1 percent and a 5.94 percent change around a normalized weight of -0.8. The largest swing experienced by a weight of 22 percent is quite large and should be reduced by either lowering the capacitive coupling from V_{train} to the floating nodes or by designing the capacitive coupling from V_{train}

to the floating nodes to match more closely.

Chapter 5

Sigmoidal and Error Units

A little learning is a dangerous thing;

Alexander Pope
An Essay on Criticism, 1711

Neural networks can compute very complex nonlinear mappings. They owe this ability in part to the nonlinear transfer function of the processing element or sigmoidal unit. Typically, these sigmoidal units are compressive, monotonically increasing, saturating functions of their inputs. I have designed and tested a circuit that has a sigmoidal transfer characteristic. In contrast, the units of the error-processing layer in the Pineda algorithm have a linear transfer function. Their output is modulated by the derivative of the sigmoid (see Equation 1.2); this derivative is a "bump function" of the input; that is, the derivative of a sigmoidal function is maximum near zero input and minimum at the two extremes—resembling the generic shape of a bump.

The sigmoidal (X units) and the linear-error units (Y units) are self-scaling relative to their inputs. If more synapses are added, the output currents of the other units are adjusted

(or adapted), different levels of external input are applied, or the chip heats up, the total level of input current changes, but the output does not; the differential current divided by the total is the only relevant quantity to the sigmoidal and error units. This self-scaling feature is a property of normalized differential current mode circuits.

This chapter presents the normalized current-mode sigmoidal circuit I use in the neural-network analog VLSI chip. I analyze the affect of κ on the circuit, show how the sigmoid uses a normalized input, show how the gain of the sigmoid is changed at design time and at run time, and show how the output of the sigmoid can be scaled. I also present the implementation of the error-layer processing element and show how this circuit operates in the normalized mode. Finally, I analyze the dynamics of these circuits and analyze how they differ from simple **resistance–capacitance** (RC) dynamics.

5.1 Forward Propagation: The Sigmoidal Unit

The sigmoidal circuit performs a monotonically increasing, compressive, saturating function on its input. The i^{th} unit in the X layer performs the following function in steady state:

$$x_i = \sum_{j \neq i} w_{ij} f(x_j), \tag{5.1}$$

where x_i is the normalized input to the i^{th} sigmoid and $f(x_j)$ is the output of the j^{th} sigmoidal unit. The full description of this equation is given in Chapter 6. This section deals with the function $f(x)$, which increases monotonically and saturates at both extremes of input.

5.1.1 Sigmoidal-Circuit Implementation

The sigmoid circuit uses the differential pair described in Chapter 3 and uses a stacked-diode pair or triplet as the input; see Figures 5.1 and 5.2. For each of these circuits, the equation describing the output of the sigmoid is given by

$$z = f(x) = \frac{(1+x)^\beta - (1-x)^\beta}{(1+x)^\beta + (1-x)^\beta}, \tag{5.2}$$

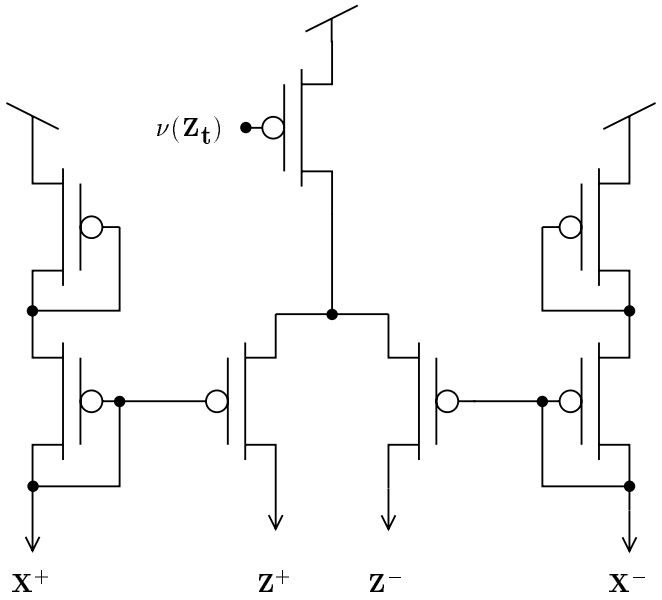


Figure 5.1 Diode-pair sigmoidal circuit. The normalized input is $x \equiv \frac{X^+ - X^-}{X^+ + X^-}$. The normalized output is $z \equiv \frac{Z^+ - Z^-}{Z^+ + Z^-}$. Using these normalized variables, z is a sigmoidal function of x with slope of two at the origin.

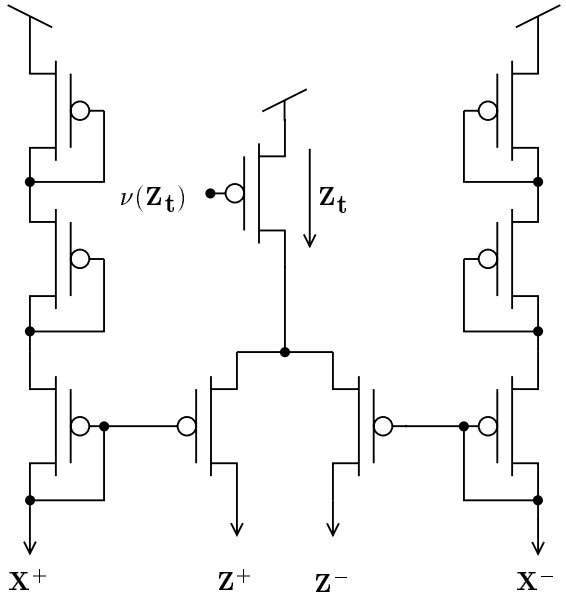


Figure 5.2 Diode-triplet sigmoidal circuit. Inputs enter through X^+ and X^-, and outputs are Z^+ and Z^-. This circuit also yields a sigmoidal function, but with a slope of three at the origin.

where for the two-diode case, $\beta = 2$; where for the three-diode case, $\beta = 3$; and where for n-diodes, $\beta = n$. Of course, for n-diodes, care must be taken to avoid running out of "head room" (that is, there is a limited voltage difference between $V_{\rm dd}$ and Gnd, and each diode in the stack requires part of this voltage difference).

Figure 5.3 shows these ideal sigmoids for both the diode pair and diode triplet. The diode-triplet curve has the higher gain. Shown in Figure 5.4 is the case for 1, 2, 3, and 10 diodes. Notice that the one-diode case is nothing more than the differential pair we analyzed in Section 3.2; with $\beta = 1$ in Equation 5.2, we get $z = x$ as expected.

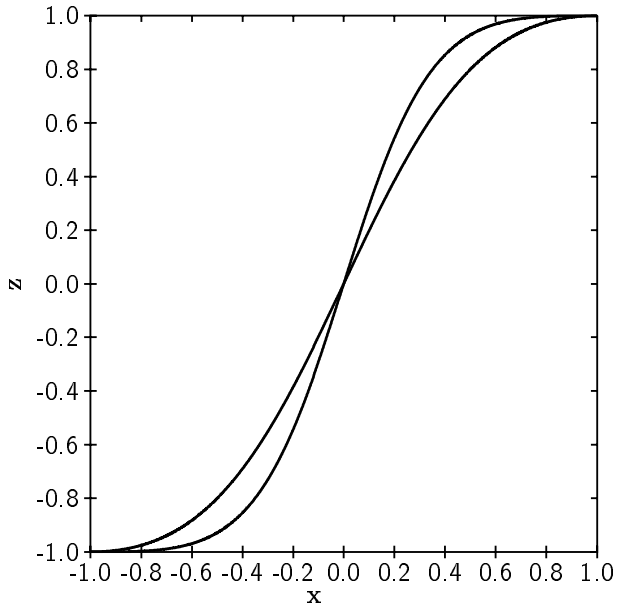


Figure 5.3 Ideal sigmoidal behavior for diode-pair and diode-triplet circuit. Transfer curves calculated from Equation 5.2 with $\beta = 2,\ 3$. The gain of each curve around $x = 0$ is β.

5.1.1.1 Effect of Kappa

When we actually measure the behavior of one of these circuits, we are quickly confronted with the inadequacy of Equation 5.2 in describing the output of the sigmoid circuit. Figure 5.5 shows a fit to data taken from the diode-triplet sigmoid of Figure 5.2 using Equation 5.2 with $\beta = 3$. The inadequacy of the fit results from the idealized equation, Equation 3.2, used to describe the subthreshold behavior of a transistor. A better equation for the FET in saturation is Equation 3.3. Using Equation 3.3, we can solve the diode-pair and diode-triplet cases for the sigmoid circuit. What we once again obtain is Equation 5.2,

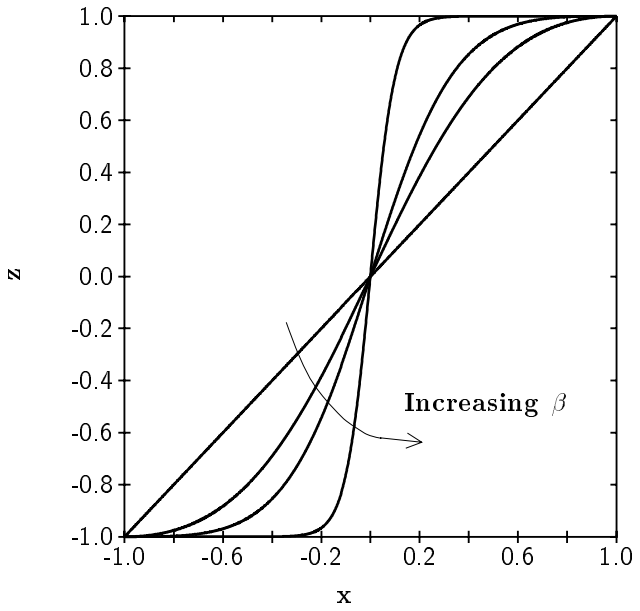


Figure 5.4 **Range of ideal sigmoidal curves. Ideal sigmoids obeying the equation $z = \frac{(1+x)^\beta - (1-x)^\beta}{(1+x)^\beta + (1-x)^\beta}$ for β = 1, 2, 3, and 10, where β = 10 is the highest gain and β = 1 is the lowest gain and is simply a normalized differential current-mode current mirror.**

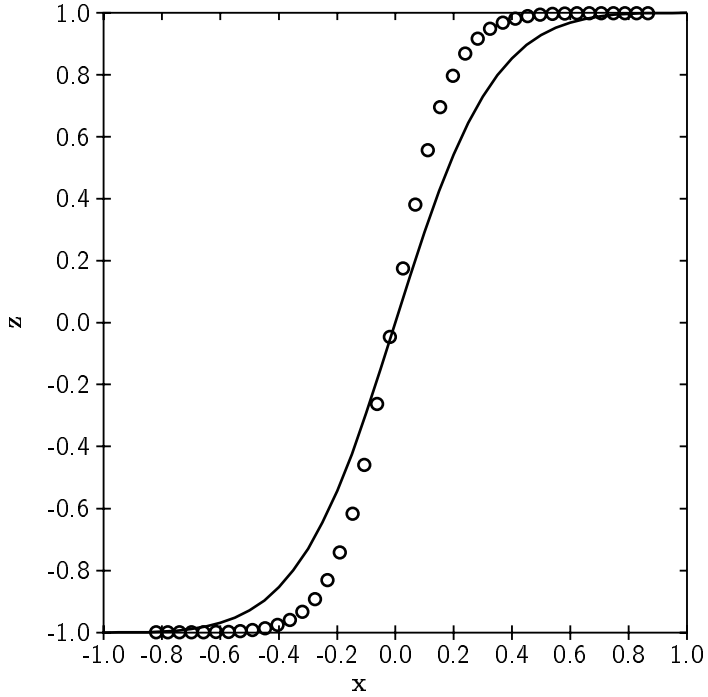


Figure 5.5 **Bad fit to diode-triplet sigmoid. Shows data of the diode-triplet sigmoid with $\nu(Z_t)$ =0.64V referenced down from $V_{\rm dd}$ and shows fit to the theoretical curve with $\kappa = 1.0$. The fit is not very good, because of the κ effect.**

but now β is a function of κ as well as n. For the diode pair, $\beta = \frac{\kappa+1}{\kappa} = \kappa\sigma(2,\kappa)$ (see Equation 3.14) and for the diode triplet, $\beta = \frac{\kappa^2+\kappa+1}{\kappa^2} = \kappa\sigma(3,\kappa)$. In general,

$$\beta(n,\kappa) = \frac{\kappa^{(n-1)} + \cdots + 1}{\kappa^{(n-1)}} = \kappa\sigma(n,\kappa), \tag{5.3}$$

where n is the number of diodes in the stack and σ is given by Equation 3.14. Figure 5.6 shows a fit to the diode-triplet sigmoidal circuit with $\kappa = 0.65$ and $\beta = 4.9$. Using the new definition for β as in Equation 5.3, Equation 5.2 fits the sigmoid quite adequately.

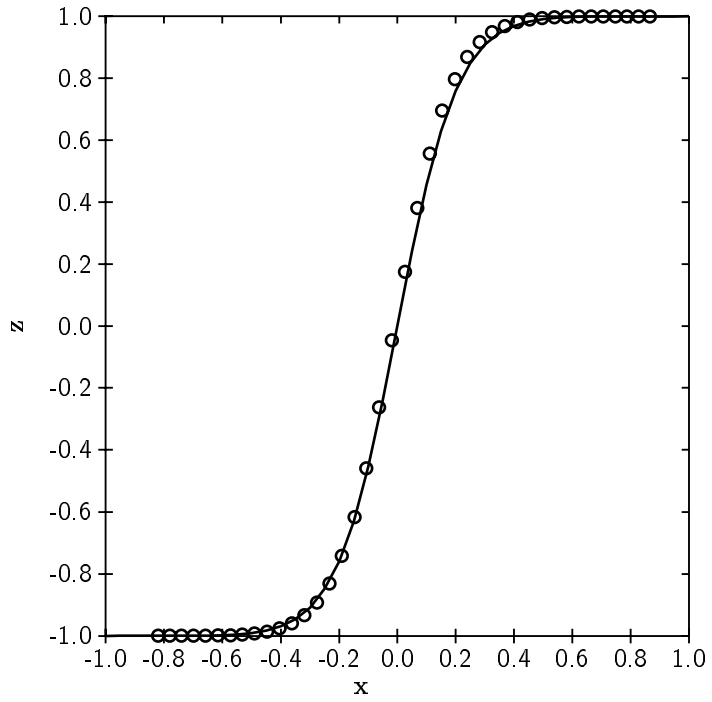


Figure 5.6 Better fit to diode-triplet sigmoid. Shows data of the diode-triplet sigmoid with $\nu(Z_t)$ =0.64V referenced down from $V_{\rm dd}$ and shows fit to theoretical curve with κ = 0.65

5.1.1.2 Normalized Operation

The sigmoid, as implemented here, is invariant to the scale of the input. Equation 5.2 is a function of the normalized input, and is independent of the absolute current level. Figure 5.7 shows the sigmoid characteristic for two levels of current input. The two levels of input current are 12.2 nA for Sig1 and 2.33 nA for Sig2. The shift in the sigmoids is due to an offset in the current-mirror pads which are feeding this test circuit (the input

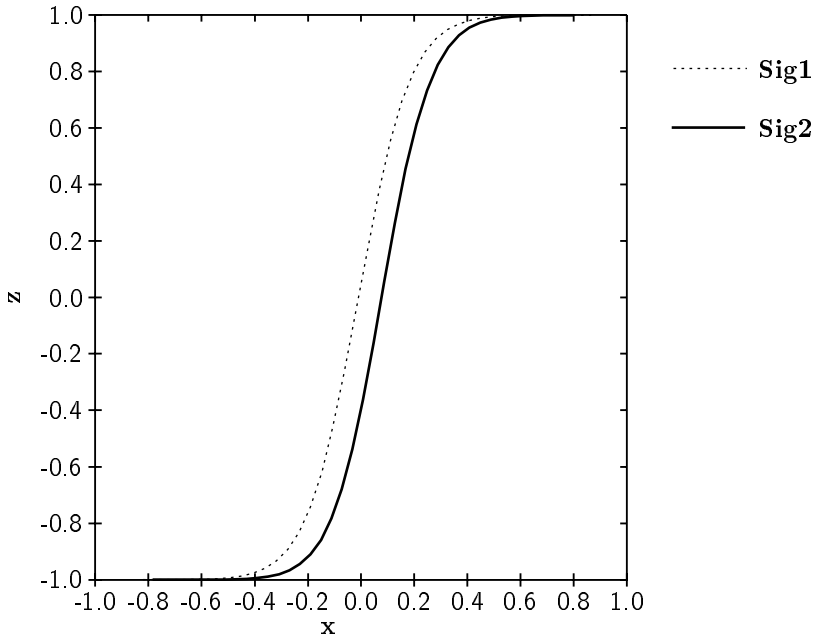


Figure 5.7 **Sigmoid data showing input normalization. Shows data of the sigmoid characteristics for the same sigmoidal circuit taken for two different input current levels. Sig1 has an input current level of 12.2 nA and Sig2 has an input current level of 2.33 nA. The shift in the sigmoid characteristics are due to offsets in the input current mirrors.**

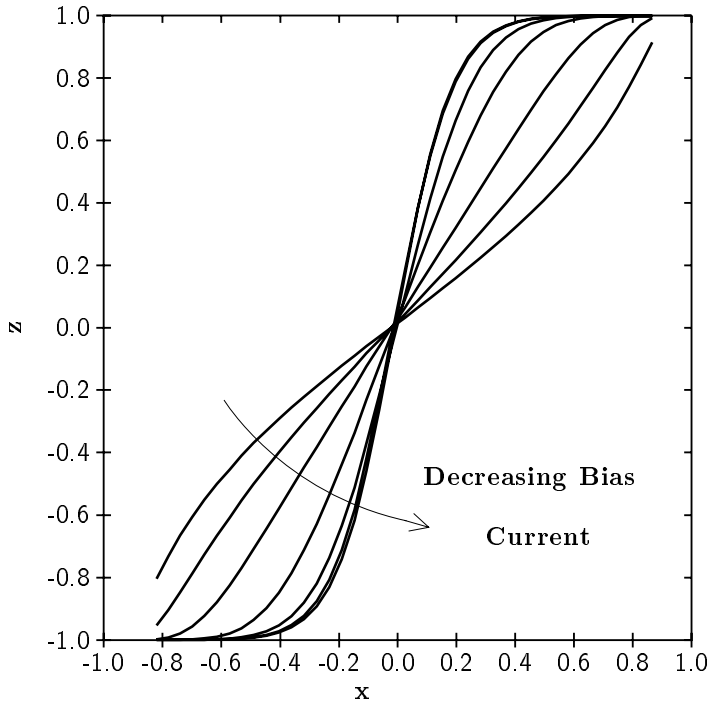


Figure 5.8 **Variable-gain diode-triplet sigmoid. Shows data from the diode-triplet sigmoid circuit for various gains of operation ranging from high gain (below threshold) to low gain (above threshold). The corresponding value of $\nu(Z_t)$ ranges from 0.64 V to 1.64V referenced down from $V_{\rm dd}$.**

current mirrors are located on different parts of the chip and were not well designed for good current matching). The importance of Figure 5.7 is the normalization of the input. This normalization is quite an important feature. The sigmoid is self-scaling relative to its input and it ignores changes in total level of input, which may arise because more synapses are added, the output currents of the other sigmoids are adjusted (or adapted), different levels of external input are applied, or the chip heats up; the differential current divided by the the total is the only relevant quantity to the sigmoid. This self-scaling feature is a property of normalized differential current-mode circuits and is an extremely powerful feature in the design of large systems.

5.1.2 Variable-Gain Sigmoidal Circuit

By operating the bias of the sigmoid (Z_t in Figure 5.2) in the moderate or strong inversion regime of the transistor, the gain of the sigmoid is reduced. Figure 5.8 shows several sigmoids with various gains. All of these sigmoids are taken from the same diode-triplet circuit with the input differential the same and only the bias, Z_t, is changed between runs. In Figure 5.8 $\nu(Z_t)$ ranges from 1.64V referenced down from $V_{\rm dd}$, which is above threshold and results in a relatively low gain, down to 0.64V referenced down from $V_{\rm dd}$, which is subthreshold and results in the highest gain. The threshold voltage for these transistors was approximately 1.0 V.

5.1.3 Variable Output-Scaling Sigmoidal Circuit

By adding an additional differential pair to the sigmoid, as shown in Figure 5.9, it is possible to cause the sigmoid to saturate at normalized outputs less than one and thereby change the gain. The way this works is that neither output branch, Z^+ nor Z^-, is able to source all the current. Since the output is normalized by the total current in both branches, and since the current in neither branch can go to zero, the saturated normalized output will approach a value less than one. The equation describing the scaled sigmoid is

$$z = f(x) \cdot z_t, \tag{5.4}$$

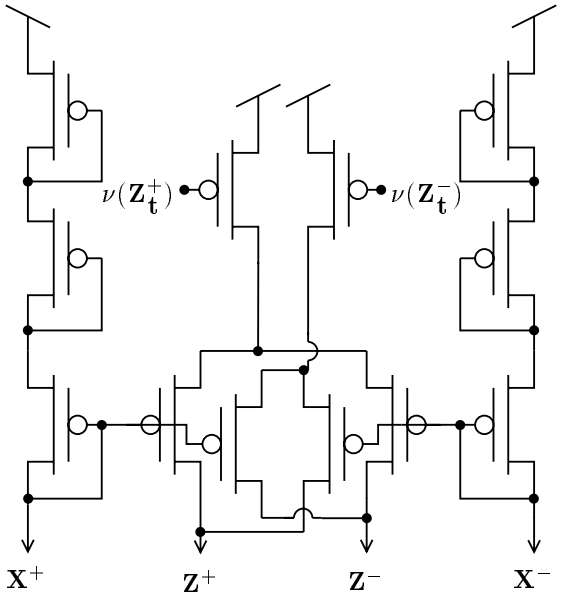


Figure 5.9 **Variable-gain sigmoidal circuit. This sigmoidal circuit is constructed by adding a cross-coupled differential pair to the original diode-triplet sigmoidal circuit. This new differential pair adds current to the opposite branch of the sigmoid, effectively reducing the gain.**

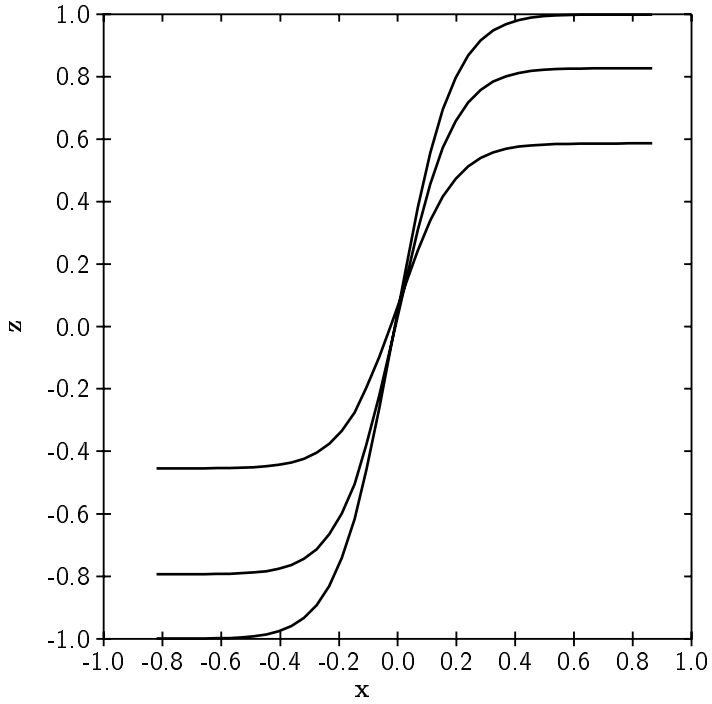


Figure 5.10 **Subthreshold data from variable-gain diode-triplet sigmoidal circuit. $\nu(Z_t^+) = 0.64V$, $\nu(Z_t^-)$ =0V, 0.54V, and 0.59V referenced down from $V_{\rm dd}$. The curve with the largest range was taken with $\nu(Z_t^-)$ =0V. The effective scaling factor z_t is 1, 0.8 and 0.6 for these three cases.**

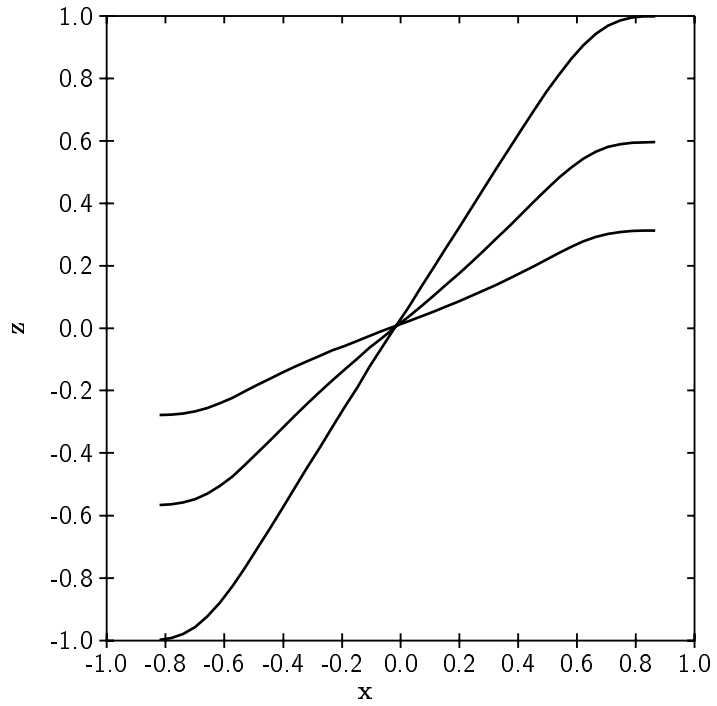


Figure 5.11 **Above-threshold data for variable-gain diode-triplet sigmoidal circuit.** $\nu(Z_t^+) = 1.24V$, $\nu(Z_t^-)$ **= 0V, 1.04V, and 1.14V referenced down from** $V_{\rm dd}$**. The curve with the largest range was taken with** $\nu(Z_t^-)$ **= 0V. The effective scaling factor is about 1, 0.6 and 0.3.**

where $f(x)$ is shown in Equation 5.2 and z_t is $\frac{Z_t^+ - Z_t^-}{Z_t^+ + Z_t^-}$. z_t is the scaling factor of this sigmoid. Figures 5.10 and 5.11 shows data from the output scaling sigmoid for below-threshold and above-threshold biasing conditions, respectively. z_t can also be negative giving us a monotonically decreasing sigmoid. Anecdotally, I have used the negative sigmoid to unlearn some patterns, but this has not been studied quantitatively.

The ability to change the saturating limit is a useful way to limit the affect of any particular sigmoid on the inputs of other sigmoids. In Chapter 6, I show how this saturation below one can be used to guarantee global asymptotic stability of the network.

5.2 Backward Propagation: The Linear-Error Unit

The dynamical system of the Y units in Equation 1.2 calls for a similar function as computed for the X layer discussed in Section 5.1. The equation at equilibrium is

$$y_i = f'(x_i)(\sum_{j \neq i} w_{ji} y_j + J_i) = f'(x_i) \cdot y_{ini}. \tag{5.5}$$

Several differences exist between this equation and Equation 5.1, though. First it is necessary to compute the derivative, $f'(x)$, of the sigmoid. Second, the weighted sum of the inputs is not passed through a sigmoid as in forward propagation; instead the resulting weighted sum is simply multiplied by the derivative of the sigmoid.

5.2.1 Linear-Error Circuit Implementation

Figure 5.12 shows the actual circuit implementation of Equation 5.5. The inputs to the ith Y unit are the outputs of the ith sigmoid, denoted as $\nu(f(X^{\pm}))$ in the circuit, as well as the weighted sum of contributions from the other Y units, $\sum_{j \neq i} w_{ji} y_j$, which is denoted in the circuit as $Y_{in}^{\pm}$. The outputs from the Y unit are $Y_{out}^{\pm}$. The function which the circuit computes is

$$\Delta Y_{out} = \frac{\Delta Y_{in}}{Y_{total}} \cdot Y_c = y_{in} \frac{Y_t}{4}(1 - f(x)^2) \approx \frac{Y_t}{4} f'(x) y_{in}, \tag{5.6}$$

where $Y_{total} \equiv Y_{in}^+ + Y_{in}^-$ and Y_t and Y_c are shown in Figure 5.12.

Figure 5.13 shows data taken from the derivative circuit by sweeping x, the input to the sigmoid, and measuring ΔY_{out}. The family of curves is obtained by varying ΔY_{in}, which

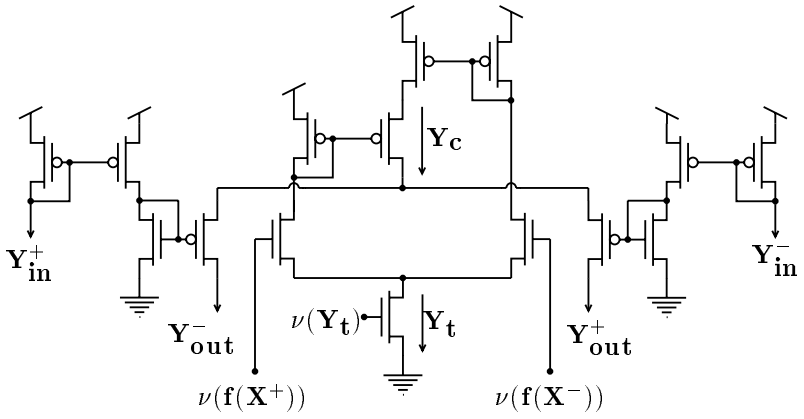


Figure 5.12 **Current-mode derivative circuit. This circuit takes input from sigmoidal output $\nu(f(X^{\pm}))$ and $Y_{in}^{\pm}$ and produces output, $Y_{out}^{\pm}$.**

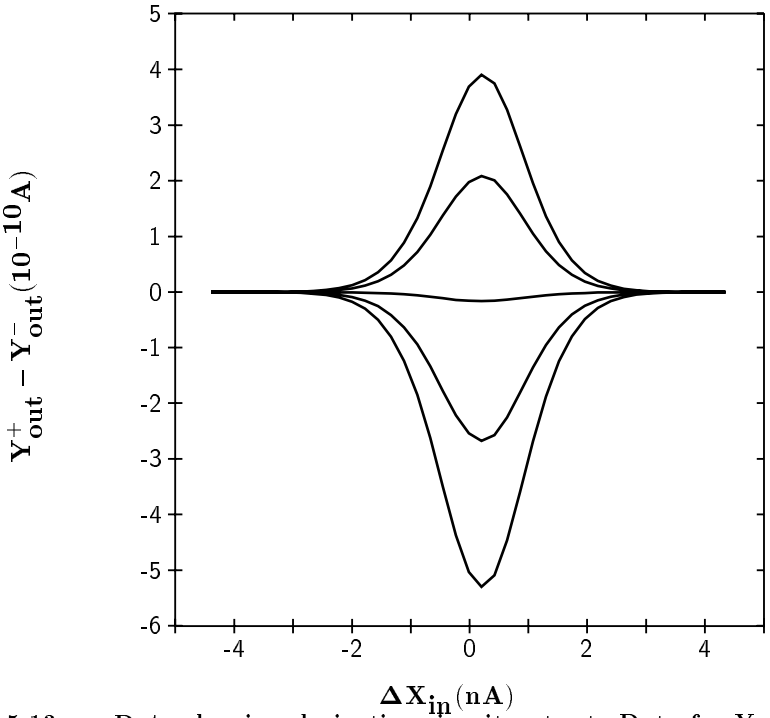


Figure 5.13 **Data showing derivative-circuit output. Data for Y units obtained by sweeping ΔX_{in} vs ΔY_{out} for the derivative circuit (Figure 5.12). The family of curves is obtained by varying ΔY_{in}, from $-4.46\,nA$ (bottom curve) to $4.42\,nA$ (top curve).**

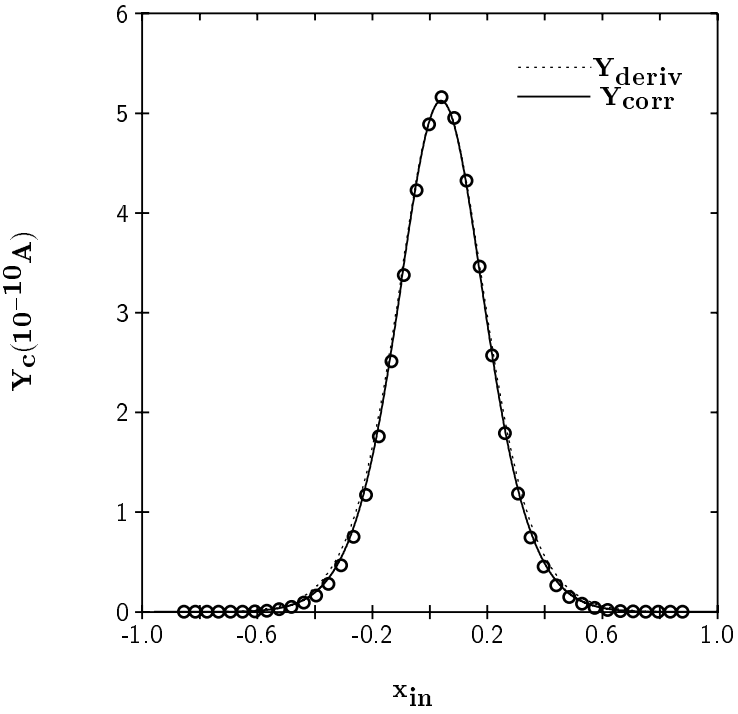


Figure 5.14 Theoretical fit to data of the tailcurrent Y_c of the derivative circuit. Two fits are shown: one is for the derivative of the sigmoid circuit (Y_{deriv} in Equation 5.8); the other fit uses the ideal current correlator equation coupled with the sigmoid equation (Y_{corr} in Equation 5.9). Both fits use $Y_t = 2.05\,nA$ and $\kappa = 0.65$ ($\beta = 4.9$). The derivative equation, Equation 5.7, is scaled by $Y_t/4\beta$.

ranges from $-4.46\,nA$ (bottom curve) to $4.42\,nA$ (top curve).

Figure 5.14 shows two theoretical fits to the tailcurrent, Y_c, of the derivative circuit. The first fit to Y_c, (seen in Figure 5.14 as the outer of the two fits), uses the derivative of the sigmoid, Equation 5.2, which is,

$$\frac{dz}{dx} = f'(x) = \frac{4\beta(1-x^2)^{(\beta-1)}}{((1+x)^\beta + (1-x)^\beta)^2}. \tag{5.7}$$

Scaling $f'(x)$ by $\frac{Y_t}{4\beta}$, we obtain our first fit to Y_c,

$$Y_{deriv} = \frac{Y_t f'(x)}{4\beta} = \frac{Y_t(1-x^2)^{(\beta-1)}}{((1+x)^\beta + (1-x)^\beta)^2}. \tag{5.8}$$

The second fit to Y_c uses the ideal current correlator equation, Equation 3.9, coupled with the sigmoid equation, Equation 5.2,

$$Y_{corr} = \frac{Y_t(1-x^2)^\beta}{((1+x)^\beta + (1-x)^\beta)^2}. \tag{5.9}$$

Both equations fit the data quite well. I assume henceforth that the Y unit actually uses the derivative of $f(x)$ to perform

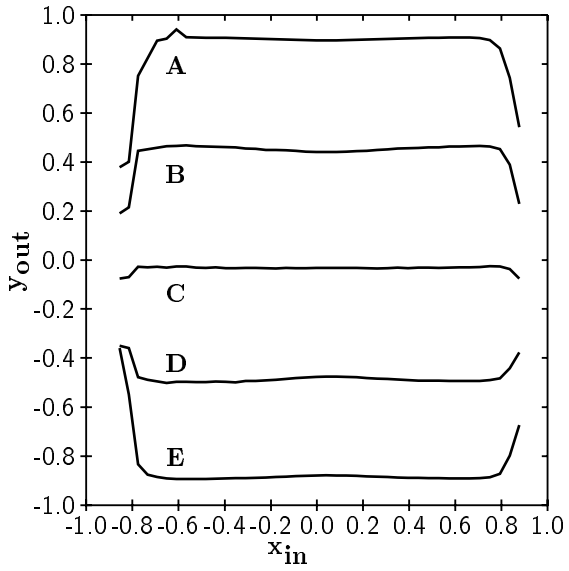
$$\frac{\Delta Y_{out}}{Y_t} = \frac{1}{4} y_{in} \cdot f'(x). \tag{5.10}$$

5.2.2 Normalized Derivative Operation

The analysis of the previous part normalizes the differential signals by Y_t, where Y_t is shown in Figure 5.12. This normalization is seen in Equation 5.10 where ΔY_{out} is normalized by Y_t, instead of the total output current, Y_c.

To calculate the normalized output, we need to normalize the differential output current by total output current, $Y_{out}^+ + Y_{out}^-$. This current is set by the tail current, Y_c, which is the quantity containing the derivative information as seen in Equation 5.6. To normalize this equation, we need to divide by Y_c, in which case Equation 5.6 becomes,

$$\frac{\Delta Y_{out}}{Y_c} \equiv y_{out} = y_{in}. \tag{5.11}$$



Curve	ΔY_{in} (nA)	y_{in}
A	4.42	0.89
B	2.22	0.45
C	0.00	0.00
D	-2.25	-0.44
E	-4.46	-0.86

Figure 5.15 **(a) Data obtained by sweeping x_{in} vs y_{out} for the derivative circuit (Figure 5.12). The family of curves was obtained by varying ΔY_{in}, from $-4.46\,nA$ (curve E) to $4.42\,nA$ (curve A). The curves are from the same raw data as in Figure 5.13, but they are normalized by the total output current, Y_c. The drooping behavior at the start and finish of each curve is caused by the outputs not settling before the data is taken. (b) Shows y_{in} used for each of the curves in (a).**

Figure 5.15(a) shows the output for the derivative circuit (as in Figure 5.13), but the output is here normalized by the total output current, Y_c. The values for y_{in} are shown in Figure 5.15(b).

It appears the derivative circuit is nothing but a current mode identity function. But remember, the output of the circuit is scaled by the derivative. The total current output is proportional to the derivative of the sigmoid. Thus, any system, which accumulates the outputs of many of the Y units at its input, is dominated by the activity of the largest such input. As we shall see in Chapter 6, this computation is suitable for the adjoint network.

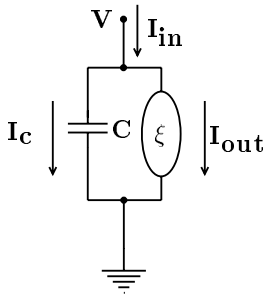


Figure 5.16 Generalized I–V characteristic element in parallel with a capacitor. If the generalized element ξ has a monotonically increasing I–V relationship, then this circuit has one stable equilibrium at $I_{out} = I_{in}$.

5.3 Dynamics

The dynamics which govern both the X and the Y units are quite complex, but with some simplifying assumptions, we can gain some insight into the relaxation of both of these units. In this section, I discuss how the equations of motion for this system are calculated and measured.

5.3.1 Generalized Model

In general, if the circuit we are considering has a capacitor in parallel with an element, whose current is monotonically increasing in the voltage across the element (see Figure 5.16), then the equation which describes the dynamics of the current through the element ξ, $\frac{dI_{out}}{dt}$, as a function of the input and output current, I_{in} and I_{out}, is

$$\frac{dI_{out}}{dt} = \frac{dI_{out}}{dV}\frac{dV}{dt} = \frac{dI_{out}}{dV}\frac{1}{C}(I_{in} - I_{out}). \tag{5.12}$$

In this generalized model, if $\frac{dI_{out}}{dV} \geq 0$ for element ξ, then the only stable equilibrium is $I_{out} = I_{in}$. In the case of element ξ being a resistor, with resistance R, then $\frac{dI_{out}}{dV} = 1/R$ and the dynamics are simply given by

$$\frac{dI_{out}}{dt} = \frac{1}{RC}(I_{in} - I_{out}).$$

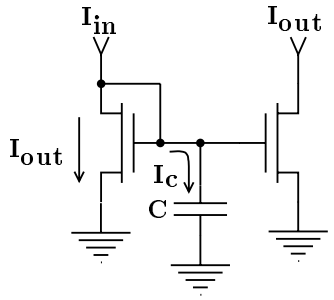


Figure 5.17 $\dot{I}_{out}$ **versus** I_{out} **for mirror connected FET. The generalized element from Figure 5.16 is now a diode-connected FET. Near equilibrium,** I_{out} **approaches** I_{in} **exponentially with the time constant set by** I_{in}**.**

5.3.2 Current Mirror

In the case where the element ξ is a diode-connected FET (see Figure 5.17) with $I_{out} = I_0 e^{\frac{\kappa V}{V_t}}$, then $\frac{dI_{out}}{dV} = \kappa \frac{I_{out}}{V_t}$ and the dynamics are described by,

$$\frac{dI_{out}}{dt} = \frac{\kappa I_{out}}{V_t C}(I_{in} - I_{out}). \tag{5.13}$$

Figure 5.18 shows $\dot{I}_{out}$ versus I_{out} for Equation 5.13. The two curves are for two different input currents. This curve intersects $\dot{I}_{out} = 0$ at the equilibrium points for Equation 5.13. Physically this analysis is only valid for $V \geq 0$, in which case, the minimum I_{out} is I_0. The only stable equilibrium for this equation is $I_{out} = I_{in}$. As I_{out} approaches I_{in}, a linear approximation to the $\dot{I}_{out}$ versus I_{out} is made. The slope around $I_{out} = I_{in}$ is $\alpha \equiv \frac{\kappa I_{in}}{C V_t}$. Thus, as I_{out} approaches I_{in}, these dynamics exhibit exponential decay-like characteristics with the time constant

$$\tau = \frac{1}{\alpha} = \frac{V_t C}{I_{in}\kappa}. \tag{5.14}$$

Notice τ is inversely proportional to the input current, I_{in}. We can use this to design a system which requires different time constants by setting the total current to different levels for the dynamical systems which require the different settling times.

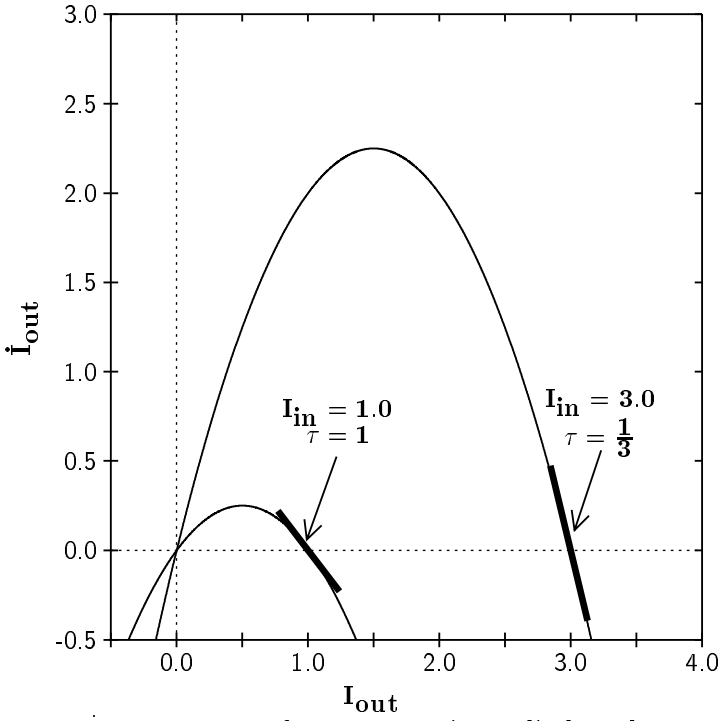


Figure 5.18 $\dot{I}_{out}$ **versus** I_{out} **for current-mirror diode. Shows relation between** $\dot{I}_{out}$ **and** I_{out} **for two different input current levels for the current-mirror dynamics. The time constant near equilibrium drops as** $\frac{1}{I_{in}}$**.**

5.3.3 Diode Stack and the κ Effect

If instead of a single diode as in Figure 5.17, we have a stack of diodes, the differential equation describing the relationship between I_{out} and I_{in} is:

$$\frac{dI_{out}}{dt} = \frac{I_{out}}{\sigma(n,\kappa)V_t C}(I_{in} - I_{out}), \tag{5.15}$$

where $\sigma(n,\kappa) = \sum_{i=1}^{n} \frac{1}{\kappa^i}$ for n diodes assuming each has the same effective κ as given in Equation 3.14. For the diode-triplet sigmoid, $\frac{1}{\sigma(3,0.65)} = 0.13$ with $\kappa = 0.65$. The time constant for an exponential decay as I_{out} approaches I_{in} is,

$$\tau = \frac{\sigma(n,\kappa)V_t C}{I_{in}}. \tag{5.16}$$

To guarantee the validity of this equation, the system must be designed such that any parasitic capacitances are negligible compared with the capacitance C.

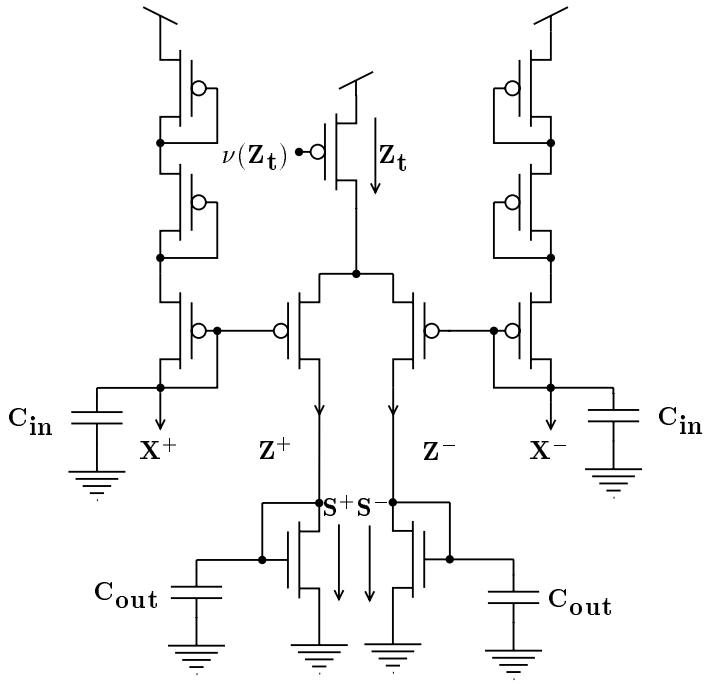


Figure 5.19 **Diode-triplet sigmoidal circuit with the associated significant capacitances, C_{in} and C_{out}. The dynamics of the sigmoid within the network are dominated by these capacitances because they are large capacitances (they are physically wires stretching the length of the chip).**

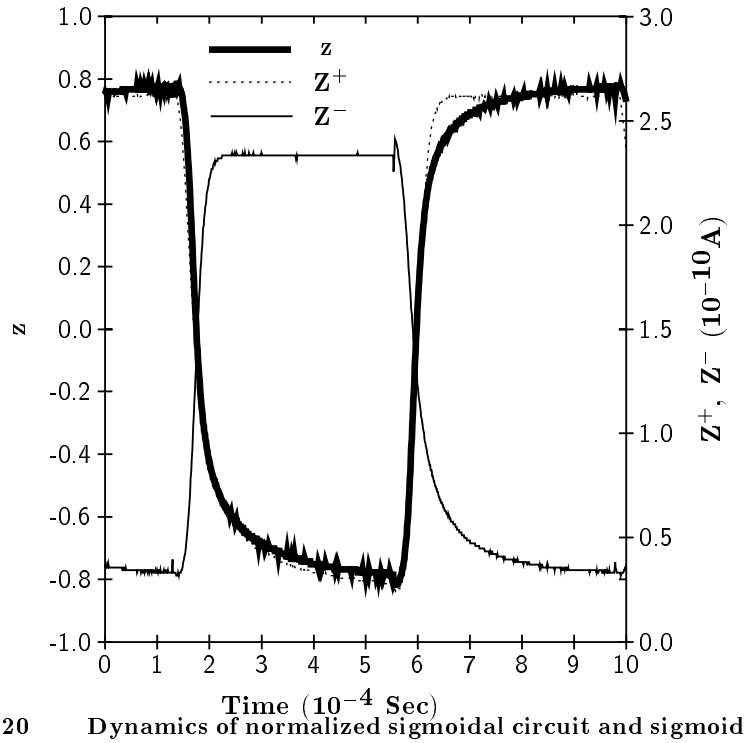


Figure 5.20 Dynamics of normalized sigmoidal circuit and sigmoidal circuit branch currents. Left axis shows normalized output for the curve labelled 'z.' The normalized curve is obtained from the two sigmoidal branch currents labelled 'Z^+' and 'Z^-.' Dynamics of both Z^+ and Z^- are shown, with right axis showing output current level. The input is a square wave provided to the linear portion of the sigmoid, as shown in Figure 5.21.

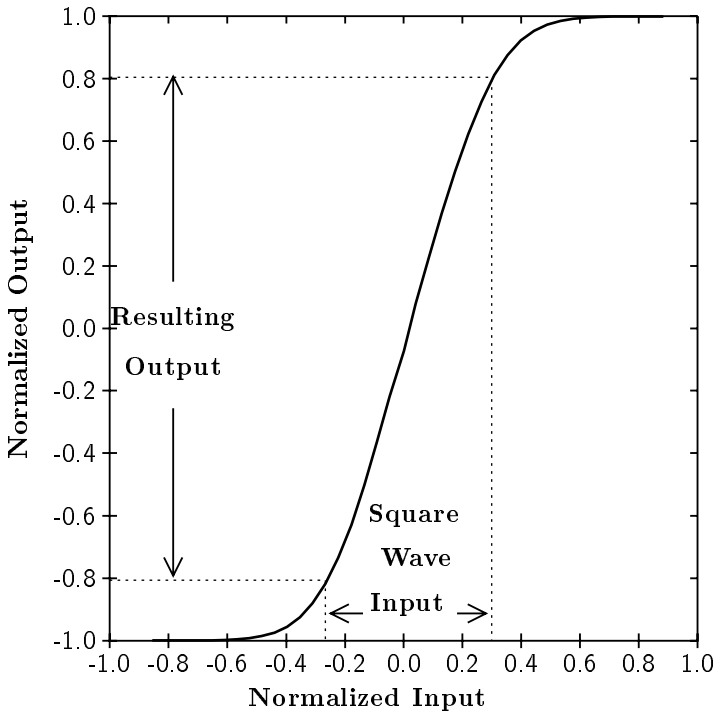


Figure 5.21 Sigmoid used to produce data in Figure 5.20. Dotted lines show the square wave input–output relationship.

5.3.4 Measurements from the X Units

Here we analyze the dynamics of the sigmoid. Figure 5.19 shows the locations of the significant capacitances which exist for the sigmoid in the network architecture. The input and output capacitances are large relative to the parasitic node capacitances, since the inputs and outputs are wires which travel much of the length of the chip. The dynamics are simplified by the fact that the sigmoid input currents are significantly larger than the output currents. Therefore, the dynamics of the sigmoid are dominated by the outputs.[1] The dynamics for the outputs of the sigmoid are shown in Figure 5.20 (measured on the right axis). Input is provided as a square wave as shown in Figure 5.21 and the outputs of this sigmoid are the measured quantities in Figure 5.20. The input current level was set large enough to ensure the input reached steady state much faster than the output for the lowest input level. The normalized output dynamics are also shown in Figure 5.20 (measured along the left axis). Notice that the slower sigmoidal output branch currents

[1] This claim is not true, though, if the normalized input is near $\pm$ 1, in which case, the input with the smaller current could potentially dominate the dynamics.

	Theory (μsec)	Z^+ Data (μsec)	Z^- Data (μsec)	Input Current (pA)
τ_{low}	500	99.7	91.0	27.8
τ_{high}	60	10.5	8.95	262
τ_{low}/τ_{high}	8.45	9.48	10.16	9.42

Table 5.1 Time constants for the positive and negative output branches of the sigmoid changing between ± 0.8 in normalized units. The ratios of the high and low time constants scales with ratio of one over the ratio of high and low currents. The difference between theory and experiment here is due mainly to errors in the measured absolute current level (this current is measured as the output of current amplifier, whose gain was inferred from layout).

dominate the behavior of the normalized sigmoid.

Assuming that $X^{\pm}$ are large compared with $Z^{\pm}$, we can calculate the dynamics of the sigmoid as follows. The dynamics of the output branch currents are described by

$$\begin{aligned} \frac{dS^+}{dt} &= \frac{\kappa}{V_t C_{out}} S^+(Z^+ - S^+), \\ \frac{dS^-}{dt} &= \frac{\kappa}{V_t C_{out}} S^-(Z^- - S^-). \end{aligned} \tag{5.17}$$

At steady state, $Z^{\pm} = S^{\pm}$. The normalized difference of the two branch currents, $Z^{\pm}$ at steady state is

$$\frac{Z^+ - Z^-}{Z_t} \equiv z = s = f(x) = \frac{(1+x)^\beta - (1-x)^\beta}{(1+x)^\beta + (1-x)^\beta}, \tag{5.18}$$

where s is the normalized difference of $S^{\pm}$. As in Equation 5.14, the time constant for Equation 5.17 near steady state are

$$\begin{aligned} \tau_x^+ &= \frac{V_t C_{out}}{\kappa Z^+}, \\ \tau_x^- &= \frac{V_t C_{out}}{\kappa Z^-}. \end{aligned} \tag{5.19}$$

The dynamics of the sigmoid is dominated by the slower of τ_x^+ and τ_x^-, which is determined by the branch with the smaller current, Z^+ or Z^-.

Table 5.1 shows the time constants calculated and measured from the data in Figure 5.20.

The calculated time constants are computed assuming the following values: the capacitances used on the outputs of the sigmoid are inferred from the layout and are roughly 400fF; κ is assumed to be 0.65; I_{in} measured at steady state are shown in Table 5.1. The discrepancy between the measured and calculated time constants is due mainly to the calculation of the current gain through the current amplifiers from the test structure to the outside world (see Section 3.6). In addition the value of the capacitance used is an approximation intended only to calculate the time constant within the correct order of magnitude. What is really of interest is the ratio of the low and high time constants. This ratio removes any error due to current level and capacitance calculations. In addition, this ratio emphasizes the main point that the time constant has a reciprocal dependence on the input current levels. Thus, by controlling the system current levels, we are able to set the time constants of each system.

Chapter 6

Network Dynamical System: An Implementation

...And learning, a mere hoard of
gold kept by a devil till sack commences
it and sets it in act and use.

Shakespeare
Henry IV, Part 2

In the preceding chapters, I introduce the elemental building blocks for a normalized current-mode learning neural network. In this chapter, I put the blocks together to build a layer of synapses and neurons. Coupled with this layer is an adjoint layer designed to collectively calculate gradient information for each weight to perform supervised training tasks. This chapter describes the entire network implemented in analog VLSI along with the associated dynamical equations, discusses stability of the X and Y layers, shows the chip architecture, and presents examples of the chip learning.

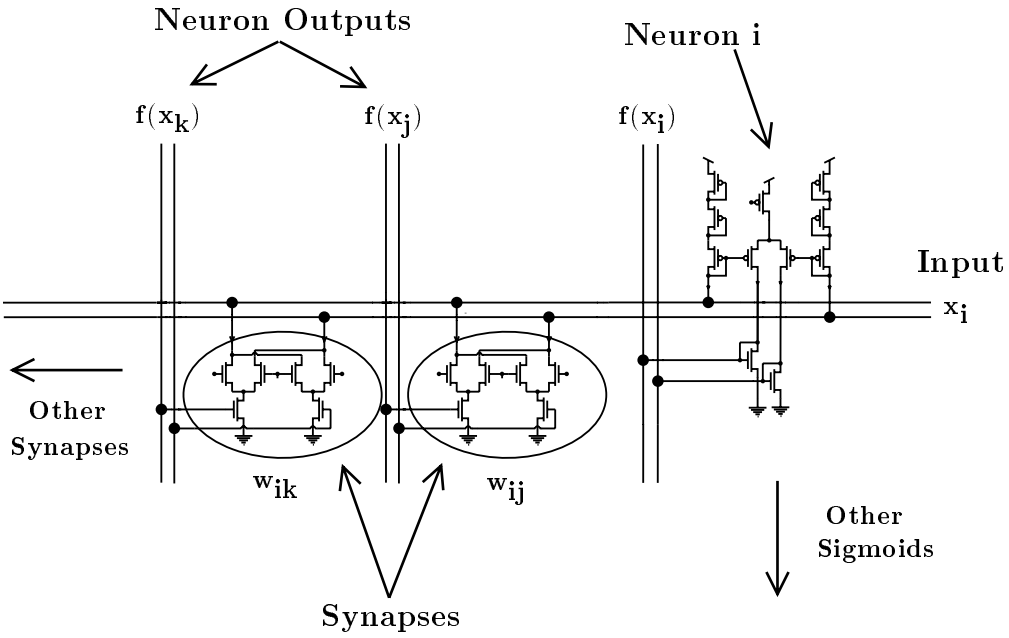


Figure 6.1 X layer with a single sigmoid and several connecting synapses.

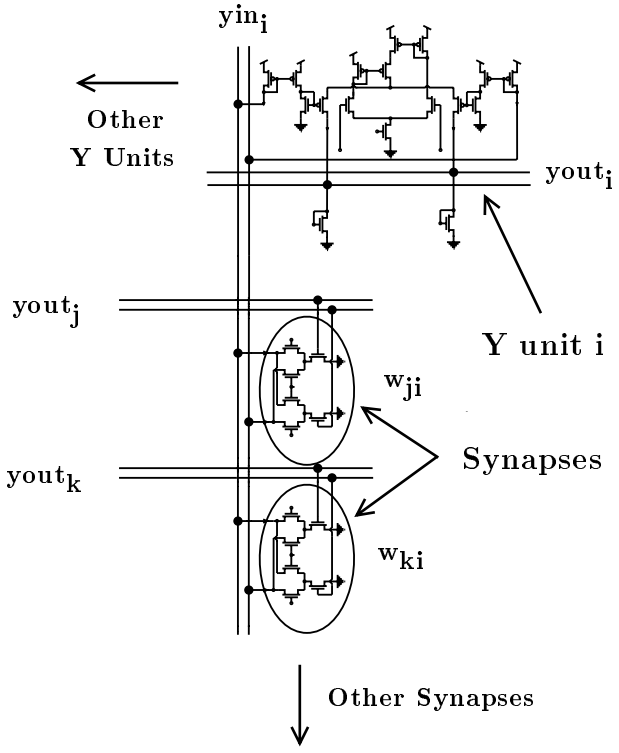


Figure 6.2 Y layer with a single Y unit and several connecting synapses.

6.1 Network Implementation in Analog VLSI

Figures 6.1 and 6.2 show the X and Y layers as implemented in silicon. The synapse is described in Chapter 4 and the X and Y units are described in Chapter 5.

6.1.1 Feedforward Layer

The X layer performs the following function in steady state:

$$x_i = \frac{1}{n-1+\alpha}\left(\sum_{j\neq i} w_{ij} f(x_j) + \alpha \cdot input_i\right), \tag{6.1}$$

where α is $\frac{I_{bias}}{X_b}$ (I_{bias} is the bias current for the external input and X_b is the bias current for the sigmoids), where $input_i$ is the normalized differential external input, and where $f(x_j)$ is given in Equation 5.2 and repeated here for convenience:

$$f(x_j) = \frac{(1+x_j)^\beta - (1-x_j)^\beta}{(1+x_j)^\beta + (1-x_j)^\beta}, \tag{6.2}$$

where $\beta = \frac{\kappa^2+\kappa+1}{\kappa^2}$. With $\kappa = 0.65$, β is approximately 4.9, which is also the slope of the sigmoid at the origin. The steady state reached in Equation 6.1 is the same equilibrium that the ideal equation in Chapter 1 reaches (see also Equation 1.1), but the dynamics of the ideal system and the implemented system are different. The dynamical system for the implemented system is derived from Equation 5.17,

$$\begin{aligned} \frac{dS_i^+}{dt} &= \gamma S_i^+ (Z_i^+ - S_i^+), \\ \frac{dS_i^-}{dt} &= \gamma S_i^- (Z_i^- - S_i^-), \end{aligned} \tag{6.3}$$

where,

$$\begin{aligned} Z_i^+ &= \frac{Z_t}{2}(1+f(x_i)), \\ Z_i^- &= \frac{Z_t}{2}(1-f(x_i)), \end{aligned} \tag{6.4}$$

are the branch currents as shown in Figure 5.19 and where $S_i^+$and $S_i^-$are the output variables of the sigmoid, as shown in Figure 5.19, and $\gamma = \frac{\kappa}{V_t C_{out}}$.

6.1.2 Error Layer

The Y units compute the following function in steady state:

$$yin_i = \frac{1}{\sum_j f'(x_j) - f'(x_i) + \beta}(\sum_{j \neq i} w_{ji} \cdot yin_j \cdot f'(x_j) + \beta \cdot error_i), \tag{6.5}$$

where $\beta \equiv \frac{8J_{bias}}{Y_t}$ (J_{bias} is the bias current for the external error input and Y_t is the bias current for the Y unit); and where $error_i \equiv \frac{Target_i - f(X_i)}{2J_{bias}}$ (the normalized differential external error input as shown in Figure 6.5).

The steady-state solution of the actual circuit implementation (see Equation 6.5), and the ideal equation (Equation 1.2) differ in the derivative prefactor and the derivative term, $f(x_j)$ multiplied by yin_j, in the summation. In Pineda's formulation, yin_i increases linearly with $f'(x_i)$, whereas in this implementation, yin_i increases monotonically with $f'(x_i)$. This implementation is not exact.

In Equation 6.5, if we are dealing with a Y unit that receives external input, and the total external input is large compared with $\sum_{j \neq i} f'(x_j)$ and is large compared with $\sum_{j \neq i} w_{ji} \cdot yin_j \cdot f'(x_j)$ which can be guaranteed by making β of Equation 6.5 much larger than one, then the steady-state output for these units is

$$yin_i = error_i. \tag{6.6}$$

Equation 6.6 is equivalent to the steady-state solution of the ideal dynamics given by Equation 1.2 for large external error inputs.

For those units that receive no external error information (that is, input units and hidden units), Equation 6.5 describes the steady-state solution. In the ideal steady-state solution, as the sigmoids begin to saturate (that is, the derivative goes to zero), the Y units' outputs tend to zero. The output Y units of this implementation also tend to zero, but the minimum is limited by the sum of all the derivatives. Also, those Y units of the implementation that receive no external error inputs tend to zero only if the derivative information of its associated sigmoid is small compared with the other derivatives. Thus, these new Y units scale their derivative by the sum of all derivatives.

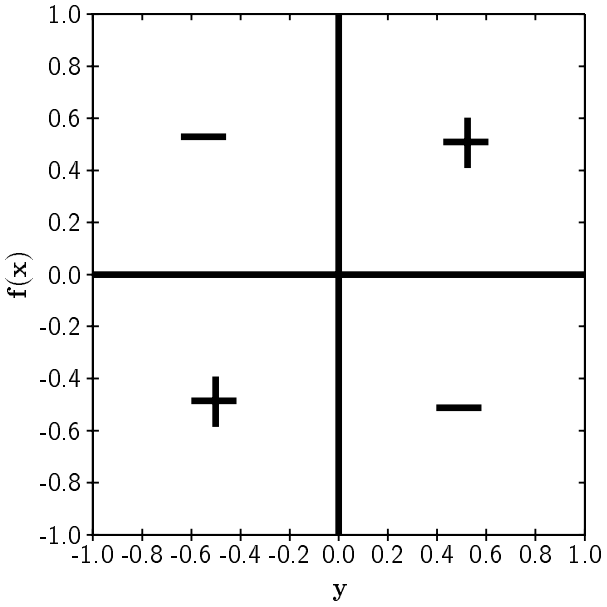


Figure 6.3 Steady-state solutions for ideal Pineda algorithm. Steady-state solutions shown with dark lines at $y = 0$ and $f(x) = 0$. Training signal is the product of y and $f(x)$.

6.1.3 Training Algorithm and Convergence

The dynamics for the weights are also somewhat different than the ideal dynamics as in Equation 1.4. The new dynamics are

$$\tau_w \dot{w}_{ij} = \text{tsgn}(yin_i \cdot f(x_j), \theta), \tag{6.7}$$

where tsgn, abbreviated from thresholded signum, is the same as that in Equation 4.3 and duplicated here,

$$\text{tsgn}(z, \theta) \equiv \begin{cases} -1 & \text{if } z < -\theta \\ 0 & \text{if } -\theta \leq z \leq \theta \\ 1 & \text{if } z > \theta, \end{cases} \tag{6.8}$$

where the threshold, θ, is either a constant, which sets the total current to a follower, as in Equation 4.4, or is set by a current-correlator follower as in Equation 4.5. Figure 4.9 shows the contours of the learning algorithm as implemented for the follower paradigm. Figure 6.3 shows the contour for Pineda's algorithm. The solid lines along the axis show the regions where the gradient information goes to zero. Notice that in the Pineda algorithm, one of two conditions must be met before the gradient information goes to zero; either the output

of the X units are zero, or the Y units are zero. Having all X units zero is the trivial case and is not a stable equilibrium. Having all Y units zero guarantees one of two conditions: either the derivatives are all zero, or the error term is zero. In a computer simulation, where the derivatives never reach zero, the only possible equilibrium is where the errors are zero.

In the original system (the Pineda algorithm), the only case in which the gradient is zero is when the Y outputs are zero. In our implementation, the Y outputs are never absolutely zero due to inherent offsets [51] and noise [16]. For this reason, I added a "dead zone" to our weight update dynamics. This is the region in Figure 4.9 encompassing the zero-gradient region. In this zero-gradient region, no weight change occurs irrespective of the noise and offsets inherent in the X and Y units. In the new thresholded signum dynamics, the qualitative behavior is similar to the original system. The advantage of the new dynamics is due to insensitivity of the new algorithm to noise. There is a noise margin built into the algorithm. As a consequence of the "dead zone," the solution does not converge on zero error, but it does converge close to zero.

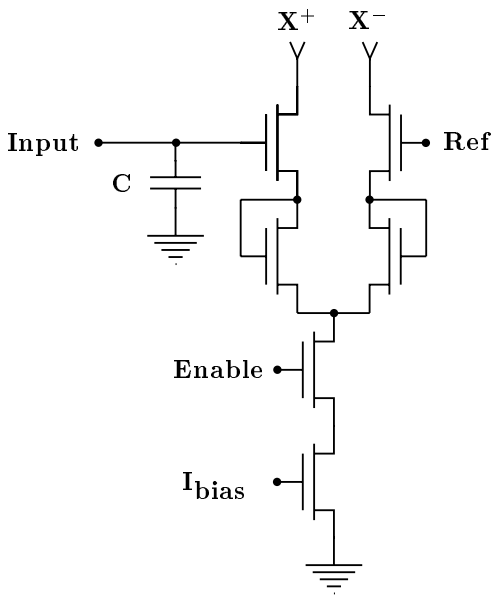


Figure 6.4 Circuitry used for external inputs.

6.1.4 External Inputs and Targets

The circuitry used for external inputs and targets is shown in Figure 6.4 and Figure 6.5. For each of these, Enable is a digital bit set to '0' (Gnd) or '1' (V_{dd}). If the Enable is '0,' no

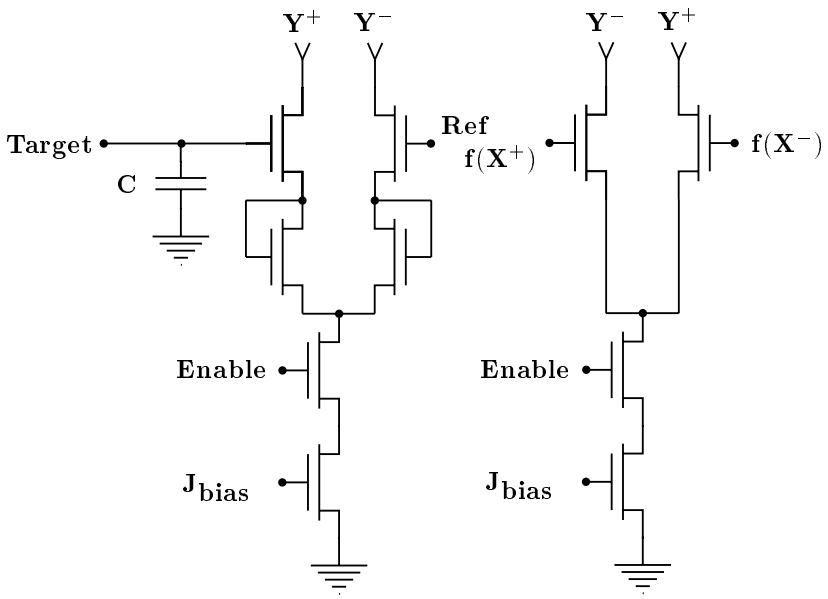


Figure 6.5 Circuitry used for external targets.

external input is provided. If the bit is '1,' a total I_{bias} current is provided by the external input. The difference between Input and Ref determines the differential signal provided as an **external input.** The sources of each input transistor are degenerate. This degeneration allows a larger allowable swing between Input and Ref before the external input saturates. By making I_{bias} large relative to the other input current ($(n-1)Z_t$), this external input dominates the sigmoids static input.

The **external error input** shown in Figure 6.5 provides external input to the Y units. The Enable bit functions the same for this circuit as for the external input to the sigmoid. The Target input, which is referenced to Ref, has $f(X^+) - f(X^-)$ subtracted from it to provide an error input to the Y layer.

6.2 Stability

Stability of the network is of concern. If we do not guarantee that the network will be stable, it may oscillate and may not perform the task we wish it to learn. To consider stability, we must consider the stability of the X and Y layers. It has been shown previously by Almeida [1] that if the X layer is stable, then the Y layer is stable as well. Additionally, there exist several stability criteria from Matsuoka, Guez, Kelly, Sugawara, and Hirsch [40, 23, 32, 62, 25] with which conditions for the stability of the X layer can derived. I use

these criteria to show under what conditions I can guarantee global asymptotic stability for the X layer.

6.2.1 Stability of X Layer

Stability requirements for the X layer deal with constraints on the weight matrix. An early condition from Hopfield [27] was that the connection strengths are symmetric. From this condition, he found a Lyapunov function and global asymptotic stability was guaranteed. I cannot ensure that our weight matrix will be symmetric and must look for other criteria.

A second type of constraint deals with the sizes of the weights in the network relative to the maximum gain of the nonlinearity. The early constraints of Sugawara [62] impose the following as a sufficient condition for global asymptotic convergence:

$$\max_i \sum_j |w_{ij}| < 1. \tag{6.9}$$

Other constraints which are less restrictive have been recently found, but the constraint in Equation 6.9 is adequate for our network. The network used by Sugawara is of the form

$$\frac{dx_i}{dt} = -x_i + \sum_{j \neq i} w_{ij} g_j(x_j) + S_i, \tag{6.10}$$

where

$$0 < g_i'(z) \leq \sup_z g_i'(z) = 1. \tag{6.11}$$

6.2.2 X Layer with Resistor-Capacitor Dynamics

If we assume that the static equilibria of the X layer were obtained from a layer which had an RC circuit establishing the dynamics, then the dynamical equations describing the static output of Equation 6.1 is

$$\frac{dx_i}{dt} = -x_i + \sum_{j \neq i} \frac{\beta \cdot w_{ij}}{n - 1 + \alpha} \left(\frac{f(x_j)}{\beta} \right) + \frac{\alpha \cdot input_i}{(n - 1 + \alpha)}, \tag{6.12}$$

where β is the maximum slope of $f(x_j)$. By now defining new variables $\hat{w}_{ij} \equiv \frac{\beta \cdot w_{ij}}{n-1+\alpha}$, $\hat{f}(x_j) \equiv \frac{f(x_j)}{\beta}$ and $\hat{x}in_i \equiv \alpha \frac{input_i}{n-1+\alpha}$, we obtain

$$\frac{dx_i}{dt} = -x_i + \sum_{j \neq i} \hat{w}_{ij} \hat{f}(x_j) + \hat{x}in_i. \tag{6.13}$$

With these new functions, we see that

$$0 < \hat{f}'(z) \leq \sup_z \hat{f}'(z) = 1 \tag{6.14}$$

is true, since the largest slope of the sigmoid in Equation 5.2 is at the origin and is β. The dynamical system of Equation 6.13 is of the same form used by Sugawara. Since the maximum value of $|w_{ij}|$ is 1,

$$\max_i \sum_j |\hat{w}_{ij}| = \frac{\beta}{n-1+\alpha} \max_i \sum_j |w_{ij}| \leq \frac{\beta}{n-1+\alpha} \max_i \sum_{j \neq i} 1 = \beta \frac{n-1}{n-1+\alpha} \leq \beta. \tag{6.15}$$

We assume $w_{ii} = 0$. Also note, from the definition of α in Equation 6.1, α is not negative. We note the bound of β is not small enough to guarantee global asymptotic stability as stipulated in Equation 6.9, since $\beta \approx 4.9$ for $\kappa = 0.65$. When training begins, though, if all of the weights are within approximately ± 0.20 or $1/\beta$ of the origin, we have guaranteed global asymptotic stability until the weights grow beyond this size.

Another technique used to guarantee stability is to reduce the overall effectiveness of any sigmoid on the other sigmoids. Using the sigmoid scaling described in Section 5.1 and Equation 5.4, the effectiveness of the sigmoid on affecting other sigmoids is reduced. The equation describing this is

$$\begin{aligned} \frac{dx_i}{dt} &= -x_i + \sum_{j \neq i} \frac{\beta w_{ij}}{n-1+\alpha} \frac{z_t f(x_j)}{\beta} + \frac{\alpha \cdot input_i}{(n-1+\alpha)} \\ \frac{dx_i}{dt} &= -x_i + \sum_{j \neq i} \tilde{w}_{ij} \hat{f}(x_j) + \hat{x}in_i, \end{aligned} \tag{6.16}$$

where z_t describes the sigmoid scaling factor and is between $\pm$ 1, where $\hat{f}_j$ and $\hat{x}in_i$ are defined as in Equation 6.13, and where the weight $\tilde{w}_{ij} = \frac{z_t \beta w_{ij}}{n-1+\alpha}$ is scaled by z_t. Here, the scaling of the sigmoid can be seen as a scale on the size of the weights in the network. It is

this insight that we use to guarantee global asymptotic stability of the X layer. Now, if we set $z_t < 1/\beta$, we now obtain,

$$\max_i \sum_j |\tilde{w}_{ij}| < 1, \tag{6.17}$$

which is the required stipulation of Equation 6.9. Thus, by using the sigmoid-scaling circuit, and by making z_t less than $1/\beta$, we are able to guarantee asymptotic global stability of the X layer. This analysis is only true if the output conductances can be considered as resistors instead of diodes and with a resistance such that $RC = 1$.

6.2.3 X Layer with Diode-Capacitor Dynamics

In reality, though, my network is composed of diode-connected FETs in parallel with capacitors. The dynamics of this layer are

$$\begin{aligned} \frac{dS_i^+}{dt} &= \gamma S_i^+ (Z_i^+ - S_i^+) \\ \frac{dS_i^-}{dt} &= \gamma S_i^- (Z_i^- - S_i^-), \end{aligned} \tag{6.18}$$

where

$$\begin{aligned} z_i &= f(x_i), \\ x_i &= \sum_j \hat{w}_{ij} s_i + input_i, \end{aligned} \tag{6.19}$$

and where $z_i \equiv \frac{Z_i^+ - Z_i^-}{Z_t}$, $\hat{w}_{ij} \equiv \frac{w_{ij}}{n-1+\alpha}$, $s_i \equiv \frac{S_i^+ - S_i^-}{Z_t}$, $input_i \equiv \alpha \frac{I_i^+ - I_i^-}{(n-1+\alpha)Z_t}$ and $x_i \equiv \frac{X_i^+ - X_i^-}{X_t}$. Also $\gamma = \frac{\kappa}{V_t C}$. The variables $S_i^\pm$ and $Z_i^\pm$ are shown in Figure 5.19. Following a method employed by Hirsch [25] and outlined in Appendix B we obtain the following criteria for stability of the X layer,

$$\sigma \sum_{j \neq i} |\hat{w}_{ji}| < 4 - \varepsilon \tag{6.20}$$

where $\varepsilon \ll 1$ and $0 \leq f'(x) \leq \sigma$. Thus, we need the gain of our sigmoid to be less than 4. By running the sigmoid in the moderate inversion regime of the transistor we can guarantee this condition. In addition, we can reduce $\hat{w}_{ij}$ by introducing sigmoid scaling to achieve the

criteria in Equation 6.20.

6.3 Overall Chip Architecture

I designed and tested a modified version of the Pineda network in a 2 μm CMOS process. The chip was fabricated through the MOSIS service. Figure 6.6 shows a schematic of the chip. This test chip contains 12 neurons, shown along the diagonal in Figure 6.6, each fully connected through the synapses to the other 11 neurons, but with no self-feedback connection. The full connectivity applies for the X units and Y units equally. All together there are 132 synapses. Each synapse contains a weight, which is a differential voltage contained on two floating nodes, and shared between two multipliers, a structure used for adaptation, a four-quadrant multiplier for the feedforward layer, another four-quadrant multiplier for the error layer, circuitry to calculate the training signal, and two differential pairs used for scanning out the weights and the training signal. The synapse is approximately 200 μm square.

The 12 fully connected neurons contain a sigmoidal unit and an error processing unit. The sigmoid unit is used for the nonlinear mapping and the error unit is used to calculate the weight updates. The sigmoidal unit and error unit along with two differential pairs for scanning out the state information of each unit occupy approximately 200 μm square, as well.

Inputs and targets are scanned in on a single line and sampled on the appropriate capacitor. Each input and each target has an associated one bit of memory called the enable bit. If an input is enabled, the associated sigmoid receives an external input. If a target is enabled, the associated error unit receives an error signal calculated from the target and the output of associated sigmoid. The inputs are scanned into the input **sample-and-hold** (s/h) circuitry and the target outputs are scanned into the target s/h circuitry. Both inputs and targets have an associated single bit memory which is also scanned in which designates whether the input is enabled, and whether the associated target is enabled. In this way, X units can be designated as input units, output units, or hidden units.

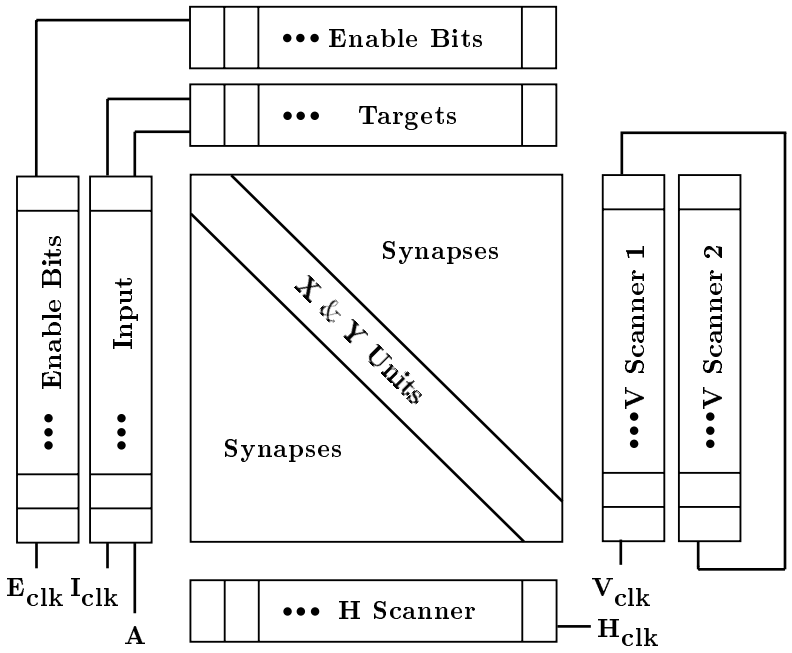


Figure 6.6 Overall learning chip architecture. Each long thin rectangular box is a scanner through which a digital bit can be clocked. The small boxes represent digital registers. Any number of 1's may be present within the scanner at any time. The digital input to a scanner may be a 0 or a 1. These inputs are not shown. The boxes labelled 'Scanner' (either H or V) are wired in such a fashion as to generate their own bit for scanning. The boxes labelled scanner are used to scan information off the chip either directly to a NEC multisync monitor (sync and blank are generated on chip) or to a host computer. H_{clk} and V_{clk} are both provided to allow easy access to any cell on the chip. Inputs and Targets are sampled and held in boxes labelled with the same name. A '1' in any register connects input 'A' (for Analog) to the associated sample-and-hold capacitor. Enable bits are scanned in and stored into the boxes labelled with the same name. These enable bits control whether an input or target is actually input into the network.

6.3.1 Recruiting Units

The X units can be recruited to perform several different tasks. They can behave as input, output, bias, or hidden units.

6.3.1.1 Input Units

By enabling the appropriate external inputs, the current sunk by the external input circuitry, shown in Figure 6.4, provides external input. If the amount of this current is large compared with the level of current supplied by the network, the external input dominates the input to a particular sigmoidal unit. In this mode of operation, this sigmoidal unit is referred to as an **input unit.**

6.3.1.2 Output Units

If the target for an associated X unit is enabled, then this sigmoid is being used as an **output unit.**

6.3.1.3 Bias Units

Many problems require **bias units.** Bias units have constant outputs, irrespective of the input pattern being applied. Bias units are implemented on the chip by dedicating several processing elements (X units) as bias units. These units receive the same external input for every input pattern presented. In addition, I specify a target for these units to match the applied input, so that the weights feeding the unit change in the direction to make the output of the bias unit more consistent with the inputs.

6.3.1.4 Hidden Units

X units which have neither the input nor target enabled are referred to as **hidden units.** These units are essential for solving tough problems. It may prove useful to actually drive a hidden unit to a specified value by enabling its target input. This technique can encourage symmetry breaking.

6.3.2 Scanners for Visualizing Network Parameters

Also shown in Figure 6.6 are scanners used to scan values off the chip. These scanners are designed to connect to a NEC Multisync monitor and are described in detail by Mead [46]. The network parameters which are monitored during training are the weights, the outputs of the X and Y layers, the inputs, the targets, and the error signals. This information is an invaluable debugging aid.

The same scanners can be used to scan values off the chip or into the chip. There is a clock for the horizontal and vertical scanners. Software was written to move anywhere onto the grid and read off the associated values.

6.4 Demonstration of Learning

This section describes the process of training the network. To train, I supply the inputs and associated targets. With a UV source turned on, the gradient signals generated on the chip cause the weights to change such that the difference between the output units and desired target is minimized.

6.4.1 Control of the Learning Rate

It is necessary to control the learning rate on the chip. The learning rate is the constant of proportionality, by which the weights change in the learning rule. In my case, the learning rate is τ_w. There are several parameters I can control to change this rate. These parameters are length of pattern-presentation time, UV intensity, power supply rail levels, and the weight reference voltage. I discuss each of these parameters.

6.4.1.1 Remove Learned Patterns (RLP)

By presenting each pattern to the chip for a controlled amount of time, I do not affect the τ_w of Equation 6.7, but I do change the amount a weight changes on each pattern presentation. When training a problem with several patterns, it is sometimes necessary to change selectively the time of presentation. For those patterns that have learned the correct output below some threshold in square error, I present the pattern to the network for a short time. For those patterns with a square error above the threshold, I present the pattern to

the network for a longer time. I call this technique **Remove Learned Patterns** (RLP), since learned patterns are effectively removed from the training epoch. I use this technique use to train the two-input parity problem on the chip discussed later in this section.

6.4.1.2 Controlling UV Intensity

By controlling the intensity of UV exposure to the chip, I can control the learning rate. I use filters, which are a proprietary glass from UVP, Inc., which allow transmission of UV. By stacking several filters, I control the level of UV intensity reaching the chip. I also reduce this intensity by moving the UV source farther from the chip. I place the source three or four cm from the chip for much of the training presented here.

6.4.1.3 Power Supply Rail

Another control I have over the size of the weight change is the power supply rail. By changing the supply voltage, I am able to change linearly the size of the weight modification when I present a pattern to the chip for a fixed time.

6.4.1.4 Weight Reference

The weight reference is used to balance weight increments with weight decrements. By moving the weight reference closer to Gnd than to $V_{\rm dd}$, I am able to decrease the weight decrements and increase the weight increments. Moving the weight reference closer to $V_{\rm dd}$, I get the opposite effect.

6.4.2 Training Examples

Here I show training data from the chip for four different training runs. For all four cases, the output unit was chosen arbitrarily as X unit number 11. There are two input units for each of the training examples, which are arbitrarily chosen as X units 2 and 3. There are two bias units for each of the runs, as well, which are arbitrarily chosen as units 5 and 6 (unit 5 is a negative bias, that is it always receives a negative input, and unit 6 is a positive bias). The training data is plotted as number of pattern presentations versus the square error of the output. For the error plots, the square error shown can range from 4.0 down to 0. An error less than one implies the output is on the same side of zero as the required

target. In all cases, I assume the problem is solved if the square error is less than 0.9 for all patterns. I use this threshold for the Remove Learned Patterns algorithm as well.

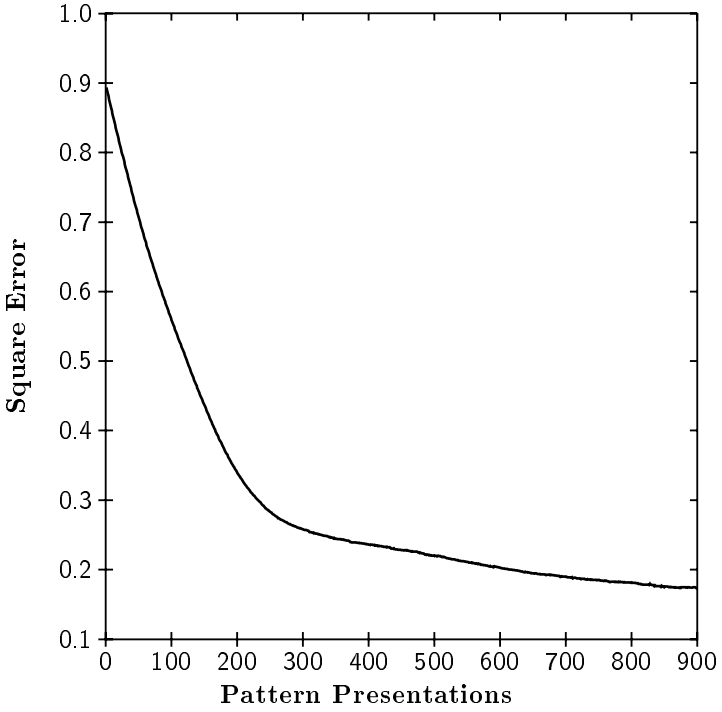


Figure 6.7 Square error versus pattern presentations for one pattern. For this curve the input pattern is '00' and the associated output is '0.' For consistency with multiple pattern training runs, the time axis is in units of pattern presentations, with each pattern presentation lasting 30 seconds.

6.4.2.1 One Pattern

The first task I put to the chip is to learn a single pattern from input to output. The input pattern is '00,' where '0' refers to a normalized input of minus one. The associated output for the network was '0.' This task is easy for the network and it requires none of the hidden units. Figure 6.7 shows the chip learning this single pattern. The pattern was presented to the chip and the resulting output was measured as a function of time. The error shown is the target minus the output squared.

6.4.2.2 Two Patterns

A more difficult problem of two patterns was next presented to the network. The patterns presented to the network were '00' and '01' with the associated outputs '0' and '1,' respec-

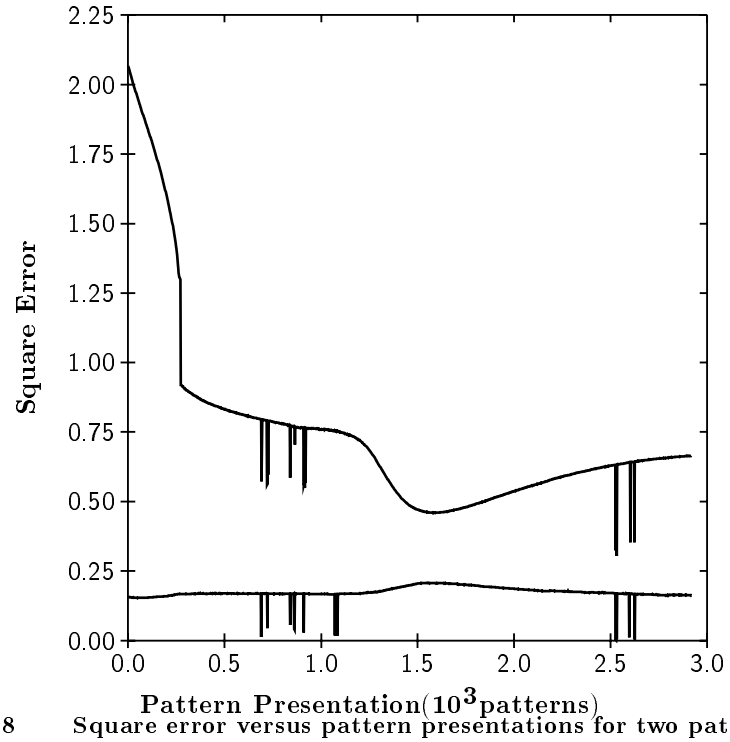


Figure 6.8 Square error versus pattern presentations for two patterns. For this curve the input patterns are '00' and '01' and the associated outputs are '0' and '1,' respectively. The time axis is in units of pattern presentations, with each pattern presentation lasting 30 seconds. The bottom curve is the '00' input pattern error curve.

Pattern	Inputs	Output
1	00	0
2	01	1
3	10	1
4	11	0

Table 6.1 Two-input parity problem. '0' is represented by a -1 in the network. '1' is represented by a 1.

tively. Figure 6.8 shows the error curve of training with two patterns following the training session with one pattern.

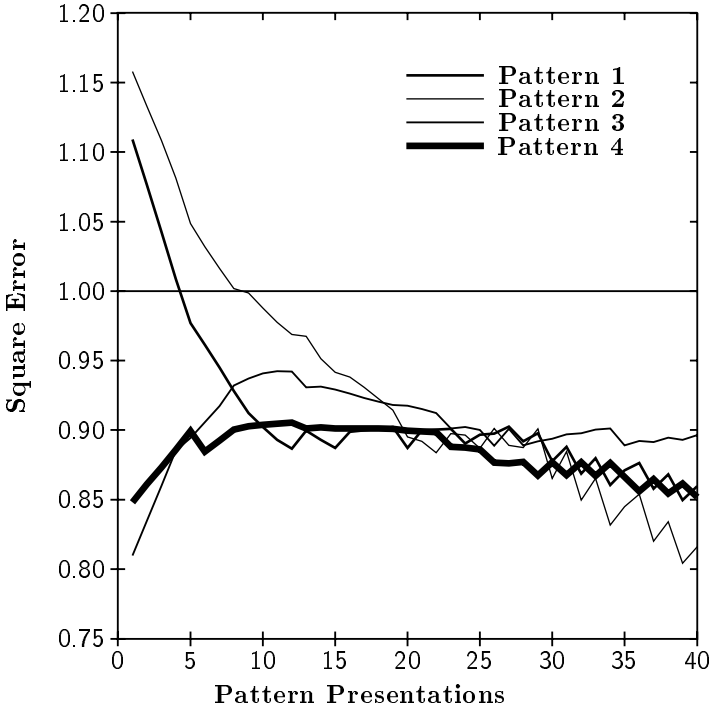


Figure 6.9 Shows square error versus pattern presentations for four patterns. The four patterns are from the two-input parity problem presented sequentially to the network. The remove learned patterns (RLP) threshold was set to 0.9. The jaggedness in the curves is caused by the RLP paradigm.

6.4.2.3 Two-Input Parity Problem

Next the **two-input parity** (XOR) problem was presented to the network. This problem has inputs and targets as shown in Table 6.1. Originally the task was not solvable for the

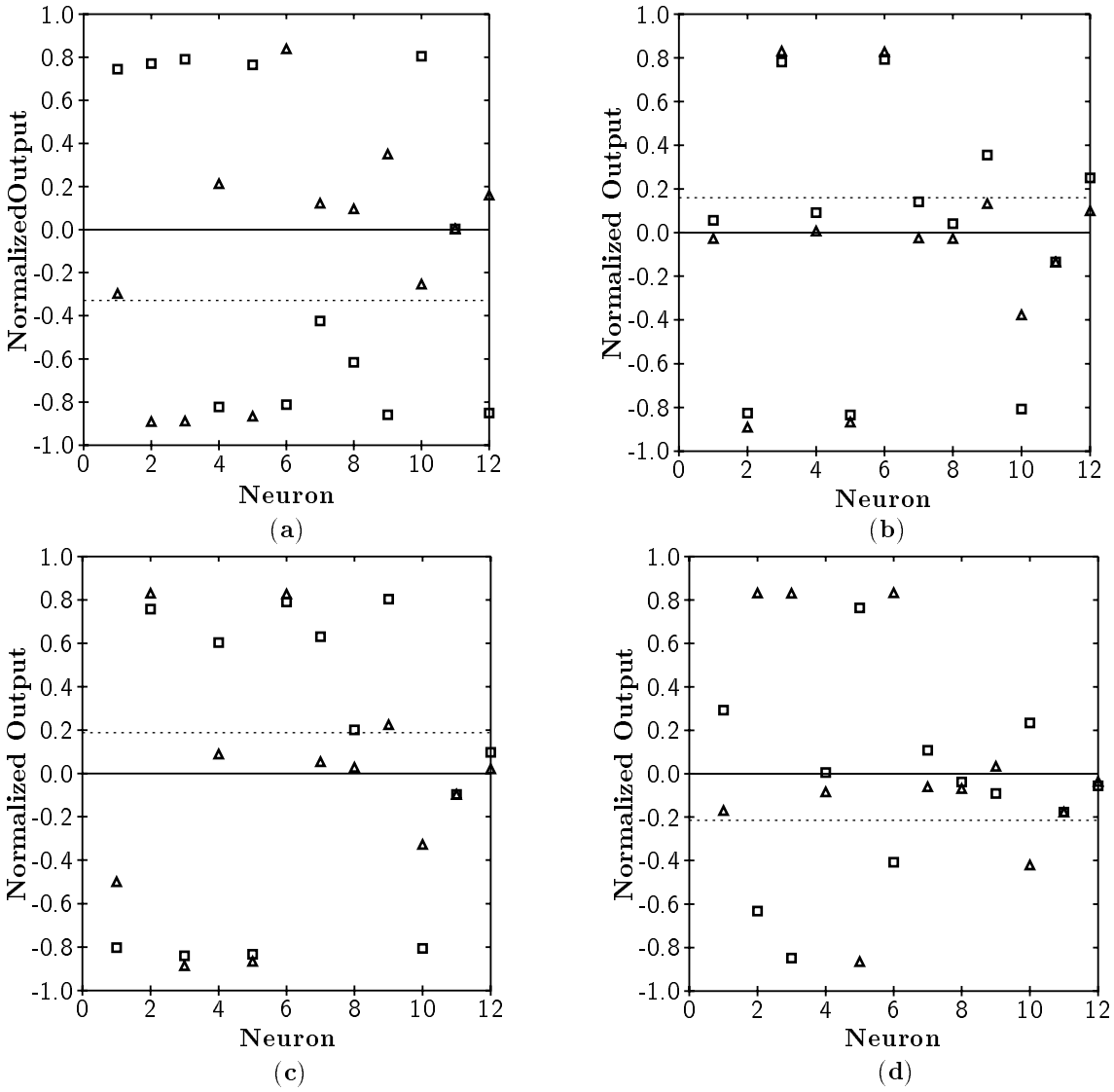


Figure 6.10 Output and gradient information for the four patterns of the two-input parity problem. The outputs are labelled with triangles, and the gradients are shown with boxes. The gradients are for the synapse which connect to the output node. Dotted line shows the value of the Y unit output. Figures (a) through (d) shows the response to inputs '00,' '01,' '10,' and '11,' respectively. Input units are triangles 2 and 3. Notice the input units show the appropriate input values. Bias units are triangles 5 and 6; 5 is a negative bias; 6 is a positive bias. Triangle 11 is the untrained output. All other units are hidden units.

network. Three of the patterns would be learned easily, but it was difficult for the network to solve the complete problem. I introduced into the learning paradigm the concept of removing learned patterns (RLP) discussed previously in this section. I implement RLP by showing a pattern to the network and subsequently reading the output. If the square error is less than some threshold, then the network is shown the next pattern. If the error is greater than the threshold, then the network is shown this pattern a fixed amount of time. This procedure is used until all patterns exhibit a square error less than the threshold at which time each pattern is shown to the network for a fixed time. RLP has the effect of reducing the learning rate for learned patterns.

Using RLP, I was able to get the XOR problem trained on the network. Figure 6.9 shows the training for four patterns. I use a square error threshold of 0.9 for RLP. Figure 6.10 shows the values of gradients and outputs for the four patterns of the XOR problem. This data was taken before any training was done.

6.4.2.4 Catastrophic Training Event

I trained the XOR problem for a long time after the network had learned. Figure 6.11 shows a catastrophe in the learning process. It is hypothesized that the network's input units received large enough signals from other X units to effectively cancel the inputs they were receiving externally.

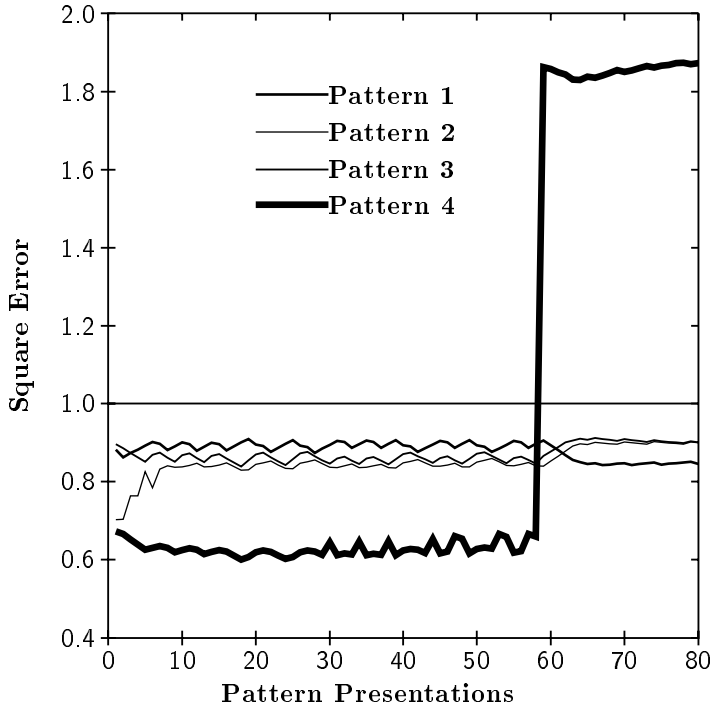


Figure 6.11 Catastrophic event during training. Shows square error versus pattern presentations for four patterns of the two-input parity problem presented sequentially to the network. The catastrophic event probably is due to the input units receiving inputs from other X units and effectively cancelling their external inputs.

Chapter 7

Conclusions

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

Winston Churchill, 1941

This chapter discusses the goals met by this research, novel ideas developed along the way, and directions for further research.

7.1 Goals

I designed, built, and tested a supervised learning neural network analog VLSI chip. The learning chip addresses the goals set forth in Section 1.2. The feedforward nonlinear mapping proceeds uninterrupted in tandem with the training process, which proceeds in real-time. The weight updates are calculated as part of a network and performed in parallel. The weights are nonvolatile in the absence of a UV source. The chip is run in the subthreshold mode of the transistor and consumes very little power.

7.2 Novelties

The work presented here is built upon the work of many others before me. I mention work that is novel for clarity and preciseness.

At the algorithmic level, I made modifications to Pineda's recurrent algorithm [53] to make it more suitable to the demands of the medium. I changed the weight update dynamics to give more immunity to the noise present in analog VLSI systems [51, 16]. Also, the details of the dynamics of the processing layer are new. The dynamics are different, because instead of RC dynamics as implied by the original Pineda system[53], a diode replaces the resistor. These new dynamics lead us to new stability criteria for the feedforward dynamics.

At the circuit level, I made several new contributions. The entire system is a novel implementation. The use of floating gates which use UV-activated conductances for control of synaptic weight storage is new as well. The designs of the sigmoid and modulated sigmoid are inventions. The circuitry controlling the bang-bang learning including the current-correlator follower is novel and was inspired by a need for a "dead zone" in the weight update for noise immunity. The method of using the current correlator to perform the derivative calculation of the sigmoid is also new. Other interesting novel circuits are contained in Appendix A. These include a CMOS subthreshold four-quadrant eight-transistor sine-approximation circuit and a subthreshold differential current inverter.

At the implementation level, there are also several novel ideas. The use of a multisync monitor and on-chip scanners to view on-chip parameters was pioneered by Mead [46]. I use this idea to monitor the weights and gradient information during training and feedforward processing. This tool was an invaluable debugging aid. The idea of calculating the gradient information collectively in the same fashion as the forward network is not new, but the implementation here is.

These innovations have led me to realize some future directions of research to follow to obtain more robust and simpler designs of analog VLSI learning systems.

7.3 Recommendations

Much of the work here has suggested new and better ways of constructing a remote low-power learning system. Here I outline some of these suggestions.

Do not use a UV source. It is not compatible with the desire for a remote learning system. It is impractical to have a UV source for every chip residing at remote locations. In addition, UV is quite cumbersome to deal with. The dangers of UV sources also abound. The UV source is dangerous if not properly shielded. Also, ozone created by the source can be harmful if the experiments are not conducted in well-ventilated rooms. On the chip, the UV light complicates sensitive circuitry by generating minority carriers whereever the UV light is not shielded. This includes circuitry around the periphery of the chip, such as pads or sense amplifiers. More specifically, if the learning task involves visual inputs, such as those used by the silicon retina, the use of UV interferes with the desire to sense visible light on the chip. It is for these reasons that structures that use tunnelling or hot-electron injection are much better suited for remote-learning chips.

Construct sparsely connected architectures, which are better suited for large-scale neural networks. Fully-connected architectures are interesting for their mathematical simplicity, but impractical as the number of neurons grows. For this reason, solutions will be specific to the task, rather than general purpose and will demand more locally-connected architectures.

Specific changes to the network implemenation I recommend are varied and numerous. Two of the most germane recommendations are outlined here. Remove the derivative information being passed to the error network and provide an explicit weight zeroing method. The network could have recruited hidden units more effectively if the error propagation was not diluted by the derivative information. Qualitative results suggest removing the derivative information altogether from gradient-descent algorithms does not prevent the system from converging and sometimes even helps the system converge. I believe this change would help the convergence of the network on the tougher training tasks. The second recommendation of providing weight zeroing structures is critical for practicality of using the chip. This chip relied on each weight decaying to the same analog level, but since there were variations in conductive and capacitive coupling, the zeroing by decay simply did not happen. This lack of zeroing made it very difficult to start the training over.

Overall, the results of this research look very promising. The goals of a learning chip which adapts as it is processing information has been challenging, but I am certainly much closer to attaining this goal in a robust and viable way.

Appendix A

Other Current-Mode Circuits

In this appendix, I introduce some new current-mode circuits. The four-quadrant sine-approximation circuit uses the current-correlator circuit discussed in Section 3.3. The inverted-differential-pair circuit is an alternative to the differential-pair circuit discussed in Section 3.2.

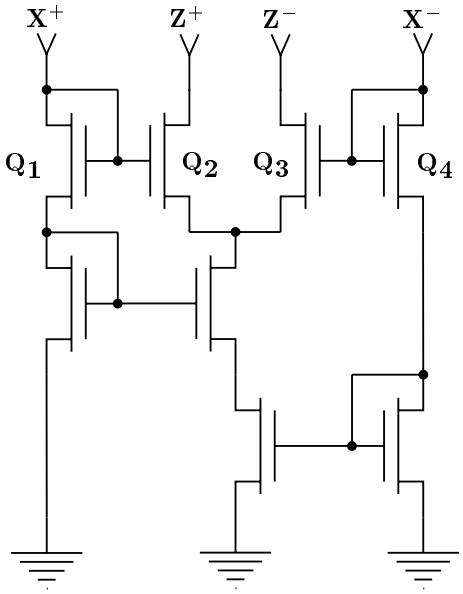


Figure A.1 **The four-quadrant sine-approximation circuit. It receives inputs through X^+ and X^- and provides outputs through Z^+ and Z^-. z has an error of approximately two percent from an ideal sine.**

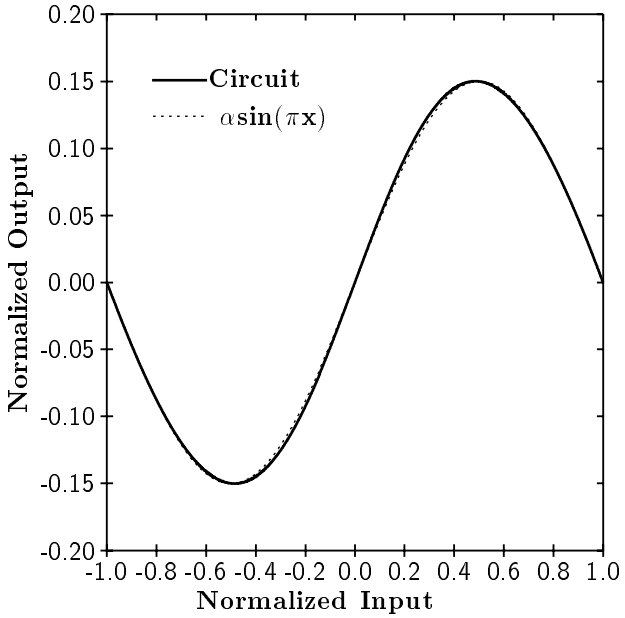


Figure A.2 The ideal four-quadrant sine-approximation circuit compared with $\alpha sin(\pi x)$. This circuit ideally obeys Equation A.1

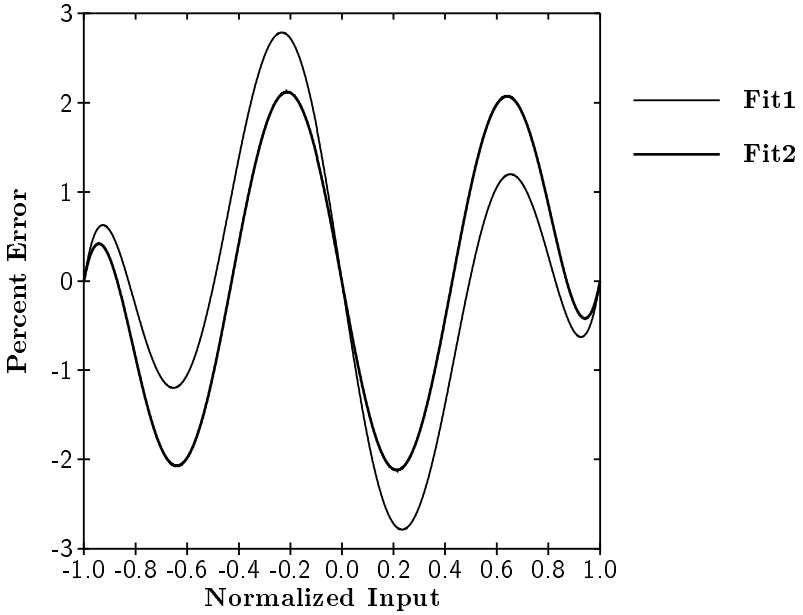


Figure A.3 Error in the four-quadrant sine-approximation circuit. The error is defined in Equation A.2 for two different values of α. Fit1 is with $\alpha = 0.1501$, which is the maximum for Equation A.1 and Fit2 is with $\alpha = 0.1516$, which is near minimax error.

A.1 The Four-Quadrant Sine-Approximation Circuit

Using the current-correlator circuit discussed in Section 3.3 as the bias to a differential pair, we can obtain an approximation to a sine function. Figure A.1 shows the sine circuit. Figure A.2 shows the output of the ideal function of the sine circuit compared[1] with $\alpha sin(\pi x)$ over the same period, where α is used to scale the sine to fit to the circuit output. The equation the sine circuit computes is

$$z = \frac{1}{2}\left(\frac{x - x^3}{1 + x^2}\right), \tag{A.1}$$

where both z and x are normalized by the total input current. z attains a maximum (and minimum) at $x = \pm\sqrt{\sqrt{5} - 2} \approx \pm 0.486$, where $z \approx 0.1501$. Figure A.3 shows the difference between the sine circuit approximation and $\alpha sin(\pi x)$ for $\alpha = 0.1501$ (Fit1 in Figure A.3) the maximum of the Equation A.1 and for $\alpha = 0.1516$ (Fit2 in Figure A.3) where we achieve near minimax error. The error used is given by

$$Error = \frac{\alpha sin(\pi x) - z}{\alpha} \cdot 100, \tag{A.2}$$

where z is the output of the circuit given in Equation A.1. The maximum deviation from ideal is about $\pm$ two percent for the near minimax case. The **minimax error** is defined as the minimization of the maximum error. The use of near in **near minimax error** is used to mean the search for minimax is performed iteratively and approximately rather than analytically.

Seevinck [57] reports a current-mode sine circuit built with **bipolar junction transistors** (BJTs), which has better accuracy, but at the expense of more transistors. Seevinck also has a sine circuit which performs Equation A.1 which uses seven BJTs.

The circuit shown here is designed for a CMOS FET technology. The advantage of this circuit over Seevinck's is that no extra bias is required. In my circuit, the input sets the scale for the output. This scaling may or may not be desirable depending on the application.

[1] The sine- approximation circuit is the only circuit that I have not fabricated in this dissertation. Thus, I am not showing data, but rather theoretical curves that should approximate closely the circuit.

A.1.1 Intuition for Four-Quadrant Sine-Approximation Circuit

Intuitively, the sine circuit has three places where the output is zero. When the inputs are equal, the output is zero since Z^+ and Z^- are equal. When the inputs are very different, the output of the correlator becomes small and the output also becomes small.

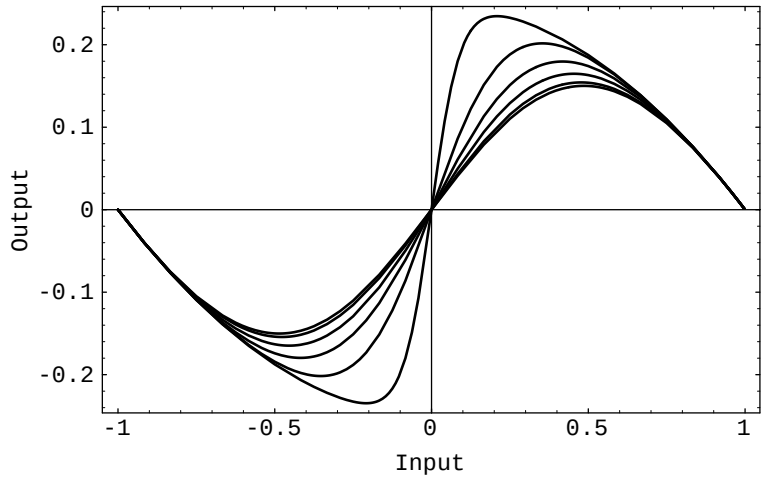


Figure A.4 **Family of curves for the sine-approximation circuit obtained by varying κ. The curve which is most skewed relative to an ideal sine wave (it also has the largest amplitude) has $\kappa = 0.1$. The family of curves has κ increasing as the curves drop in amplitude. κ is 0.1, 0.3, 0.5, 0.7, 0.9, and 1.0 for the curves shown.**

A.1.2 Effect of κ

If now we use the FET equation which takes κ into account (Equation 3.3), we get the following relationship for the sine circuit,

$$\frac{1+\hat{z}}{1-\hat{z}} = \left(\frac{1+x}{1-x}\right)^{1+1/\kappa}, \tag{A.3}$$

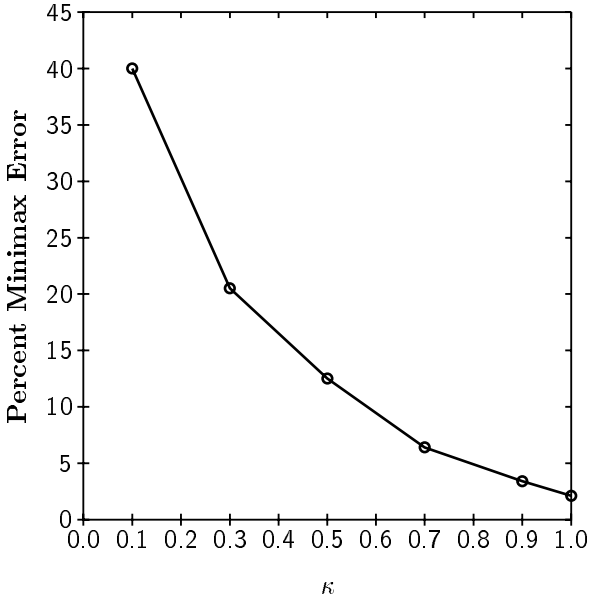


Figure A.5 Dependence of near minimax error on κ.

where $\hat{z} \equiv \frac{Z^+ - Z^-}{Z_c}$, $Z_c = \frac{X_t}{2}(1 - x^2)$, and $x \equiv \frac{X^+ - X^-}{X_t}$. Solving for $\hat{z}$, we find,

$$\hat{z} = \frac{-1 + x + \left(\frac{1+x}{1-x}\right)^{1/\kappa}(1+x)}{1 - x + \left(\frac{1+x}{1-x}\right)^{1/\kappa}(1+x)}. \tag{A.4}$$

Now define $z \equiv \frac{Z^+ - Z^-}{X_t}$. Then,

$$z = \frac{1}{4}\left(\frac{-1 + x + \left(\frac{1+x}{1-x}\right)^{1/\kappa}(1+x)}{1 - x + \left(\frac{1+x}{1-x}\right)^{1/\kappa}(1+x)}\right)(1 - x^2). \tag{A.5}$$

For $\kappa = 1.0$, Equation A.5 reduces to the sine-approximation equation used previously (Equation A.1). Figure A.4 shows the behavior of the ideal circuit for various values of κ. Shown in the Figure A.4 are six curves for values of κ ranging from 0.1 (largest amplitude) to 1.0 (smallest amplitude). Figure A.5 shows how the near minimax percent error increases as κ becomes smaller. A reasonable range for κ is between 0.6 and 0.9, where the near minimax error remains below 10 percent.

For two percent accuracy, we can place each of the transistors in its own well, with the well tied to each individual source. Actually, only four separate wells are needed. One well for Q_1 and Q_4 each of Figure A.1. One well only for both Q_2 and Q_3, and one well for all

the remaining transistors. With this circuit, we get the behavior simulated by $\kappa = 1.0$ and given in Equation A.1.

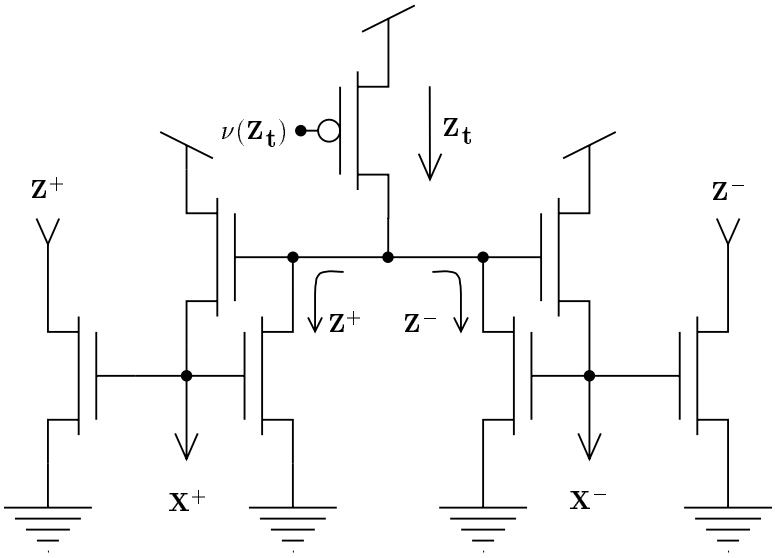


Figure A.6 The inverted-differential-pair circuit. The inputs are X^+ and X^- and the outputs are Z^+ and Z^-.

A.2 The Inverted-Differential-Pair Circuit

Figure A.6 shows the inverted-differential-pair circuit. The inverted-differential-pair circuit takes its inputs as X^+ and X^-; its outputs are Z^+ and Z^-. The circuit computes

$$z = -x, \tag{A.6}$$

where x and z are the normalized differential quantities of $X^\pm$ and $Z^\pm$ respectively. It has several advantages over the standard differential pair. First, inputs and outputs have the same direction so no additional current-mirror circuitry is required, unlike a standard differential pair. It uses the two-transistor current conveyor introduced by Boahen [8] and is a close relative of the winner-take-all circuit of Lazzaro [36], but performs a very different function. In the unipolar mode of operation, it performs

$$\frac{Z^+}{Z_t} = \frac{1}{\left(\frac{X^+}{X^-}\right)^\kappa + 1} \quad \text{and} \quad \frac{Z^-}{Z_t} = \frac{1}{\left(\frac{X^-}{X^+}\right)^\kappa + 1}. \tag{A.7}$$

Data taken from this circuit is shown in Figure A.7 along with theoretical fits given by Equation A.7.

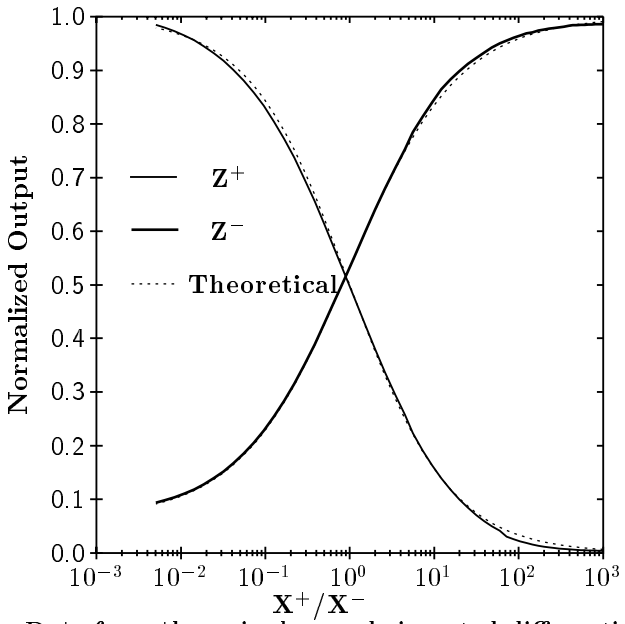


Figure A.7 Data from the unipolar-mode inverted-differential-pair circuit. The input to the circuit is $\frac{X^+}{X^-}$ and the outputs $Z^{\pm}$ are normalized by Z_t. Theoretical fits using Equation A.7 with $\kappa = 0.73$ for Z^+ and $\kappa = 0.67$ for Z^-.

This circuit may be useful in gain-control applications where one input sets the average of a time varying signal; the time varying signal is the other input. The output then reports how different one input is from the other in a logarithmic sense over several orders of magnitude, with a relatively linear range over about two orders of input magnitude.

Appendix B

Stability of the X Network

To guarantee global asymptotic stability for the X network, we follow the same approach used by Hirsch [25]. Hirsch proves the following theorem

Theorem B.1 *Assume there is a constant $-\mu < 0$ such that each Jacobian matrix $A = DF(y)$ has the property that $\langle A\xi, \xi\rangle \leq -\mu\|\xi\|^2$ for all $\xi \in \Re^n$. Then the dynamical system $\dot{x} = F(x)$ is globally asymptotically stable.*

In our case, the dynamical system we are considering is

$$\begin{aligned} \frac{dS_i^+}{dt} &= \gamma S_i^+(Z_i^+ - S_i^+) \equiv F_i^+, \\ \frac{dS_i^-}{dt} &= \gamma S_i^-(Z_i^- - S_i^-) \equiv F_i^-. \end{aligned} \tag{B.1}$$

See Section 6.1 for a description of this system. The vector F_i is a concatenation of vectors F_i^+ and F_i^-. Thus the Jacobian A has the following form:

$$A \equiv \left(\begin{array}{ccccc|ccccc} \frac{\partial F_i^+}{\partial S_i^+} & \cdots & \frac{\partial F_j^+}{\partial S_i^+} & \cdots & & \frac{\partial F_i^-}{\partial S_i^+} & \cdots & \frac{\partial F_j^-}{\partial S_i^+} & \cdots & \\ \vdots & \ddots & & & & \vdots & \ddots & & & \\ \frac{\partial F_i^+}{\partial S_j^+} & & \cdots & & & \frac{\partial F_i^-}{\partial S_j^+} & & \cdots & & \\ \vdots & & & \ddots & & \vdots & & & \ddots & \\ \hline \frac{\partial F_i^+}{\partial S_i^-} & \cdots & \frac{\partial F_j^+}{\partial S_i^-} & \cdots & & \frac{\partial F_i^-}{\partial S_i^-} & \cdots & \frac{\partial F_j^-}{\partial S_i^-} & \cdots & \\ \vdots & \ddots & & & & \vdots & \ddots & & & \\ \frac{\partial F_i^+}{\partial S_j^-} & & \cdots & & & \frac{\partial F_i^-}{\partial S_j^-} & & \cdots & & \\ \vdots & & & \ddots & & \vdots & & & \ddots & \end{array}\right). \tag{B.2}$$

Hirsch further shows that the conditions of the theorem are valid if the largest eigenvalue of $\frac{1}{2}(A + A^T)$ is $\leq -\mu$. By Gerschgorin's circle theorem this is obtained by the sufficient condition

$$A_{ii} + \frac{1}{2}\sum_{j\neq i} \|A_{ij} + A_{ji}\| \leq -\mu. \tag{B.3}$$

For our dynamical system, this gives us two sufficient conditions to guarantee global asymptotic stability. These conditions are

$$\begin{aligned} \gamma[Z_i^+ - 2S_i^+ + \frac{1}{4}\sum_{j\neq i} |f'(x_j)\Delta S_j \hat{w}_{ji}|] \leq -\mu, \\ \gamma[Z_i^- - 2S_i^- + \frac{1}{4}\sum_{j\neq i} |f'(x_j)\Delta S_j \hat{w}_{ji}|] \leq -\mu. \end{aligned} \tag{B.4}$$

For this condition to imply global asymptotic, we need to show that S_i^+ follows Z_i^+ closely and that S_i^- follows Z_i^-. We have not been able to show this conclusively, but only in special cases.

What we have done, is assume that the S_i's follow the Z_i's closely to see what conditions result. Assume S_i^+ follows Z_i^+ closely, such that $\frac{Z_i^+ - 2S_i^+}{Z_t} < -1 + \varepsilon$ for $0 \leq \varepsilon \ll 1$. Then we need to guarantee

$$\frac{1}{Z_t}\frac{1}{4}\sum_{j\neq i} |f'(x_j)\Delta S_j \hat{w}_{ji}| < 1 - \varepsilon. \tag{B.5}$$

We know that $0 \leq f'(x_j) \leq \sigma$. Also, we know $0 \leq \frac{|\Delta S_j|}{Z_t} \leq 1$. From this we find

$$\sigma \sum_{j \neq i} |\hat{w}_{ji}| < 4 - \varepsilon. \tag{B.6}$$

We can guarantee stability by adjusting the maximum gain of the sigmoid and the sum of the magnitude of the weights to which the neuron outputs. Remember though, if we cannot guarantee that S_i^+ follows Z_i^+ closely or the same for S_i^- and Z_i^-, then we have no guarantee for global asymptotic convergence.

References

[1] L.B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In M. Caudil and C. Butler, editors, *Proceedings of the IEEE First International Conference on Neural Networks*, pages 609–618, San Diego, CA, 1987.

[2] J. Alspector, B. Gupta, and R.B. Allen. Performance of a stochastic learning microchip. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 748–760, San Mateo, CA, 1989. Kaufmann.

[3] A.G. Andreou and K.A. Boahen. Synthetic neural circuits using current-domain signal representations. *Neural Computation*, 1:489–501, 1989.

[4] R.G. Benson and T. Delbrück. Direction-selective silicon retina that uses null inhibition. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, San Mateo, CA, 1992. Morgan Kaufmann.

[5] R.G. Benson and D.A. Kerns. UV-activated conductances allow for multiple time scale learning. *IEEE Transactions on Neural Networks*, 4(3):434–440, May 1993.

[6] K. Boahen and A. Andreou. A contrast-sensitive silicon retina with reciprocal synapses. In S. Hanson J. Moody and R. Lippmann, editors, *Advances in Neural Information Processing Systems, Volume 4*. Morgan Kaufmann, Palo Alto, CA, 1991.

[7] K.A. Boahen, A.G. Andreou, and C.A. Mead. A generalization of the translinear principle to the ohmic region of MOS transistor operation. *To appear in Electronic Letters*, 1993.

[8] K.A. Boahen, A.G. Andreou, P.O.Pouliquen, and A. Pavasovič. Architectures for associative memories using current-mode analog MOS circuits. In C. Seitz, editor, *Proceedings of the Decennial Caltech Conference on VLSI*. California Institute of Technology, MIT Press, 1989.

[9] L.R. Carley. Trimming analog circuits using floating-gate analog MOS memory. *IEEE Journal of Solid-State Circuits*, 24(6):1569–1575, December 1989.

[10] G. Cauwenberghs. A learning analog neural network chip with continuous–time recurrent dynamics. In *Advances in Neural Information Processing Systems*, San Mateo, CA, 1994. Morgan Kaufmann. *To appear*.

[11] P. Cleaveland. Memory chip stores data in analog form. *I&CS Control Technology for Engineers and Engineer Management*, 64(2):81–82, 1991.

[12] M. Cohen and A.G. Andreou. Current-mode subthreshold MOS implementation of the Herault–Jutten autoadaptive network. *IEEE Journal of Solid-State Circuits*, 27(5):714–727, May 1992.

[13] T. Delbrück. "Bump" circuits for computing similarity and dissimilarity of analog voltages. Computation and Neural Systems (CNS) Memo 10, Caltech, Pasadena, CA, 1991.

[14] T. Delbrück. A silicon network for motion discrimination that uses spatio-temporal interpolation. In *Society of Neuroscience Abstracts*, 1991. Session 143.8.

[15] T. Delbrück and C.A. Mead. Silicon adaptive photoreceptor array that computes temporal intensity derivatives. In *Proc. SPIE 1541*, volume 1541-12, pages 92–99, San Diego, CA, July 1991. Infrared Sensors: Detectors, Electronics, and Signal Processing.

[16] T. Delbrüück. *Investigations of analog VLSI visual transduction and motion processing.* PhD thesis, California Institute of Technology, 1993.

[17] D.J. DiMaria, F.J. Feigl, and S.R. Butler. Capture and emission of electrons at 2.4 eV-deep trap level in SiO_2 films. *Physical Review B*, 11(12):5023–5030, 15 June 1975.

[18] S. Eberhardt, T. Duong, and A. Thakoor. A VLSI analog synapse 'building block' chip for hardware neural network implementations. In L. H. Canter, editor, *Proceedings of IEEE 3rd Annual Parallel Processing Symposium*, pages 257–267, March 1989.

[19] F. Faggin and C.A. Mead. VLSI implementation of neural networks. In S. F. Zornetzer, J. L. Davis, and C. Lau, editors, *An Introduction to Neural and Electronic Networks*. Academic Press, New York, 1990.

[20] B. Gilbert. Translinear circuits: A proposed classification. *Electronics Letters*, 11(1):14–16, 1975.

[21] B. Gilbert. A monolithic 16-channel analog array normalizer. *IEEE Journal of Solid-State Circuits*, SC–19(6), 1984.

[22] L. A. Glasser. A UV write-enabled PROM. In Henry Fuchs, editor, *Chapel Hill Conference on VLSI (1985)*, pages 61–65, Rockville, MD, 1985. Computer Science Press.

[23] A. Guez, V. Protopopsecu, and J. Barhen. On the stability, storage capacity, and design of nonlinear continuous neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, SMC–18:80–87, 1988.

[24] J. G. Harris, C. Koch, and J. Luo. A two-dimensional analog VLSI circuit for detecting discontinuities in early vision. *Science*, 248:1209–1211, 1990.

[25] M. W. Hirsch. Convergent activation dynamics in continuous time networks. *Neural Networks*, 2:331–349, 1989.

[26] M. Holler, S. Tam, H. Castro, and R. Benson. An electrically trainable artificial neural network (ETANN) with 10240 "floating gate" synapses. In *The International Joint Conference on Neural Networks*, volume 2, pages 191–196, Washington D.C., June 1989.

[27] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 79:2554–2558, April 1982.

[28] J. J. Hopfield. Neurons with graded response have collective properties like those of two-state neurons. *Proceedings of the National Academy of Sciences USA*, 81:3088–3092, May 1984.

[29] J. J. Hopfield. The effectiveness of analogue "neural network" hardware. *Network: Computation in Neural Systems*, 1(1):27–40, January 1990.

[30] J. J. Hopfield and D. W. Tank. Neural computations of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[31] P. Horowitz and W. Hill. *The Art of Electronics*. Cambridge University Press, Cambridge, 1980.

[32] D. G. Kelly. Stability in contractive nonlinear neural networks. *IEEE Transactions on Biomedical Engineering*, 37:231–242, 1990.

[33] D.A. Kerns. *Experiments in Very Large Scale Analog Computation*. PhD thesis, California Institute of Technology, 1993.

[34] D.A. Kerns, J. Tanner, M. Sivilotti, and J. Luo. CMOS UV-writable non-volatile analog storage. In Carlo H. Séquin, editor, *Advanced Research in VLSI*, pages 245–261, University of California Santa Cruz, 1991. MIT Press.

[35] J. Lazzaro and C.A. Mead. A silicon model of auditory localization. *Neural Computation*, 1:47–57, 1989.

[36] J. Lazzaro, S. Ryckebusch, M.A. Mahowald, and C.A. Mead. Winner–take–all networks of O(n) complexity. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 703–711. Morgan Kaufmann, San Mateo, CA, 1988.

[37] R.P. Lippman. An introduction to computing with neural nets. *IEEE ASSP*, 4(2), April 1987.

[38] R. F. Lyon and C.A. Mead. An analog electronic cochlea. *IEEE Transactions. Acoustics, Speech, Signal Processing*, 36:1119–1134, 1988.

[39] M.A. Mahowald. *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function.* Ph.D. thesis, California Institute of Technology, Pasadena, California, 1992.

[40] K. Matsuoka. Stability conditions for nonlinear continuous neural networks with asymmetric connection weights. *Neural Networks*, 5:495–500, 1992.

[41] M. McLuhan. *Understanding Media.* 1964.

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

[43] C.A. Mead. Adaptive retina. In C. A. Mead and M. Ismail, editors, *Analog VLSI implementation of neural systems*, pages 239–246. Kluwer Academic Publishers, Boston, MA, 1989.

[44] C.A. Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, 1989.

[45] C.A. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78:1629–1636, 1990.

[46] C.A. Mead and T. Delbrück. Scanners for use in visualizing analog VLSI circuitry. *Analog Integrated Circuits and Signal Processing*, 1:93–106, 1991.

[47] C.A. Mead and M.A. Mahowald. A silicon model of early visual processing. *Neural Networks*, 1:91–97, 1988.

[48] A. Moopenn, T. Duong, and A.P. Thakoor. Digital-analog hybrid synapse chips for electronic neural networks. In D. S. Touretzky, editor, *Advances in Neural Information Processing 2*, pages 769–776, San Mateo, CA, 1990. Morgan Kaufmann.

[49] E.H. Nicollian and J.R. Brews. *MOS (Metal Oxide Semiconductor) Physcis and Technology.* John Wiley & Sons, 1982.

[50] Y. Nissan-Cohen, J. Shappir, and D. Frohman-Bentchkowsky. Dynamic model of trapping-detrapping in SiO_2. *J. of Applied Physics*, 58(6):2252–2261, 15 September 1985.

[51] A. Pavasovič. *Subthreshold Region MOSFET Mismatch Analysis and Modeling for Analog VLSI Systems.* PhD thesis, The Johns Hipkins University, 1990.

[52] A. Pavasovič, A.G. Andreou, and C.R. Westgate. Characterization of CMOS process variations by measuring subthreshold current. In C.O. Ruud and R.E. Green, editors, *Non-Destructive Characterization of Materials, vol. IV*. Plenum Press, 1991.

[53] F.J. Pineda. Dynamics and architecture for neural computation. *Journal of Complexity*, 4:216–245, 1988.

[54] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*. MIT Press, 1986.

[55] E. Säckinger and W. Guggenbühl. An analog trimming circuit based on a floating-gate device. *IEEE Journal of Solid-State Circuits*, 23(6):1437–1440, December 1988.

[56] F.M.A. Salam and Y. Wang. A real time experiment with a 50 neuron CMOS analog silicon chip, with on chip digital-learning. *IEEE Transactions on Neural Networks*, 2(4):461–464, July 1991.

[57] E. Seevinck. *Analysis and Synthesis of Translinear Integrated Circuits*, volume 31 of *Studies in Electrical and Electronic Engineering*. Elsevier Science Publishers B. V., The Netherlands, first edition, 1988.

[58] MOSIS Service. MOSIS parametric test results, run N32A, technology SCNA 2.0μm CMOS, 1993.

[59] MOSIS Service. MOSIS parametric test results, run N32X, technology SCPE 2.0μm CMOS, 1993.

[60] M.A. Sivilotti, M.R. Emerling, and C.A. Mead. A novel associative memory implemented using collective computation. In Henry Fuchs, editor, *1985 Chapel Hill Conference on Very Large Scale Integration*, pages 329–342, Rockville, MD, 1985. Computer Science Press.

[61] M.A. Sivilotti, M.R. Emerling, and C.A. Mead. VLSI architectures for implementation of neural networks. In J. Denker, editor, *Neural Networks for Computing (Snowbird 1986)*. American Institute of Physics, 1986.

[62] K. Sugawara, M. Harao, and S. Noguchi. On the stability of equilibrium states of analogue neural networks. *Transactions of the IECE*, J–66–A:258–265, 1983.

[63] S.M. Sze. *Physics of Semiconductor Devices*. Wiley & Sons, New York, second edition, 1981.

[64] D. W. Tank and J. J. Hopfield. Simple "neural" optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, CAS–33(5):533–541, May 1986.

[65] R. Tawel, R. Benson, and A.P. Thakoor. A CMOS UV-programmable nonvolatile synaptic array. In *The International Joint Conference on Neural Networks*, volume 1, pages 581–585, Seattle, WA, July 1991.

[66] E. Vittoz, H. Oguey, M. Maher, O. Nys, E. Dijkstra, and M. Chevroulet. Analog storage of adjustable synaptic weights. In U. Ramacher and U. Rückert, editors, *VLSI Design of Neural Networks*, pages 47–63. Kluwer Academic Publishers, 1991.

[67] E. A. Vittoz. Analog VLSI signal processing: Why, where and how? *To be Published in: Journal of VLSI Signal Processing*, 1993.

[68] E.A. Vittoz and X. Arreguit. CMOS integration of Herault–Jutten cells for separation of sources. In *Workshop on Analog VLSI and Neural Systems*, Norwell, MA, 1989. Kluwer Academic Press.

[69] L. Watts. *Cochlear Mechanics: Analysis and Analog VLSI*. PhD thesis, California Institute of Technology, 1993.