Pulse Stream VLSI Circuits and Techniques for the Implementation of Neural Networks

Alister Hamilton

Thesis submitted for the degree of
Doctor of Philosophy
The University of Edinburgh
March 1993

Acknowledgements

I would like to acknowledge the help and support I have received from many people during the time taken to complete the research for this thesis.

- I would like to thank both of my supervisors, Alan Murray and Martin Reekie, for their help, advice and support over the period of this research work. Thanks also to Professor Peter B. Denyer for early discussions on VLSI circuits.
- Special thanks to Donald J. Baxter and Stephen Churcher who I have worked closely with for over three years and who have added immeasurably to the effectiveness and enjoyment of the work reported herein.
- I would also like to thank Peter J. Edwards and W. Henry Bruce, the former for help with the Virtual Target Algorithm software, the latter for microcontroller information and general PC *support*. Thanks to Robin J. Woodburn for proof reading this thesis. Last, but not least, thanks to all those within the the Neural Network Research Group and the Imager Group at the University of Edinburgh, Department of Electrical Engineering, including those now at VLSI Vision Limited and elsewhere.

Declaration

The work presented in this thesis was carried out by the author, unless otherwise stated.

Alister Hamilton

Abstract

The recent dramatic increase in research activity in the study of artificial neural networks has resulted in non-linear systems that are capable of tasks such as classification, optimisation and content addressable memory. These successes, together with the desire to understand and mimic biological neural systems, have led to interest in the implementation of neural networks in both analogue and digital VLSI.

This thesis describes the pulse stream methodology for signalling, arithmetic and communication in VLSI neural networks. A review of conventional VLSI implementations of neural networks by case study highlights the significant contributions to date in the areas of digital, mixed signal and analogue neural networks. A review of pulsed VLSI implementations of neural networks highlights research activity that is most closely related to that contained within this thesis.

Several pulse stream neuron and synapse circuits have been developed and implemented in VLSI to test their operation. Methods for communicating neural state information between chips have been developed and implemented. The use of automatic set up procedures for analogue VLSI circuits are seen as essential to the development of large pulse stream neural networks and have been investigated. These circuits comprise *The Edinburgh Pulse Stream Cell Library*. The experience gained in developing this cell library has resulted in the development of a large pulse stream neural network chip, EPSILON (Edinburgh's Pulse Stream Implementation of a Learning Oriented Network), which has been demonstrated solving vowel recognition using *real world* data.

A recommendation for the use of pulse stream circuits for various categories of neural network based on the cells presented in this thesis has been made and forms the main contribution to knowledge of this work.

Table of Contents

Chapter 1

Introduction

Recent increased activity in neural network research has been stimulated by the observation that even simple biological organisms can perform complex computational tasks that conventional computing techniques have been unable to master adequately. This observation has helped inspire many network types that, while sharing the parallelism of biological networks, do not attempt to exactly mimic biological behaviour. Learning in these networks tends to be further removed from the current limited understanding of biological learning in large networks. This approach has been greatly facilitated by the advent of readily accessible high speed digital computers.

A more *bottom up* approach has resulted in scientists trying to understand directly how biological systems work, by observing nerve and low level processing functions, in an attempt to mimic their behaviour. Both approaches have provided exciting results, but there remains much progress to be made.

Artificial neural networks comprise a variety of architectures, some of which provide *instantaneous* responses, other networks need time to respond and are characterised by their time-domain or dynamic behaviour. Neural networks also differ from one another in their learning procedures that establish how and when the weights within the network change. Finally, networks operate at different speeds and learning efficiencies and therefore differ in their ability to respond accurately to input stimulus.

Neural networks, in contrast to conventional computers, are capable of learning associations, patterns or functional dependencies. Learning replaces the functional programming required for conventional computing. Users of neural networks do not specify the functional algorithm to be computed by each node of the network, rather they select what they believe to be the best architecture to solve their problem, the characteristics of the neurons and weights, and the learning procedure to be used. The application of inputs to the network, and the resultant training procedure, cause the network to form internal representations of the data that allow the network to solve the problem. The knowledge stored within the network is not necessarily evident in terms of a rule base or decision boundaries. Neural networks are therefore often used for their ability to form internal representations of data and generate decision boundaries within data that may not be otherwise apparent to the system designer.

While the use of conventional computing platforms has facilitated neural network research, the use of dedicated digital VLSI allows much greater computational bandwidth by designing systems that are matched to the requirements of neural network simulation. The use of analogue VLSI allows complex functionality in limited area by the careful use of transistor operating modes. Analogue systems are constrained by resolution, noise and system tolerances - which must be present in biological systems, and present distinct challenges, or perhaps opportunities to the VLSI neural network designer.

The remainder of this introduction outlines the distinctions between neural network learning schemes that have implications for analogue or pulsed VLSI implementations. It also reviews pulse stream signalling techniques and the remainder of the contents of this thesis.

1.1. A Classification of Neural Networks by Learning Procedure

Neural networks are distinguished in this thesis by their learning mechanism which helps define their suitability for implementation in *Pulsed* Neural Network VLSI. The essentially analogue nature of the pulsed circuits used to perform neural computation results in performance tolerances on circuit operation and implies that circuits will have limited precision and be affected by noise. These analogue effects have implications for network performance.

The first class of neural network is defined by the use of *off-line algorithmic techniques* to calculate a weight set. In this class, the neural network is not in the learning loop and therefore the learning procedure is **not** driven by errors produced by the network. The Hopfield training algorithm [1] is an example of this class which assigns weights in a network according to Equation 1.1

$$T_{ij} = \begin{cases} \sum_{s=0}^{M-1} V_i^s V_j^s & i \neq j \\ 0, i = j; 0 \leq i,j \leq M-1 \end{cases} \tag{1.1}$$

where T_{ij} is the connection weight from neuron i to neuron j and V_i^s which may be ± 1 is element i of the exemplar s. Since this algorithmic approach to neural learning takes no account of the characteristics of any analogue or pulsed VLSI it is likely that it will be least suited to this method of implementation.

The second classification category encompasses those networks whose learning mechanism is driven by errors between the network output and a target output. The best example of this is the back propagation algorithm [2] used in multi-layer feedforward networks. Figure 1.1 illustrates the basic principle involved in this learning mechanism where the output error is used to adjust weights within the network in order to minimise

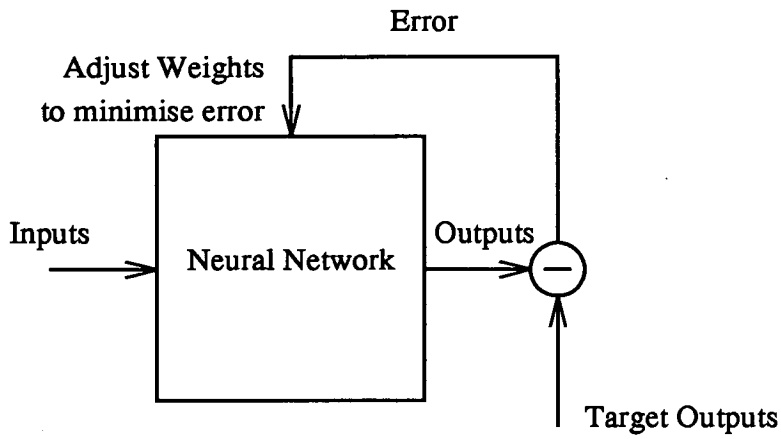


Figure 1.1 Neural Network Learning : Output Error Driven
Example : Multi Layer Perceptron.

the error. This category of neural network is likely to be most suited to analogue or pulsed VLSI implementations. Since the VLSI device may be physically placed in the learning loop shown in Figure 1.1 then the *error driven learning algorithm* can compensate for individual component tolerances within the network.

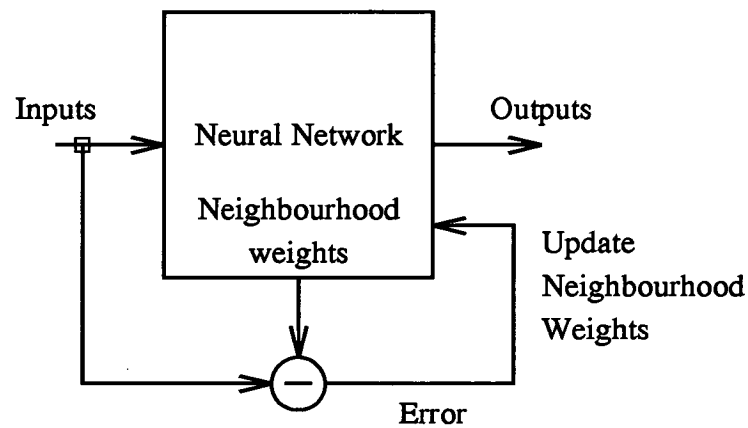


Figure 1.2 Neural Network Learning : Error Driven
Example : Kohonen Network.

The final category is a special case of *error driven learning*, where the error is defined between the input vector and the weight set. The Kohonen Network[3] is an example of this class, and the error driven learning mechanism is illustrated in the simplified diagram

of Figure 1.2. This special case of *error driven learning* is unlikely to produce good results using analogue or pulsed VLSI since the output units are not in the error feedback path. Any source of error resulting from the processing performed within the network will therefore not be compensated for naturally.

1.2. Choice of Neural Network for this study

These considerations have therefore resulted in *conventional error driven neural networks* being used by the author for the evaluation of pulsed neural networks within this thesis. On first consideration, this class of networks appear to be best suited to pulsed implementation. Pulsed neural networks applied to the Kohonen network as an example of the special case *error driven learning* has been studied by Baxter[4], who concludes that pulse stream neural network implementations are not suited to this class of network.

1.3. Pulse Stream Signalling

Pulsed representation of signals for the VLSI implementation of neural networks was first proposed by Murray and Smith in 1987[5] and later demonstrated using digital techniques[6] The pulsed nature of the neural state data has proved a robust method of data representation for VLSI implementation and by mixing analogue and digital VLSI techniques compact circuits for neural network functions have since been developed. This thesis presents these developments resulting in a series of recommendations for the implementation of Pulsed VLSI Neural Networks based upon *The Edinburgh Pulse Stream Cell Library* and the experience gained in the successful development of a large pulsed VLSI neural network.

As a necessary precursor to the main body of this thesis, therefore, this section highlights the fundamentals of pulse stream signalling and systems.

Neurobiological systems are known to operate, at least for the greater part, on pulse stream principles, and communications systems have used PAM (Pulse Amplitude Modulation), PWM (Pulse Width Modulation) and PCM (Pulse Code Modulation) for data transmission for some time.

Pulse stream encoding was first used and reported in the context of neural integration in 1987[5, 7], and has since been used by a number of other groups (see, [8-15] for example). The underlying rationale is simple:-

- analogue computation is attractive in neural VLSI, for reasons of compactness, potential speed, asynchronousness and lack of quantisation effects.
- analogue signals are far from robust against noise and interference, are susceptible to process variations between devices, and are not robust against the rigours of inter-chip communication.

- Digital silicon processing is more readily available than analogue.
- Digital signals are robust, easily transmitted and regenerated, and fast.
- Digital multiplication is area- and power- hungry.

These considerations all encourage a hybrid approach, seeking to blend the merits of both digital and analogue technology. The pulse-stream technique uses *digital* signals to carry information and control *analogue* circuitry, while storing further *analogue* information in the time dimension, as will be described below. A number of possible techniques exist, for coding a neural state $0 < S_i < 1$ on to a pulsed waveform V_i with frequency ν_i , amplitude A_i and pulse width δ_i. [16] Of these, 3 are relevant to the systems described in this thesis and are illustrated in Figure 1.3.

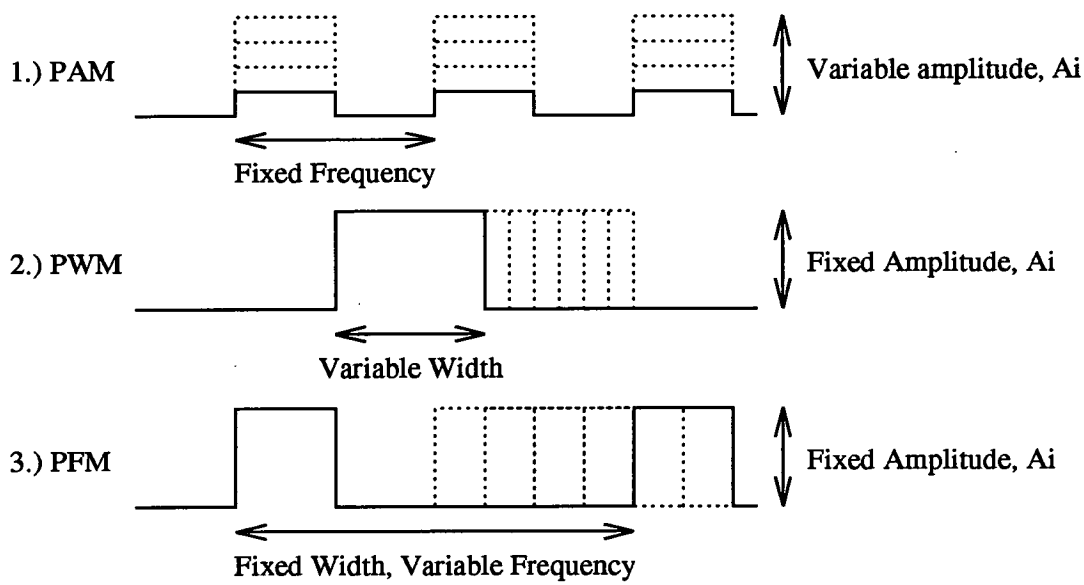


Figure 1.3 Pulse Stream Signalling Techniques : PAM (Pulse Amplitude Modulation), PWM (Pulse Width Modulation) and PFM (Pulse Frequency Modulation).

1.3.1. Pulse Amplitude Modulation (PAM)

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

1.3.2. Pulse Width Modulation (PWM)

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

1.3.3. Pulse Frequency Modulation (PFM)

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

In summary, pulsed techniques can code information across several pulses or within a single pulse. The former enjoys an advantage in terms of accuracy, while the second sacrifices accuracy for increased bandwidth. The ensuing thesis describes analogue VLSI devices that use a combination of the above techniques, moving between the analogue and digital domains as appropriate, to optimise the robustness, compactness and speed of the associated network chips.

1.4. Thesis Aims.

This thesis describes how a series of disparate circuit ideas and techniques were invented, explored and refined, and developed into a significant pulse stream neural system implemented in VLSI, and used in realistic demonstrations. The result of this research experience has allowed the author to propose a thesis of how pulse stream VLSI circuits and techniques may be applied to the implementation of neural networks.

The thrust of this thesis is, however, an exploration of the options within the pulse stream methodology. While the system alluded to above is undoubtedly impressive, it is the underlying **signalling, arithmetic** and **communication** options that form the central pillar of the thesis. The aim from the outset, therefore, was to explore the possibilities within pulsed methods, and thus to arrive at a taxonomy and series of recommendations.

1.5. Thesis Overview.

A global view of *conventional* techniques used for the VLSI implementation of neural networks is presented in Chapter 2. The use of the term *conventional* covers a wide range of techniques and styles and has simply been used here as a means of describing all implementations other than *pulsed* neural networks. The use of a case study approach to this review concentrating on functional VLSI allows the significant contributions in the field of digital, analogue and mixed signal implementations to be considered in some detail. Significant implementational details are highlighted for later reference in relation to the authors' own work.

Chapter 3 reviews the the range of pulsed techniques that have been applied to the VLSI implementation of neural networks. This more local view of VLSI for neural networks represents a body of work closely related to that presented in this thesis. Here also, digital, analogue and mixed signal techniques have been used and the most significant contributions are presented in case study form.

The Edinburgh Pulse Stream Cell Library, presented in Chapter 4, is a collection of synapse, neuron, inter-chip communication and associated circuits that have been designed by the author and others working in the Department of Electrical Engineering at the University of Edinburgh. The details of circuit rationale, design and results from HSPICE simulation, where appropriate, are presented here.

All the cells presented in Chapter 4 have been fabricated and the results from working VLSI are presented in Chapter 5. The experience and confidence gained from two functional VLSI test devices, at circuit and neural network system level, have resulted in the informed choice of cells from the library in implementing a large demonstrator chip. The resultant EPSILON (Edinburgh's Pulse Stream Implementation of a Learning Oriented Network) chip is a fully functional pulse stream neural network device with a maximum of 120 neural inputs, 30 neural outputs and 3600 fully programmable synapses. EPSILON is capable of operating in a variety of network architectures with a variety of programmable input and output modes.

The success of the EPSILON chip has resulted in the development of a system environment based on two EPSILON chips that allows the investigation of various neural network architectures. This system, outlined in Chapter 6, has resulted in the solution of a vowel classification problem using *real world* analogue data.

Chapter 7 draws together the experience gained from the Cell Library, VLSI test devices and EPSILON at both chip and system levels. A series of recommendations are presented in the light of this experience and that from the VLSI implementations reviewed in Chapters 2 and 3, as to when and where cells from the Edinburgh Pulse Stream Cell Library should be used.

Appendix 1 contains the mathematical derivation of the transfer characteristics of the pulse stream neuron circuits developed by the author and referenced in Chapters 4 and 5. Appendix 2 contains the circuit diagram of the phase frequency detector used in the phase lock loop described in Chapter 4. Appendix 3 contain the SPICE decks and net lists of all the circuits developed by the author and described in this thesis, while Appendix 4 contains the VLSI layout of these cells. Finally, appendix 5 contains a list of the author's published papers during the period of this research work.

Chapter 2

A Review of Conventional VLSI Implementations of Neural Networks

The aim of this chapter is to evaluate the major achievements to date in the *conventional* VLSI implementation of neural networks. The term *conventional* is used here to describe all VLSI implementations other than those using *pulsed* techniques.

On first consideration VLSI implementations of Neural Networks fall into two broad categories, namely digital and analogue. However, on closer inspection this boundary becomes less distinct. For example, some implementations use a mixed signal approach where analogue techniques are used for efficient implementation of functions such as multiplication and addition while digital techniques are used to facilitate easy interface to conventional digital processor environments or for robust signal communication across chip boundaries.

A distinction can also be drawn between network implementations by their degree of programmability. While some designers have implemented devices to solve a variety of neural network problems, others have implemented devices with fixed functionality. Indeed some researchers attempt to mimic biological functionality directly, while others use more abstracted models of neural behaviour.

For these reasons and for the purposes of review, conventional VLSI neural network implementations have been grouped into digital, mixed-signal, analogue programmable and analogue fixed function categories. While this review is not comprehensive, it highlights the significant activities in this field of research at present. Exemplars have been used in each of the above implementation categories to highlight the functional and practical advantages of the approach.

In reviewing *conventional* VLSI implementations of neural networks, pulsed implementations of neural networks and hence the work reported in the rest of this thesis will be placed in context.

2.1. Digital Implementations

In this section of the review design considerations such as arithmetic precision, bit-serial or bit parallel structures and systolic architectures are discussed in relation to the implementation of neural networks in VLSI.

Digital VLSI is a relatively mature technology and is supported by a wide range of computer aided design tools that allow rapid and reliable implementation. While transistor geometries in CMOS VLSI continue to shrink and chips increase in size and complexity the performance of digital VLSI will continue to improve. In addition, the design of digital neural networks that interface to existing computer architectures will provide a powerful simulation environment for the development of neural network algorithms and applications. It is for these reasons that the digital implementation of neural networks is attractive to the system designer.

This section of the review will not consider digital implementations that do not make use of custom VLSI. For this reason the impressive work of Aleksander[17] in developing WISARD, for example will not be reviewed here. The continuing improvement in parallel digital signal processor technology that facilitates the implementation of expandable computing engines capable of computation rates in excess of 800MFLOPS (Meridian's Powerstack[18], for example) likewise will not reviewed here.

2.1.1. Arithmetic Precision

One of the first problems in the digital implementation of neural networks is determining how many bits are required to represent physical states, parameters and variables in order to ensure learning and generalisation performance. As in any digital computer, there are memory and performance trade-offs between using, for example, floating or fixed point arithmetic.

In neural networks research a number of papers have appeared that consider the arithmetic requirements of various neural network algorithms. For example, the back-propagation training algorithm applied to the real world task of phoneme classification for a continuous speech recognition system[19] required 16 bit weight values to achieve training and classification results comparable to 32 bit floating point precision. In this example weight and bias values are scaled separately and rounding is used rather than truncation to reduce the precision of intermediary values.

Other research has led to forms of *reduced precision arithmetic*[20] where the smooth neuron activation function has been replaced by a "staircase" function so that the neuron is in any one of five states. This technique reduces multiplication to a simple shifting operation.

2.1.2. Bit-Serial versus Bit-Parallel Architectures

The Bit-Serial approach to VLSI design [21] is distinguished by the communication strategy employed. Digital signals are transmitted bit sequentially along single wires, as opposed to bit-parallel architectures which use the simultaneous transmission of words on

a parallel bus. This strategy leads to efficient communication within and between chips and provides outstanding advantages where communication issues dominate, as in many signal processing applications. Although bit-parallel designs have a lower computational latency, bit-serial designs lend themselves to pipelining and thus make optimal use of high clock rates.

The combination of reduced precision arithmetic and a bit-serial architecture has resulted in an efficient digital implementation of a neural network accelerator board[22].

2.1.3. Systolic Architectures

A systolic system [23] consists of a set of interconnected cells or processing elements (PEs) each capable of performing some simple operation. Information in a systolic system flows between cells in a systolic fashion and communication with the outside world occurs only at the boundary cells. Systolic architectures may be linear, rectangular, triangular or hexagonal to make use of higher degrees of parallelism. Data flow in a systolic system may be at multiple speeds in multiple directions. Unlike pipelined architectures where only results flow, both inputs and partial results flow in a systolic array.

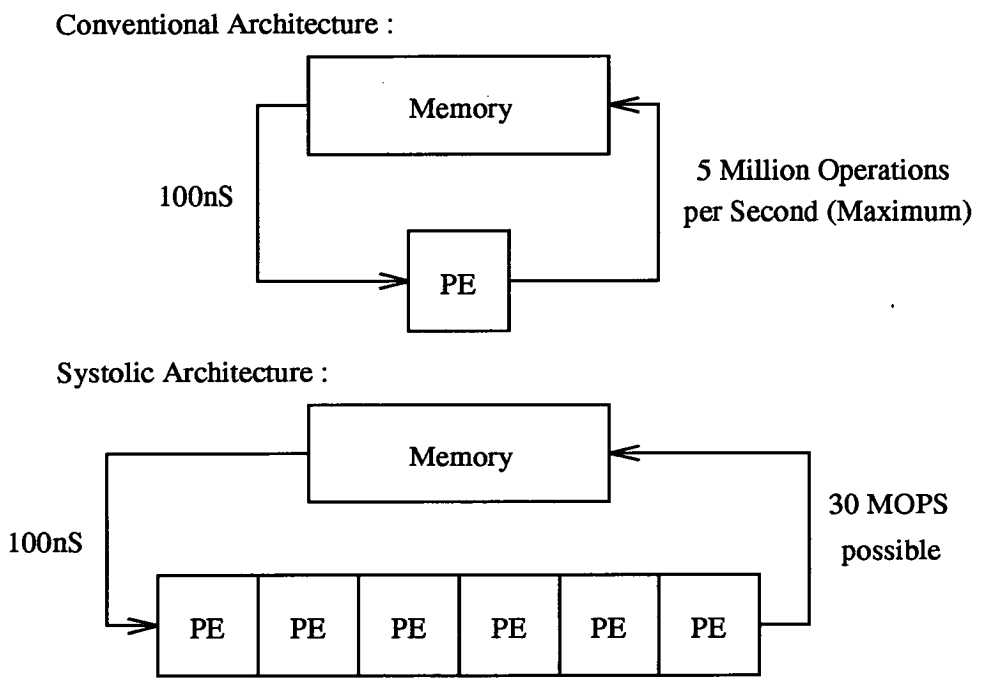


Figure 2.1 Basic Principle of a Systolic System

Systolic architectures have been developed to balance computation time with input output times of computer memory (or other *host*). In the example shown in Figure 2.1 the

conventional system is capable of computing results only as quickly as data can be supplied. The computational bottleneck in this system is memory access. If it is assumed that at least two bytes of data are read from or written to the memory for each operation then the maximum computation rate is 5 million operations per second. Orders of magnitude improvement in computational throughput are achievable if multiple computations are performed per memory access. The second diagram in Figure 2.1 illustrates this point where the systolic architecture with 6 processing elements is capable of a six fold increase in computation rate. The use of systolic architectures allows the designer to match the computation architecture to the memory structure so that computation time is balanced with data input/output rates.

The work of Blayo[24] is one example of systolic architectures applied to neural network VLSI.

2.1.4. A Digital Implementation Case Study : The Adaptive Solutions CNAPS Neurocomputer Chip

The development of the CNAPS (Connected Network of Adaptive Processors) was motivated by the desire to develop commercial hardware suitable for implementing any neural network algorithm and able to interface to existing computer architectures. The designers envisage that CNAPS would be considered the *microprocessor* of neural network computing. The architecture is general enough to implement every neural network algorithm examined by the designers (learning and non-learning) and many feature extraction computations including digital signal processing, pattern recognition and rule processing.

The chip architecture [25] consists of a number of simple Processor Nodes, PNs, operating in a SIMD (single instruction multiple data) configuration. Each PN has three buses, two 8 bit buses INbus and OUTbus, used for data input to and output from each PN, and a 31 bit bus, PNCMD, controlled by a sequencer to broadcast commands to individual PNs. In addition, there is a 2 bit inter-PN bus allowing connection to the nearest neighbouring PN. One possible multi-PN configuration depicting the bus interconnects is shown in Figure 2.2

The sequencer contains the program store, sequencing logic and input and output buffers for directly accessing a host memory when operated as a coprocessor. The sequencer has several basic command groups and controls the INbus and OUTbus including directing data to and from a hosts' system memory or parallel input/output port from off board.

Each PN contains a weight memory, adder, multiplier, logic unit and register file, a brief specification of each is given in table 2.1.

Each neuron in a network is represented by a state in a PN and is called a CN or Connection Node. In a basic configuration, one PN is used for each CN. In the case where there

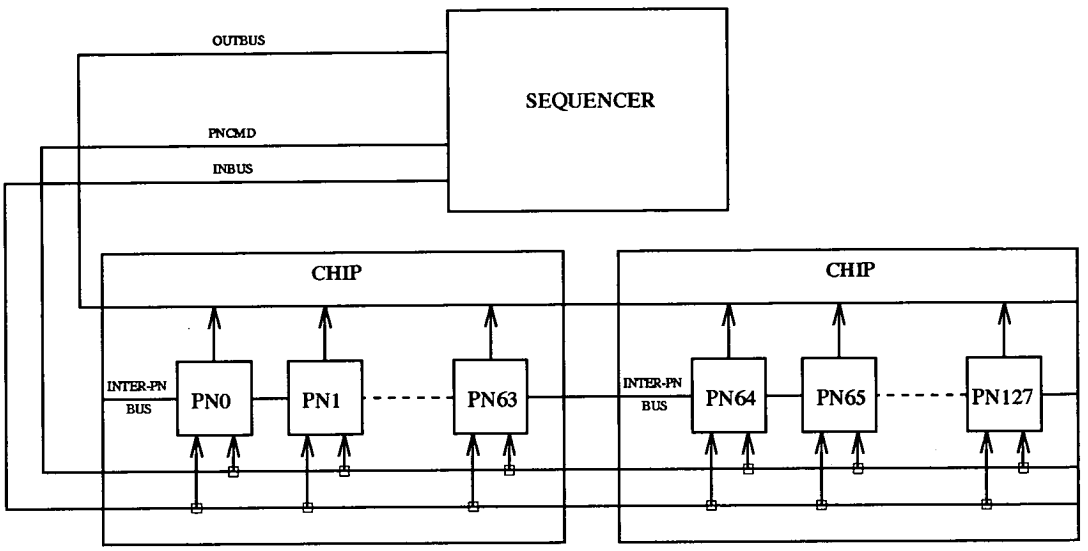


Figure 2.2 CNAPS Multi-PN Configuration.

Processing Node (PN) Functionality	
Weight Memory	4K Bytes. 8 or 16 Bit Modes
Adder	16 or 32 Bit Modes
Multiplier	Various Modes. 9 × 16 bit 2's Complement produces a 24 bit result
Shifter and Logic Unit	16 Bit data
Register File	32 16 Bit registers Some dedicated e.g. data width control
Internal Buses	2 16 Bit Buses

Table 2.1 Internal Contents of each Processing Node (PN).

are more CNs than PNs, each PN is used to emulate more than one CN. A group of PNs which the designers call a layer, takes a vector, multiplies it by a matrix of weights and creates a new vector. Any neural network architecture can be emulated using this function. Totally interconnected networks, for example the Hopfield network, use the output vector as the next input. Feed-forward networks can be thought of as several layers feeding each other successively.

A simple two layer network may be mapped onto an array of PNs as shown in Figure 2.3. Once CN0 - CN3's outputs have been computed, CN0's output is transmitted onto the

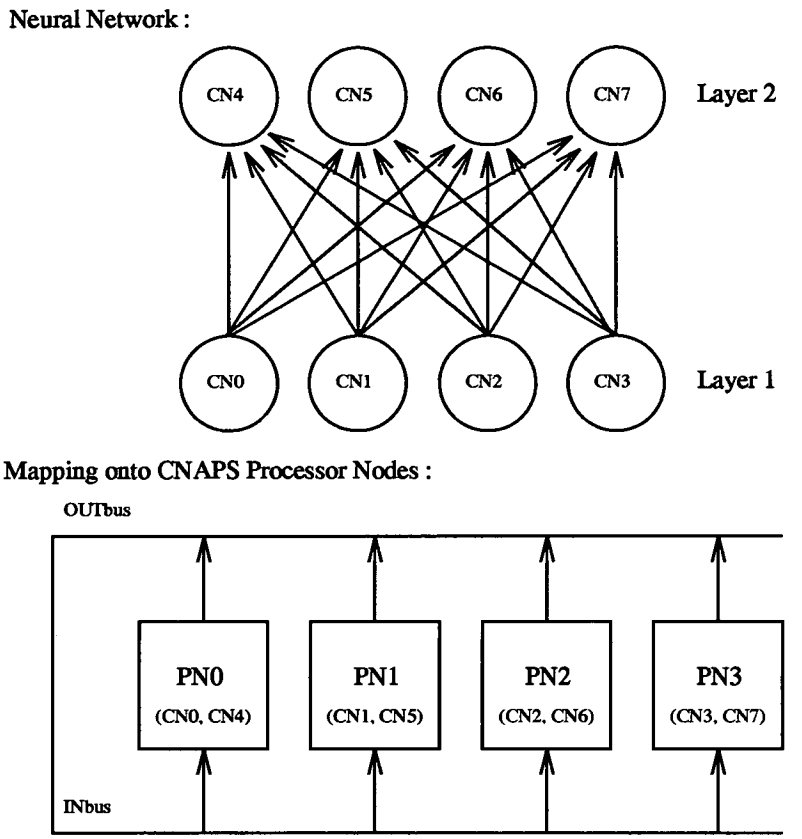


Figure 2.3 Processor Node (PN) Allocation for a Feedforward Network

OUTbus. It is then read into the INbus of all the PNs and used to calculate one connection for each CN in the second layer. Each first layer CN is broadcast in turn. Therefore n^2 connections are computed in n clock cycles.

An eight chip configuration has been used to perform the back-propagation training algorithm [26] on a network with 1900 inputs, 500 hidden nodes and 12 output nodes. Since the CNAPS chip is restricted to fixed point integer arithmetic the position of the binary point must be chosen. In this application, the weights range from -8 to +8 with four bits to the left of the binary point, including a sign bit, and twelve to the right. Input and output vectors are represented as 8 bit unsigned integers with binary point to the left. The learning rate is represented by an 8 bit integer with two bits to the left of the binary point, and the error by an 8 bit signed integer at the output layer and the same representation as the weights at the hidden layer.

A speed of 2.3 Giga training weight updates per second or 9.6 Giga feed forward connections per second was observed. This result has been compared to the same task on other computers and the results shown in table 2.2.

Machine	MCUPS	MCPS	Weights
SUN 3	0.034	0.25	fp
SAIC SIGMA-1		8	fp
WARP	17		fp
CRAY 2	7		fp
CRAY X-MP		50	fp
CM-2 (65,536)	40	182	fp
GF-11 (566)	901		fp
8 CNAPS CHIPS	2,379	9,671	16 bit int

Table 2.2 Back-Propagation Performance of various computers compared to CNAPS for a network with 1900 inputs, 500 hidden units and 12 output units. fp = floating point, int = integer.

The Adaptive Solutions CNAPS chip is therefore a very powerful general purpose digital neural computing chip. It is implemented in $0.8\mu m$ CMOS technology on a die $26.2mm \times 27.5mm$. The backpropagation benchmark problem highlights the speed of the device in comparison to other computer configurations. Back propagation implementations typically use 32-bit floating point arithmetic, however the designers quote Baker[27] who has shown that 16-bit integer weights are sufficient for backpropagation training and much less precision required for use after training.

2.2. Mixed Signal Implementations

Analogue and digital techniques are combined in mixed signal implementations in order to exploit the potential of both techniques. The introduction of analogue techniques introduces a tolerance into the system due to the mismatch of analogue parts. System design considerations must therefore ensure that these tolerances do not adversely affect the algorithm performance.

2.2.1. A Mixed Signal Implementation Case Study : AT & T Bell Laboratories ANNA Chip

The designers of ANNA (Analogue Neural Network Arithmetic unit) have adopted a mixed analogue and digital design to exploit the low computational accuracy they claim is typically required by neural network algorithms, while addressing system integration issues which call for a digital interface[28, 29]. The relatively low resolution of the chip, 6 bit weight and 3 bit state representation, has proved capable of handwritten digit recognition with reported classification error rates of 5.3%. This figure compares with a

classification error rate of 4.9% when the network was simulated using floating point precision and back propagation training and a human performance of 2.5% on the same data[30].

ANNA Chip Features	
Number of Neurons	8
Inputs per Neuron	16 to 256
Weights	4096
Bias Units	256
Weight Accuracy	6 bit
State Accuracy	3 bit
Input State Data Rate	120 Mbits per second
Output State Data Rate	120Mbits per second
Maximum Computation Rate	10GC per second
Total Weight Refresh	110μs
Clock Rate	20MHz
Technology	0.9μm CMOS
Die Size	4.5mm x 7mm

Table 2.3 ANNA Chip Features

The ANNA chip comprises of eight neurons with either 64, 128 or 256 inputs. The input neural state information which has a resolution of 3 bits (2 bits magnitude and one bit sign), is presented to a multiplier array via a barrel shifter at a rate of 120Mbits/s. The weight values are held in RAM and converted to analogue values on chip for multiplication at data rates approximately equal to the access times of static RAM. The magnitude of the weight value is stored dynamically at each synapse or multiplier while the sign bit is stored as a digital value.

The multiplier is implemented as a multiplying digital to analogue converter (MDAC), a simplified schematic of which is illustrated in Figure 2.4. The weight magnitude controls the current, I, while the weight and state sign bits are exclusive OR'ed to control the sign of the multiplication. Current is either dumped onto the bus or removed from the bus depending on the sign. The magnitude of the current is determined by the analogue weight value and the digital state bits X0 and X1 as indicated in Table 2.4.

The weight load circuitry is of interest in this design due to the fast weight refresh times achieved. The basic circuit diagram of the weight refresh circuit is shown in Figure 2.5. An on chip DAC generates a current IDAC proportional to the weight magnitude that is *copied* to the synapse circuit. The voltage required to generate IDAC appears across the

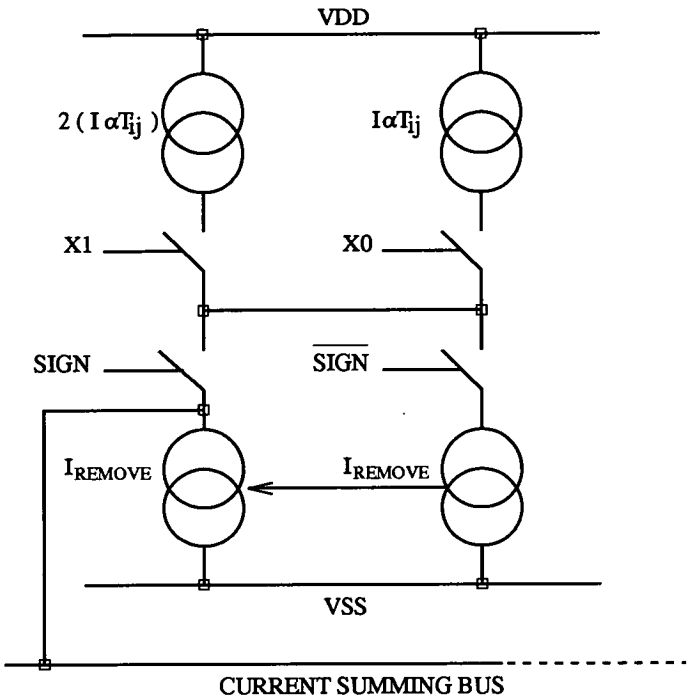


Figure 2.4 Simplified Schematic of MDAC used in ANNA

X1	X0	SIGN	CURRENT SUMMING BUS
0	0	Don't Care	No Change
0	1	1	$+ (I\alpha T_{ij})$
1	0	1	$+2 \times (I\alpha T_{ij})$
1	1	1	$+3 \times (I\alpha T_{ij})$
0	1	0	$I_{REMOVE} = - (I\alpha T_{ij})$
1	0	0	$I_{REMOVE} = -2 \times (I\alpha T_{ij})$
1	1	0	$I_{REMOVE} = -3 \times (I\alpha T_{ij})$

Table 2.4 Operation of the MDAC used in ANNA

storage capacitor, C, and is held there until the next refresh cycle. The use of *current-mode* techniques in this circuit gives a very fast update time measured in terms of nanoseconds as indicated by the weight refresh times quoted in Table 2.3.

The neuron circuit scales the current output from the MDAC multiplier circuits and converts it into a 3 bit digital representation. Successive-approximation DAC techniques have been employed in the neuron circuitry, the nonlinear sigmoid function is a result of

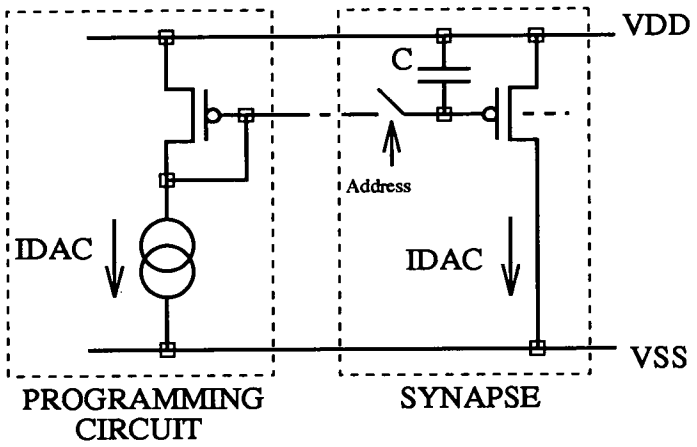


Figure 2.5 ANNA Weight Refresh Circuitry

the overload characteristics of the converter. A scaling factor permits control of the slope of the nonlinear function.

As mentioned earlier, ANNA has been used to perform high speed character recognition. All training was performed off-chip using a computer simulation and the back-propagation learning algorithm. A five layer network model was used with the final layer being implemented using a conventional digital signal processing device (DSP) in order to achieve the necessary resolution for good discrimination. The throughput of the chip on this problem was 1000 characters per second or 130M connections per second. Although this performance is less than the maximum possible performance of the chip, due to the inefficient use of the chip architecture it compares favourably with the 20 characters per second achieved when the problem was evaluated on a DSP.

2.3. Analogue Implementations

Analogue VLSI design techniques are not as highly automated as those used in digital VLSI design. The simulation of large systems becomes more difficult and great care is required in the design, layout and testing of devices. The advantages of using analogue techniques, however, are considerable. By careful exploitation of transistor characteristics operations such as multiplication and addition, which form the basis of many neural network models, can be performed very efficiently with few components.

2.3.1. A Programmable Analogue Implementation Case Study : The Intel ETANN Chip (80170NX)

The Intel 80170NX Electrically Trainable Neural Network (ETANN) is a fully analogue neural network chip with non-volatile EEPROM memory technology for analogue storage of synaptic weights. The chip contains 10,240 synapses organised into two arrays and has 64 input and 64 output neurons. The use of two synapse arrays allows the implementation of Hopfield networks [31] and networks capable of processing and producing sequences of patterns[3].

ETANN Chip Features	
Number of Output Neurons	64 Variable Gain
Inputs per Neuron	128 Analogue (maximum)
	10,240 Synapses
Synapse	Analogue Non-Volatile Memory
	4 Quadrant Gilbert Multiplier
Multiplication Accuracy	6Bit × 6Bit
Synapse Arrays	Two 80 × 64 arrays
Chip Area	12.2mm × 7.5mm
Technology	1μm CMOS EEPROM
Performance	2000MCPS Maximum

Table 2.5 ETANN Chip Features

A summary of the main features of the ETANN device is given in Table 2.5. The synapse circuit used in this device is an NMOS version of the Gilbert Multiplier[32] capable of 4 quadrant multiplication. The analogue weight memory is stored as a differential voltage at each synapse as shown in Figure 2.6 and the input neural state is represented by a differential voltage ΔV. The result of the multiplication $\Delta Iout$ is added to a summing node connected to other synapses in the synaptic column.

A weight change is achieved by an iterative process whereby the weight voltage for an individual synapse cell is read off chip via a multiplexor and compared to the desired value. A programming voltage to move the weight closer to the required value is then applied and the weight voltage is measured again. This process continues until the required weight voltage is achieved. The width of each programming pulse is in the range 10μs to 1ms and between 12V and 20V in height. As a result of this iterative process no definitive time can be given for weight update.

The 6 Bit weight resolution quoted in Table 2.5 [33] is a simplification of the capability of the storage medium. Weight accuracies of 7 Bits are quoted[34] when weight values

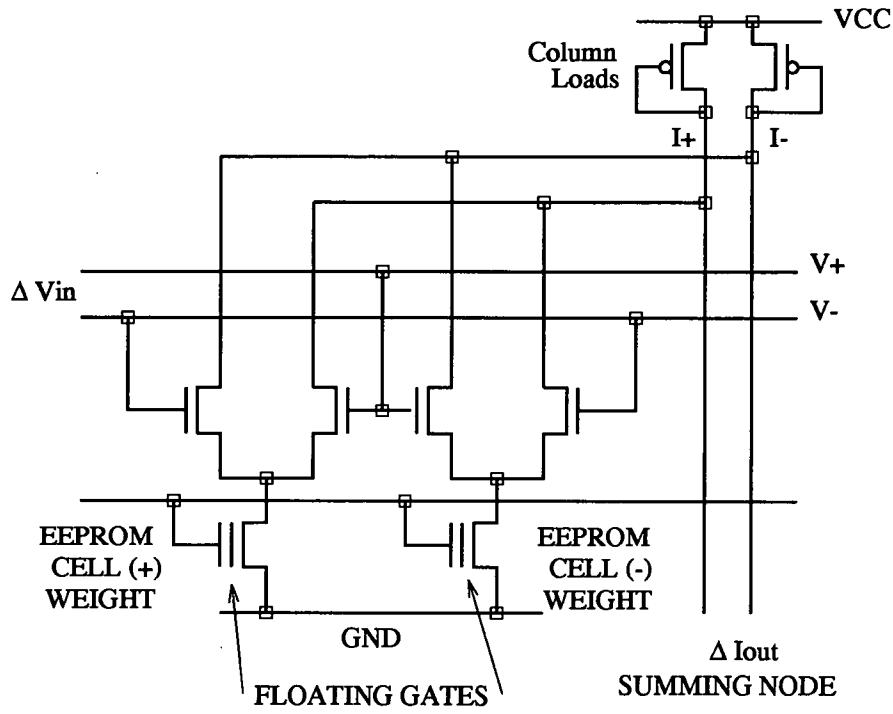


Figure 2.6 ETANN Synapse : Gilbert Multiplier and Non-Volatile Weight Memory

are measured directly after writing. For long term weight retention a figure closer to 4 Bit accuracy has been estimated[34].

The core of the sigmoid shaping circuitry is shown in Figure 2.7. The relationship between the differential sum current from the column of synapses and the output voltage of this circuit is sigmoidal. In addition, the gain of the circuit can be controlled from a linear function to a high gain threshold by controlling the voltages SMIN and SMAX[35]. The output from this circuit is level shifted and buffered before being passed off chip.

ETANN has been used as the basis for systems implementations. For example, the Mod 2 Neurocomputer[36] is a neural network processing system incorporating individual neural networks as subsystems in a layered hierarchical architecture. Mod 2 is designed to support parallel processing of image data at real-time rates.

2.3.2. A Fixed Function Analogue Implementation Case Study : Mead's Silicon Retina

The work of Carver Mead[37] and his associates is derived from the viewpoint that the nervous system of even very simple animals contains computing paradigms that are orders of magnitude more effective than those so far developed by engineers and

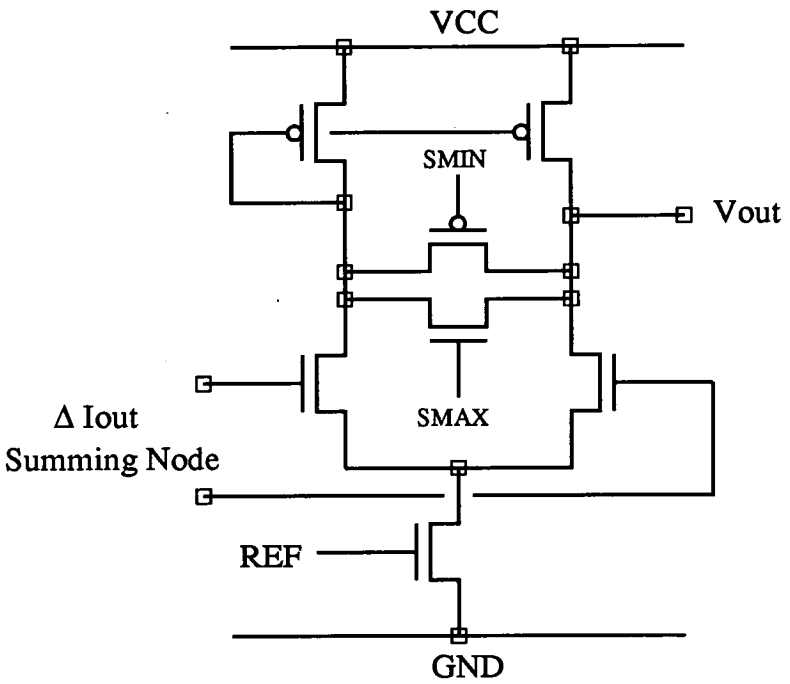


Figure 2.7 ETANN Neuron : Sigmoid Generation Circuitry

scientists. On this basis, Mead has set out to study and implement the early processing functions carried out in biological structures such as the retina[38] and cochlea[39]. Mead's retina chip [38] for example, generates outputs in real time that correspond directly to signals observed in the corresponding levels of biological retinas.

Most of the circuits used in these implementations use standard CMOS operating in the subthreshold region. The advantages of operating the transistors in this region are primarily very low power consumption, good current source properties over almost the entire operating voltage range and an inherent exponential nonlinearity which is an ideal computational primitive for many applications.

The biological retina is capable of operating over many orders of magnitude of illumination. The processing that is performed in nature to achieve this relies on lateral inhibition to adapt the system to a wide range of viewing conditions to produce an output that is independent of absolute illumination level. The lateral inhibition mechanism also performs spatial edge enhancement.

A simplified plan of the silicon retina is shown in Figure 2.8. Each node or pixel on the resistive grid contains a photoreceptor and a transconductance amplifier operating in a follower mode which drives the resistive grid. The output of the pixel is the difference between the output of the photoreceptor and the voltage stored on the capacitance of the

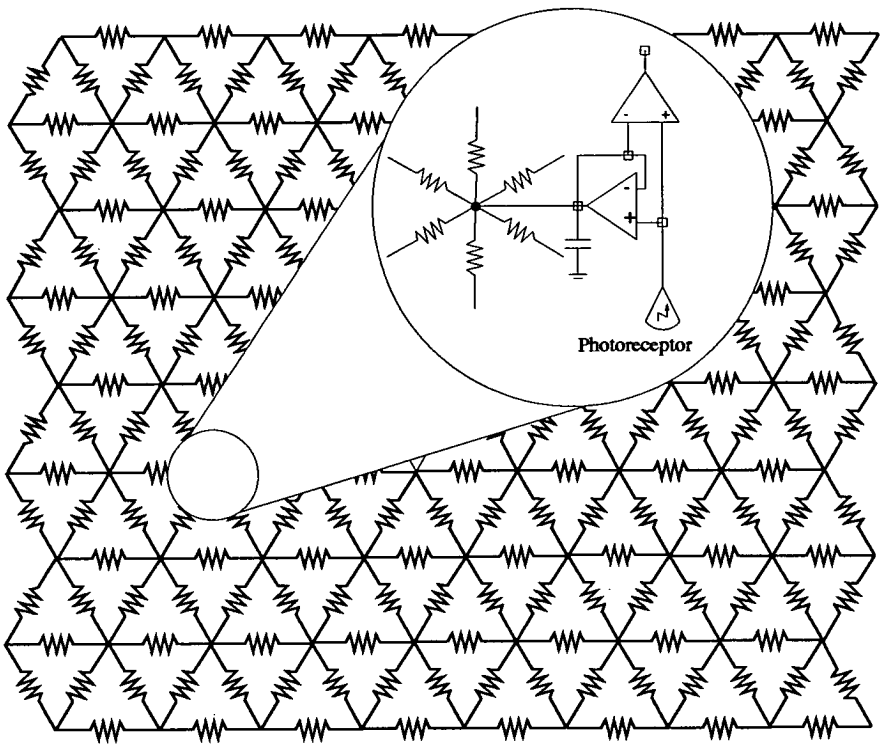


Figure 2.8 Silicon Retina : Resistive Network and Pixel Element.

resistive network. This output is at a maximum when the image is changing rapidly in either space or time.

The photoreceptor takes the logarithm of the incident-light intensity and transmits that value to the resistive network. The resistive network, which corresponds physically to a layer of cells within the retina, performs spatial and temporal averaging of the photoreceptor output.

Experiments on the silicon retina have yielded results remarkably similar to those obtained from biological systems[38]. In engineering terms, the primary function of the computation performed by the silicon retina is to provide automatic gain control that extends the useful operating range of the system. This makes the sensor sensitive to input changes independent of the viewing conditions. In addition, the gain control structure also performs other functions such as computing contrast ratio and edge enhancement. This preprocessing therefore has important consequences for the subsequent representation and processing of image data.

Given that this device implements a specific biological function and operates in a highly parallel real-time mode then discussion of system performance in terms of connections per second is not relevant here. Further extensions of this work are reported in the

literature, for example, contrast sensitive [40] and direction selective [41] retinas. This work is very distinctive and the medium of analogue VLSI proves extremely powerful for implementation of these structures.

2.4. Discussion

Systems using custom digital VLSI neural networks devices are capable of outperforming conventional computing platforms on benchmark neural network problems. These devices typically offer sufficient precision and flexibility to allow neural algorithm development and simulation. Their increased performance has great advantages for neural system development offering faster performance and having the capability to emulate large networks.

The mixed-signal ANNA chip has been used to implement an existing structure for a specific character recognition problem. The architecture and problem used has allowed the use of reduced precision arithmetic which lends itself to analogue implementation. The use of a digital interface has allowed interface to standard digital SRAM for weight load, for example. The use of current-mode techniques for weight load ensures the load time of individual weights is matched to SRAM access times. For these reasons, ANNA is an excellent example of a mixed-signal implementation integrated to system level. Computational bottlenecks have been avoided at system design time and impressive performance figures result.

The use of EEPROM technology in the Intel ETANN chip makes individual weight load time very slow. Each weight programming pulse is between 10μs and 1ms long, and several pulses may be required to get the desired weight. For this reason, chip in the loop learning would take an excessive amount of time. Therefore it is an advantage if learning is performed on a host computer and weights downloaded to the chip. However, once weights have been loaded, the performance of the network measured in connections per second is good.

The work of Mead is distinctive and impressive. By studying biological systems new methods for processing information are being developed. This work has generated much interest and further developments are eagerly awaited.

Chapter 3

A Review of Pulsed VLSI Implementations of Neural Networks

The term *pulsed neural networks* is used here to define a class of implementation strategies that use some form of pulse coding to represent neural state and/or weight data. A number of techniques have been used, for example stochastic bit streams[42], pulse-density modulation[43] detailed models of the precise behaviour of biological neurons[10] and dendritic structures[44], or techniques similar to those outlined in the remainder of this thesis using pulse frequency [45, 46] or pulse width modulation[47].

As in *conventional* implementations of neural networks, pulsed neural network fall into several implementation categories using a variety of analogue and digital VLSI techniques. For example, stochastic logic neural networks[42] employ digital techniques, while pulse-density modulation circuits[43] have been implemented using either digital or mixed signal techniques. Models of biological neurons[10, 44] have been implemented using predominantly analogue techniques.

The review of pulsed neural networks has been selective in order to highlight the principle strategies and achievements at present in this area.

3.1. Stochastic Logic Neural Networks

The term stochastic logic is used to describe a digital circuit that performs pseudo analogue operation using stochastically coded pulse sequences. The coding circuit that codes a digital or analogue value into a stochastic pulse sequence is shown in Figure 3.1.

A digital value, X, is compared with a random number, X_i, generated from a uniform probability distribution between the values of 0 and X_{max}. The comparator produces an output pulse whenever $X > X_i$. The resulting output sequence therefore represents the value X.

The multiplication and addition of stochastic pulse sequences can be performed using the circuits of Figure 3.2. The output firing probability of the AND gate is equal to the product of the input firing probabilities, P_A and P_B, while the output of the OR gate is approximately equal to the sum of P_A and P_B, except in the high firing rate region where *pulse collisions* are too frequent.

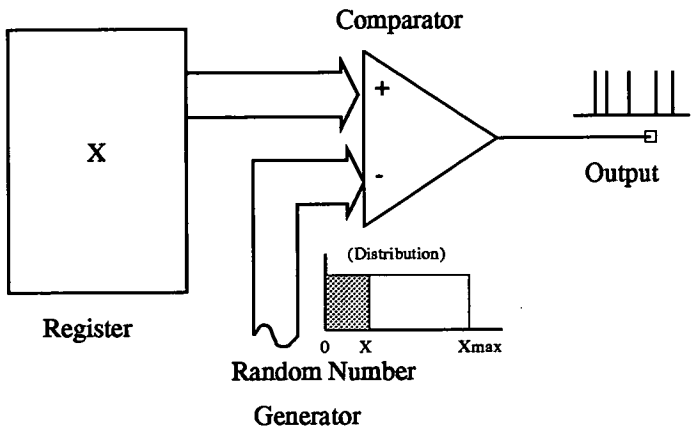


Figure 3.1 Stochastic Pulse Sequence Coding Circuit

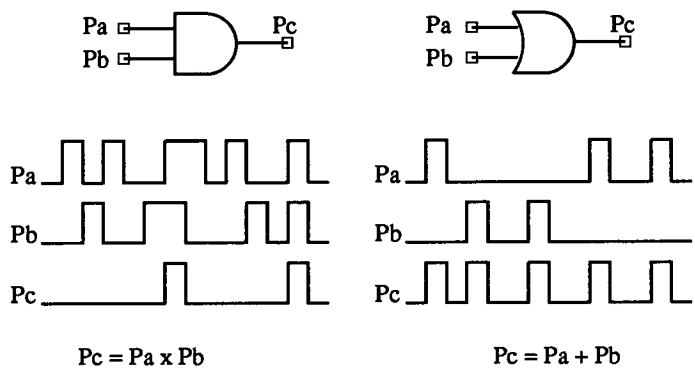


Figure 3.2 Stochastic Multiplier and Adder Circuits

These principles have been used to design stochastic neural networks[42]. A single stochastic neuron circuit is shown in Figure 3.3. Each weight is accessed from the weight memory in turn and loaded into Counter 1. Counter 1 allows the implementation of a local learning rule, the weight value can be modified depending on the current value of the neuron input and output and the gain term multiplied together by the gate AND 1. The resultant stochastic pulse stream representing the weight is multiplied by the neuron input signal by gate AND 2. This output either increments or decrements Counter 2 representing the aggregated activity value, depending upon the sign of the weight. The latch allows update of the neuron output values at a specified time. Comparator 2 is fed by a second random number generator whose characteristic defines the saturation

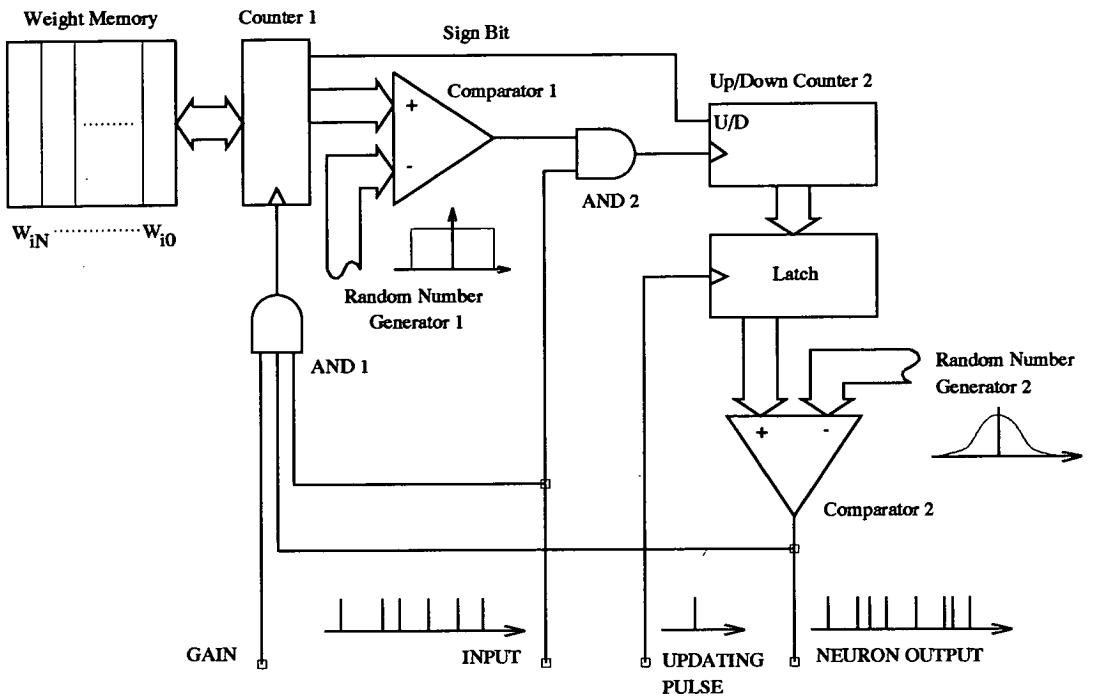


Figure 3.3 Stochastic Neuron Circuit

characteristic of the neuron. An OR gate could replace Counter 2 and Comparator 2 since an OR gate adder has a naturally saturating sigmoidal characteristic, however the circuit illustrated provides improved learning characteristics[42].

Neural networks implemented with stochastic logic suffer coding noise due to the probabilistic coding of signals. The effects of this noise have been discussed in relation to associative memory models and optimisation problems[42] and methods of limiting these effects have been found.

As seen from the diagram of Figure 3.3, stochastic pulse coding is well suited to a digital implementation reducing adders and multipliers to relatively simple functions. In addition synaptic plasticity has been implemented in an efficient manner.

3.2. Pulse Density Modulation Neural Networks

Pulse density modulation represents numbers as streams of digital bits in a statistical manner that is distinct from stochastic and conventional bit-serial representations. The value of the pulse density number is defined by the relation between the number of 0's (+) and 1's (-) in a given time window[43]. Furthermore, all the values in the pulse density modulation representation are fractional values between -1 and +1. Therefore the value of a bit stream including N zeros and M ones is $(N-M)/(N+M)$ and, a number

with P bits can represent P + 1 values. The value of the pulse-density stream is continuous so that a sample of P bits taken at any time will represent the value. All values are statistical in nature rather than exact values, however this data representation has proven suitable for neural networks[43]. Figure 3.4 shows the number 0.5 represented in pulse-density format.

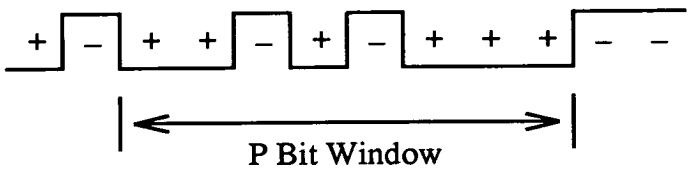


Figure 3.4 Pulse-density Number Format. N = 6, M = 2, Number = 0.5.

Pulse density networks have been implemented in both analogue and digital realisations[43].

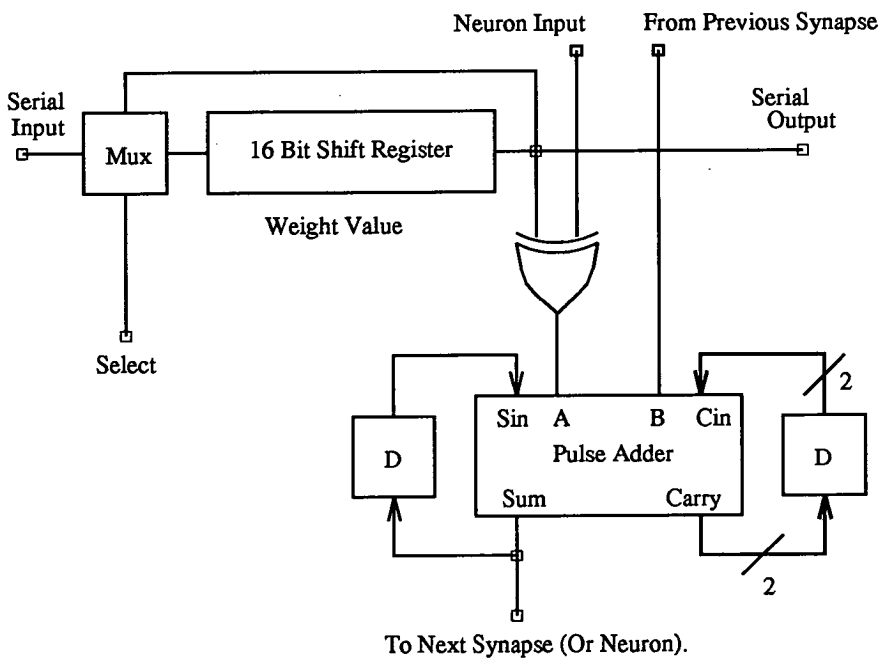


Figure 3.5 Pulse-density Digital Synapse Circuit

3.2.1. Digital Implementation

The digital realisation implements multiplication of neural state by synaptic weight using an Exclusive-OR gate as shown in Figure 3.5.

Synaptic weights are fed onto the chip in a serial manner and stored in a recycling shift register. Weights are stored in pulse-density format. The result of the multiplication performed by the Exclusive-OR gate is added to the result from the previous synapse in a column and fed onto the next synapse or the neuron.

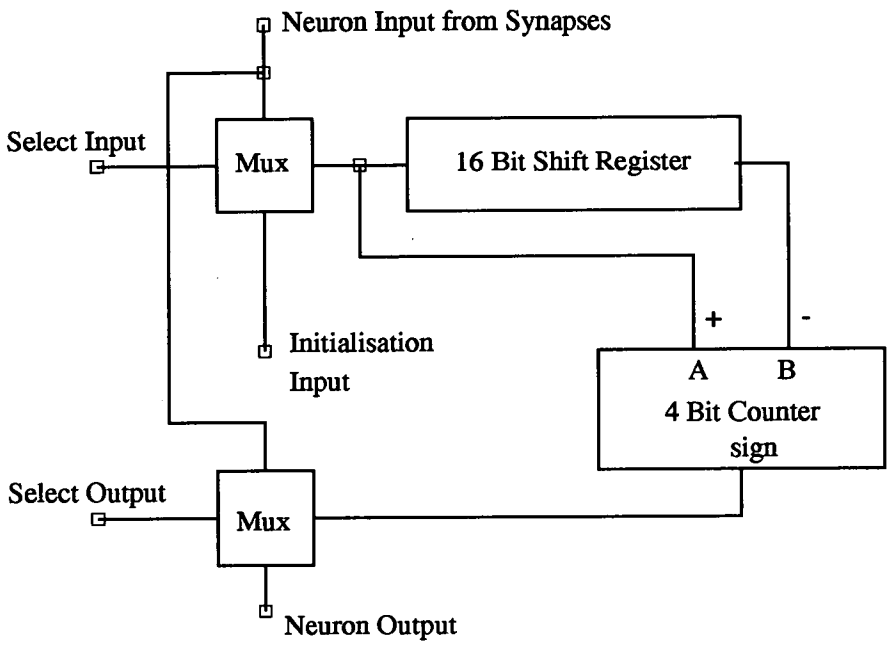


Figure 3.6 Pulse-density Digital Neuron Circuit

The neuron circuit implements a step function non-linearity by simply counting the number of -1's (-) and +1's (+) in a given time window and setting the output to zero if the result is positive and to 1 if it is negative. In practice this is achieved using the neuron circuit of Figure 3.6 which feeds the result into a 4 bit counter and tests via input A for +1's and then via input B for -1's. In addition, the computation performed by the synaptic array may be fed off-chip directly and cascaded with other such devices to form a larger network.

3.2.2. Mixed Signal Implementation

The mixed signal implementation again uses similar digital structures for weight storage and an Exclusive-OR gate to perform multiplication of two pulse density modulated waveforms. In place of the digital pulse adder used in Figure 3.5 a switched capacitor

structure has been employed to dump packets of charge on to a summing bus as the result of the multiplication is fed out of the Exclusive-OR gate.

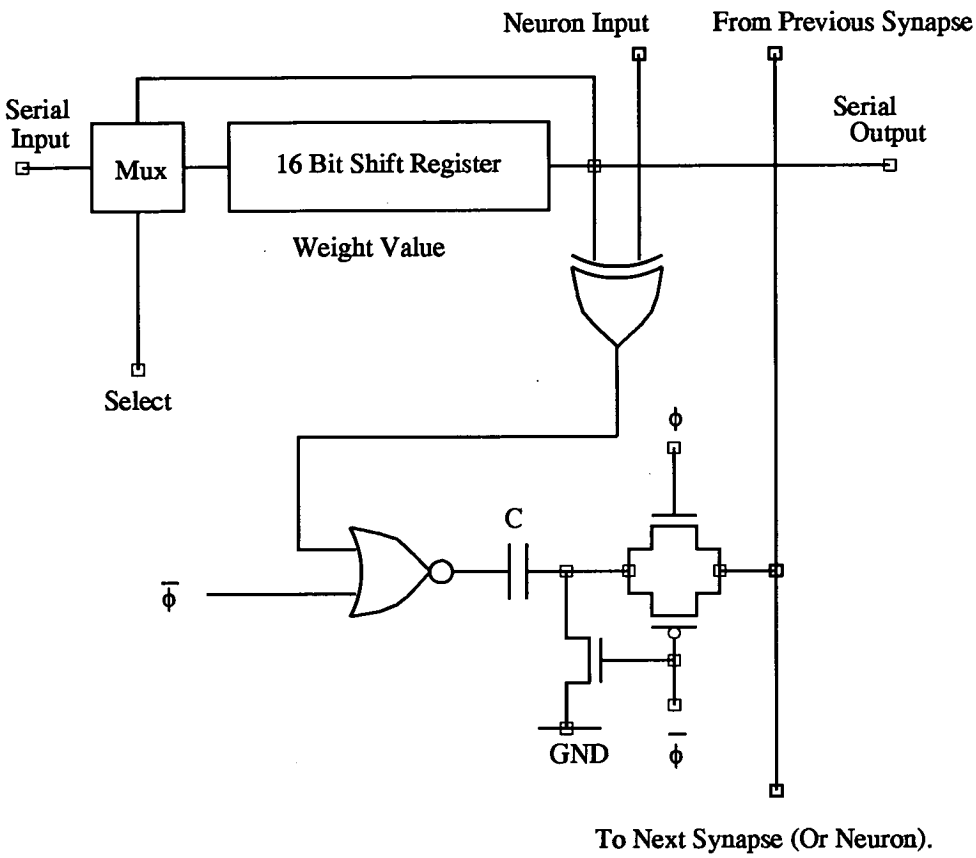


Figure 3.7 Mixed-Signal Pulse-density Synapse Circuit

A two-phase clocking scheme has been employed which connects both ends of the capacitor C in Figure 3.7 to ground during one phase of the clock. During the other phase, the output of the NOR gate connects VDD or ground onto the left hand end of the capacitor for Exclusive-OR outputs of 0 and 1 respectively while connecting the right hand end to the summing bus.

A simplified schematic of this arrangement showing the capacitors from a number of synapses connected to the summing bus is shown in Figure 3.8. The neuron body, has a capacitive input stage so that charge on the synapse capacitors is shared with this capacitor such that $V_{C1} = (3 \times VDD \times C + 2 \times 0V \times C) / (6 \times C) = VDD/2$ assuming C = C1 in this example. The result of this computation for each bit is integrated within the neuron body over the complete bit window and a non-linear output stage performs the sigmoid transfer characteristic and pulse output.

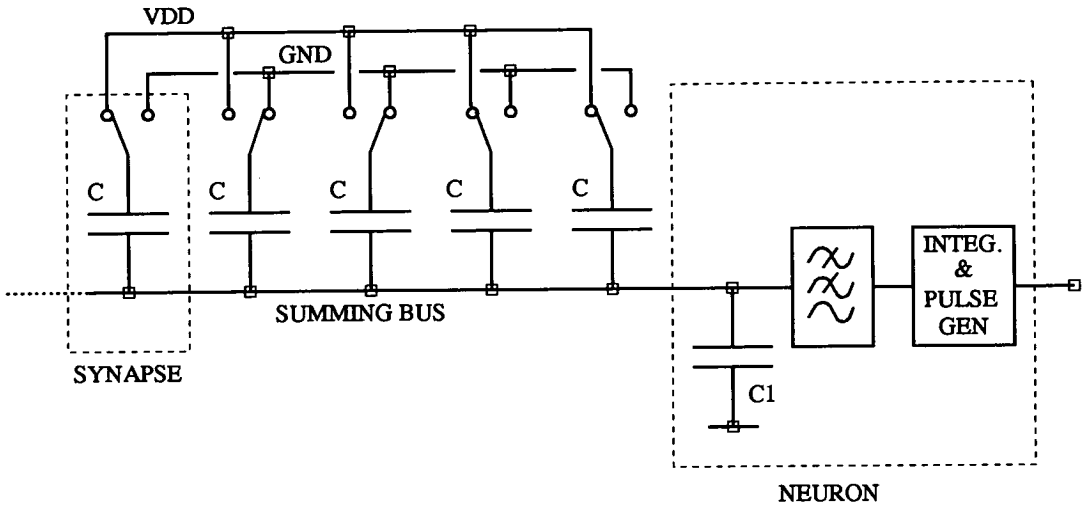


Figure 3.8 Simplified Mixed-Signal Pulse-density System

It is estimated[43] that 100 fully connected mixed signal neurons or 90 digital neurons can be integrated onto a $1cm^2$ die using $1\mu m$ technology.

3.3. Programmable Impulse Neural Circuits

The emphasis of the research work performed by Meador and his group[10] is placed upon how the known characteristics of natural neuron function can be applied to overcome the requirement for large circuit area and high power dissipation which limit the integration capability of analogue neural VLSI. It is hoped that this approach will increase functionality by way of programmability and ultimately allow the implementation of on-chip adaption[48].

The neuron circuit, shown in Figure 3.9, models the short term memory (STM) neuron dynamics of biological neurons. The mathematical formalism used to describe the neuron dynamics can be shown to be equivalent to the STM equations of Hopfield, shunting, and adaptive resonance networks[10].

The synapse circuits in this example are fixed connections which can be either excitatory or inhibitory but not both simultaneously. The synapses are implemented using two transistors, one acting simply as a switch, the other as a current limiter controlled by the voltage references V_{rp} and V_{rn}. By controlling the operation region of the transistors via these voltage references and the switching points of the comparator circuit, the network can be operated in different summing and shunting modes.

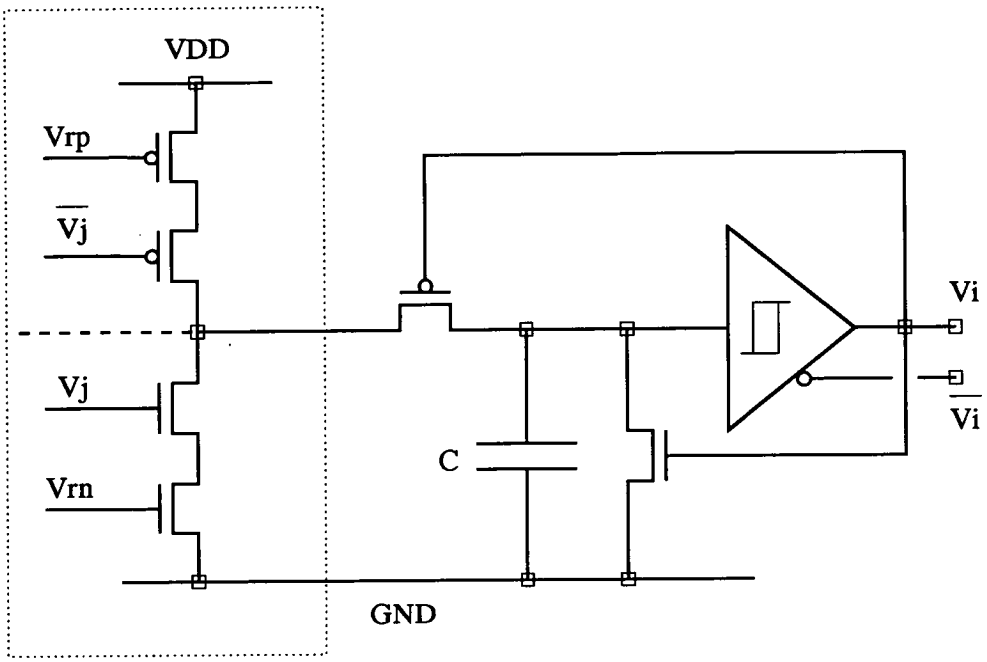


Figure 3.9 Impulse Neuron Circuit with two fixed synapses

The neuron circuit performs a temporal integration of the excitatory and inhibitory activity arriving at the neuron input from the synapse array. Once the integrated activity reaches the upper switching threshold of the comparator circuit, the neuron outputs a pulse. The duration of the pulse is determined by the time the neuron takes to *self-deplete*. Self-depletion describes the mechanism by which the neuron output pulse discharges the integration capacitor to the lower threshold of the comparator circuit. This is a function of the comparator, integration capacitor and discharge transistor.

The issue of long term memory (LTM) capable of implementing an adaption algorithm has led to the development of a programmable synapse using EEPROM floating-gate technology for non-volatile analogue weight storage. The circuit developed, shown in Figure 3.10, uses 5 transistors, two of which share a common floating gate. The floating gate structure is programmed by applying positive or negative pulses to V_{pp} resulting in an excitatory or inhibitory value of I_d. With V_m adjusted so that the floating gate structures operate in the sub-threshold region, the value of I_d can be varied over several orders of magnitude.

Adaptive circuits using competitive pulse Hebbian learning based on the circuits outlined above have been suggested[48] and simulation of small networks using capacitive weight storage shows encouraging results.

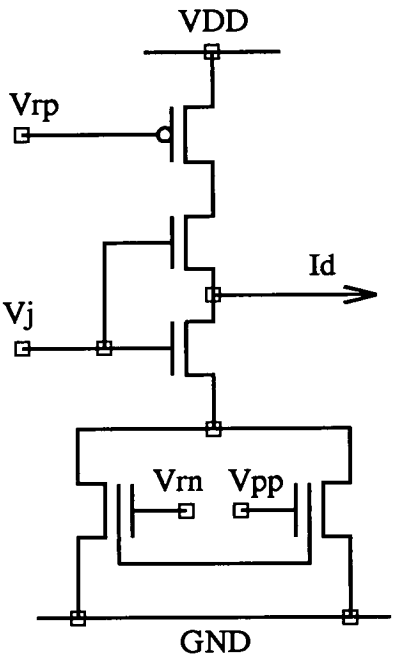


Figure 3.10 Programmable Synapse Circuit using Floating-Gate Weight Storage

3.3.1. Neuromorphic Circuits for Dynamic Signal Processing

In many problems such as the control of physical systems dynamic signal processing is required. However, most conventional models of neural networks do not possess dynamic signal processing capabilities. By investigating the control structures used in the vertebrate brain for coordinating motor activity, Elias[44] has produced circuit architectures whose structure and connectionism produce the ability to process dynamic signals.

Figure 3.11 illustrates the structure used to model the processing performed by a spatially extensive dendritic tree which collects and processes the majority of afferent signals, propagating the result to the neuron body. The computation performed by this structure can be understood with reference to the equivalent circuit of part of the dendritic tree illustrated.

Distributed along the dendritic branches, at the cross points, are synapses which either inject or withdraw a fixed current from the distributed RC network. The synaptic connections therefore do not have programmable strength. A digital impulse signal arriving at the NMOS transistor removes charge at a node while an impulse signal arriving at the PMOS transistor adds charge to a node. The RC network introduces a time constant and signal attenuation across the network so that an excitatory input in one part of the network would have a different dynamic effect from one introduced at another part of the network. The neuron may be a simple thresholding device. The operation of this

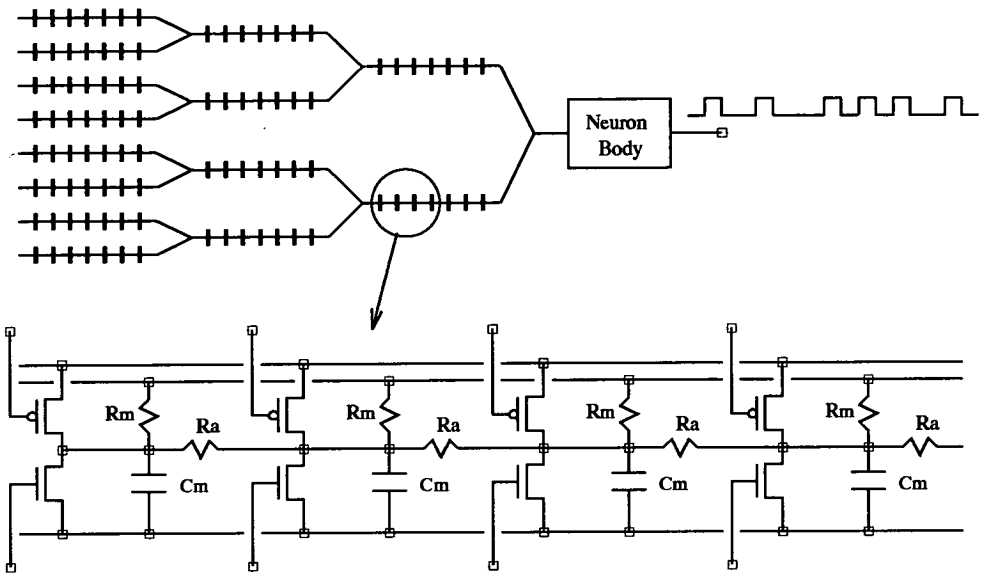


Figure 3.11 Dendritic Tree Model and Equivalent Circuit.

network can be illustrated by the example of Figure 3.12 where an input sensor is connected to the network in order to categorise symmetrical and asymmetrical patterns.

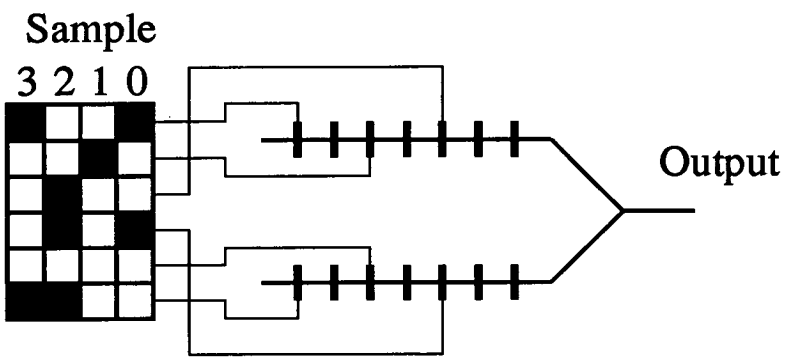


Figure 3.12 Input Sensor Patterns 0 - 3 Connected to Dendritic Tree

Since the synapses are not programmable, it is the manner in which input data is connected to the network that determines overall performance. Genetic algorithms have been used to train networks of artificial dendritic trees[49] by determining the best connection between sensor and network to achieve the desired processing. The sensor data illustrated in samples 0 - 3 is connected to the dendritic tree so that an asymmetrical input will

produce different signals in each of the dendritic branches and therefore result in a positive or negative voltage trajectory at the output. A symmetrical input, in this example sample 3, will produce signals of equal and opposite magnitudes on the two dendritic branches and therefore cancel at the output and produce no voltage output signal.

3.4. Pulsed Implementations : Closer To Home.

This section of the review highlights work by others whose methodology is most closely matched to that of Edinburgh group.

3.4.1. A Global Clocking System for Pulse Stream Networks.

Neural networks have been developed using a global clocking scheme to perform multiplication of a pulse stream input by a signed digital weight[50]. The operation of the synapse multiplier can be understood with reference to Figure 3.13.

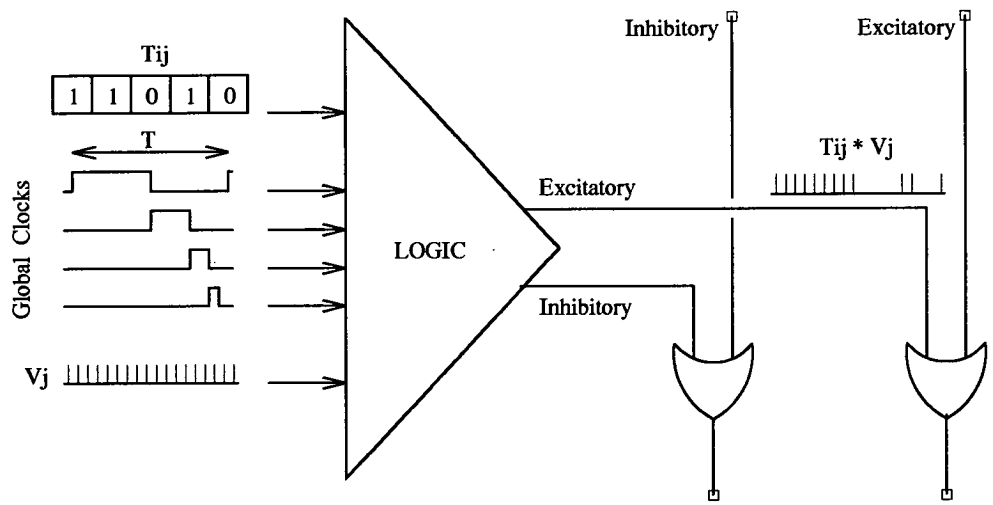


Figure 3.13 Pulse Stream Synapse : Global Clocks and Digital Weight Storage

An asynchronous pulse stream signal arriving at a synapse input is gated onto an inhibitory or excitatory output line depending on the sign of the weight and for a duration that is a function of the weight magnitude. The concept of a time frame, T, is used in order to gate the input signal for a fraction of the period T, depending on the weight magnitude, onto one of the outputs. OR gates are used to sum the pulse stream output from the synapse onto to an excitatory or inhibitory summing line. In order to avoid a large loss of information due to pulse collisions at the OR gate the pulse stream signals should be relatively sparse.

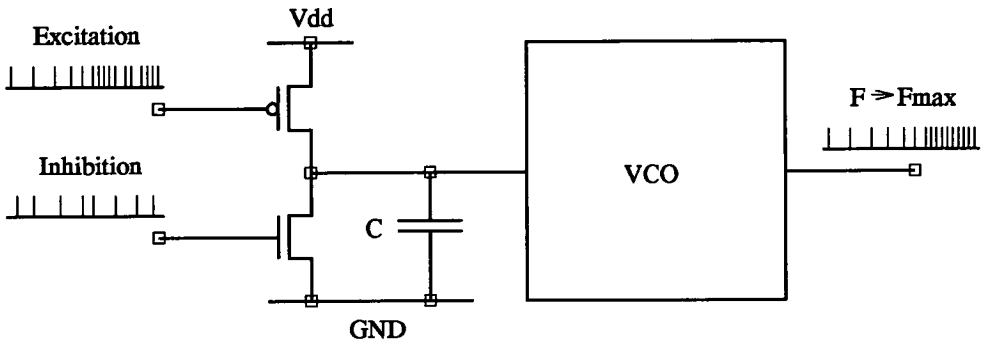


Figure 3.14 Pulse Stream Neuron : Integrator and Voltage Controlled Oscillator (VCO).

The inhibitory and excitatory summing buses enter the neuron, shown in Figure 3.14, and either dump or remove small amounts of charge from an integrating capacitor, C, at the arrival of an inhibitory or excitatory pulse respectively. The integrated voltage is used to drive a voltage controlled oscillator which produces constant width output voltage pulses at a frequency which is a function of the voltage on C.

Devices implementing the synapse structure have been designed and tested [6] with the integrator and oscillator implemented off-chip. Small problems using 6 or fewer neurons to perform simple content addressable memories and lateral inhibition networks have been implemented[6].

3.4.2. Pulsed Multi-Layer Networks with Guaranteed Compute Time.

In order to design pulsed analogue neural network hardware for multi-layer perceptron (MLP) neural networks with a guaranteed compute time the concept of pulse width modulation and a master clocking system has been investigated[51]. All neural input states to the network are synchronised to the leading edge of the master clock and are pulse width modulated over the duration of half the master clock period.

The equivalent circuit of Figure 3.15 shows how an output current I_{out} is generated from the difference between a constant current source and a linearly controllable current source. The switch on the output modulates the output current, the resultant charge packet added or removed from the output summing node representing the multiplication of $T_{ij} \times V_j$. In practice, the constant current source has been implemented by the transistor M1, while M2, operating in the linear region, implements a linearly controllable current source. In order to balance this circuit when no pulses are present, the transistor M4 has been added to remove the difference current.

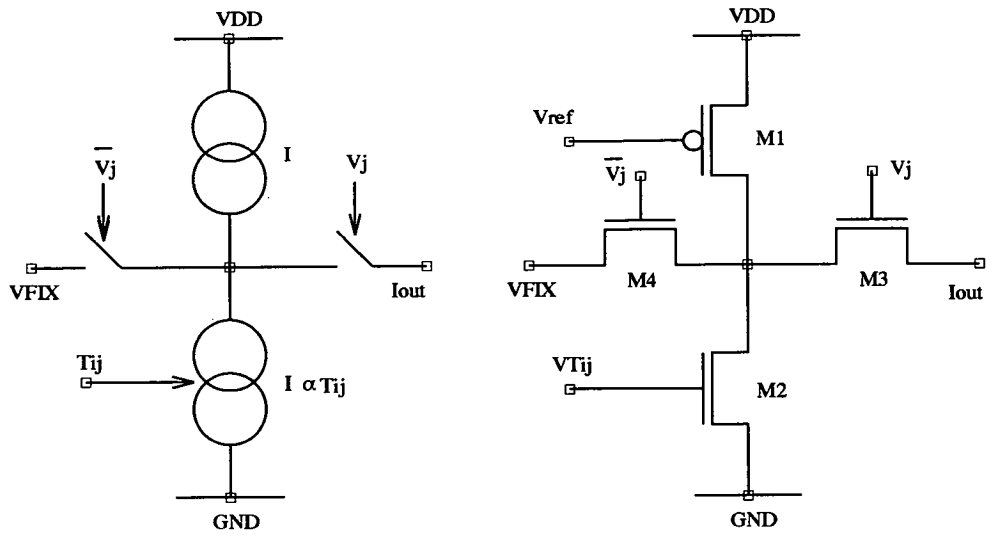


Figure 3.15 Pulse Width Synapse : Equivalent and actual circuits

The resultant currents from the synapse circuits are summed by an integrator structure and passed to a differential stage that has naturally saturating characteristics. Pulse width modulated outputs are generated from the output of the differential stage by performing a linear voltage to pulse width transformation.

This pulsed methodology has been implemented on a VLSI test chip and the circuits have been shown to function correctly[47]. However, a small design error has prevented these chips from being used to implement MLP networks. This methodology is capable of operating at clock rates of 1MHz and for a chip with 10,000 synapses over 10^{10} operations per second are possible.

3.4.3. Switched Capacitor Pulse Stream Neural Networks

The techniques of pulse stream neural state signalling and switched capacitor techniques combine in the work of Brownlow[45] to produce an elegant system for neural computation.

A simplified schematic of the system is shown in Figure 3.16. The weight voltage, stored dynamically on a capacitor, is buffered by a voltage follower and feeds into a switched capacitor arrangement. When a pulse arrives at the synapse the buffered weight voltage sets the voltage on capacitor, C. This charge is transferred to the integrator structure during the low period following the input pulse. Charge is therefore transferred to the integrator circuit in proportion to the multiplication of the weight voltage and the input pulse frequency. By varying the weight voltage around V_{ref} both excitation and inhibition can

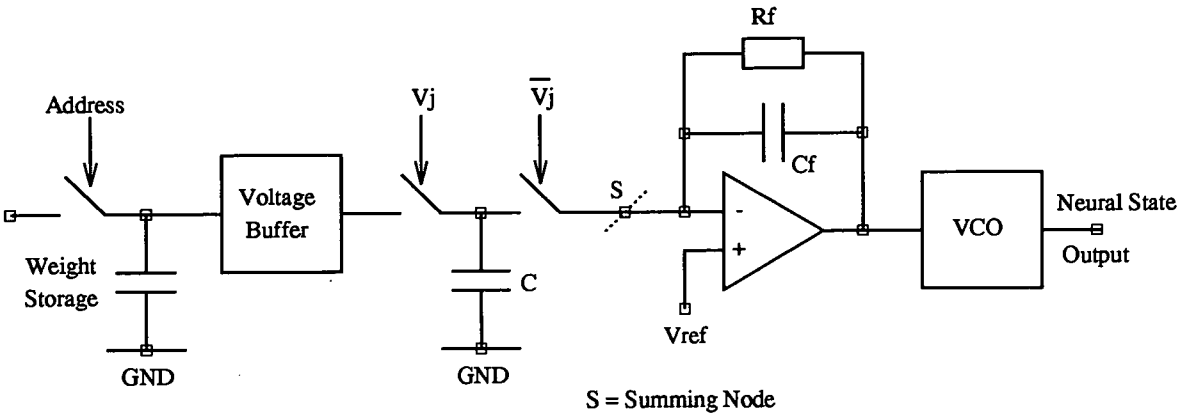


Figure 3.16 Asynchronous Switched Capacitor Synapse and Neuron

be achieved. In practice a switched capacitor arrangement is used to implement the leaky integrator structure.

A test chip integrating an array of synapses, but not the Voltage Controlled Oscillator (VCO) circuit of Figure 3.16, has been fabricated and tested. The synapse area is small, $65\mu m \times 65\mu m$ when implemented in $2\mu m$ Digital CMOS technology, and 144 synapses and 12 neurons have been implemented on a 3mm x 3mm die.

This device has been tested on a robot localisation problem[52] where real time pattern association is required. The task of the navigational sub-system is to determine the position of the robot within a room using time-of-flight infra red scanner measurements taken from a scanner mounted on the robot. The scanner performs a 360° rotation taking time-of-flight measurements every 30°. These readings have been recorded for 24 locations within the room shown in Figure 3.17 and been used to train two chips so that neuron 0 responds to location 0, neuron 1 to location 1 and so on.

During the navigation of the robot, the most recent scanner data is fed as the neural state inputs to the array. The approximate position of the robot is then given by the maximally responding neuron. The response of the VLSI devices for the robot in the position shown in Figure 3.17 is shown in Figure 3.18. Also illustrated in Figure 3.18 is the response from a model of the matching procedure performed algorithmically on a SUN workstation with weight accuracy of 8 bits. These results show that the scalar product calculated by the VLSI devices are within 1.2% of those computed by the SUN workstation.

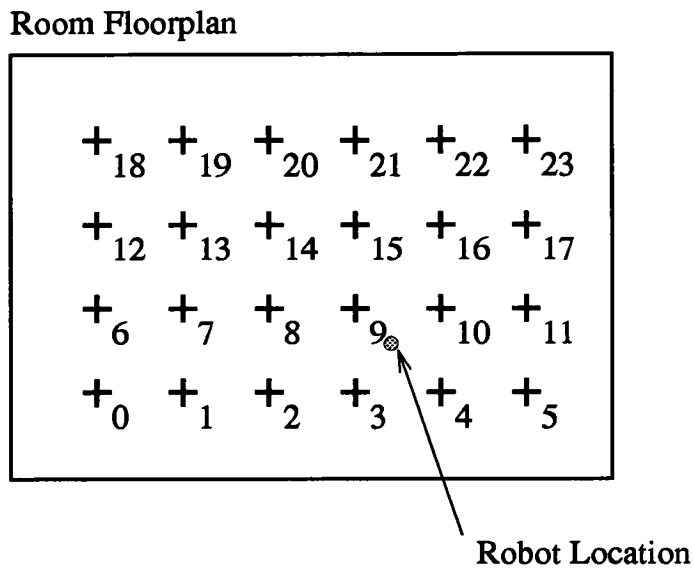


Figure 3.17 Simulated Test Environment

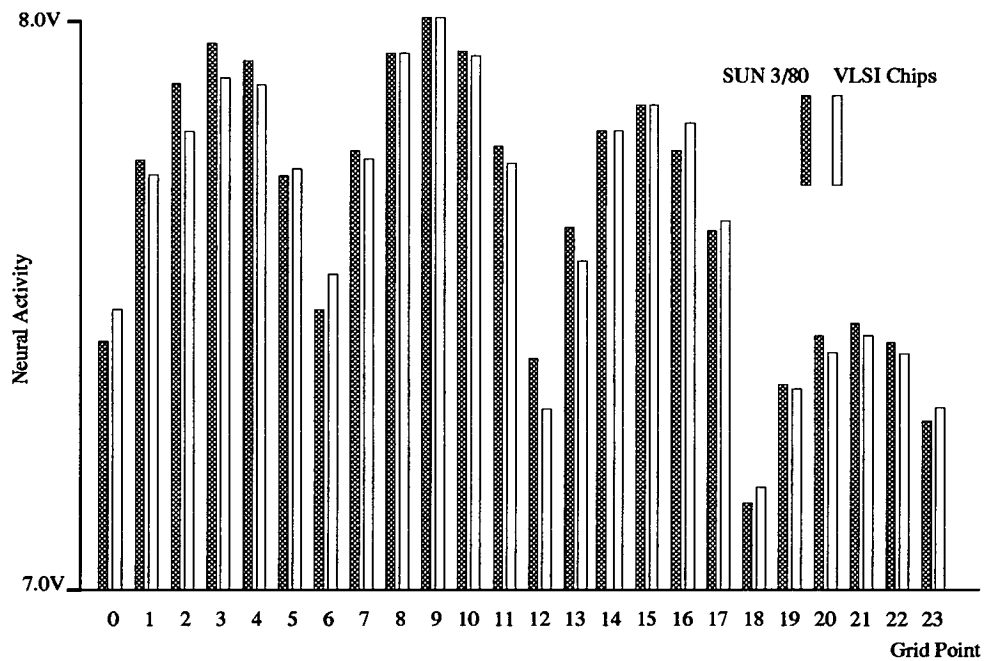


Figure 3.18 Mobile Robot Localisation : VLSI Results compared to Workstation Results

These devices have therefore proved successful in implementing neural applications. The relatively good accuracy demonstrated in the application problem is due to the good matching of capacitors across chip. The main limitation with this system, however, is its inability to scale to networks with large fan in due to the requirement for ever larger integration capacitors.

3.4.4. Pulsed Hopfield and BAM Networks using Oscillatory Neurons.

The work of Linares-Barranco[46] uses a voltage controlled oscillator (VCO) to implement the neuron function and a conventional transconductance multiplier circuit[46, 53] similar to that used in Intel's ETANN chip[34] and reviewed in the previous chapter, to implement the synapse.

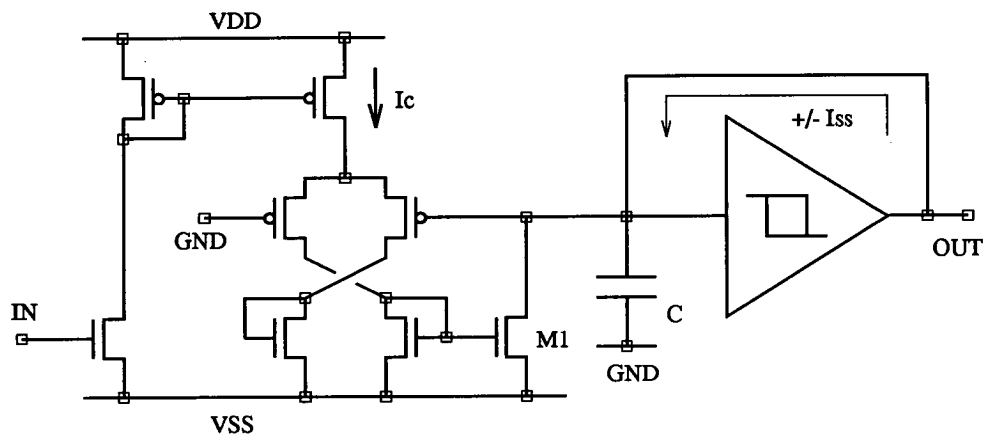


Figure 3.19 Neuron Circuit : Non-Linear Resistor Circuit Details and Comparator

The differential stage in the neuron circuit of Figure 3.19 acts like a non-linear resistor that sinks current through M1 only when the voltage on the capacitor exceeds 0V. The circuit oscillates when the input voltage, IN, is adjusted so that $I_c < I_{ss}$, otherwise there is no oscillation. The neuron circuits oscillate at a fixed frequency determined by I_{ss} and reference voltages within the comparator circuit that set the switching points. The oscillator is either on or off.

The transconductance multiplier circuits used in the synapse have analogue weight storage and dump or remove charge from a summing bus when a pulse arrives at the synapse input. The resultant charge is integrated and used to drive the neuron circuit.

These devices have been used to implement a 5 neuron Hopfield network[46] used to store the pattern **10101**. The network was run for various input patterns and was found to converge to the stored pattern, or the inverse **01010** , in every case. A 6 neuron Binary

Associative Memory (BAM) has also been implemented with 3 neurons per layer and convergence to the stored pattern was also observed.

3.5. Discussion

The pulsed neural networks presented in this review highlight the range and diversity of implementation possibilities in this category of neural network implementation. While stochastic and pulse density neural networks appear closely related it is likely that the temporal nature of the data representation will lead to slower synaptic multiply and add times when compared to more conventional digital techniques outlined in the previous chapter. However, their simple structure lends itself to high VLSI integration densities and their application in solving *real world* problems is eagerly awaited.

Similarly, the research work being carried out in the implementation of structures that more closely model their biological exemplars is of great interest. The work of Elias is very distinctive, providing new techniques for performing dynamic signal processing.

The techniques used to implement pulsed networks that most closely resemble those outlined in the rest of this chapter have been presented. The early work of Smith, for example, proved the pulse stream technique is viable, but the use of a global clocking scheme is clumsy.

The more recent work of Brownlow has proved highly successful and the robot localisation application is one of the most significant applications attempted using pulse stream neural networks. However, the problem of scalability due to the limitation on the size of the integration capacitor and operational amplifier drive capability remains a problem.

The synchronous network of Tombs gives impressive performance for feed forward networks, however, due to a small design error it has not been possible to use this chip for the intended network implementations.

At present pulsed neural networks have been used to implement only very simple problems with typically (apart from the work of Brownlow) less than 10 neurons. These have served to highlight the possibility of such networks in solving larger problems, but their credibility will naturally be enhanced when they are demonstrated solving *real world* problems.

The objective of the remainder of this thesis, therefore, is to present a library of synapse, neuron and ancillary circuits or cells for pulsed neural networks, whose performance and characteristics have been fully tested in functional VLSI. From this library, recommendations will be made for large pulsed neural systems and a large pulsed neural network chip presented solving a *real world* problem using *real world* data.

Chapter 4

The Edinburgh Pulse Stream Cell Library

This Chapter presents the *Edinburgh Pulse Stream Cell Library*, a collection of circuits designed to implement *Pulsed* neural networks in VLSI. The pulse stream signalling techniques used in these circuits are the pulse width and pulse frequency techniques outlined in Chapter 1. The Library contains a number of synapse, neuron and associated circuits and a strategy for multiplexing neural state data on and off chip. While the requirement for synapse and neuron circuits is obvious, it has been shown that multiplexing interconnections is necessary for the implementation of very large neural networks that exhibit poor locality[54].

The concept of inter-chip communication of neural state information has been considered by other workers. Frequency division multiplexing for communication between layers in a multi-layer perceptron implemented in analogue VLSI has been suggested[55], and the use of asynchronous pulse coded data to communicate over a parallel bus has been assessed in relation to accuracy and dynamic range[56]. While synchronous multiplexing of pulsed data has been suggested by some researchers[16, 47], the approach adopted here is the asynchronous serial transmission of pulsed information.

Since *The Edinburgh Pulse Stream Cell Library* contains work carried out by the author and a number of research colleagues, the individual contributions from these colleagues are referenced within this Chapter. Before considering individual Library Cells, the implementation of analogue capacitors on a digital technology requires consideration.

4.1. Implementation of Capacitors Using Digital CMOS.

The circuits described in this chapter have all been designed and implemented using European Silicon Structures' (ES2) 2μm and 1.5μm Digital CMOS Technology. The use of digital memory as a means for storing data has been precluded due to the unacceptably large area that it requires to implement. Although other workers have implemented non-volatile analogue memory using EEPROM technology[57-59], or used additional on chip refresh circuitry local to each synapse for medium term storage[60], the technique adopted here has been to store analogue weight voltages directly as a voltage on a capacitor and refresh them from off-chip memory via a digital to analogue convertor. In many of the circuits described in this chapter, therefore, voltages representing synaptic weights or the result of some computation are stored on capacitors. Since capacitors are not

supplied as standard components in a single polysilicon, double metal digital process, the capacitive properties of transistors have been used to implement these parts.

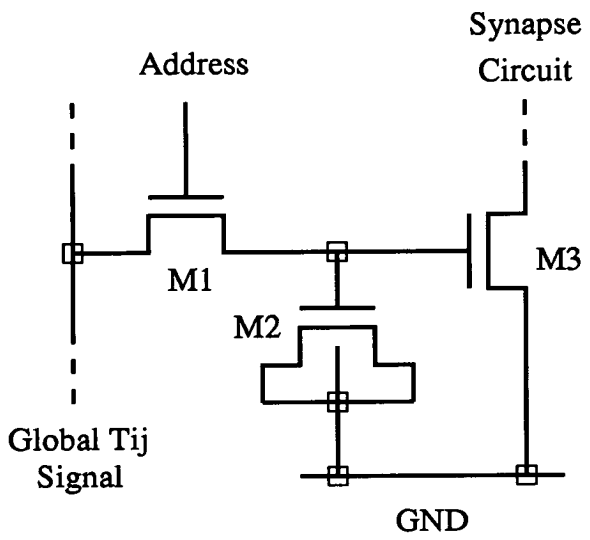


Figure 4.1 Synaptic Weight Storage Circuit

A typical weight storage node is illustrated in Figure 4.1 This weight storage node comprises the gate of an N-type transistor whose drain and source are connected to 0V. This configuration forms a capacitor which, provided that the gate is biased above the transistor threshold voltage (typically 0.9V for the 2μm process), has a capacitance very nearly equal to the oxide capacitance. The storage node can be addressed by switching transistor M1 on and the voltage on the Global T_{ij} Signal will charge or discharge the node to the appropriate weight voltage. In practice, the NMOS transistor will only pass relatively *low* voltages with a 5V Address signal. Should a larger range of weight voltages be required, a full transmission gate could be used to replace M1.

The size of the transistor M2 is chosen bearing in mind the leakage currents from the storage node, the oxide capacitance of the technology used, and the rate at which each node can be refreshed. The main sources of leakage are the current through transistor M1 to the Global T_{ij} Signal when M1 is switched off, and the leakage current through the reverse biased pn junction at the source of M1 where it connects to the gate of M2. The tendency for capacitive coupling of transient signals from the Address signal switching or from switching within the synapse circuit to corrupt the stored voltage has also to be considered.

From HSPICE simulation of the addressing transistor, the leakage current through the reverse biased pn junction is at worst 10fA. Since the ES2 transistor models do not

model subthreshold operation, then an intelligent guess at the current through transistor M1 is the best that can be achieved. A very conservative figure of 1pA has been assumed. If a refresh time of 2μS is assumed and 1800 synapses are to be refreshed (this number relates to the number of synapses refreshed in the final EPSILON chip), then the period between refresh is

$$2\mu S \times 1800 = 3.6mS \tag{4.1}$$

If an excitatory (or inhibitory) weight range of 1V is assumed and it is required to hold this to within 1% of its value, then since

$$I = C.\frac{\delta V}{\delta T} \tag{4.2}$$

$$C = 1 \times 10^{-12} \times \frac{3.6 \times 10^{-3}}{1 \times 10^{-2}} = 0.36pF \tag{4.3}$$

In practice, a capacitance of 0.75pF might be implemented at the storage node.

The capacitor symbols used in the circuit diagrams of this chapter have in practice been implemented with structures and analysis of this type.

4.2. Edinburgh's Pulsed Synapse Circuit Designs

This section of the thesis details the design of three pulsed synapse circuits developed within the VLSI Neural Network group at Edinburgh University.

4.2.1. A Pulse Magnitude Modulating Synapse Design

The Pulse Magnitude Modulating Synapse developed by Churcher [61, 62] operates effectively as a switched current source or sink, the *magnitude* of the output current being modulated by the synaptic weight. The basic circuit, illustrated in the first diagram of Figure 4.2, is capable of operating in pulse width or pulse frequency input modes.

A synaptic weight, stored dynamically on a capacitor, controls a current source. When a pulse arrives at the input, V_j, current is either added to or removed from the capacitor, C, depending on the magnitude of the current source in relation to the fixed balance current. Excitation is achieved when a net flow of current onto the capacitor occurs, resulting in a rise in activity voltage; inhibition is produced by a net removal of charge from the capacitor and a resultant reduction in activity voltage.

This circuit has been developed to operate in a self-depleting manner illustrated in the second diagram of Figure 4.2. Once the charge integrated on the dendritic summing node reaches the firing threshold of the receiving neuron, the neuron fires outputting a pulse on V_i and discharging C via the current I_r.

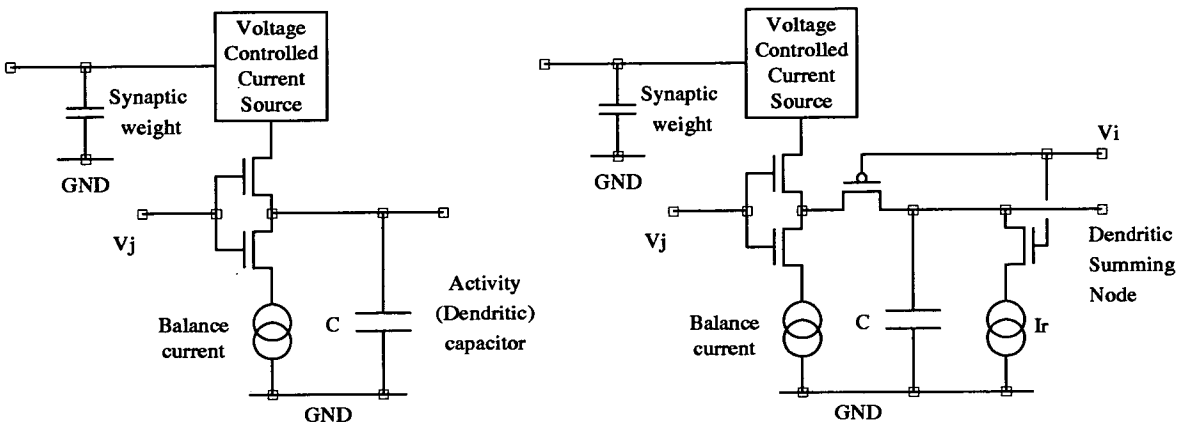


Figure 4.2 Pulse Magnitude Modulating Synapse : Principles of Operation
Basic Synapse and Synapse with self-depletion

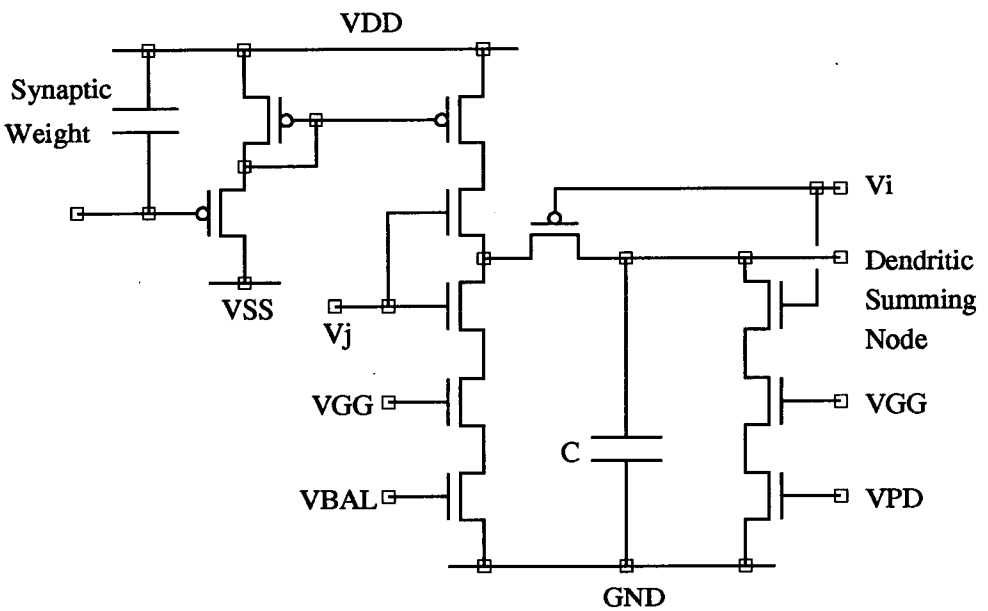


Figure 4.3 Pulse Magnitude Modulating Synapse : Circuit Details

The complete circuit diagram of the synapse is shown in Figure 4.3. The voltages VBAL and VPD set the balance current and I_r, respectively. A cascode current stage has been used and is controlled by the bias voltage, VGG, to prevent coupling of switching transients between adjacent synapse circuits. Simulation results from this circuit have not been included here due to the difficulty reported by Churcher[62] in obtaining good

HSPICE results.

4.2.2. A Pulse Stream Synapse using Pulse Width Modulation.

The pulse stream synapse presented here and designed by the author uses pulse width modulation techniques to perform multiplication on individual pulses. A functional specification of the pulse stream synapse has been developed and a summary of the desirable characteristics is presented in Table 4.1.

Synapse Circuit Specification	
Synapse Function	Presynaptic Neural State × Synaptic Weight
Input Neural State	Pulse Stream, Duty Cycle 0% - 50%
Synaptic Weight	Analogue Value, Stored at Synapse
Synapse Output	Analogue Integrated Activity Value, VX_i, 0V - Vdd
Cascadability	Without Redesign
Silicon Layout Area	Small
Power Consumption	Low

Table 4.1 Pulse Stream Synapse Circuit Specification

Multiplication of the incoming pulse stream by the synaptic weight can be performed directly in a number of ways. Since neural state information is encoded as a stream of pulses of constant width whose repetition rate represents neural state information, the result of the multiplication is obtained by integrating over time. Multiplication can be performed on each pulse directly by using the synaptic weight to modulate either the height or the width of each individual pulse. Alternatively, the incoming pulse stream can be used to gate currents defined by the synaptic weight into (or out of) a summing node. The method chosen for study here is to modulate the width of each pulse directly. The circuit is based on a circuit idea suggested in[63] which has been modified to greatly improve multiplier linearity.

The operation of the pulse width modulation circuit can be explained with reference to the simplified circuit diagram of Figure 4.4. The weight voltage controls the supply voltage, V_{supply}, to a two-transistor CMOS inverter M1/M2. V_{supply} is inversely proportional to the synaptic weight voltage. Presynaptic input pulses, V_j, occur asynchronously at the input to this inverter, with a constant width *Dt* and a frequency determined by the state of neuron *j*. The ability of the inverter to discharge its output node is weaker than its charging ability due to the addition of transistor M3. This discharge is linear over most of the discharge range due to transistor M3 operating in its saturated region and therefore as a constant current sink. Prior to the arrival of a pulse at its input, the inverter output is

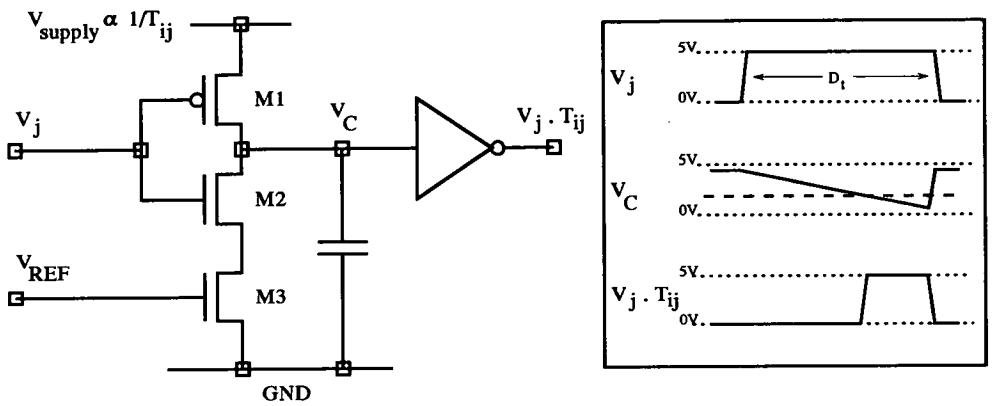


Figure 4.4 Simplified Synapse Multiplier Circuit and Timing Diagram

static at the voltage determined by the synaptic weight. When the pulse arrives at the inverter input, the output node, V_C discharges through transistors M2 and M3. At the end of the input pulse, a rapid recharge occurs via transistor M1.

The second inverter performs a threshold operation via its switching threshold, the voltage at which the inverter switches between high and low outputs. The length of the output pulse is determined by how long the discharge node spends below the switching threshold of the second inverter.

The output is therefore a pulse whose width is equal to the input pulse width, *Dt* , multiplied by a factor $0 \leq T_{ij} \leq 1$. The linearity of the multiplier characteristic is determined by the linearity of the buffer stage that converts the weight voltage, T_{ij}, into the supply voltage, V_{supply}. This stage has been implemented as an active resistor inverter (shown in the complete circuit diagram of Figure 4.6) which has a limited output voltage range and very low gain. The linearity of the pulse width multiplier circuit is shown from HSPICE simulations in Figure 4.5 showing T_{ij} against pulse width multiplier for typical process parameters.

The useful weight range for this circuit is approximately $1V \leq T_{ij} \leq 3.5V$. Potential noise problems with high T_{ij} voltages exist where the output of the buffer approaches the switching threshold of the second inverter. For this reason, the transistors in the second inverter been chosen to give a typical switching threshold of 2V, leaving some headroom in the dynamic range of T_{ij}.

The complete synapse circuit is shown in Figure 4.6, showing details of the buffer circuitry and additional transistors to allow the implementation of both excitation and inhibition within one circuit. The method of implementing inhibition requires more

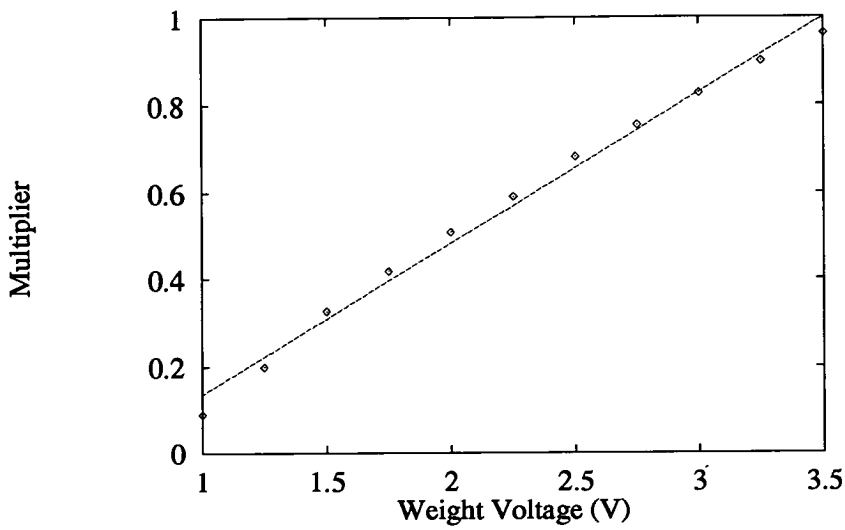


Figure 4.5 Pulse Width Multiplier Linearity : HSPICE Simulation Results

explanation.

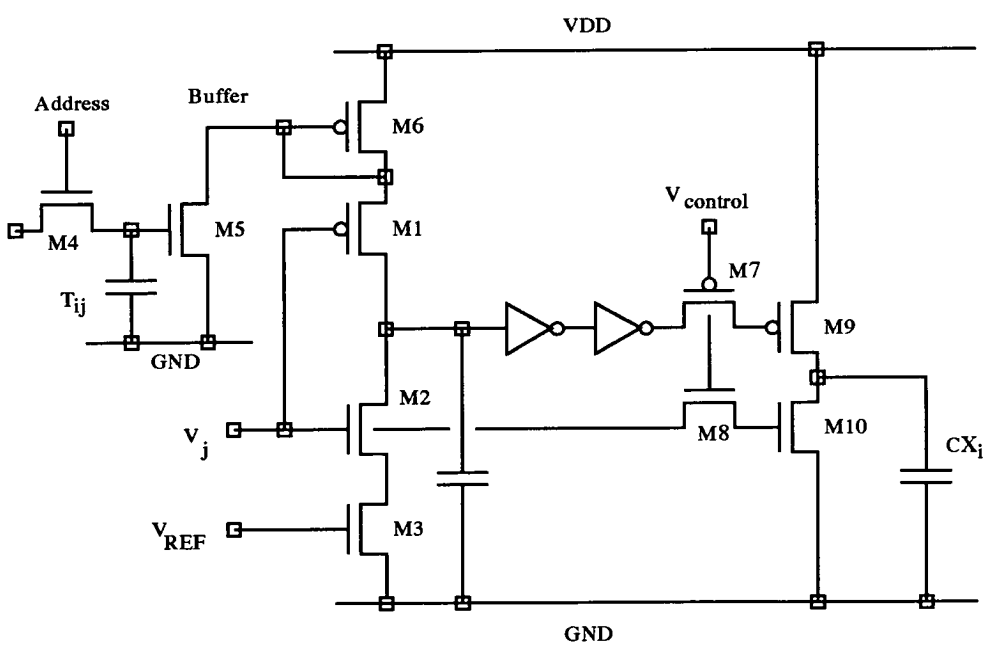


Figure 4.6 Pulse Stream Synapse Circuit using Pulse Width Modulation

If we arrange that the transistors M9/M10 are always either fully open circuit or saturated, then they are a switched current source and sink respectively, whose associated currents are controlled by the transistor widths and lengths W_9, W_{10}, L_9, and L_{10}.

Therefore, a pulse on the gate of M9 dumps a packet of charge of value

$$Q_{dumped}(T_{ij}) = \int I_9 \, dt = I_9 \, Dt \times T_{ij} \text{ coulombs,} \tag{4.4}$$

while a pulse on the gate of M10 removes

$$Q_{removed} = \int I_{10} \, dt = I_{10} \, Dt \text{ coulombs.} \tag{4.5}$$

The net charge added to the activity x_i is therefore

$$Q_{total}(T_{ij}) = Q_{dumped}(T_{ij}) - Q_{removed}. \tag{4.6}$$

If we choose values of W_9, L_9, W_{10} and L_{10} such that $Q_{dumped}(2.2V) = Q_{removed}$, then $Q_{total}(2.2V) = 0$. This effectively splits the range of T_{ij} such that a value $T_{ij} > 2.2V$ will result in an increase in x_i proportional to $(T_{ij} - 2.2V)$ and a value $T_{ij} < 2.2V$ will result in a decrease in x_i proportional to $(2.2V - T_{ij})$.

A column of these synapses, with the associated distributed capacitors on the drain connections of M9 and M10 will aggregate the total activity from all neurons connecting to neuron *i* to represent x_i by a slowly varying voltage. The rise in the voltage representing x_i caused by a single pulse passing through synapse T_{ij} is:-

$$\Delta V(x_i) = \frac{Q_{total}(T_{ij})}{C_{total}(x_i)} \tag{4.7}$$

As the number of synapses in the column is increased, the capacitance $C_{total}(x_i)$ in the denominator of (4.7) increases proportionately, and therefore the individual contributions to $\Delta V(x_i)$ become less significant. Naturally, as more synapses are added, more terms of the form $Q_{total}(T_{ij})$ are added, and the synapse is therefore 100% cascadable, both topologically and electrically.

To ensure that transistors M9 and M10 remain in saturation, two additional devices, M7 and M8, are incorporated, as shown in Figure 4.6. This incurs little penalty in silicon area, as they have the additional effect of reducing the need for the lengths of M9 and M10 to be large to restrict their source-drain currents. These devices act as voltage attenuators, as for instance, the gate voltage on M10 cannot be driven above $V_{drain}(8) - V_{tn}$. The effectiveness of the attenuation process is increased by the *body effect*, whereby a MOS device has its threshold raised as its source rises above the substrate potential. The attenuated pulses on the gates of M9 and M10 ensure that these transistors operate in the subthreshold region, and are therefore essentially always saturated, and always operating as current source and sink respectively.

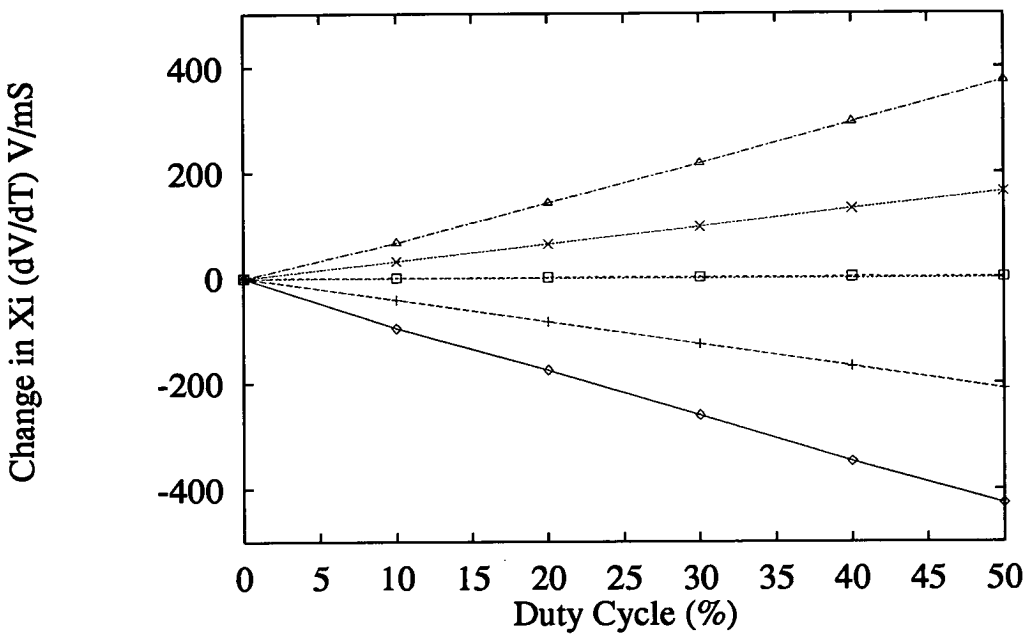


Figure 4.7 HSPICE Simulation Results for Synapse Circuit
Weight Values 1V, 1V6, 2V2, 2V8 and 3V4. Duty Cycle Steps of 10%

The linearity of the complete synapse is illustrated by the HSPICE simulations of Figure 4.7 which show change in activity voltage for the weight range $1V \rightarrow 3.4V$ and duty cycles from 0% to 50% in steps of 10%. The weight range $1V \rightarrow 2.2V$ gives a net decrease in activity representing inhibition while the range $2.2V \rightarrow 3.4V$ gives a net increase in activity representing excitation. The typical peak power consumption for an individual synapse ranges from 1mW for a weight voltage of 1V to 1.6mW for a weight voltage of 3.4V. References[63-68] contain further information on this synapse circuit.

4.2.3. A Distributed Feedback Synapse

The heart of this synapse design, shown in Figure 4.8 and developed by Baxter[69, 70, 4], is the multiplier formed by transistors M1, M2 and M3. Transistors M1 and M2 output a current proportional to the weight voltage, T_{ij}, which is then pulsed by the switch transistor, M3, controlled by the incoming neural state, V_j. The resulting output current, integrated over a period of time, represents the multiplication of T_{ij} by V_j.

The operation of the multiplier can be explained with reference to the MOSFET transistor characteristic equations. The equation of interest here is that for the drain-source current,

I_{DS} , for a MOSFET in the *linear* or *triode* region:-

$$I_{DS} = \frac{\mu_0 C_{ox} W}{L}\left[(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right] = \beta\left[(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right] \quad (4.8)$$

Here, C_{ox} is the oxide capacitance/area, μ_0 the surface carrier mobility, W the transistor gate width, L the transistor gate length, and V_{GS}, V_T, V_{DS} the transistor gate-source, threshold and drain-source voltages respectively.

This expression for I_{DS} contains a useful product term:- $\beta \times V_{GS} \times V_{DS}$, however, it also contains two other unwanted terms in $V_{DS} \times V_T$ and V_{DS}^2 .

In order to **remove** these non-linear terms a second identical MOSFET can be employed, as shown in Figure 4.8, leaving the output current defined by

$$I_3 = \beta (V_{GS1} - V_{GS2}) V_{DS1} \quad (4.9)$$

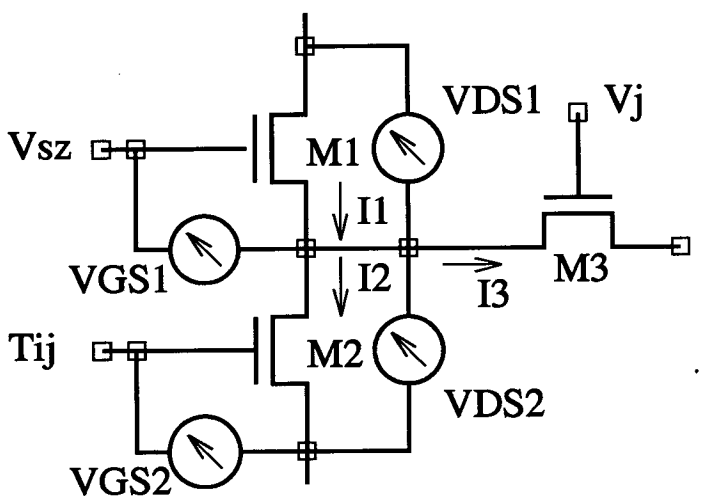


Figure 4.8 Transconductance Multiplier Circuit : Basic Principles

This is a fairly well-known circuit, called a **Transconductance Multiplier**. It was reported initially for use in signal processing chips such as filters[71] and later in[72, 73]. In our implementation we have used this circuit as a voltage controlled current source by clamping V_{DS2} to a constant value and therefore I_3 is linearly dependent on V_{GS2} which represents the synaptic weight.

However, this analysis ignores the *body effect* , whereby the threshold voltages of transistors M1 and M2 are not equal and therefore do not cancel. This results in equation 4.10.

$$I_3 = \beta (V_{GS1} - V_{GS2} + V_{TM2} - V_{TM1}) V_{DS1} \quad (4.10)$$

In addition, the output current of an individual synapse is dependent on β which is itself dependent on process parameters.

In order to minimise variations in performance of this circuit across chip due to these problems, additional buffer transistors M4 and M5 have been added as shown in Figure 4.9. The operational amplifier at the foot of each postsynaptic column provides a feedback signal, V_{outi}, that controls the current in all the buffer stages in that column of synapses so that it balances the current being sourced or sunk by the multipliers. In order to achieve good matching these additional transistors should be placed physically close to the multiplier transistors. Solving for V_{outi} then yields

$$V_{outi} = \frac{\beta_{TRANS}}{\beta_{BUF}}(V_{Tij} - (V_{bias} - V_{ref} - (V_{TM1} - V_{TM2}))) + V_{bias} + V_{ref} + (V_{TM4} - V_{TM5}) \quad (4.11)$$

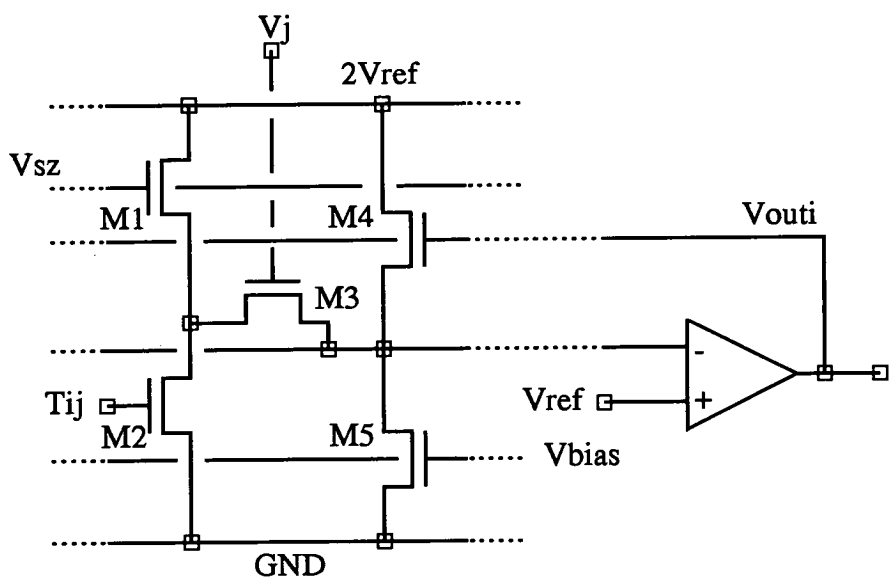
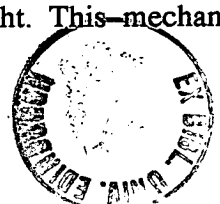


Figure 4.9 Distributed Feedback Synapse Circuit

Since the transistors are well matched, the β terms should cancel each other out. V_{outi} is therefore a linear function of the weight voltage V_{Tij}. The voltage output of the operational amplifier therefore represents a "snapshot" of all the weights switched in at a particular moment in time. This output voltage is then integrated over time to form the neural activity.

An on-chip feedback loop has been used to determine automatically the value of V_{SZ} and hence the synaptic zero weight. This mechanism compensates for process variations



between chips.

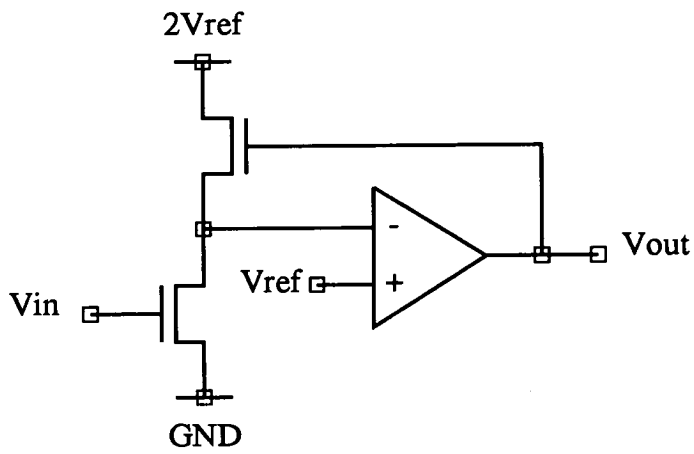


Figure 4.10 Feedback Circuit for Determination of Bias Voltages

This circuit is also used for the derivation of bias voltages used in the voltage integrator circuit described in the next section. If the input V_{in} represents a zero weight voltage, then the output, V_{out}, can be used to drive V_{sz} in Figure 4.9.

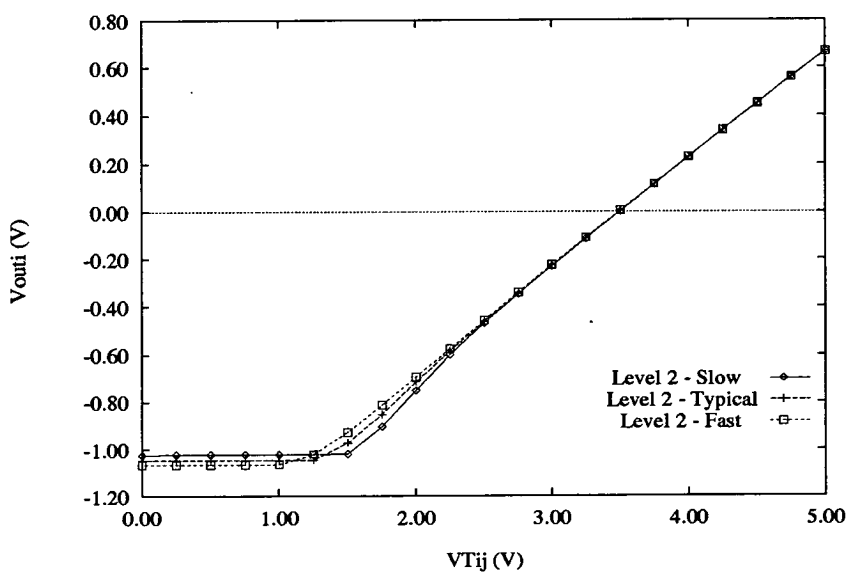


Figure 4.11 Distributed Feedback Synapse : HSPICE Simulation Results

In Figure 4.11 the output voltage of the operational amplifier is plotted against weight voltage with transistor M3 switched on for HSPICE simulations.

4.3. Voltage Integrator Circuit

As the output voltage of the operational amplifier represents a "snapshot" of all the weights switched in at a particular moment in time, the feedback operational amplifier needs to be followed by a voltage integrator, to obtain the aggregated neural activity. This integrator is composed of a differential stage and cascode current mirrors (Figure 4.12). The current, I_{ext}, through these current mirrors is determined off-chip to minimise the effects of process variation on the integrators' output current range. The output current from the integrator is directly proportional to the difference between the signals V_{outi} and the reference voltage V_{oz}. As the integrator capacitor has been implemented as a NMOS transistor any variations in the gain of the differential stage are tracked by the variations in the integration capacitance. Thus the rate of change of voltage will remain the same over all process variations.

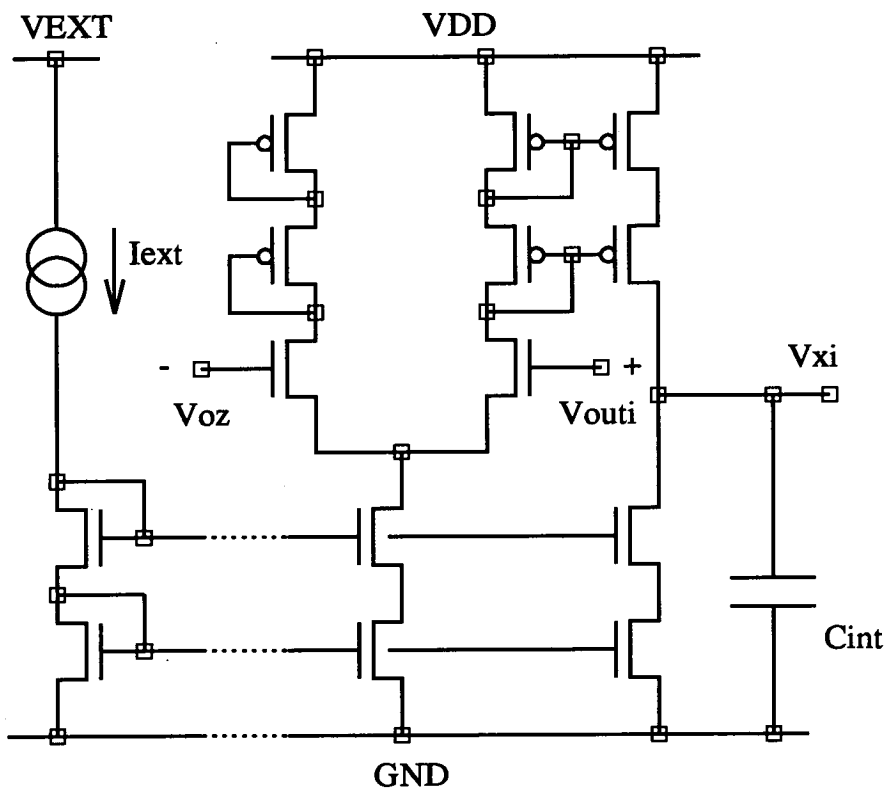


Figure 4.12 Integrator Circuit

The reference voltage V_{oz} is derived from a feedback circuit of the type illustrated in Figure 4.10. In this case the input V_{in} is the voltage V_{bias} of Figure 4.9, and the output V_{out} drives V_{oz}.

4.4. Edinburgh's Pulsed Neuron Circuit Designs

This section of the thesis details the design of three pulsed neuron circuits developed within the VLSI Neural Network group at Edinburgh University.

4.4.1. A Current Controlled Oscillator Neuron Design

The current controlled oscillator neuron developed by Churcher[74] is an attempt to implement a pulse based neuron that occupies a small silicon area in order to achieve high integration densities. The model used is, in some respects, similar to that of a biological neuron and to the work of Meador[10].

The neuron operates by aggregating current pulses from an array of synapses onto the capacitor shown in Figure 4.13. Once sufficient charge has accumulated so that the voltage on the capacitor exceeds the upper threshold of the comparator circuit, an output pulse is generated. The duration of the output pulse is determined by the capacitor size, the discharge current I_r and the switching thresholds of the comparator circuit.

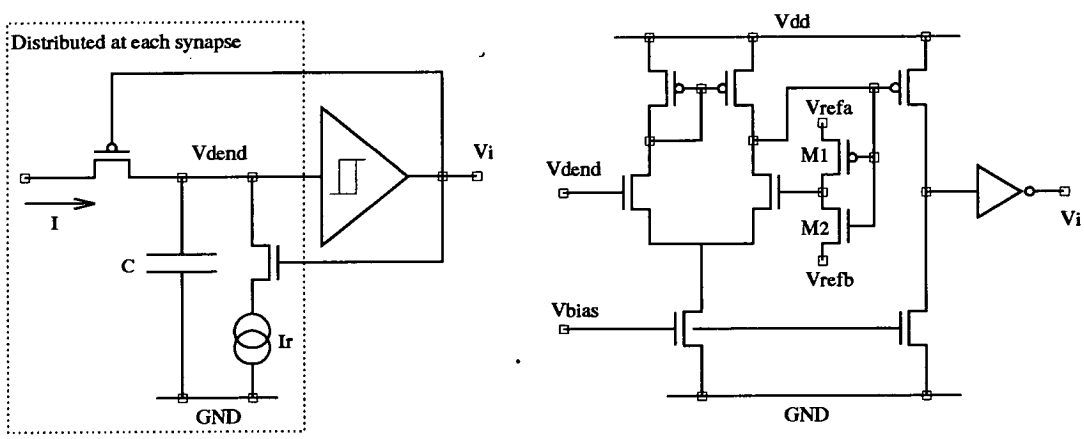


Figure 4.13 Current Controlled Oscillator Neuron : Principles and Comparator Circuit Details

The operation of the circuit is illustrated by the timing diagram of Figure 4.14. For practical reasons, the capacitor, C, and the transistors controlling the charge/discharge process have been distributed at each synapse location. The details of the synapse circuit used in this system have been outlined in the previous section.

The comparator circuit details are are shown in Figure 4.13 where hysteresis has been introduced by the feedback arrangement of transistors M1 and M2, and the reference voltages V_{refa} and V_{refb}. The comparator circuit is of a standard design, V_{bias} sets the differential gain and slew rate of the circuit, however, modifications have been made to the feedback mechanism that controls the hysteresis of the circuit. The feedback circuit formed by transistors M1 and M2 would normally be derived from the output of the inverter V_i but this configuration was found to give spurious oscillations at low input currents to the integrating capacitor. This problem has been solved by taking the feedback from the output of the differential stage. In addition, the differential stage has a low gain, to ensure stability at high input current levels and a wide operating range.

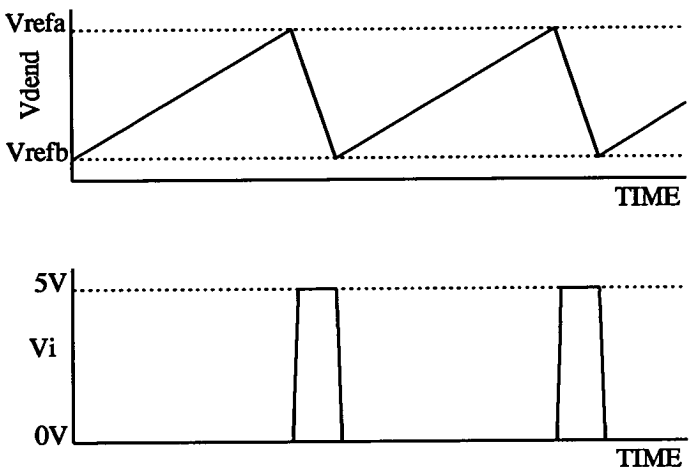


Figure 4.14 Current Controlled Oscillator Neuron : Timing Diagram

4.4.2. A Pulse Width Neuron Circuit Design

In order to produce pulse width modulated input or output neural states a circuit has been developed by Churcher [62] whose transfer characteristic is programmable. The circuit simply consists of a conventional comparator whose non-inverting input (+) has a capacitor used to store an analogue input voltage when the circuit is used as an input neuron, or integrated neural activity voltage when used as an output neuron. The inverting input (-) is fed a symmetrical ramp input signal whose characteristics can be varied.

The operation of the circuit with a linear ramp input signal is illustrated in Figure 4.15. The ramp input, normally high, is linearly ramped to 0V. When the ramp signal passes the voltage stored on the capacitor, the output of the comparator switches high. Once the ramp signal reaches 0V it is ramped back up to 5V. When the ramp signal passes the voltage stored on the capacitor again, the output of the comparator switches low. An

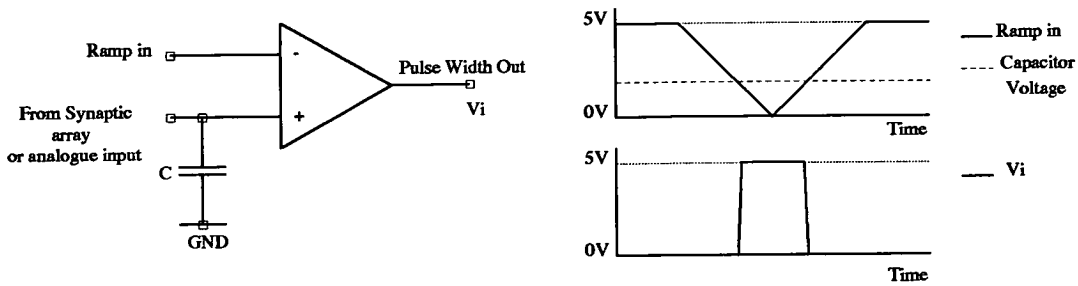


Figure 4.15 Pulse Width Neuron : Schematic and Waveform Diagram

output pulse signal V_i is therefore produced whose width is linearly proportional to the voltage stored on the capacitor. The ramp waveform is distributed to all input or output neurons. The symmetry of this waveform ensures that all pulse centres are coincident rather than pulse edges and this prevents the large power transients that occur with synchronous digital edges.

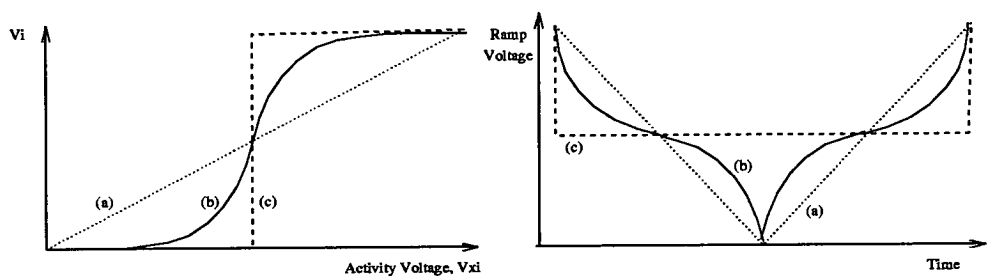


Figure 4.16 Pulse Width Neuron : Various neuron transfer characteristics and required ramp input waveforms. (a) Linear, (b) Sigmoid and (c) Threshold.

By varying the characteristics of the ramp waveform, various neuron transfer characteristics can be obtained. Figure 4.16 shows several typical neuron transfer characteristics and the ramp signals required to obtain them using this neuron circuit. In practice, these ramp waveforms have been generated off chip by storing the ramp signals in RAM and sequencing these values through a digital to analogue converter (DAC). The comparator circuit is almost completely invariant to processing variations across chip, and as a result of the use of digital RAM for ramp signal storage, the pulse width neuron is also largely invariant to process variations. The programmability and performance of this technique

is at the expense of the additional external RAM and DAC circuitry required.

4.4.3. Pulse Stream Neuron Circuit Design

A pulse stream neuron circuit has been developed by the author that produces a fixed width pulse output with a sigmoid activity to duty cycle characteristic. In addition, feedback mechanisms have been introduced to allow the output pulse width of the neurons to be set from an off-chip oscillator reference. Further development of this circuit has enhanced the performance to facilitate a variable gain sigmoid characteristic under electronic control. A summary of the circuit design specification is given in Table 4.2.

Neuron Circuit Specification	
Input Activity Range	0V - Vdd
Output Pulse Width	1μs
Output Duty Cycle Range	0% - 50%
Transfer Characteristic	Sigmoidal. Electronically variable Gain
Silicon Layout Area	Small
Power Consumption	Low

Table 4.2 Pulse Stream Neuron Circuit Specification.

4.4.3.1. Pulse Stream Neuron Circuit Basic Principles

The basic circuit details of a pulse stream neuron circuit are shown in Figure 4.17. The pulse width output is determined by the rate at which the capacitor is charged between the upper and lower threshold limits of the comparator, while the pulse spacing is determined by the discharge rate.

The operation of the circuit is summarised in Table 4.3. The output duty cycle, defined as the pulse width output multiplied by the pulse frequency, is a function of the currents I_H and I_L. In order to obtain a constant pulse width output to meet the specification in Table 4.2, I_H should be set to charge capacitor C in 1μs. The pulse frequency is controlled by I_L which is a function of the input activity voltage, VXi. The control of I_L should be such that the input activity voltage to output duty cycle transfer characteristic is sigmoidal.

A circuit to perform the required transformation from input duty cycle, VXi, to output current, I_L, is shown in Figure 4.18. It is a differential stage with a tail current equal to the capacitor charging current I_H. The current I_L is "copied" and used to discharge the capacitor C in Figure 4.17. I_L varies from a maximum of $I_L = I_H$ when VXi = Vdd

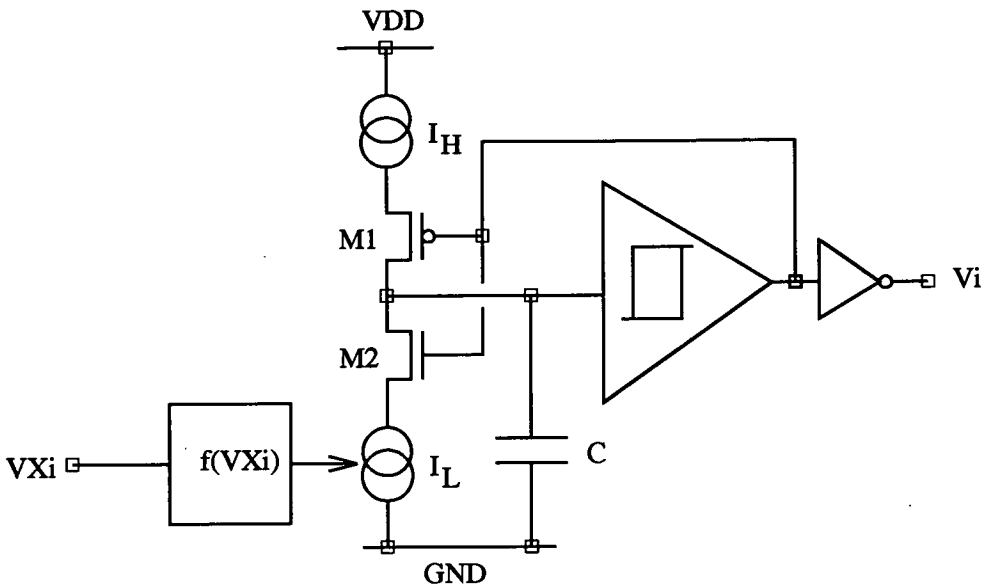


Figure 4.17 Pulse Stream Neuron : Basic Principles

Neuron Circuit Operation			
V_i	M1	M2	C
HIGH	ON	OFF	Charging via I_H
LOW	OFF	ON	Discharging via I_L

Table 4.3 Pulse Stream Neuron Circuit Operation.

producing a 50% duty cycle output, to $I_L = 0$ when VXi = 0 producing a 0% duty cycle or permanent 0V output. The voltage VMID sets the midpoint of the transfer characteristic and is set to $\frac{VDD}{2}$.

The design process to create a circuit with the desired transfer characteristic relies on the following analysis. Assuming the transistors M3 and M4 in the differential stage of Figure 4.18 are in saturation their currents can be described by equations 4.12 and 4.13,

$$I_{M3} = \frac{\beta_{M3}}{2} (V_{SGM3} - V_T)^2 \tag{4.12}$$

$$I_{M4} = \frac{\beta_{M4}}{2} (V_{SGM4} - V_T)^2 \tag{4.13}$$

where V_{SG} and V_T are the source-gate and threshold voltages. The transconductance

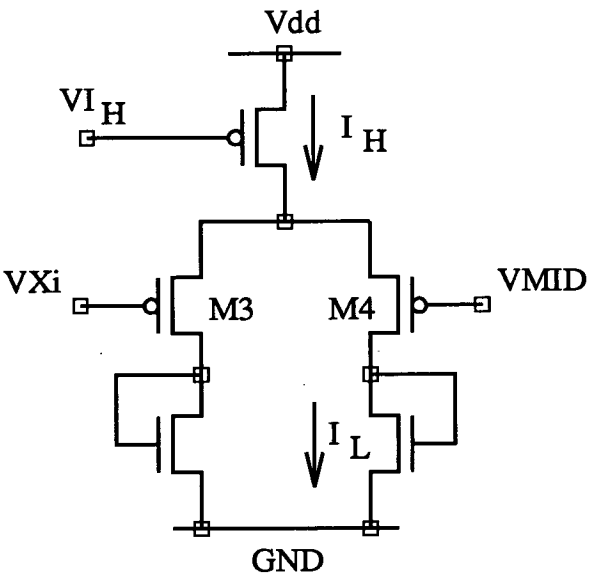


Figure 4.18 Pulse Stream Neuron Input Stage

parameter β is given in terms of physical parameters as,

$$\beta \approx (\mu_0 C_{ox}) \frac{W}{L} \text{ amps/volts}^2 \tag{4.14}$$

where μ_0 is the surface carrier mobility of the channel for the transistor (cm^2/volt. seconds), and C_{ox} is the capacitance per unit area of the gate oxide (F/cm^2). The variables W and L are the width and length of the device set by the designer. If we define the geometries of transistors M3 and M4 so that $\beta_{M3} = 2\beta_{M4}$ and substitute β for β_{M4}; the differential input voltage is

$$V_{ID} = V_{SGM3} - V_{SGM4} = \left[\frac{2I_{M3}}{\beta}\right]^{1/2} - \left[\frac{4I_{M4}}{\beta}\right]^{1/2} \tag{4.15}$$

and

$$I_H = I_{M3} + I_{M4} \tag{4.16}$$

where it has been assumed that M3 and M4 are matched. Substituting Equation 4.16 into Equation 4.15 and forming a quadratic equation allows the solution of I_{M4} and hence I_L in Equation 4.17.

$$I_L = I_{M4} = \frac{I_H}{3} + \frac{\beta V_{ID}^2}{18} \pm \frac{I_H}{9}\left[\frac{12\beta V_{ID}^2}{I_H} - \frac{2\beta^2 V_{ID}^4}{I_H^2}\right]^{1/2} \tag{4.17}$$

The regions in which Equation 4.17 is valid is defined by $V_{ID} < \left[\frac{2I_H}{\beta_x}\right]^{1/2}$ where $\beta_x = \beta_{M4}$

for positive values of V_{ID}, otherwise $\beta_x = \beta_{M3}$. The equation for the duty cycle in terms of I_L and I_H is defined in Equation 4.18.

$$\text{DutyCycle} = \text{PulseWidth} \times \text{PulseFrequency} = \frac{I_L}{I_H + I_L} \tag{4.18}$$

The relationship between differential input voltage and duty cycle output has therefore been established by substituting Equation 4.17 into Equation 4.18. This function has been plotted in Figure 4.19.

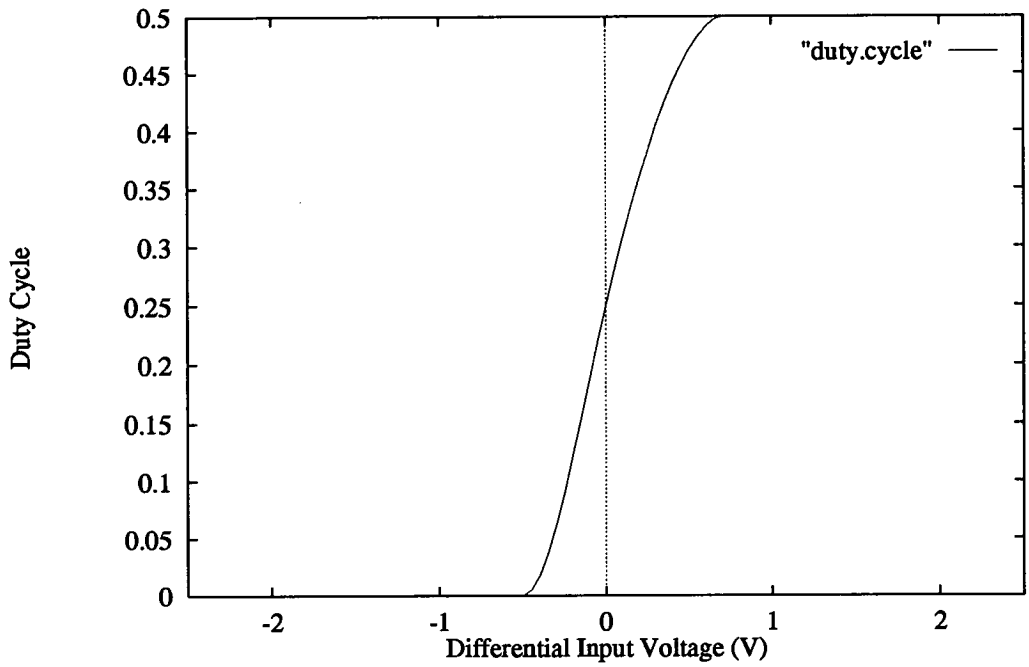


Figure 4.19 Pulse Stream Neuron Transfer Characteristic.
Y-axis duty cycle range $0 \rightarrow 0.5$
represents actual duty cycles in the range $0 \rightarrow 50\%$.

The slope of the transfer characteristic is of interest here and it can be found by differentiating Equation 4.18 with respect to V_{ID} and setting $V_{ID} = 0$, giving the differential gain of the neuron as

$$\frac{\delta \text{DutyCycle}}{\delta V_{ID}} \left(V_{ID} = 0 \right) = \frac{1}{8} \left[\frac{3\beta}{I_H} \right]^{1/2} \tag{4.19}$$

The gain of the transfer characteristic is therefore proportional to the square root of β and inversely proportional to the square root of I_H. These parameters can be set by the

designer at the design stage. A circuit with electronically programmable gain is described in the next section.

The complete circuit for a pulse stream neuron is shown in Figure 4.20.

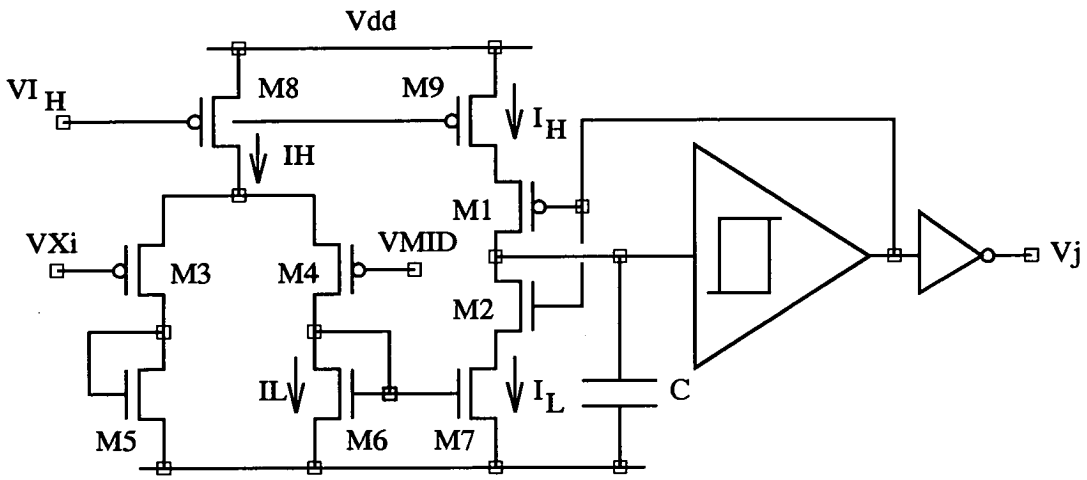


Figure 4.20 Pulse Stream Neuron Circuit

The non-linear transfer function commonly used for multi-layer perceptron architectures, for example, is expressed in Equation 4.20 in terms of pulse stream variables.

$$\text{DutyCycle} = \frac{1}{1+e^{(V_{ID})}} \tag{4.20}$$

Although the relationship of Equation 4.18 is not mathematically equivalent to Equation 4.20 it has many fundamental similarities. It is monotonically increasing, has no discontinuities and saturates at either end of the V_{ID} range. These characteristics are essential for the correct operation of the back-propagation learning procedure.

4.4.3.2. A Pulse Stream Neuron with Electrically Adjustable Gain

In order to have more control over network dynamics in back propagation networks, the activation function of Equation 4.21 is suitable provided the *temperature* can be controlled electronically.

$$V_i = \frac{1}{1+e^{(V_{ID})/T}} \tag{4.21}$$

The *Temperature*, T of the sigmoid varies the gain of the sigmoid function. A low value of T gives the sigmoid function a high gain, whereas a high value of T results in a more gently varying function. The gain of the Pulse Stream Neuron Circuit, described in Equation 4.19 for $V_{ID} = 0$, is a function of the square root of β and the inverse of the square

root of I_H. The gain of the neuron circuit can therefore be controlled electrically by varying I_H. However, only small variations in gain are possible due to the square root function.

In practice, the pulse width of the resulting pulse stream has to be maintained, so that I_H must remain constant. To vary the gain the current through the differential transistors, M3 and M4 should be varied. In addition, the current I_L should have a maximum value equal to I_H and a minimum value equal to zero.

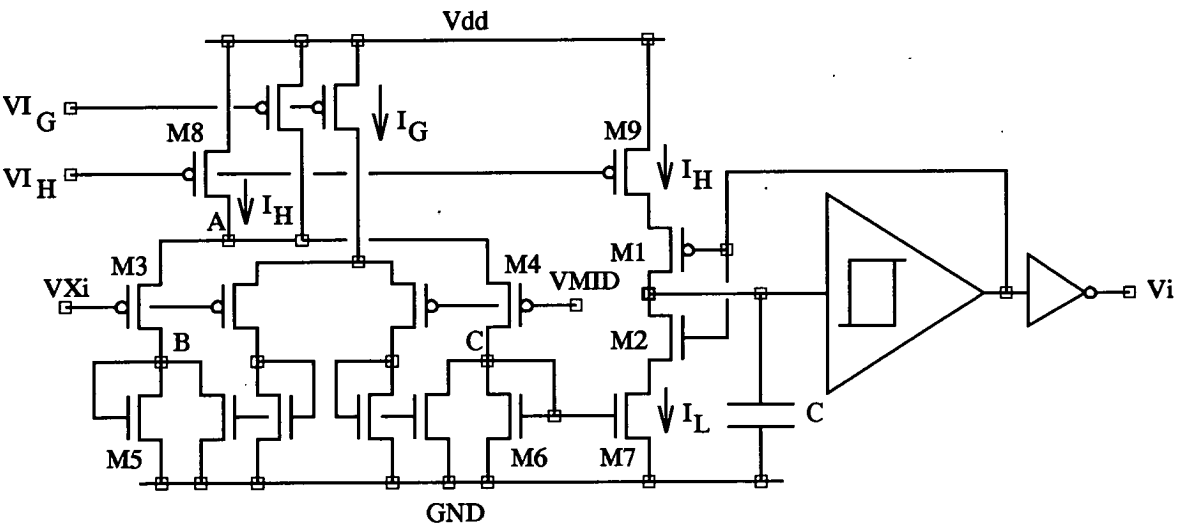


Figure 4.21 Neuron with Electrically Adjustable Gain

The circuit shown in Figure 4.21 performs this function. It uses an additional differential stage to control the current through transistors M3 and M4. Additional current, I_G, is injected into the differential stage at node A, and removed again at nodes B and C. This mechanism controls the gain of the transfer function while maintaining the mid and end points of the sigmoid characteristic.

The current through transistor M4, assuming that $\beta_{M3} = 2\beta_{M4}$, is described by Equation 4.22,

$$I_{M4} = \frac{I_{TAIL}}{3} + \frac{\beta V_{ID}^2}{18} \pm \frac{I_{TAIL}}{9}\left[\frac{12\beta V_{ID}^2}{I_{TAIL}} - \frac{2\beta^2 V_{ID}^4}{I_{TAIL}^2}\right]^{1/2} \qquad (4.22)$$

where $I_{TAIL} = I_H + I_G$. The regions in which Equation 4.22 is valid is defined by $V_{ID} < \left[\frac{2I_{TAIL}}{\beta_x}\right]^{\frac{1}{2}}$ where $\beta_x = \beta_{M4}$ for positive values of V_{ID}, otherwise $\beta_x = \beta_{M3}$. The

current I_L is equal to the current I_{M4} minus the current removed from node C by the inner differential stage. Assuming the ratio of the β's of the inner differential stage is the same as that for the outer stage, and assigning this to be β_i, Equation 4.23 describes the current removed from node C.

$$I_C = \frac{I_G}{3} + \frac{\beta_i V_{ID}^2}{18} \pm \frac{I_G}{9}\left[\frac{12\beta_i V_{ID}^2}{I_G} - \frac{2\beta_i^2 V_{ID}^4}{I_G^2}\right]^{1/2} \tag{4.23}$$

Similar restrictions apply to the valid regions of equation 4.23 as to those of equation 4.22. The current I_L is therefore defined in Equation 4.24.

$$I_L = \frac{I_H}{3} + \frac{\left[\beta - \beta_i\right] V_{ID}^2}{18} \pm \left[\begin{array}{l} \frac{I_{TAIL}}{9}\left[\frac{12\beta V_{ID}^2}{I_{TAIL}} - \frac{2\beta^2 V_{ID}^4}{I_{TAIL}^2}\right]^{1/2} \\ -\frac{I_G}{9}\left[\frac{12\beta_i V_{ID}^2}{I_G} - \frac{2\beta_i^2 V_{ID}^4}{I_G^2}\right]^{1/2} \end{array}\right] \tag{4.24}$$

The gain of the transfer characteristic can be found by substituting equation 4.24 into equation 4.18 and differentiating with respect to V_{ID}. Setting $V_{ID} = 0$, the differential gain of the neuron is defined by equation 4.25.

$$\frac{\delta \text{DutyCycle}}{\delta V_{ID}} \left(V_{ID} = 0 \right) = \frac{1}{8}\left[\left[\frac{3\beta \left(I_H + I_G \right)}{I_H^2}\right]^{1/2} - \left[\frac{3\beta_i I_G}{I_H^2}\right]^{1/2}\right] \tag{4.25}$$

The form of equations 4.24 and 4.25 are similar to those of 4.17 and 4.19, however the gain of the transfer function can now be controlled by adjusting I_G.

4.4.4. Comparator Circuit Design

In order to minimise the area occupied by the neuron circuit and to reduce the amount of reference voltages required, a simple comparator circuit has been used. Two different VLSI technologies for the implementation of the fixed and variable gain pulse stream neuron circuits have been used requiring the design of two different comparator circuits.

The original circuit, shown in Figure 4.22 was implemented in 2μm digital CMOS. The input stage is a voltage follower circuit followed by two digital inverters. The feedback transistors M1A and M2A provide hysteresis. However, when simulated using 1.5μm digital CMOS design rules, this circuit did not switch properly under certain process combinations. For this reason the second circuit was used. In this circuit, the transistors M1B and M2B provide hysteresis by fighting the switching action of the transistors in the input stage.

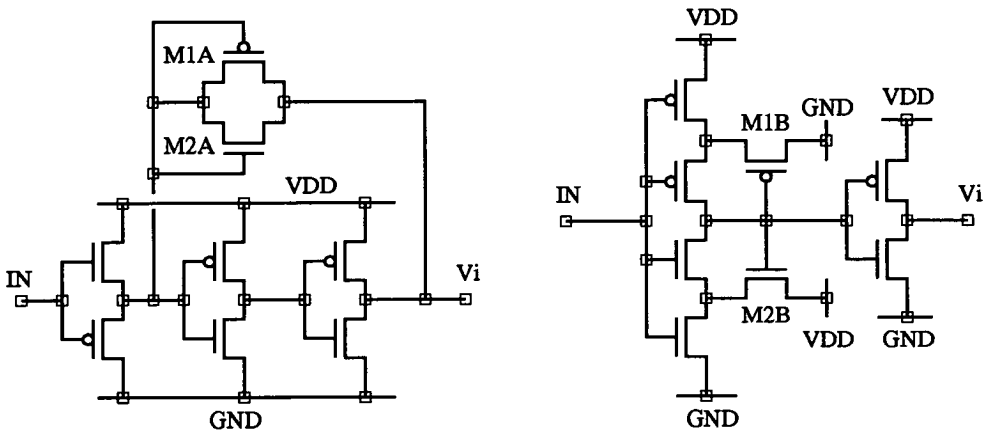


Figure 4.22 Comparator Circuit Details : 2μm and 1.5μm Versions

4.4.5. Automatic Pulse Width and Gain Setting Mechanisms

In developing pulse stream neuron circuits it is envisaged that systems may be devised that use more than one chip or operate in different ambient conditions. However, manufacturing process variations and environmental conditions such as temperature will lead to variation in circuit performance between devices. Therefore it is important in analogue circuit design to introduce mechanisms that minimise the effects of these variations.

A mechanism that automatically sets the pulse width output and gain of of the neuron circuit has been developed. The technique employs two phase lock loops, one for controlling the pulse width and one for controlling the gain. The phase lock loops are identical, only the signals used to control them are different.

The block diagram of Figure 4.23 shows the components of the locking mechanism. A charge pump phase lock loop[75] which allows accurate phase tracking with a passive RC loop filter has been used. A well-known sequential logic Phase Frequency Detector (PFD[76, 77]) is used for phase/frequency detection. Since it has memory to compare frequency as well as phase, the PFD is free from locking to second or third harmonics.

The outputs of the PFD are the signals UP and DOWN and dump or remove charge from the loop filter. The resulting voltage controls either the pulse width or gain control currents in a reference pulse stream neuron. Lock is achieved when the output of the reference pulse stream neuron matches the input waveform. Both UP and DOWN are deactivated to a high level when the loop is in a perfectly locked state. Under no circumstances are both signals activated together. The resultant control voltages from the phase lock loop, VI_H or VI_G, are distributed across chip to control the pulse widths and gains of all other pulse stream neurons.

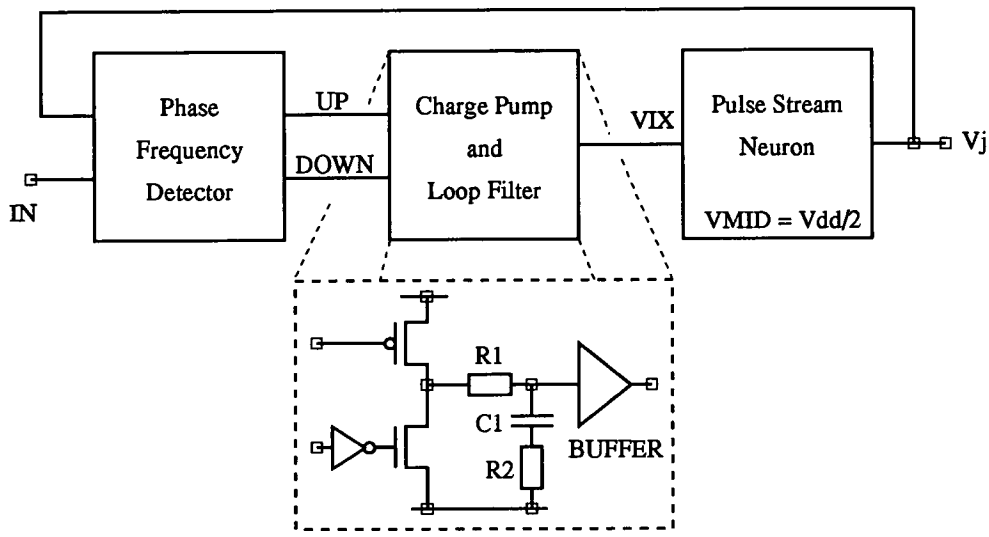


Figure 4.23 Phase Lock Loop Schematic

Phase Lock Loop Configuration					
Pulse Width Control			**Gain Control**		
IN	VIX	VXi	IN	VIX	VXi
50% Duty Cycle	VI_H	VDD	> 25% Duty Cycle	VI_G	VMID + δ

Table 4.4 Phase Lock Loop Configuration
for Pulse Width and Gain Control

The circuit configuration details for pulse width and gain control are given in Table 4.4. For pulse width control, the reference pulse stream neuron operates at 50% duty cycle, since VX_i = VDD, and VI_H adjusts the pulse width to obtain lock to the 50% duty cycle input reference. For gain control, a point on the sigmoid characteristic is chosen and VX_i is set to this voltage VMID + δ, as illustrated in Figure 4.24. The desired duty cycle required for this activity voltage is fed from off-chip to the input of the phase lock loop, and VI_G is adjusted to obtain lock. By varying the the input duty cycle, the gain of the neuron can be varied.

SPICE level simulation of the complete feedback network presents problems because a small time step is required to simulate the sequential logic in the PFD, and yet a long simulation time is required to allow the phase lock loop to converge. The loop filter formed by R1, C1 and R2 have therefore been implemented off-chip to allow variation of the

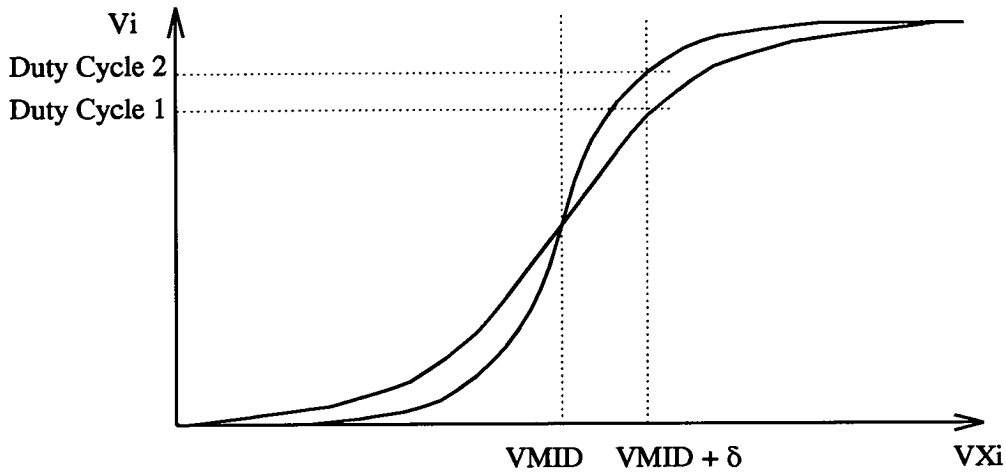


Figure 4.24 Neuron Gain Control : Principles

filter characteristics and to allow the feedback mechanism to be overridden.

4.5. Inter-chip Communication Strategies for Pulse Stream Neural Networks.

Pulse Stream Neural Networks encode neural state information in the time domain as a series of pulses whose spacing represents the neural state. The inter pulse spacing at the neuron output is determined by the input activity voltage V_{xi}. The relationship between the input activity voltage and the output duty cycle is determined to a large extent by the characteristics of the input stage of the neuron circuit.

Since the input stage characteristics of neurons on different chips will vary slightly due to manufacturing variations, an activity voltage producing a pulse stream on one chip will not necessarily produce the same pulse steam output on another chip. For this reason, the pulse stream output of each neuron has been used to communicate neural states.

4.6. Transmitting Data - Encoding Pulse Streams

In order to determine a suitable inter-chip communication scheme, a number of possibilities have been considered and their hardware and performance characteristics have been assessed.

It is assumed in this analysis that duty cycles in the range 0.5% to 50% are to be encoded, and that the pulse width is τ. It is also assumed that the neuron address can be communicated directly to the receiver over an address bus, or can be implied by assuming a fixed sequence of data transfer. The addressing mechanism is therefore independent of the communication scheme employed.

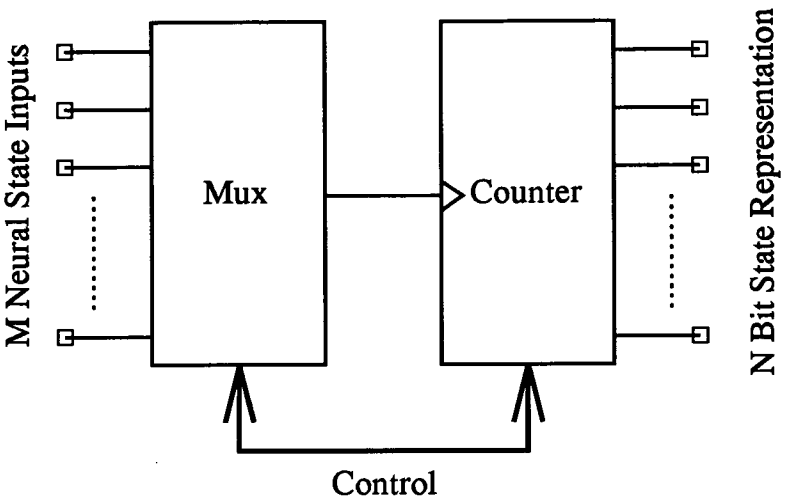


Figure 4.25 Scheme 1 : Pulse Stream Data Encoding

Figure 4.25 illustrates one possibility. Each of the M input pulse streams are multiplexed in turn onto the clock line of a counter for a period equal to 200τ, the period of the lowest possible duty cycle. The counter counts the number of pulses arriving at the clock signal during this time period. The output of the counter can be transferred from the chip as a direct measure of the state of the transmitting neuron.

A variation on this scheme, shown in Figure 4.26, is to use a counter at each transmitting neuron, all enabled for a period of 200τ and then multiplex the duty cycles off the chip at a rate limited only by the bandwidth of the inter-chip link.

Instead of allowing a fixed interval for the counter to count pulses arriving at the clock input, the configuration of Figure 4.27 allows the *space between pulses* to be measured. A *Pulse Space Encoder* outputs a pulse that represents the space between consecutive neuron output pulses. The data transmission time now depends on the data being transmitted and is faster the greater the average duty cycle over the link. This transmission strategy transmits a *snapshot* of the current duty cycle and avoids the averaging operation inherent in the previous schemes.

The Pulse Space Encoder can be implemented as a finite state machine as illustrated by the waveform and state diagrams of Figure 4.28. The input to the Pulse Space Encoder is a pulse stream selected by the multiplexor. The output is a pulse whose width represents the space between the pulses of the input pulse stream. Since the Pulse Space Encoder is very simple, requiring only 4 states, it can be implemented using 2 D-type flip-flops. This compares with the N D-type required for a N bit counter.

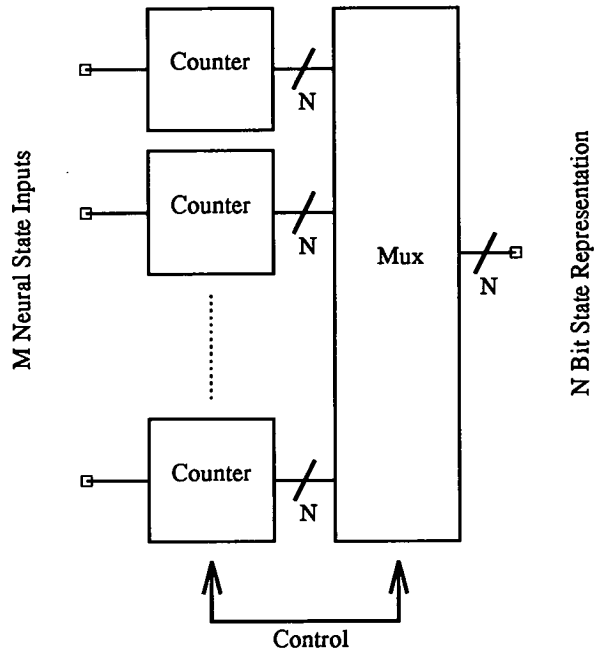


Figure 4.26 Scheme 2 : Parallel Pulse Stream Data Encoding

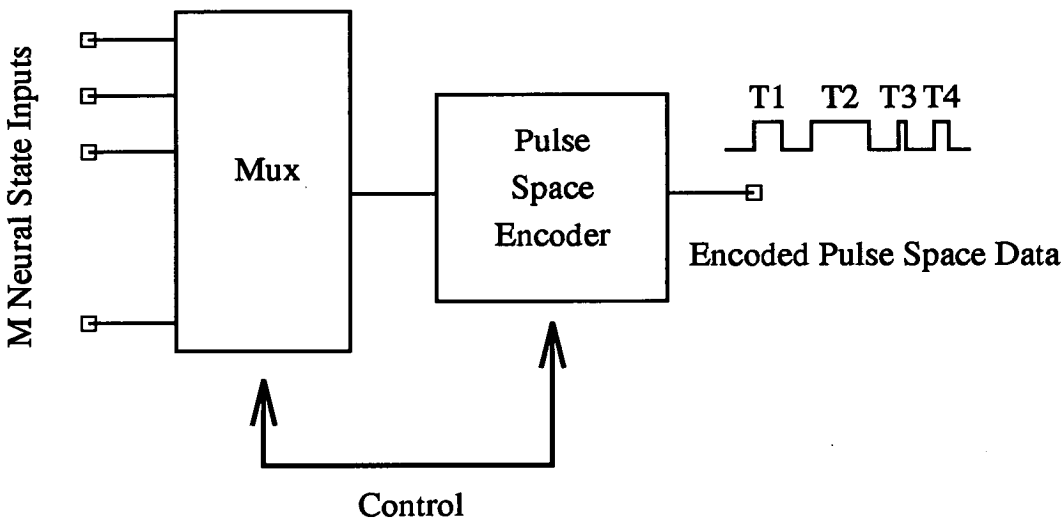


Figure 4.27 Scheme 3 : Serial Pulse Stream Data Encoding

T1, T2, T3 and T4 represent the time between output pulses of neuron 1, 2, 3 and 4,

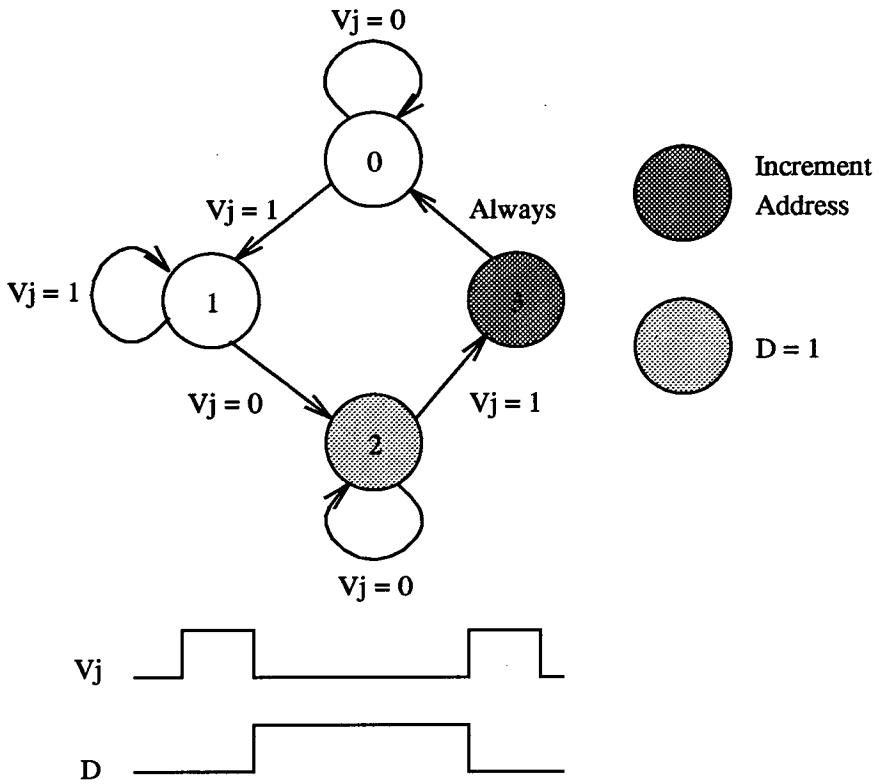


Figure 4.28 Pulse Space Encoder : State and Waveform Diagrams

The three schemes outlined here are compared in Table 4.5. The counters require 7 bits to represent duty cycles between 0.5% and 50%, although additional circuitry is required to derive the clocks for these counters to ensure the required resolution over the entire duty cycle range[78]. The time δ is the time required to transmit one data word over the communication link.

Inter-Chip Communication Schemes : Performance Comparison					
Scheme	**Mux**	**Counters**	**Pulse Space Encoder**	**Chip pin count**	**Communication Time**
1	Serial	1, 7 Bit	None	7+	$M \times 200\tau + M\delta$
2	Parallel	M, 7 Bit	None	7+	$200\tau + M\delta$
3	Serial	None	1	1+	$M \times 2\tau \rightarrow M \times 200\tau$

Table 4.5 Inter-chip Communication Schemes : Hardware requirements and performance comparison.

The choice of inter-chip communication scheme is a trade-off between hardware complexity and speed of data communication. Therefore, although Scheme 2 offers the fastest communication time, it requires a 7 bit counter dedicated to each transmitting neuron. Scheme 3 offers a data dependent communication rate that, at best is two orders of magnitude faster than scheme 1, and at worst has approximately the same communication rate. This inter-chip communication scheme (3) was therefore chosen due to the simple hardware requirements, the low chip pin count and the inherently asynchronous nature of scheme. It is interesting to note, although not strictly relevant, that inter-pulse timing is used in many biological systems, rather than pulse frequency - see for example[79].

4.6.1. Controlling the Communication Interface

The communication system has been designed to allow a chip to communicate its neural states to another chip, or chips, and to allow communication to and from standard microprocessor peripheral parts. In order to keep the chip pin count to a minimum, the address information relating to the neural state being transmitted is implied rather than being sent explicitly. This imposes the restriction on inter-chip communication that the number of neural states being transmitted is predefined at both the transmitter and the receiver. In order to ensure that the transmitting and receiving chips remain in step with each other a handshake mechanism has been employed in the communication interface. The handshake mechanism also facilitates easy interface with standard microprocessor peripheral parts, and the transmission of neural states from a transmitting chip to a number of receiving chips.

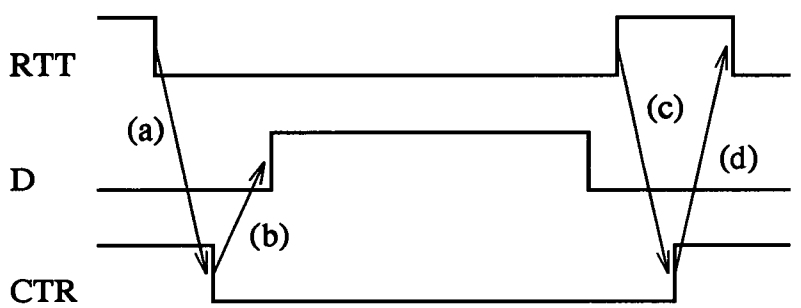


Figure 4.29 3-Wire Handshake
RTT - Request To Transmit. CTT - Clear To Transmit. D - Data.
(a) Transmitter requests to send data. (b) Receiver ready to receive data.
(c) Transmitter signals end of data. (d) Receiver signals receipt of data.

4.6.1.1. The 3-Wire Handshake

In order to guarantee that two independent finite state machines remain in step with each other a 3-wire handshake mechanism is required. One of the three wires carries the pulse width modulated information relating to pulse spacing and has been named *Data*, or D. The other two perform the handshake operation. One of these is used by the transmitter to interrogate the receiver to check either on its readiness to receive data, or that it has received data sent and has been named *Request To Transmit* , or RTT. The other wire allows the receiver to confirm readiness to receive data, or to confirm receipt of the data sent and has been named *Clear To Transmit* , or CTR. The operation of the handshake mechanism is illustrated in Figure 4.29.

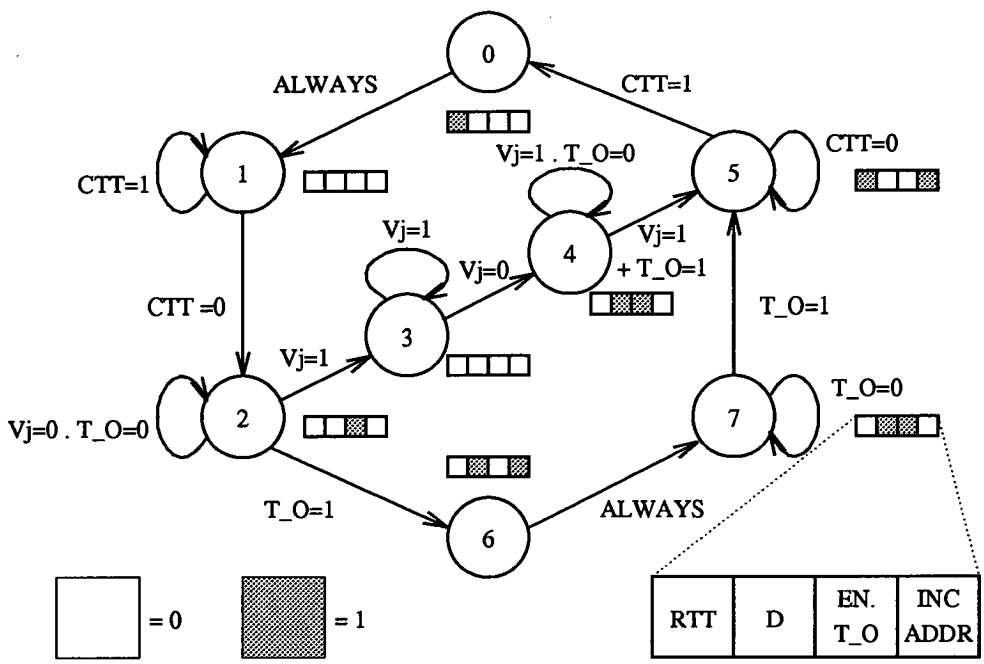


Figure 4.30 Transmitter State Diagram
RTT - Request To Transmit. CTT - Clear To Transmit. D - Data.
Vj - Pulse Stream State being transmitted. T_O - Time-out.
En. T_O - Enable Time-out. Inc Addr - Increment neuron address.

4.7. Transmitter and Receiver Controllers

The signals used to control the communication interface have been derived from two separate and independent state machines. A transmitting state machine controls the Pulse Space Encoder, associated multiplexor and RTT signal while a receiving state machine controls the CTT signal, the Pulse Stream Regeneration Circuit and associated

demultiplexor. The state machines are initialised to a known state with a common *Reset* signal and thereafter the handshake ensures correct transfer of data.

Figure 4.30 shows the transmitter state diagram developed from the Pulse Space Encoder state diagram of Figure 4.28. Duty cycles less than 0.5% are rounded up to 0.5% by the use of a time-out mechanism started after the detection of the first pulse from the current transmitting neuron. If a second pulse does not arrive before the end of the time-out period, the data signal D is taken low by the transmitter state machine thereby performing the rounding operation. The transmitter state diagram requires 8 states and therefore only 3 D-type flip-flops.

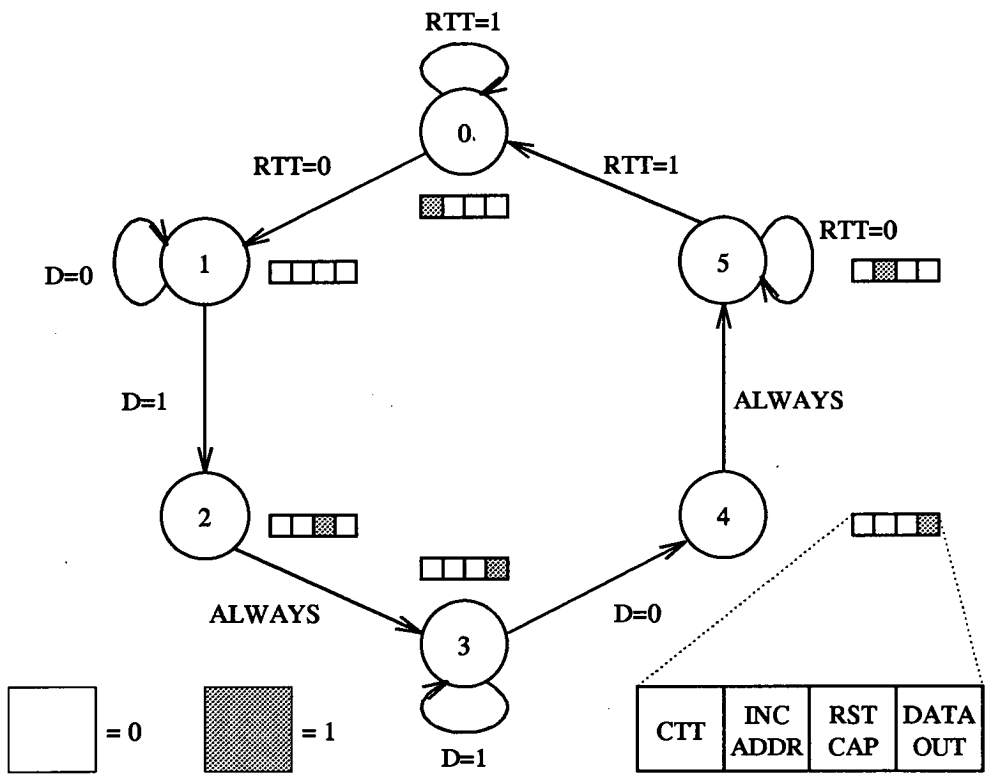


Figure 4.31 Receiver State Diagram
RTT - Request To Transmit. CTT - Clear To Transmit. D - Data.
Rst Cap - Reset Receiving Capacitor. Inc Addr - Increment neuron address.
Data Out - Regenerated Pulse Stream.

The receiver state diagram shown in Figure 4.31 has been implemented using 6 states and similarly requires only 3 D-type flip-flops for full implementation.

4.7.1. Receiving Data - Regenerating Pulse Streams

Communicating neural states using the pulse width modulation technique outlined earlier allows the use of analogue circuit techniques to implement the pulse stream regeneration function. The circuit is essentially an oscillator whose output pulse spacing is matched to the input pulse width modulated signal.

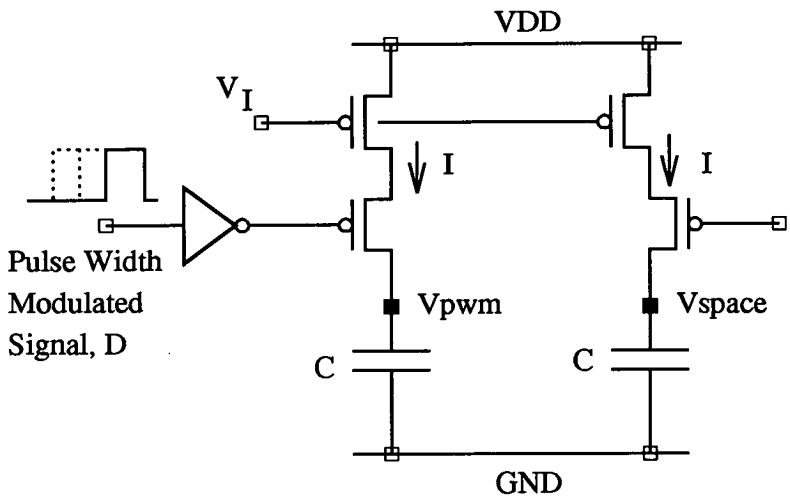


Figure 4.32 Pulse Stream Regeneration - Pulse Space Control Circuit

The pulse spacing information is stored on the receiving chip between updates over the communication link as charge on a capacitor. The storage capacitor is charged by a constant current for the duration of the incoming pulse as shown in Figure 4.32. The resulting voltage, V_{pwm}, provides a reference against which the output of an identical timing circuit, V_{space}, can be compared to determine pulse spacing. The pulse stream regeneration circuit continues to output pulse streams with this pulse spacing until the communication link updates V_{pwm}. Since the two circuits are physically close to each other, the capacitances, currents, and therefore the resulting space between pulses should be well matched.

In order for the pulse stream regeneration circuit to have a constant pulse width output, τ, an off-chip reference clock is used to control a switched capacitor feedback arrangement where a charging current, I, is set to charge a capacitor, C, to a reference voltage, V_{REF}, in time τ. The details of this circuit are shown in Figure 4.33.

During time T_a the capacitor, C, is charged by the current, I. If the resulting voltage on the capacitor exceeds V_{REF} the output of the comparator goes low, if it is less than V_{REF} the comparator output goes high. The switched capacitor arrangement controlled by the clock signals T_b and T_c adjusts the current by a small amount, increasing it when the

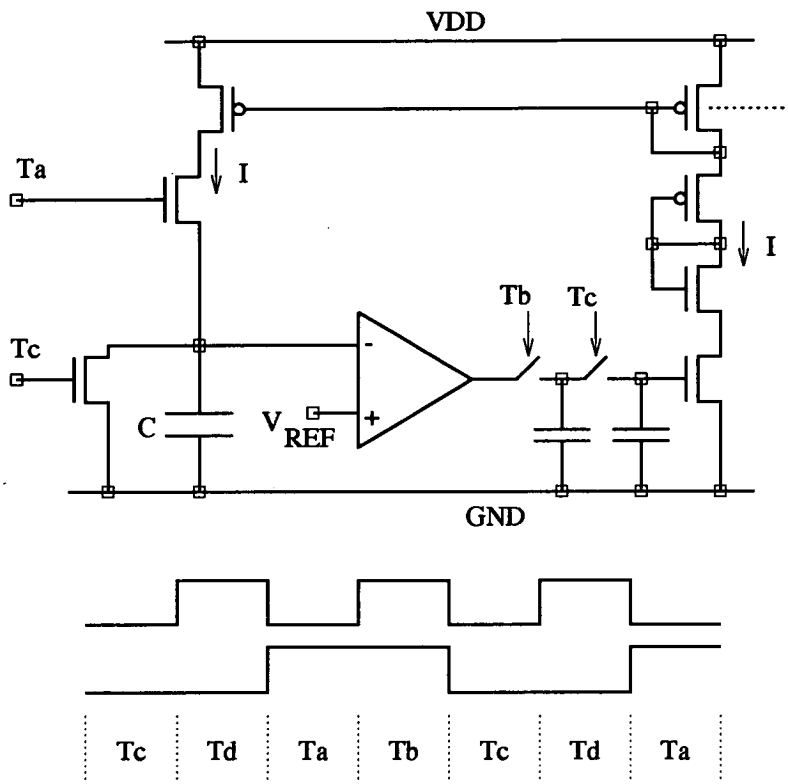


Figure 4.33 Pulse Stream Regeneration - Pulse Width Control Circuit

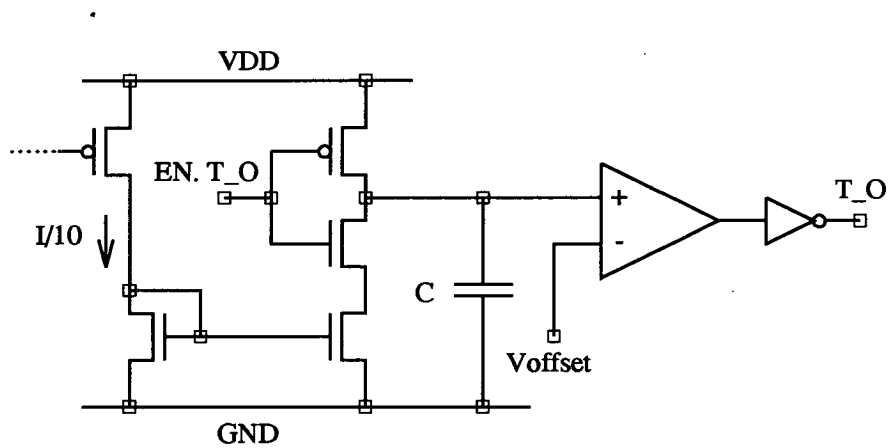


Figure 4.34 Time-out Mechanism : Circuit Details
En. T_O - Enable Time-out. T_O - Time-out.

comparator output is high and decreasing it when the comparator output is low. The current, I, therefore is adjusted over a number of clock periods so that it charges the capacitor, C, to V_{REF} in time T_a. This current is mirrored to the pulse stream regeneration circuit and used in similar circuits to control the output pulse width.

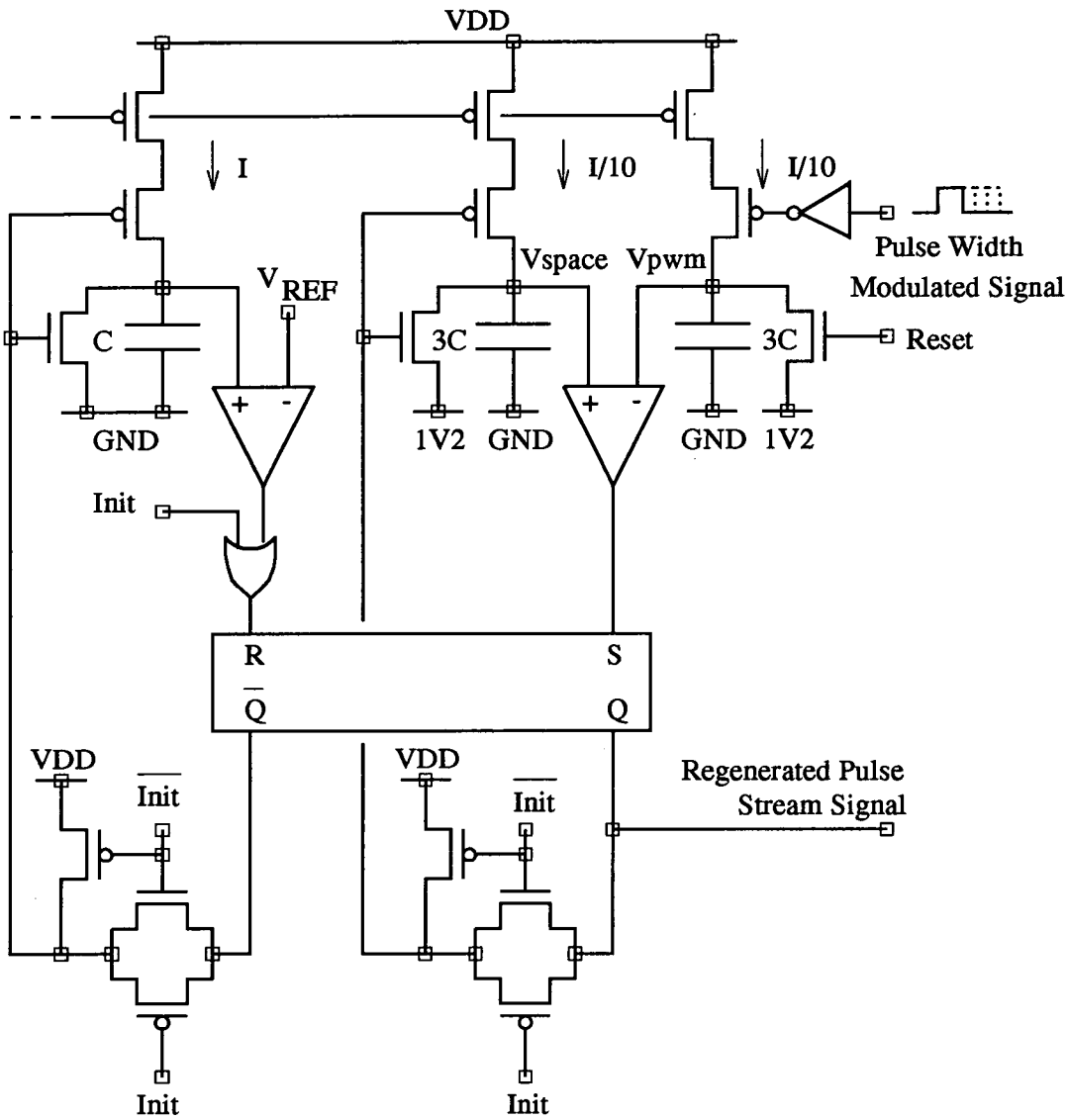


Figure 4.35 Pulse Stream Regeneration Circuit
Init = Pulse Width Modulated Signal **OR** Reset

The time-out mechanism uses the current, I, derived by the pulse width control circuit to control a capacitive discharge. By scaling I, and careful selection of discharge capacitor and comparator switching point, the circuit of Figure 4.34 may be designed to generate an output time-out signal, T_O, after the enable signal, EN. T_O, has been held for a period

greater than that of a 0.5% duty cycle.

The complete circuit diagram of the pulse stream regeneration circuit is shown in Figure 4.35. The circuit is initialised to a known state while the pulse width modulated signal is being stored as the voltage V_{pwm}. Once V_{pwm} has been stored, the voltage V_{space} begins to rise from the initial value of 1.2V. When V_{space} reaches V_{pwm}, the output of the comparator *sets* the S-R latch. This re-initialises V_{space} to 1.2V ready for the next pulse space measurement and starts the pulse width measurement. The current, I, set by the pulse width control circuit, charges the capacitor, C, until it reaches V_{REF}. The comparator then *resets* the S-R latch and the pulse space measurement restarts. The Q output of the S-R latch therefore forms the regenerated pulse stream.

In order to achieve the range of duty cycles, from 0.5% to 50%, the currents and capacitances in the pulse space control circuitry have been scaled as indicated in Figure 4.35.

4.8. Conclusion and Discussion

A number of pulse based circuits have been presented in this chapter to implement neural functions. In addition, support circuits have been developed in order to allow automatic circuit bias and inter-chip communication. Although HSPICE simulation results may indicate good circuit performance, the *acid test* of any circuit is how well it performs when implemented in VLSI. The skill and experience of an analogue VLSI designer lies in the thorough pre-fabrication circuit simulation and design that should anticipate any potential design problems. In practice, as will be shown in the following chapter, unexpected design problems often occur, their resolution resulting in a more mature and experienced analogue VLSI designer.

Chapter 5

Integrated Pulse Stream VLSI Results

This chapter introduces three VLSI devices that have been fabricated and tested from the pulse stream cell library described in Chapter 4. The chips are presented in chronological order, the first two being test devices used to evaluate different circuit design strategies and as such implement relatively small neural network structures. These test devices have allowed the considered choice of circuits from the cell library to implement a large pulse stream demonstrator chip entitled EPSILON (Edinburgh's Pulse Stream Implementation of a Learning Oriented Network). EPSILON is capable of operating in a variety of modes and of solving different neural network problems.

5.1. The VLSI Test Environment

A simplified schematic of the automated test environment used to exercise all the chips is shown in Figure 5.1.

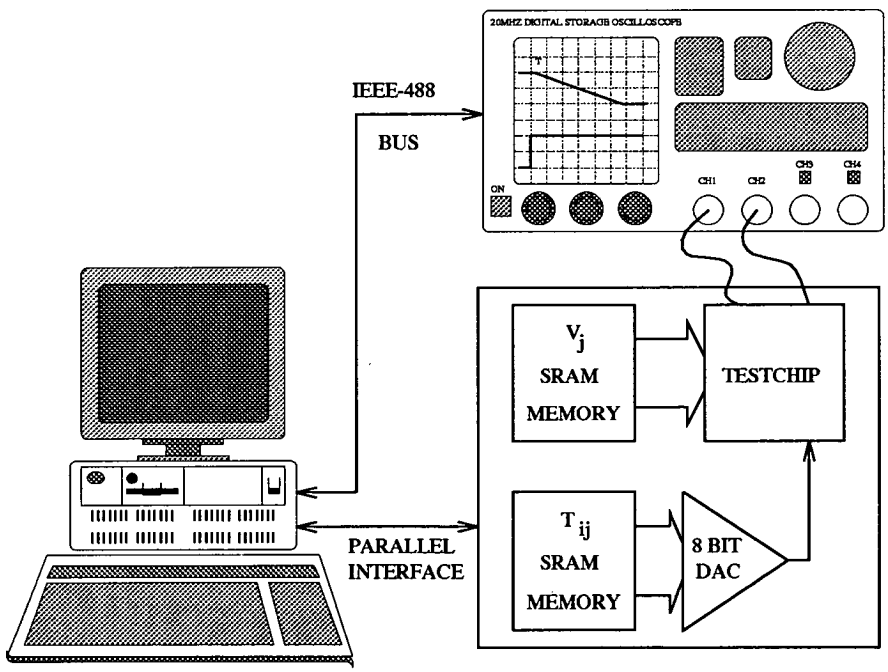


Figure 5.1 Automated Test Environment

Neural states and synaptic weights are loaded from the IBM PS/2 to a test board via a standard parallel interface card and stored locally in Static Random Access Memory (SRAM). The control circuitry continually cycles through the synaptic weight memory refreshing the on-chip capacitive weight storage nodes via an 8-bit Digital to Analogue Converter (DAC).

Neural state information described as a programmable duty cycle in the software environment on the IBM PS/2 is converted to bit patterns and loaded into the Neural State SRAM on the testboard. Each data output line from the Neural State SRAM represents a pulse stream input to the synapse array. By sequencing through all memory locations a pulse sequence can be fed into the synaptic array.

A 20MHz Digital Storage Oscilloscope has been used for data capture. The data captured by the oscilloscope is fed back to the IBM PS/2 over an IEEE-488 Interface for display, storage and manipulation.

The software environment developed for the IBM PS/2 allows algorithms to be developed to test the synaptic array on the test chip and to implement neural algorithms.

5.2. Test Chip A : A Pulse Stream Synapse using Pulse Width Modulation

The pulse stream synapse using pulse width modulation described in Chapter 4 has been fabricated on test chip A using ES2's 2μm Digital CMOS technology. This chip was the first designed at Edinburgh using purely analogue circuits for pulse stream neural networks and therefore proved the design route and the capability of the digital process for analogue circuit design.

The layout for the synapse circuit was performed by hand using the MAGIC design tool, each synapse occupying a silicon area of 174μm by 73μm. An array of 10×10 synapses and additional support circuitry for weight addressing and signal multiplexing was placed on chip. All digital circuitry was compiled using the SOLO 1400 silicon compiler. A photograph of a section of the synaptic array on the test chip is shown in Figure 5.2.

The performance of an individual synapse circuit as a two-quadrant multiplier has been evaluated by measuring the change in activity voltage at the output of the synaptic column for various weight voltage and input state duty cycles. These results have been obtained by measuring the rate of change of activity voltage with time from the data captured by the oscilloscope and are shown graphically in Figure 5.3. The linearity of the pulse multiplier circuit is illustrated in these results as the even spacing between lines for any given duty cycle.

This set of results shows that the circuit performs as predicted by the HSPICE simulation results of Figure 4.7 in Chapter 4. The performance of the circuit as a two-quadrant multiplier is excellent. The difference in rate of change of activity voltage between the

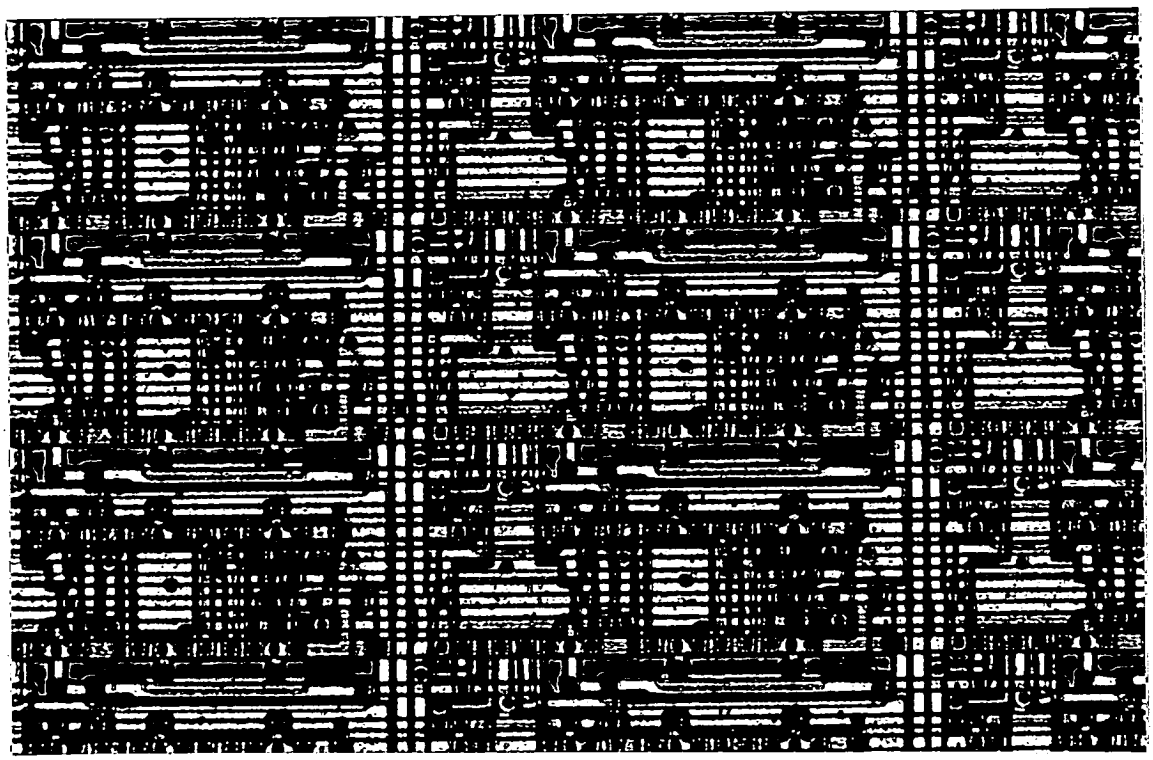


Figure 5.2 Chip Photograph of Synapse Circuit

HSPICE simulation results and those from VLSI is due to a voltage follower buffer circuit introducing an additional capacitive load for the synapse to drive.

All 100 synapses from one chip have been characterised using the test environment by measuring the change in activity voltage for a range of duty cycle input values for five different weight values over the range $1V \rightarrow 3.4V$. The linear regression algorithm has been used to find the best fit line from the resulting set of data points. The results for a single weight voltage are tabulated in Table 5.1 showing the variation in circuit performance across chip. This set of results is typical of any chosen weight value. The minimum, mean and maximum values of activity change for all measured values of duty cycle are tabulated. The variation of these minimum and maximum values from the mean are expressed for each value of duty cycle as a percentage of the mean activity change for 50% duty cycle.

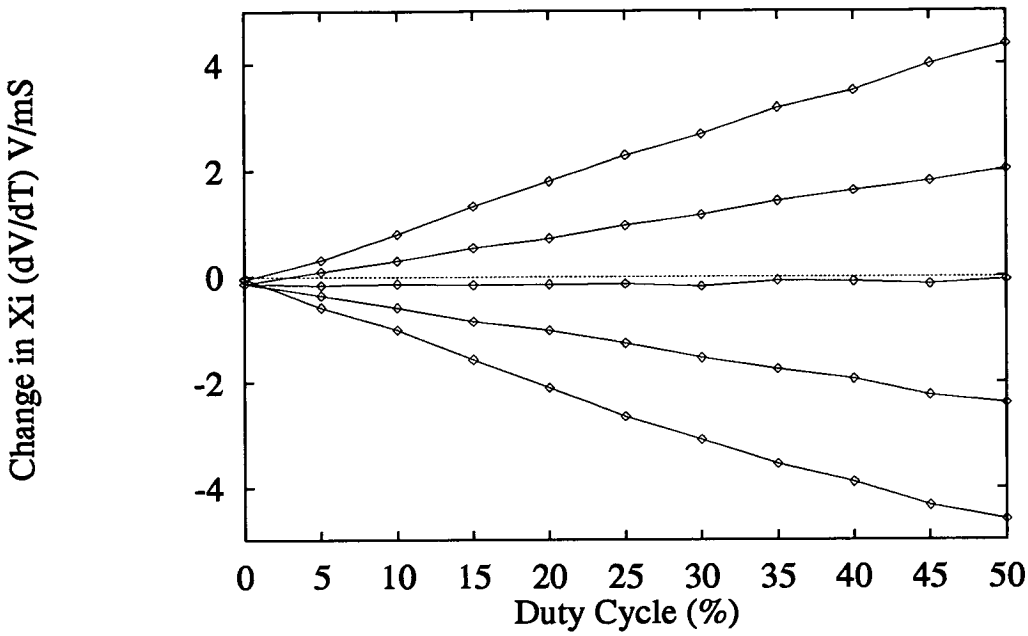


Figure 5.3 Measured Multiplier Characteristic of a Single Pulse Width Modulation Synapse. Weight Values 1V, 1V6, 2V2, 2V8 and 3V4. Duty Cycle Steps of 5%

Table 5.1 demonstrates that although an individual synapse circuit can perform as predicted by the HSPICE simulation given in Chapter 4, the variation in process parameters across the chip causes significant variation in circuit performance. Variation of this magnitude may present problems for the system designer if they cannot be compensated for by learning algorithms. Since these variations will vary between devices it may be necessary to have the chip in the learning loop to allow the learning procedure to compensate for individual chip variations.

5.2.1. Capacitive Weight Storage Leakage

An approximate measure of the leakage currents from the storage node has been made indirectly using the following technique. The weight voltage is set to the maximum excitatory value and the weight on an individual synapse updated. The weight refresh mechanism is then disabled and a pulse stream input is applied to the synapse and the activity voltage observed. As the weight voltage decays, and hence the weight value goes from being excitatory to being inhibitory, the response from the synapse becomes less. By measuring the time taken for the output to change from being excitatory to having zero

Synapse Circuit Results					
Duty Cycle (%)	Change in VX_i V/mS			Variation	Std Dev
	Min	Mean	Max		
0	-0.18	-0.05	0.57	-2.3%/+9.2%	±0.92%
5	0.23	0.51	0.81	-5.0%/+5.3%	±2.15%
10	0.59	1.14	1.73	-9.8%/+10.5%	±4.51%
15	1.12	1.79	2.50	-11.5%/+14.4%	±5.74%
20	1.49	2.33	3.33	-14.9%/+17.8%	±7.01%
25	1.88	2.88	4.14	-17.8%/+22.4%	±8.58%
30	2.11	3.42	4.81	-23.3%/+24.7%	±10.18%
35	2.53	4.04	5.95	-26.8%/+33.9%	±12.41%
40	2.76	4.48	6.49	-30.6%/+35.7%	±14.30%
45	3.11	5.10	7.31	-35.3%/+39.2%	±16.46%
50	3.38	5.63	8.16	-40.0%/+44.9%	±18.06%

Table 5.1 Pulse Width Modulation Synapse Circuit Performance Variation
$VT_{ij} = 3.4V$

response, an approximate measure of the leakage currents can be calculated.

The measured time from disabling the refresh mechanism to a zero activity response was 360 seconds. If we assume a constant current discharge, then

$$I = 0.75 \times 10^{-12} \times \frac{1.2}{360} = 2.5fA. \quad (5.1)$$

This measured current indicates the over-specification of the weight storage circuit described in section 4.1.

5.2.2. Pattern Associator Network

In order to demonstrate the principle of a learning algorithm compensating for circuit variation a simple pattern associator network [80] has been implemented using test chip A.

The pattern associator network is one of the most basic network architectures and has been used widely in distributed memory modelling. This architecture has been chosen to demonstrate the operation of an array of pulse stream synapses in a small network using a local learning rule. Both the Hebb Rule[81] and the Delta Rule [82] have been used to train such networks, however, since the Delta Rule is related to the Perceptron

convergence procedure of Rosenblatt[83], it has been chosen for this demonstration.

The pattern associator consists of a set of input neurons driving an array of synapses connected to output neurons and learns associations between input patterns and output patterns. In this demonstration a set of 9 input neurons drives an array of 36 synapses connected to 4 output neurons. The network is to be trained to associate the input pattern A with the output of neuron 0, input pattern B with the output of neuron 1 and so on. The training patterns are shown as the first four inputs to the network in Figure 5.4. If the nine inputs are arranged as a 3 x 3 array, as illustrated in Figure 5.4, then the ' inputs can be visualised as two diagonal lines, a horizontal line and a vertical line.

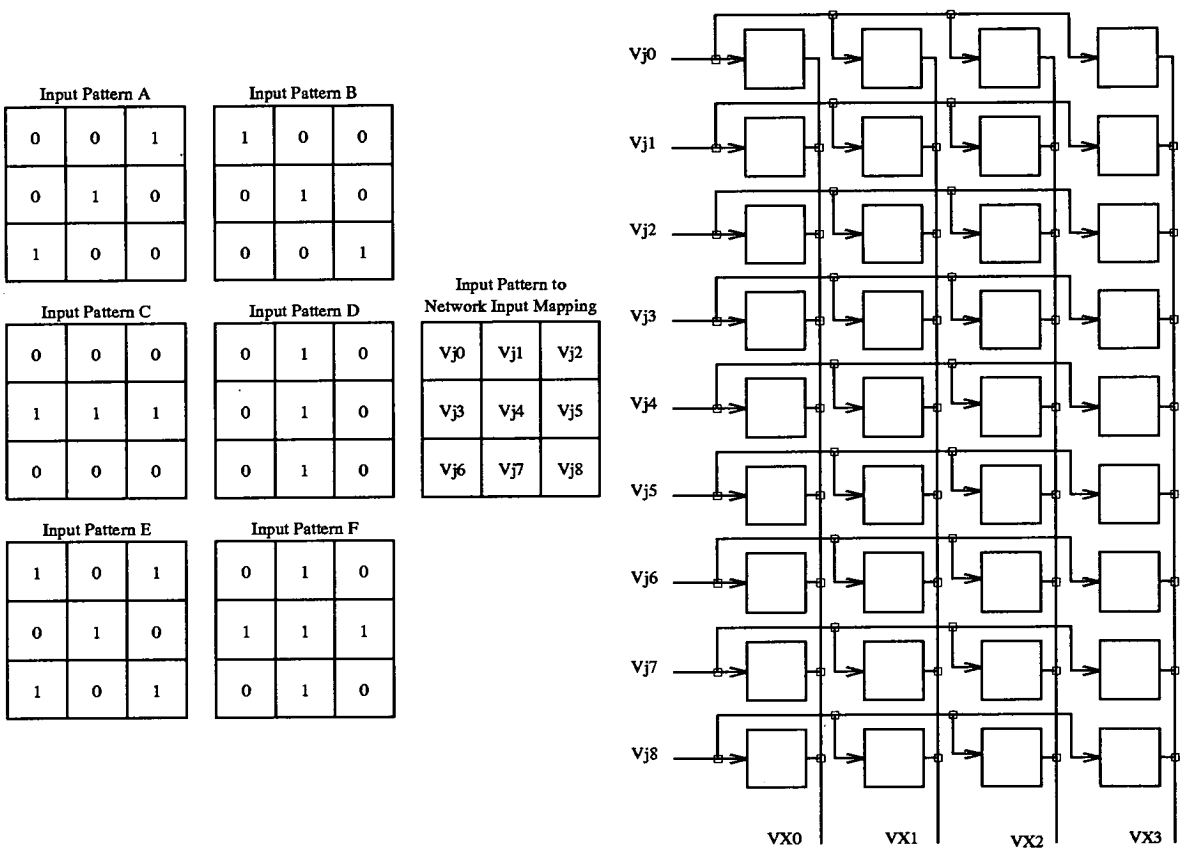


Figure 5.4 Input patterns illustrated on a 3×3 grid, mapping of grid points onto network inputs and block diagram of network showing inputs and activity voltage outputs. Each Square represents one pulse stream synapse circuit.

The Delta Rule, in its simplest form can be written

$$\Delta w_{ij} = \varepsilon \times e_i \times v_j \tag{5.2}$$

Delta Rule Results										
Input Patterns						Weight Voltages				Syn-
A	B	C	D	E	F	Neuron 0	Neuron 1	Neuron 2	Neuron 3	apse
0	1	0	0	1	0	-0.54	0.49	-0.46	-0.59	0
0	0	0	1	0	1	-0.42	-0.28	-0.39	1.00	1
1	0	0	0	1	0	1.01	-0.29	-0.27	-0.53	2
0	0	1	0	0	1	-0.35	-0.36	0.79	-0.54	3
1	1	1	1	1	1	-0.32	-0.45	-0.34	-0.68	4
0	0	1	0	0	1	-0.35	-0.36	0.79	-0.54	5
1	0	0	0	1	0	1.01	-0.29	-0.27	-0.53	6
0	0	0	1	0	1	-0.42	-0.28	-0.39	1.00	7
0	1	0	0	1	0	-0.54	0.49	-0.46	-0.59	8
Change in VX_i in V/mS					A	3.91	-3.13	-2.73	-4.3	
					B	-14.1	3.9	-6.25	-4.7	
					C	-4.3	-12.1	3.13	-6.25	
					D	-14.5	-2.7	-15.6	3.91	
					E	2.73	2.34	-12.5	-2.73	
					F	-7.42	-17.97	1.56	2.73	

Table 5.2 Delta Rule Results : Training patterns are Inputs A → D.
Weights are expressed as a Voltage in the range −1.2V → 1.2V.
Input States are of duty cycle 0% or 25%

where e_i, the error for unit i, is given by

$$e_i = t_i - a_i \tag{5.3}$$

the difference between the training input to unit i and its obtained activation.

The synapse array is initialised with a set of approximately zero weights. Each input pattern is applied to the synapse array in turn and the output of the first column of synapses is observed. The binary input has been represented by a 25% duty cycle for a "1" input and a 0% duty cycle for a "0" input. The resulting change in activity value is used as a measure to indicate the output neuron switching on or off. A threshold value has to be exceeded before the output neuron can be said to be truly "1" or "0".

Since the first synapse column is to be trained to recognise the input pattern A, the target t_1 is "1" for the pattern A and "0" for all others. The resultant error e_1 is calculated and the weights in that column modified according to equation 5.2. In practice, the value of ε

is a function of time, decaying linearly to zero. The process is repeated for all output neurons until zero error is observed for all input/output combinations.

The resultant weights taken from the hardware demonstrator are displayed in Table 5.2. The weights are tabulated as voltages in the range $-1.2V \rightarrow 1.2V$. A positive voltage represents an excitatory weight and a negative voltage represents an inhibitory weight.

The output responses from each column of synapses after training, expressed as a rate of change of activity voltage in V/mS, are tabulated in Table 5.2 for each input pattern. When input patterns A $\rightarrow$ D are presented to the network after learning is completed, the responses to each of the inputs, expressed as a change in VX_i, are shown in outputs A $\rightarrow$ D. This clearly demonstrates that learning has been successfully completed. Input pattern E is a combination of patterns A and B and results in outputs 0 and 1 responding with a positive change in VX_i. Similarly input pattern F is a combination of patterns C and D and results in both outputs responding.

5.2.3. Test Chip A : Conclusions and Discussion

This device has proved successful in demonstrating correct analogue circuit functionality from circuits designed using the manufacturer's SPICE data. It has proved the storage capability of capacitors implemented using transistors as described in Chapter 4 and given a rough measure of leakage currents. In addition, it has illustrated the problems of process variation and quantitive measurements have been made.

A simple pattern associator network has been implemented with this chip and shown to be capable of compensating for variations in individual chips providing the chip is in the learning loop. It is important to note that while this result is interesting, it does not indicate that any level of process variation may be tolerated by such a network trained using error driven learning. This simple example has been restricted to binary input data and therefore does not exploit the complete functionality of the analogue synapse circuit.

While the problem of process variation in the synapse circuit design can be compensated for to some degree, it is desirable to minimise this variation. For this reason, the distributed feedback synapse circuit described in Chapter 4 and demonstrated working on Test Chip B is of great interest.

5.3. Test Chip B

The second test chip contains three distinct functional blocks. The first is the self-depleting neuron and pulse magnitude modulating synapse design configured as a 10×10 fully interconnected synapse array. The second is the distributed feedback synapse, integrator and pulse stream neuron circuit with associated phase lock loop, also organised in a 10×10 array. The third is the inter-chip communication strategy implemented with an

independent transmitting and receiving state machine and the pulse stream regeneration circuitry.

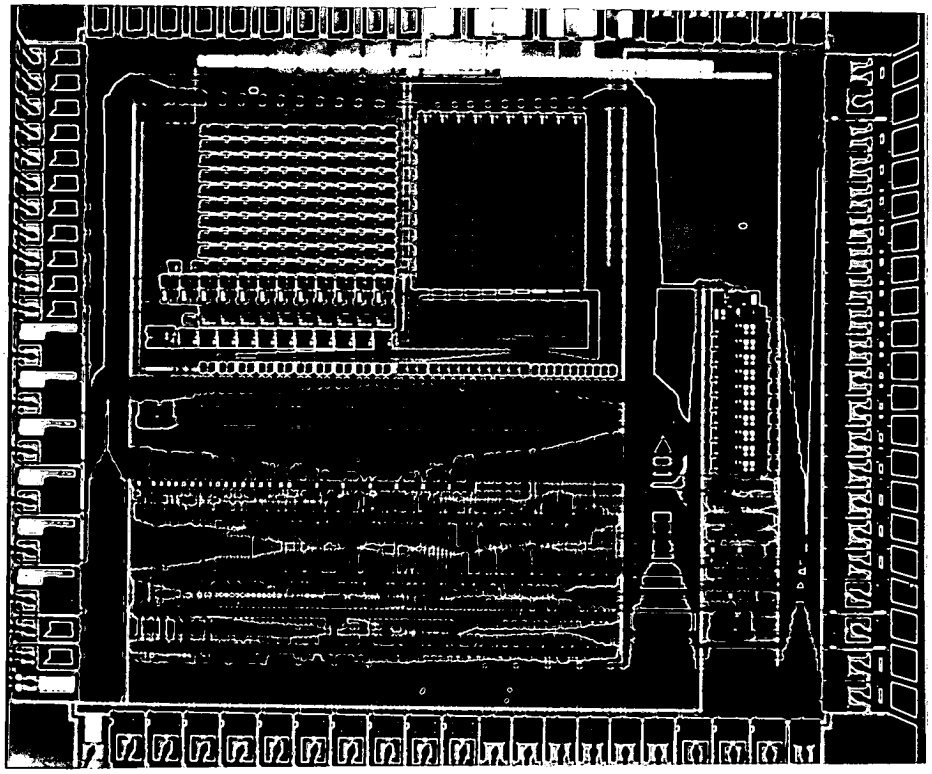


Figure 5.5 Chip Photograph of Test Chip B.

All digital support circuitry for weight addressing, signal multiplexing and state machine generation was implemented using the SOLO 1400 silicon compiler. ES2's 2μm Digital CMOS technology was used. A photograph of the chip is shown in Figure 5.5.

Figure 5.6 illustrates the floorplan of Test Chip B showing the placement of the individual circuit elements on the die.

The physical size of each of the circuit components used is given in Table 5.3. The initials in the table refer to the circuit designer.

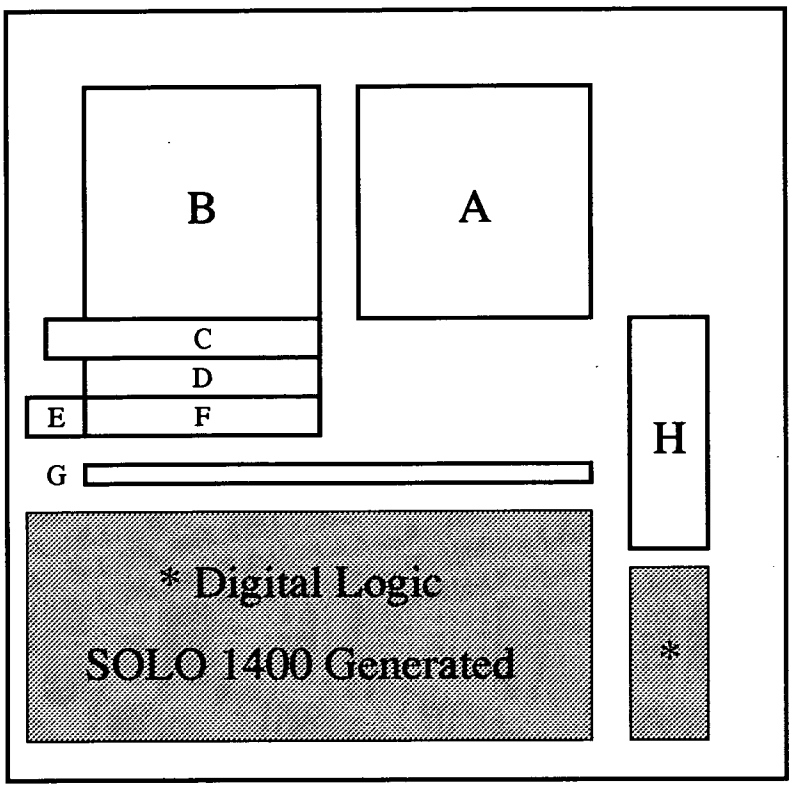


Figure 5.6 Chip Floorplan of Test Chip B.
A - Pulse Magnitude Modulating Synapse array and Self-Depleting Neurons
B - Distributed Feedback Synapse array. C - Operational amplifiers.
D - Integrators. E - Phase Frequency Detector and PLL Reference Neuron.
F - Pulse Stream Neurons (Fixed Gain). G - Signal Multiplexing Circuits.
H - Pulse Stream Regeneration Circuitry.

5.3.1. Pulse Magnitude Modulating Synapse and Self-Depleting Neuron

The transfer characteristics of the Pulse Magnitude Modulating Synapse and Self-Depleting Neuron circuits have been measured by Churcher[62] over 7 functional devices. In order to characterise the network, 5 synapses within any column were given a fixed weight value in order to provide a constant current drive to the self-depleting neuron. The remaining 5 synapses in the column were loaded with a variable weight voltage. A single duty cycle was fed into all input states. This configuration allowed the weight and input state values to be varied over their entire range. The mean characteristics are plotted in Figures 5.7 and 5.8.

Figures 5.7 illustrates the approximately linear response of the circuits for low input duty cycles up to values of approximately 15%, while Figure 5.8 illustrates a linear response

Cell Layout Sizes in ES2's 2μm CMOS		
Cell	**Size (Height × Width)**	**Designer**
Pulse Magnitude Modulating Synapse	130μm × 140μm	S.C.
Self-Depleting Neuron	100μm × 140μm	S.C.
Distributed Feedback Synapse	130μm × 165μm	D.J.B.
Operational Amplifier	250μm × 165μm	D.J.B.
Integrator	200μm × 165μm	D.J.B.
Phase Lock Mechanism (Phase Frequency Detector, Current Set and Reference Neuron)	173μm × 446μm	A.H.
Pulse Stream Neuron (Fixed Gain)	165μm × 165μm	A.H.
Pulse Stream Regeneration Circuitry	139μm × 628μm	A.H.

Table 5.3 Cell Layout Sizes and Designer.
S.C. - Stephen Churcher. D.J.B. - Donald J. Baxter. A.H. - Alister Hamilton.

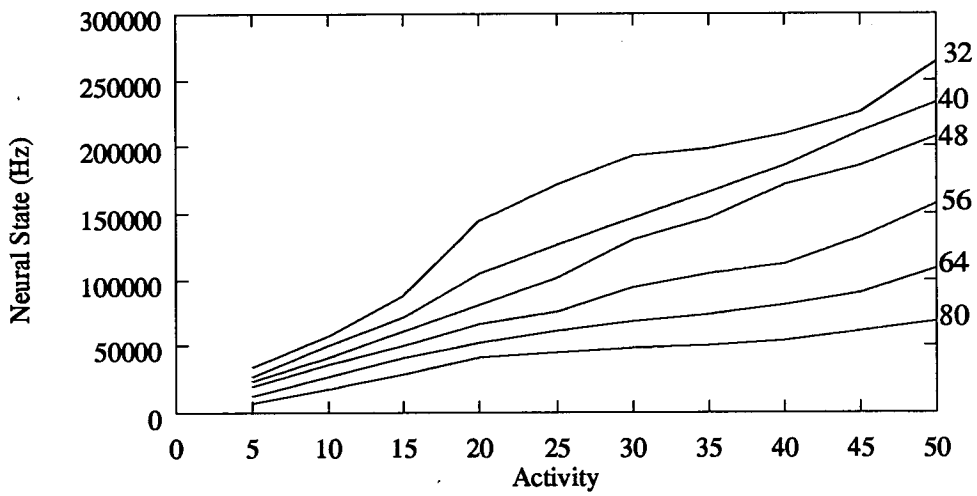


Figure 5.7 Pulse Magnitude Modulating Synapse and Self-Depleting Neuron.
Output States vs Input States (Activity 0 - 50%)
for different weight values (0 = 0V, 80 = 1.6V) averaged over 7 chips.

for high weight voltages. Erratic behaviour was observed at higher input state duty cycles and lower weight voltages.

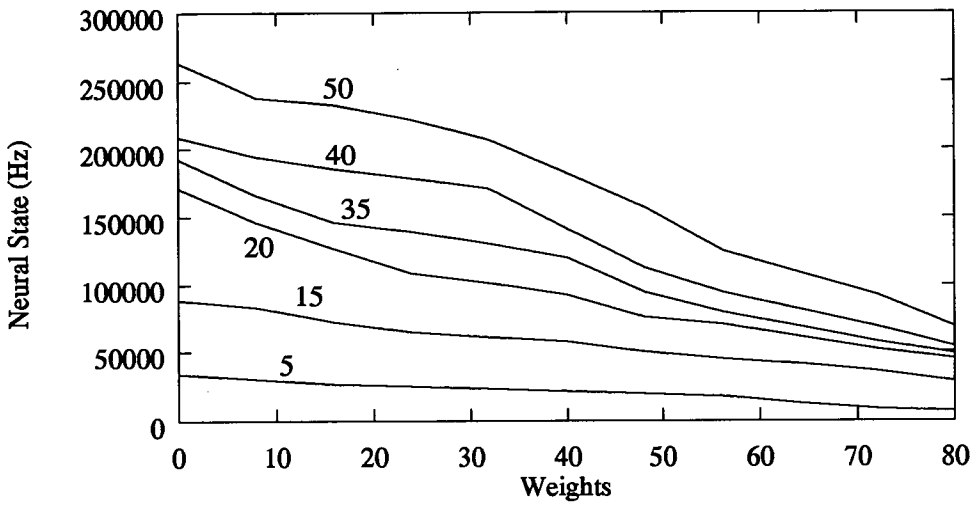


Figure 5.8 Pulse Magnitude Modulating Synapse and Self-Depleting Neuron.
Output States vs Weight Values (0 = 0V, 80 = 1.6V)
for different Input States (Activity 0 - 50%) averaged over 7 chips.

Large amounts of jitter from the output neurons was observed while taking these results. Churcher[62] attributes the cause of this jitter to power supply noise affecting the switching points of the comparator circuit within the self-depleting neuron circuit. This conclusion is borne out by the observation from Figure 5.7 that low pulse duty cycle inputs that have inherently fewer digital noise inducing edges give linear results while high duty cycles do not. This effect is accentuated by the experimental procedure where all input states and therefore switching edges are synchronised. Further evidence is provided by the results of Figure 5.8 where lower weight voltages that produce higher currents to be switched into the summing node give correspondingly worse results.

In addition to these problems, Churcher[62] required a lengthy and delicate set-up procedure in order to achieve correct circuit functionality.

Due to the above observations, the non-linear synapse transfer characteristic, the indeterminate activation function of the neuron and the resultant problems with mapping existing learning procedures onto this architecture, no further investigation has been made of this network type within the Edinburgh group.

5.3.2. Distributed Feedback Synapse, Operational Amplifier and Integrator Circuit

The array of transconductance multiplier synapses has been tested in both static and dynamic modes. Figure 5.9 shows static measurements of the output of the operational amplifier measured for a range of input weight voltages with the switch transistor, M3 in Figure 4.9 in Chapter 4, permanently switched on. The results have been taken from 8 chips, with 10 columns of 10 synapses measured on each chip. Figure 5.9 is directly comparable to the HSPICE simulation results given in Chapter 4, Figure 4.10. This comparison shows excellent matching between HSPICE and VLSI results.

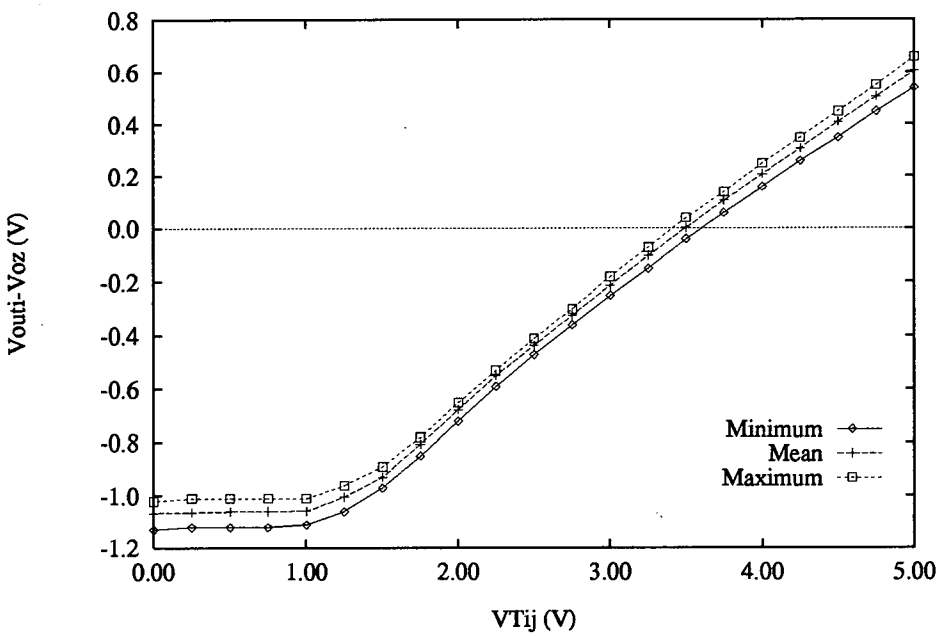


Figure 5.9 Distributed Feedback Synapse : Static VLSI Results

The two-quadrant multiplier performance measured over 8 chips, with 7 columns of 6 synapses is shown in the dynamic measurements of Figure 5.10. The integrator circuit is included in this test since *rate of change* of activity voltage is measured. The rate of change of activity output has been plotted against input state settings at intervals of 5% duty cycle. The mean value of each measurement over the test sample has been plotted together with the error bars showing ± one standard deviation.

The performance of the two-quadrant multiplier deviates from the expected results at the 0% duty cycle input, and at duty cycles of approximately 40% and greater. The high duty cycle deviations can be explained by the limitation of the test environment where duty cycle inputs of 40% and 45% could not be generated. The 0% duty cycle deviations are

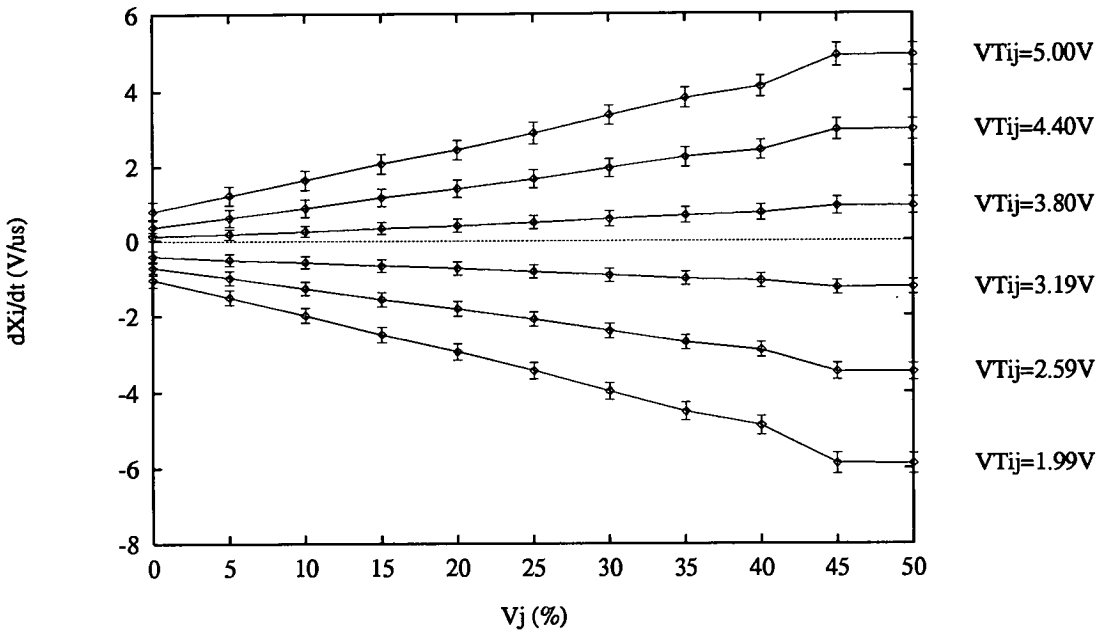


Figure 5.10 Distributed Feedback Synapse : Dynamic VLSI Results
Double Sweep

caused by an error introduced by the integrator circuit. Measurements of the integrator performance indicate an offset problem that has been found to be due to imprecise mirroring of currents between legs of the integrator circuit. This source of error is due to the layout of the integrator current mirror circuits causing transistor mismatches. This problem is therefore not a circuit problem as such, more an indication of the attention to detail required in layout for accurate analogue circuit performance. It was anticipated, and later results verify, that more careful layout of the integrator circuit could minimise this source of error. The maximum standard deviation for this set of results is ±6.1%.

5.3.3. Pulse Stream Neuron with Fixed Gain

The input activity voltage to output duty cycle characteristic was measured for 3 neurons on each of 8 devices. The results of these measurements are shown graphically in Figure 5.11. The voltage VMID was set to 2.5V. The maximum, minimum and mean measured duty cycles are plotted. The vertical error bar represents ± one standard deviation from the mean value and gives a graphical indication of the spread of results. This graph shows the predicted sigmoid activity voltage to duty cycle characteristic approximately centred on VMID. The relatively small variation in circuit performance across a sample

of 8 devices is also demonstrated. The spread of results is at a maximum at high values of VXi. This is due to imprecise current mirroring across chip due to variations in threshold voltage.

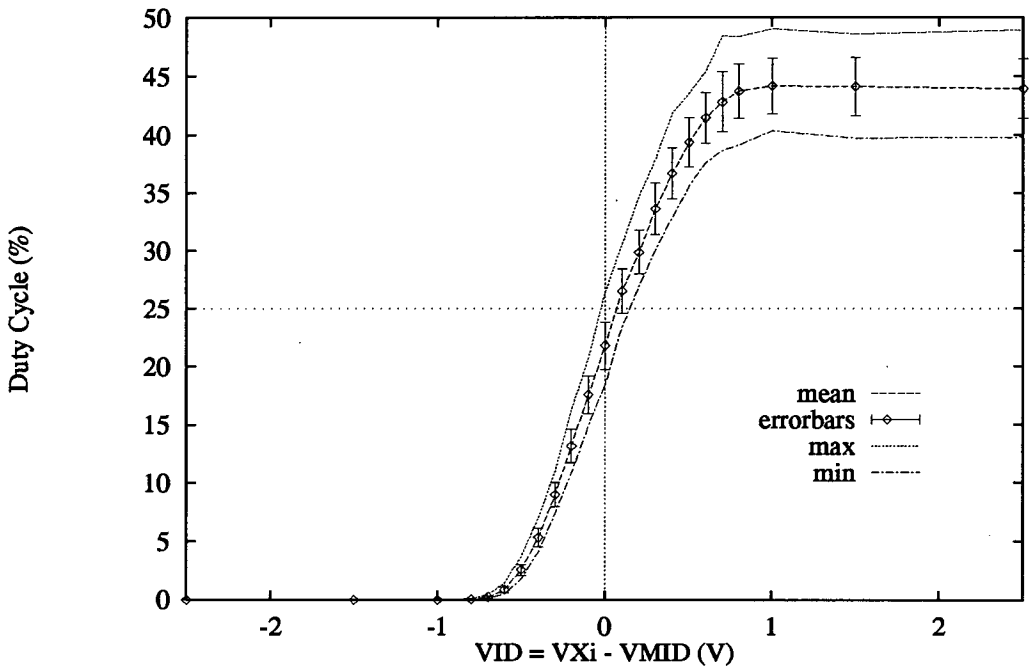


Figure 5.11 Fixed Gain Pulse Stream Neuron : VLSI Results

These results are summarised in Table 5.4. The minimum, mean and maximum values of duty cycle for selected values of VID are tabulated. The variation of these minimum and maximum values from the mean are expressed for each value of VID as a percentage of the mean duty cycle for VID = 2.5. The standard deviation of each set of points for each tabulated VID point is also expressed as a percentage of the mean duty cycle for VID = 2.5.

The gain of the transfer characteristic for each of the 24 neurons has been measured and compared against the theoretical value derived in Chapter 4 and presented here in Equation 5.4. The parameters used to derive the gain of the transfer characteristic have been taken from the HSPICE typical model from ES2 or set at design time and are given in Table 5.5.

$$\frac{\delta \text{DutyCycle}}{\delta V_{ID}} (V_{ID} = 0) = \frac{1}{8} \left[\frac{3\beta}{I_H} \right]^{1/2} \tag{5.4}$$

Fixed Gain Pulse Stream Neuron : VLSI Results Variation measured on 3 neurons from each of 8 Chips.					
VID	Duty Cycle (%)			Variation	Std Dev
	Min	Mean	Max		
-0.5	1.79	2.57	3.72	-1.78%/+2.60%	±1.09%
0.0	18.44	21.80	26.43	-7.63%/+10.53%	±4.65%
0.5	35.63	39.38	41.53	-8.54%/+9.51%	±4.88%
1.0	40.39	44.20	49.07	-8.66%/+11.07%	±5.42%
1.5	39.75	44.15	46.63	-10.0%/+10.14%	±5.65%
2.5	39.83	43.98	48.96	-9.42%/+11.33%	±5.78%

Table 5.4 Fixed Gain Pulse Stream Neuron : VLSI Results
Variation measured on 3 neurons from each of 8 Chips.

Transistor Parameters Defining β and I_H Value	
μ_0	$175 cm^2$/volt. seconds
C_{ox}	$863.9 \times 10^{-10} F/cm^2$
W	$4\mu m$
L	$10\mu m$
I_H	$2\mu A$

Table 5.5 Transistor Parameters defining β and I_H Value

A comparison of the gain results from theoretical, HSPICE simulation and measured VLSI data are presented in Table 5.6. This table shows good correspondence between the HSPICE simulation predictions of gain and the mean value measured from VLSI. The difference between the theoretical gain and the HSPICE and measured results can be explained by the simplified analysis performed in Chapter 4 where channel length modulation effects have been ignored.

The duty cycle for each neuron at VID = 0 compared to the theoretical value of 25% has been measured giving a mean offset of -3.2%. The standard deviation for this figure is ±2.04%. This offset is due to imprecise mirroring of currents across chip.

Pulse Stream Neuron : Gain Results			
Theoretical	HSPICE	Measured	
		Mean	Std Dev
37.65	45.2	44.79	±2.72

Table 5.6 Fixed Gain Pulse Stream Neuron : Comparison of Gain Measurements

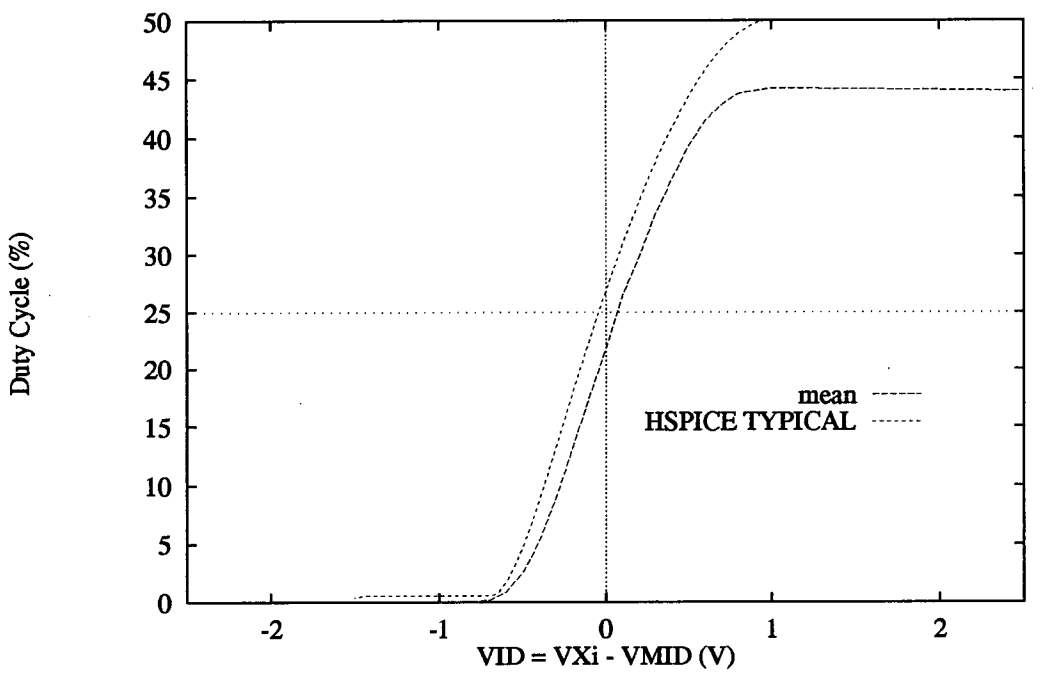


Figure 5.12 Comparison of HSPICE Simulation and Test Chip B Results

A comparison of the Pulse Stream Neuron transfer characteristic on Test Chip B and HSPICE simulation data is shown in Figure 5.12. The mean of the measured data from the 24 neurons is compared with HSPICE simulation results using typical process parameters. Figure 5.12 demonstrates the good correspondence between simulated and actual data, the main source of error in VLSI is due to the imprecise mirroring of currents across chip.

The operation of the phase lock loop is demonstrated in Figure 5.13 which shows data captured from a digital oscilloscope sampling at 100MHz. The bottom trace shows the

reference clock while the upper trace shows the output of the reference neuron. The two signals are locked together with an output pulse width of 1μS.

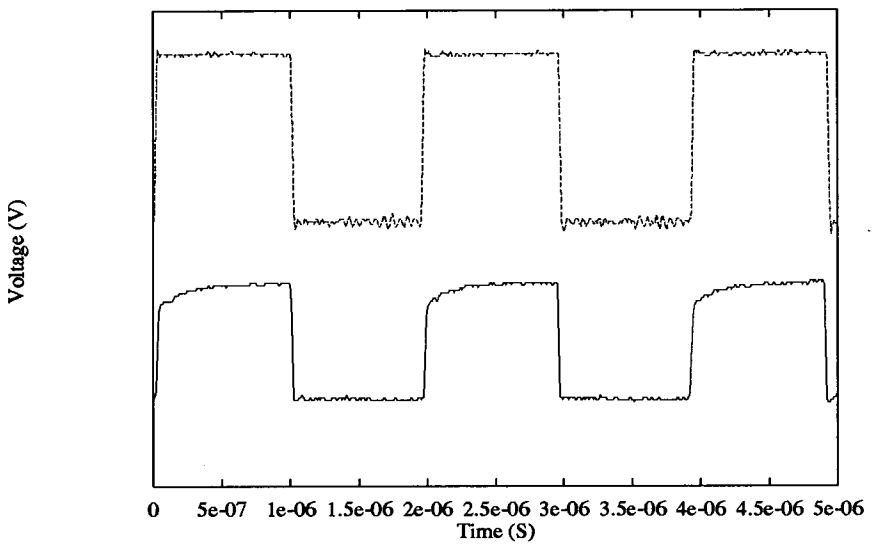


Figure 5.13 Oscilloscope Trace of External Clock Reference (Bottom Trace) and Pulse Stream Neuron Circuit (Top Trace) Phase Locked.

It was anticipated that physically adjacent neurons might lock to each other due to mechanisms such as coupling through power supply rails. Three adjacent neurons were used to test for lock up. While the centre neuron was allowed to oscillate at a predefined duty cycle, the inputs of the remaining two neurons were controlled by a linear ramp voltage. The centre neuron was observed during this process and no sign of lock up was observed.

5.3.4. Pulse Stream Network Operation

The oscilloscope photograph of Figure 5.14 illustrates the operation of the distributed feedback synapse, operational amplifier, integrator and pulse stream neuron circuit on test chip B. The top trace shows a pulse stream input, the second trace the integrator output and the third trace the resulting output pulse frequency from the pulse stream neuron.

The synapse array has been loaded with a uniform excitatory weight value and the input pulse stream causes the integrator output to ramp upwards. The resultant integrator output voltage causes the output neuron to switch on, eventually saturating at a duty cycle output of 50%.

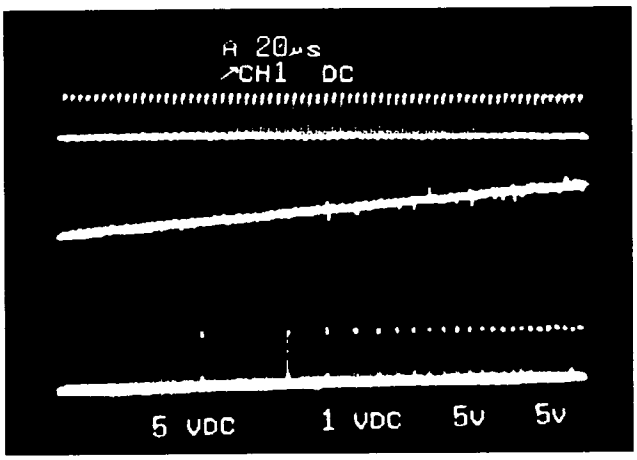


Figure 5.14 Pulse Stream Network Operation
Input pulse stream (Top Trace). Integrator activity (Second Trace).
Output pulse stream (Third Trace).

It is interesting to note at this stage that the "update algorithm" associated with this form is distinct from that of other analogue neural forms. Conventional networks express the neural activity x_i as the result of the synaptic summation of Equation 5.5.

$$x_i(t) = \left[\sum_{j=0}^{j=n-1} T_{ij} V_j(t) \right] \tag{5.5}$$

In the pulse stream network presented here, the result of each pulse arriving is to either add or subtract a small package of charge to the integration capacitor, and thus increment or decrement x_i. The update algorithm for the voltage x_i with respect to a single pulse is therefore:-

$$x_i(t + \Delta t) \approx x_i(t) + \delta \times \left[\sum_{j=0}^{j=n-1} T_{ij} V_j(t) \right] \tag{5.6}$$

where δ is controlled by the characteristics of the voltage integrator.

5.3.5. Inter-Chip Communication Scheme.

The operation of the inter-chip communication scheme has been tested as a mechanism for transferring data between chips and between a chip and a host computer port, in this case the IBM PS/2.

In order to test the handshake and data transfer mechanism a single chip had its transmission port connected to its receiver port using the three wire connection scheme. The transmitting and receiving state machines were clocked from separate clock sources to ensure independent operation. The results from this test are shown in Figure 5.15 where the output of a single neuron is shown in the top trace, RTT in the second, RTR in the third and the encoded signal, D, in the final trace. A clock frequency of 2MHz has been used for the state machines and the data transfer is completed in approximately 12μs for the input duty cycle of approximately 15%.

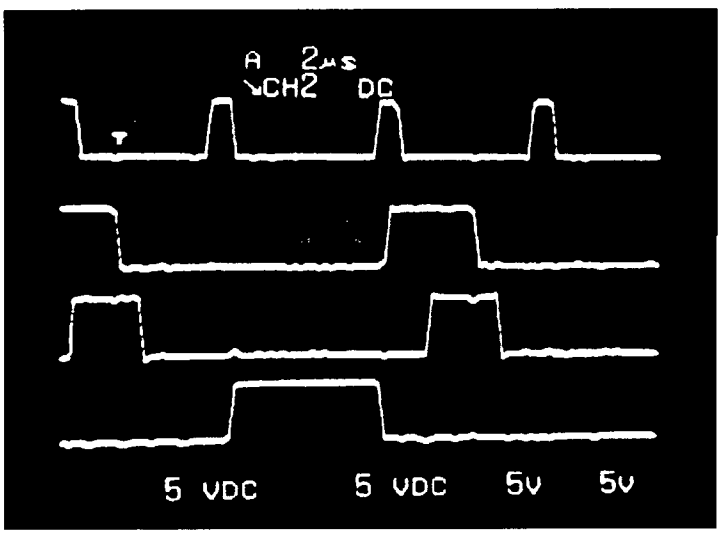


Figure 5.15 Inter-Chip Communications : VLSI Results
Top Oscilloscope Trace : Transmitting Neuron Output. Second Trace : request to transmit (RTT). Third Trace : ready to receive (RTR) from receiving chip. Fourth Trace : encoded data, D.

Figure 5.16 shows the transmitting state machine communicating to the IBM PS/2 which is a relatively slow device. The receiving state machine has been implemented as a software algorithm within the PS/2 and signals are fed to and from the computer over a standard parallel port. Communication proceeds at the speed of the slowest device, as in any self-timed system, in this case the data transfer operation now takes approximately 45μs. This demonstrates that the transmitting device is capable of sending data to a host computer via conventional parallel interface ports, using the same communication circuitry and protocols as developed for inter-chip communication.

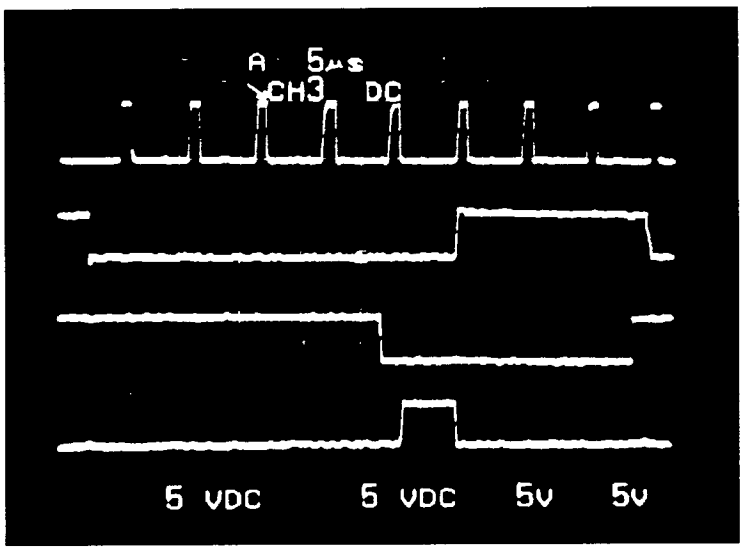


Figure 5.16 Chip to IBM PS/2 Communications : VLSI Results
Top Oscilloscope Trace : Transmitting Neuron Output. Second Trace : request to transmit (RTT). Third Trace : ready to receive (RTR) from IBM PS/2. Fourth Trace : encoded data, D.

5.3.5.1. Pulse Stream Regeneration

The pulse stream regeneration circuitry was found not to function correctly on test. The Pulse Stream Regeneration circuit was originally implemented without the initialisation circuitry controlled by the *Init* signal shown in the circuit diagram of Figure 4.35, Chapter 4. It is believed that a circuit initialisation problem was the cause of improper functionality causing the SR latch to be in an indeterminate state when the *Reset* and pulse width modulated signal arrived at the circuit input.

The addition of the initialisation circuitry avoids this indeterminate condition. This later circuit addition was simulated and the circuit layout was generated. At present this revised circuit has not been implemented in silicon.

While full circuit functionality was not observed, the pulse width outputs of the pulse stream regeneration circuit were of the correct width but of incorrect spacing. This observation indicates that the pulse width control circuitry was functioning correctly.

In addition the transmitting state machine was observed to time-out correctly at low duty cycle inputs indicating that the analogue time-out circuitry illustrated in Figure 4.34 and the time-out states within the transmitting state machine of Figure 4.30 were functioning

correctly.

5.3.6. Test Chip B : Conclusions and Discussion

This device has provided valuable test results despite the functional problems outlined in the previous sections. In particular, the characterisation of the Distributed Feedback Synapse as a circuit to minimise process variation provides data for a direct comparison with the Pulse Stream Synapse using Pulse Width Modulation implemented on Test Chip A. The comparison of worst case standard deviation in circuit performance of ±18.1% for pulse stream synapse and ±6.1% for the Transconductance Multiplier favours the latter circuit. In general, the correspondence between HSPICE simulated results and those obtained from functional VLSI is excellent, and where deviation exists, methods for overcoming the problems have been suggested.

The results from the pulse stream neuron with fixed gain deviate from HSPICE simulations due to errors in current mirroring. Careful attention to the choice of transistor geometries and operating regions should enable improved performance. The use of phase lock loop techniques has proved successful for controlling the neuron pulse width output despite the lack of HSPICE simulation results for the reasons outlined in Chapter 4. This result gives confidence for their use as a gain control mechanism for the variable gain neuron circuit.

While the results from the inter-chip communication system have proved only partially successful, the handshake mechanism has been successfully demonstrated for both inter-chip and chip to host computer modes. While no further work has been carried out by the author in this area at the time of writing, it remains an area of interest requiring further investigation. For example, transmission speed improvement, the effects of the inter-communication strategy on network dynamics, the optimal number of neurons occupying a communication channel and the number of communication channels required for a given network remain areas of great interest.

The pulse magnitude modulating synapse and self-depleting neuron circuit have proved of limited interest, however, they serve as a reminder of the attention to detail required in order to achieve robust analogue circuit performance.

5.4. EPSILON : Choice of Cells

The results from the test chips described earlier provide the detailed data used to determine the optimal choice of circuit cells to be used on EPSILON. The confidence gained from the good correlation between HSPICE simulation results and the functional VLSI test devices enabled the design team to include some circuit ideas that had not previously been proved on the test devices.

It is clear, from the results presented in this chapter from the three synapse circuits which were implemented, that the transconductance multiplier circuit should be used to implement the synapse function on EPSILON. While this circuit has good linearity and improved process tolerance over the other circuits tested, it is also capable of operating in either pulse frequency modulated or pulse width input modes. In addition to these digital input modes, a third analogue input mode has been added using the pulse width neuron design described in Chapter 4. The resulting neural input state conditioning circuitry is shown in Figure 5.17. The *Select Input Mode* signal determines digital or analogue input mode.

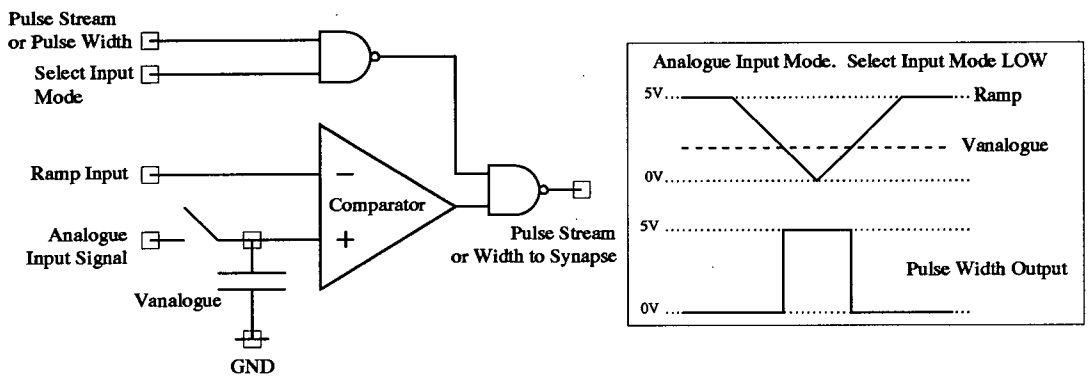


Figure 5.17 EPSILON : Neural State Input Conditioning Circuitry

The analogue input mode has been designed to simplify the interface between a host computer and EPSILON. Since the loading of analogue weights is not time critical, the interface to a computer parallel port could simply be a DAC and analogue switches. Alternatively, the output of an analogue transducer could be directly interfaced to EPSILON. This compares with relatively complex digital hardware circuitry required for pulse width or pulse stream inputs.

In order to allow cascading of EPSILON chips, and a choice of output operating modes, both the pulse stream neuron and the pulse width neuron have been implemented as output neurons. The use of the pulse width neuron offers a wider range of neuron output characteristics than that available from the pulse stream neuron. The flexibility of the pulse stream neuron has been increased by the implementation of the variable gain circuit described in Chapter 4. Direct analogue output of neural activity has not been implemented due to the requirement of adding relatively large and power hungry operational amplifiers to buffer each neural activity voltage.

Both the analogue input mode and pulse width input and output modes operate in a time stepped manner. Inputs are applied, the system is allowed to settle and the outputs are read. The use of pulse stream input and output modes allows continuous asynchronous operation of the network suitable, for example, for dynamic feedback networks.

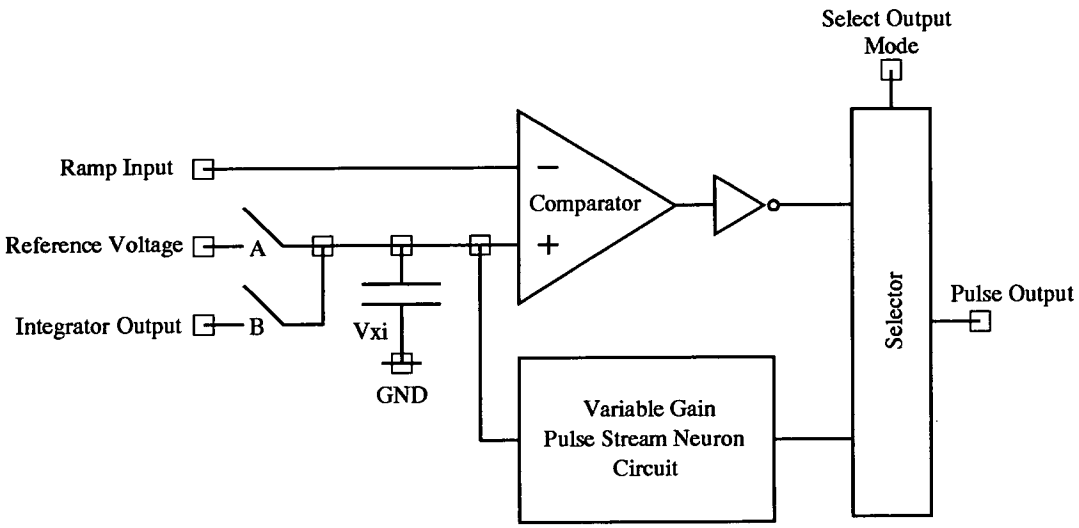


Figure 5.18 EPSILON : Neural State Output Conditioning Circuitry

The signal conditioning circuitry required at the neuron output is illustrated in Figure 5.18. In normal operation, the integrating capacitor is reset to an initial value and the integrator either charges or discharges the capacitor as a result of calculations performed within the synaptic array. In pulse stream mode, the activity capacitor feeds the pulse stream neuron directly, and the result of the computation appears as a pulse stream output. In pulse width mode, an external ramp signal is applied to the comparator circuit and a pulse width signal is output.

The gain of the sigmoid transfer characteristic of the pulse stream neuron is set by a phase lock loop as described in Chapter 4. Synaptic weight addressing is performed by a custom x and y shift register driven by a two-phase clock.

EPSILON has been implemented using 1.5μm CMOS. This change in technology is obviously not desirable since all testing has been undertaken using 2.0μm technology, however the change was imposed due to ES2 phasing out the 2.0μm process. The 1.5μm technology forced a further evaluation of the circuits used and modifications to such variables as transistor geometries in order to accomodate the poorer quality analogue design environment provided by the new process.

5.5. EPSILON : The Choice of Chip Architecture

A single layer architecture with 120 input units and 30 output units allows the configuration of a variety of network types. This configuration allowed optimal use of the silicon area funding permitted. In order to restrict the size of the chip pad count, neural input states are multiplexed onto chips in 4 banks of 30 inputs.

To create a network requiring more input units would require operating several EPSILON devices with linear input and output units and adding and scaling the outputs. Networks with arbitrary widths can be created by placing several EPSILON chips in parallel. Multilayer networks can be implemented by cascading EPSILON chips. Alternatively EPSILON can be used in a paged manner.

EPSILON Chip Specification	
Number of V_j Input Pins	30
Number of V_j Inputs to Synapse Array	120, Multiplexed in groups of 30
Input Modes	Analogue, Pulse Width and Pulse Stream
Number of V_j Outputs	30
Output Modes	Pulse Width and Pulse Stream
Number of Synapses	3600
Number of Weight Load Channels	2
Weight Load Time	3.6mS
Weight Storage	Voltage on a Capacitor, off-chip refresh
Connections per Second (cps)	*Minimum* 18Mcps, *Maximum* 360Mcps
Technology	1.5μm Digital CMOS
Die Size	9.5mm x 10.1mm
Maximum Power Consumption	350mW

Table 5.7 EPSILON Chip Specification

The specification for the EPSILON chip is given in Table 5.7. The main restriction on computation speed is the limitation in the performance of the operational amplifier controlling the feedback in the synapse array. This places a limit of 1μs on the minimum pulse width input to the system and therefore defines the maximum pulse frequency of a pulse stream input to be 500kHz. If we assume a range of input duty cycles between 0.5% and 50%, then a time of 200μs is required to represent the lowest duty cycle. If it is assumed that the support circuitry generating the pulse width is capable of changing the pulse width in steps of 0.1μs, then a 1% resolution can be achieved with a 10μs pulse. These considerations define the computation time of the system expressed in connections per second in Table 5.7. The use of analogue or pulse width input mode and pulse width

output mode therefore provide the fastest computation time.

5.6. EPSILON : VLSI Results

Table 5.8 gives the physical size of each of the circuit components used and the initials of the designer responsible for each circuit.

EPSILON Cell Layout Sizes in ES2's 1.5μm CMOS		
Cell	**Size (Height × Width)**	**Designer**
Variable Gain Pulse Stream Neuron	100μm × 200μm	A.H.
Phase Lock Loop	100μm × 650μm	A.H.
Pulse Width Input Neuron (Double)	100μm × 336μm	S.C.
Pulse Width Output Neuron	116μm × 86μm	S.C.
X-Shift Register	104μm × 200μm	S.C.
Y-Shift Register (Double)	100μm × 268μm	S.C.
Distributed Feedback Synapse	100μm × 100μm	D.J.B.
Operational Amplifier	262μm × 200μm	D.J.B.
Integrator Circuits	200μm × 200μm	D.J.B.

Table 5.8 EPSILON Cell Layout Sizes and Designer
S.C. - Stephen Churcher. D.J.B. - Donald J. Baxter. A.H. - Alister Hamilton.

The physical layout of the various components of EPSILON are shown in the floorplan of Figure 5.19. Figure 5.20 shows a photograph of the EPSILON device.

5.6.1. Distributed Feedback Synapse, Operational Amplifier and Integrator Circuit

The testing of the distributed feedback synapse and operational amplifier has been performed using a static test where the switching transistor, M3 in the circuit diagram in Figure 4.9 is permanently switched on. The measurements have been taken from a single column of 120 synapses on each of 6 chips. Again, these results are linear and largely process invariant. Also plotted on this graph is the output of the operational amplifier V_{outiZ} for the corresponding weight voltages with all M3 transistors switched off. Ideally V_{outiZ} should be zero for all weight values, but these results show a decrease over the active voltage range as the weight increases. This is due to power supply voltage drop along the power bus feeding the synapse array due to the resistance of the metal track. This observation has implications for the multiplier performance which are further explored in the following paragraphs on dynamic measurements.

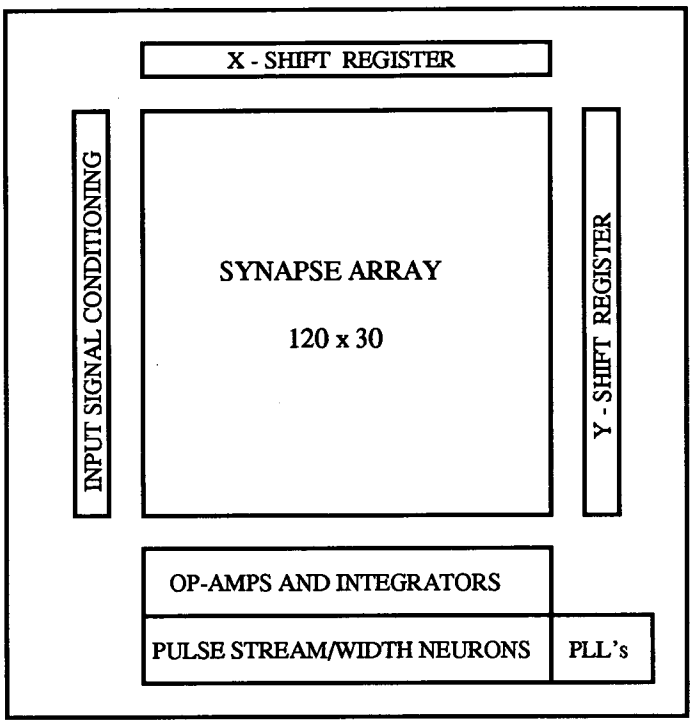


Figure 5.19 Floorplan of EPSILON Chip Core. PLL : Phase Lock Loop.

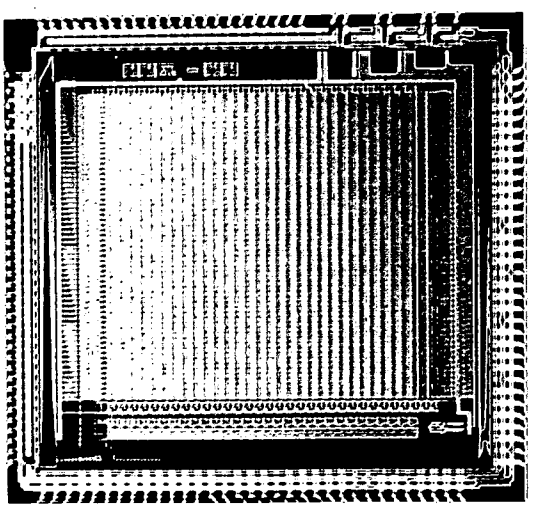


Figure 5.20 Photograph of EPSILON Chip

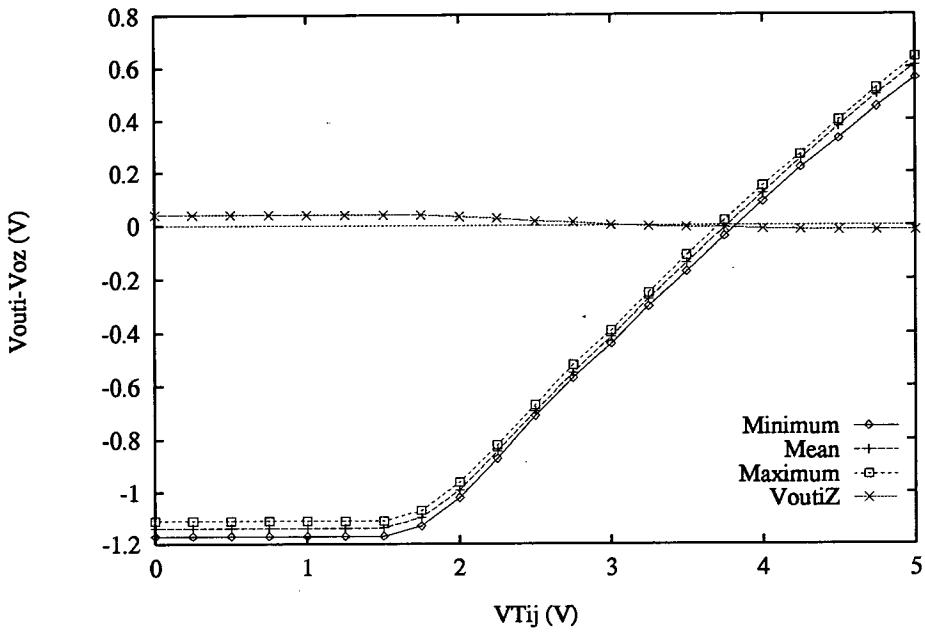


Figure 5.21 EPSILON : Distributed Feedback Synapse and Operational Amplifier : Static VLSI Results

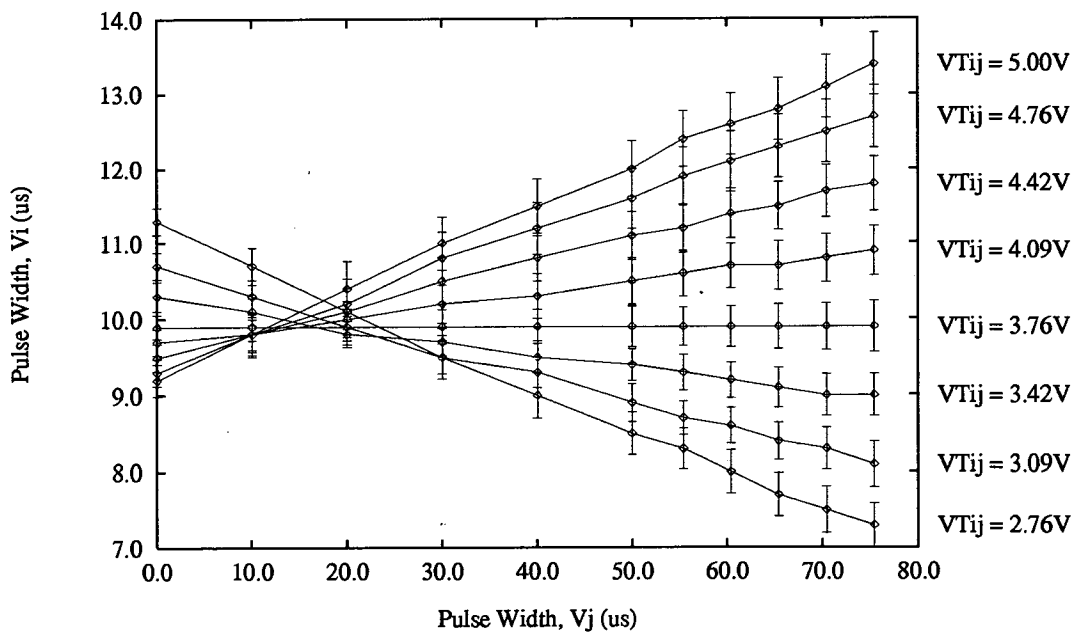


Figure 5.22 EPSILON : Distributed Feedback Synapse, Operational Amplifier and Integrator Circuit Two-Quadrant Multiplication : Dynamic VLSI Results

Operation of the distributed feedback synapse as a two-quadrant multiplier including the integrator circuit in dynamic mode provides the results depicted in Figure 5.22. The measurements have been made over an extended pulse width output using 12 synapses. Measurements were taken on 6 chips, 29 synaptic columns from each chip, and 10 sets of measurements from each column. The mean of these results and error bars representing ± one standard deviation have been plotted. The maximum standard deviation for this set of results is ±12.4%.

In this test, pulse width input and output signals have been used. The use of pulse width modulated input signals overcomes the limitation on the resolution of pulse stream generation inherent in the test environment. Since the output comparator circuits are highly process invariant, the use of a linear ramp signal on the output will produce a pulse width directly related to the output of the voltage integrator.

Prior to the application of the pulse width input to the network the activity capacitor was reset to 2.5V. The multiplier zero point is therefore at the point where there is no net activity change and results in a 10μs output pulse. The effect of the power supply variation with weight voltage can be seen from these results in the mismatch of the multiplier characteristic most notable at the zero point. This result demonstrates the sensitivity of the distributed feedback synapse to power supply variations.

There is a relative deterioration in performance of the synapse circuit between Test Chip B and EPSILON as measured by the maximum standard deviation of the dynamic measurements. This is not only as a result of moving from 2.0μm to 1.5μm CMOS, a retrograde step in terms of analogue circuit performance, it is also a function of the power supply drop across the synaptic array. The latter power supply variation may be reduced by using wider power track supplying the synaptic array. However, it should be noted here that the offset problems due to the integrator performance of Test Chip B have been alleviated on EPSILON.

5.6.2. Variable Gain Pulse Stream Neuron

The input activity voltage to output duty cycle characteristic was measured for all 30 neurons on each of 16 devices. The results of these measurements are shown graphically in Figures 5.23 and 5.24 for high and low gains respectively. In each case the phase lock loop was used to control the gain setting and the voltage VMID was set to 2.5V. The maximum, minimum and mean measured duty cycles are plotted. The vertical error bar represents ± one standard deviation from the mean value and gives a graphical indication of the spread of results.

Improved performance of the current mirrors on EPSILON in comparison to Test Chip B has resulted in the mean duty cycle more closely approaching 50%. Consequently, the

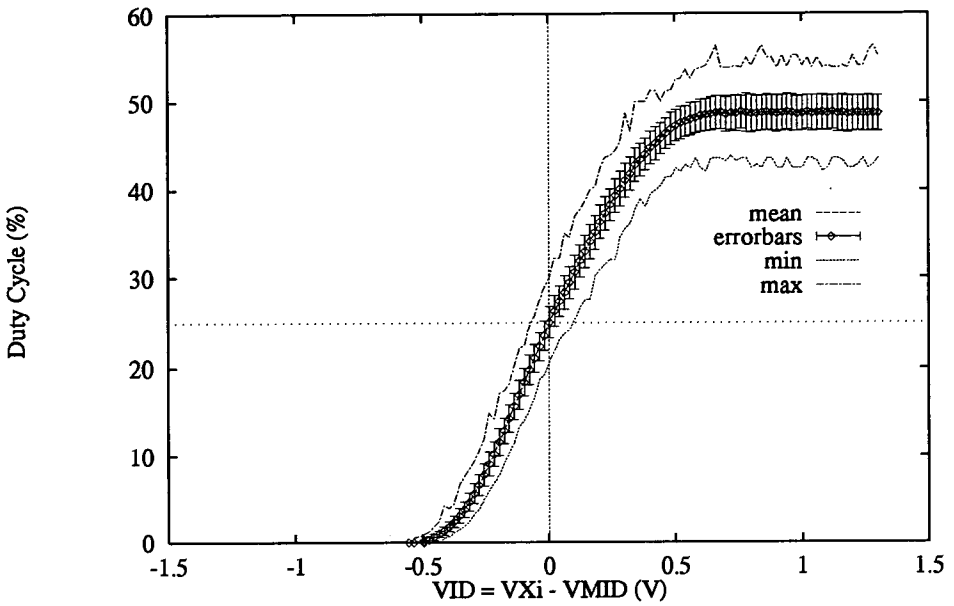


Figure 5.23 EPSILON Chip Neuron Characteristics : High Gain

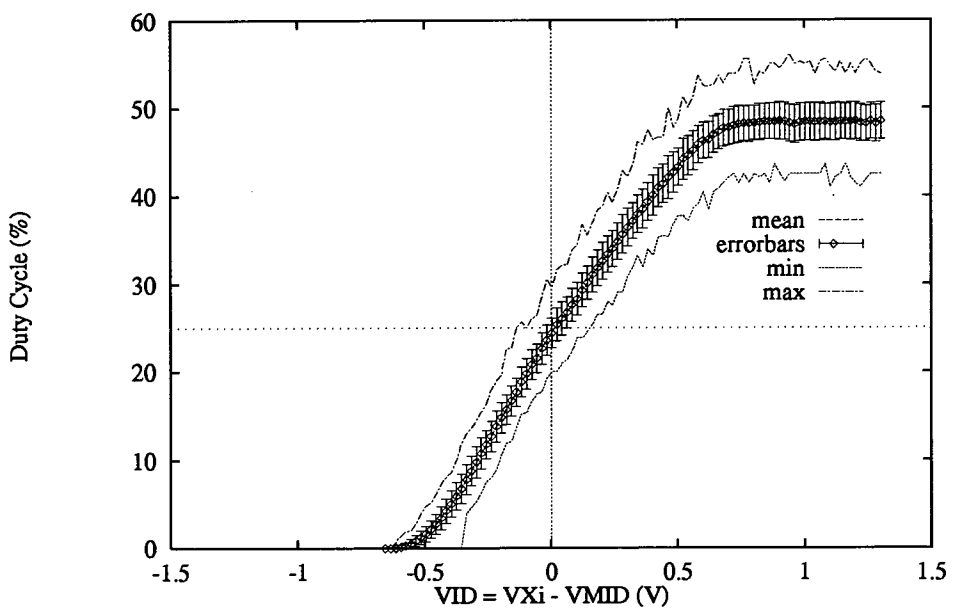


Figure 5.24 EPSILON Chip Neuron Characteristics : Low Gain

duty cycle for VID = 0 is closer to the required value of 25%.

The graph of Figure 5.25 allows comparison of the mean measured duty cycles for the two gain settings. The rotation of the characteristic about the midpoint at VID = 0 and DutyCycle = 25% is evident from these results.

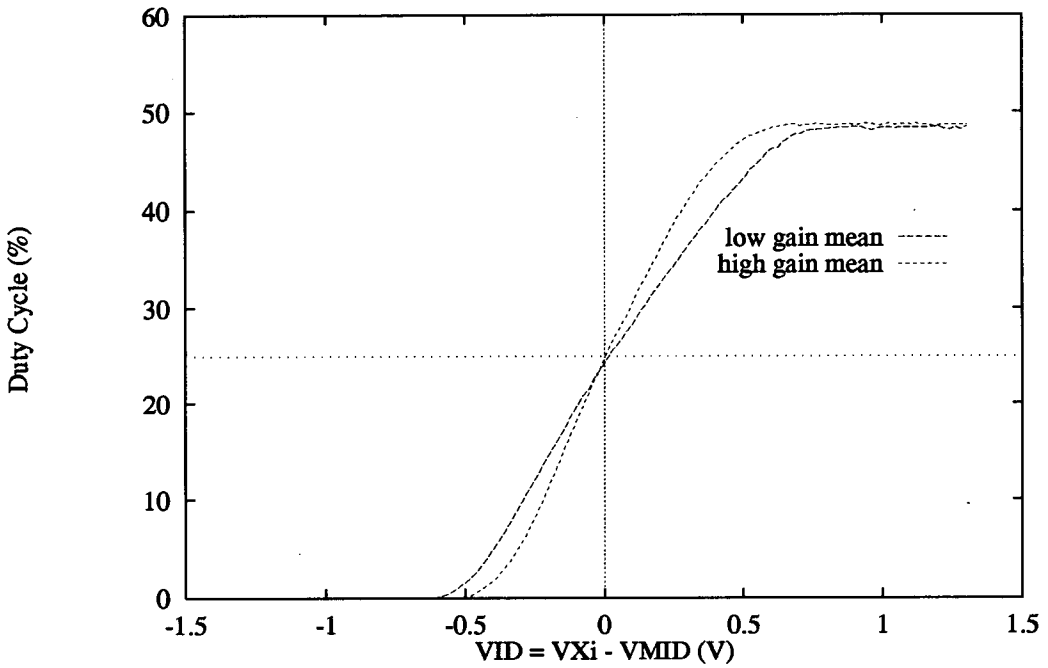


Figure 5.25 Measured EPSILON Chip Neuron Characteristics

These results are summarised in Table 5.9 for the high gain measurements. The minimum, mean and maximum values of duty cycle for selected values of VID are tabulated. The variation of these minimum and maximum values from the mean are expressed for each value of VID as a percentage of the mean duty cycle for VID = 2.5. The standard deviation of each set of points for each tabulated VID point is also expressed as a percentage of the mean duty cycle for VID = 2.5.

Comparison of the maximum standard deviation from Test Chip B, ±5.78%, with that from EPSILON, ±4.07% again indicates a slight improvement in performance.

The theoretical gain derived in Chapter 4 and repeated here in Equation 5.7 has been calculated with $I_G = 0$ and the parameters given in Table 5.10. The mean gain of the transfer characteristic for each of the 16 chips has been measured and the gain from HSPICE simulations using typical model parameters defined in Table 5.10 is also given.

Variable Gain Pulse Stream Neuron : VLSI Results Variation measured on 30 neurons from each of 30 Chips : High Gain					
VID	Duty Cycle (%)			Variation	Std Dev
	Min	Mean	Max		
-0.5	0.00	0.09	0.94	-0.18%/+1.70%	±0.44%
0.0	20.78	25.05	30.16	-8.77%/+10.5%	±3.51%
0.5	42.50	47.28	52.50	-9.68%/+10.7%	±3.59%
1.0	42.50	48.67	55.00	-12.67%/+13.0%	±4.02%
1.3	43.59	48.66	55.26	-10.42%/+13.56%	±4.07%

Table 5.9 Variable Gain Pulse Stream Neuron : VLSI Results
Variation measured on 30 neurons from each of 30 Chips : High Gain

$$\frac{\delta \text{DutyCycle}}{\delta V_{ID}} \left(V_{ID} = 0 \right) = \frac{1}{8}\left[\left[\frac{3\beta \left(I_H + I_G \right)}{I_H{}^2}\right]^{1/2} - \left[\frac{3\beta_i I_G}{I_H{}^2}\right]^{1/2}\right] \tag{5.7}$$

Transistor Parameters Defining β, β_i and I_H Value	
μ_0	238.91cm^2/volt. seconds
C_{ox}	138×10^{-9}F/cm^2
W	10μm
L	10μm
W_i	7μm
L_i	10μm
I_H	3μA

Table 5.10 Transistor Parameters defining β, β_i and I_H Value

The difference between the theoretical and HSPICE measurements can be explained by the simplified analysis performed in Chapter 4 where channel length modulation effects have been ignored. The difference between the measured results and those obtained from HSPICE is likely to be as a result of I_G in the experimental set-up not being exactly 0.

Neuron Gain Results			
Theoretical	HSPICE	Measured	
		Mean	Std Dev
71.77	83.9	67.63	±10.8

Table 5.11 Variable Gain Neuron Circuit Results

A comparison of EPSILON results with HSPICE simulation data is shown in Figure 5.26. The mean of the measured data is compared with HSPICE simulation results using typical process parameters. The correspondence between the low gain measurement and HSPICE data for $I_G = 3\mu A$ is good over almost the entire VID range. Divergence occurs at values of VID around -0.5 due to ES2's HSPICE models not modelling the subthreshold region of operation. Correspondence between the high gain measurement and the HSPICE simulation data is not so good, adding weight to the hypothesis that I_G was not in fact equal to zero during these measurements.

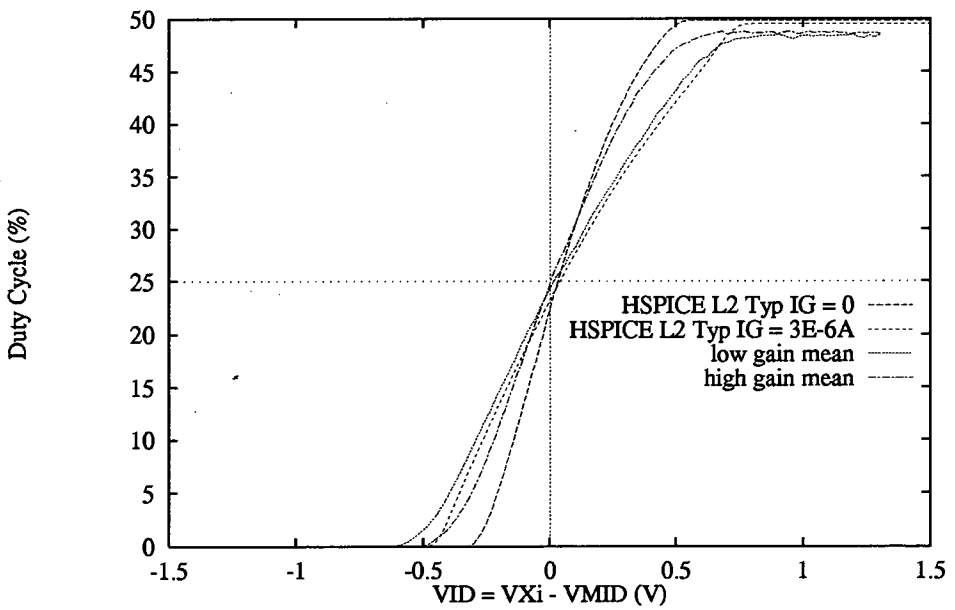


Figure 5.26 Comparison of HSPICE Simulation and EPSILON Chip Results

The operation of the phase lock loop is demonstrated by the data obtained from a 100MHz digital oscilloscope and shown graphically in Figure 5.27, which shows the input reference clock and the output of the neuron used in the loop successfully locked together.

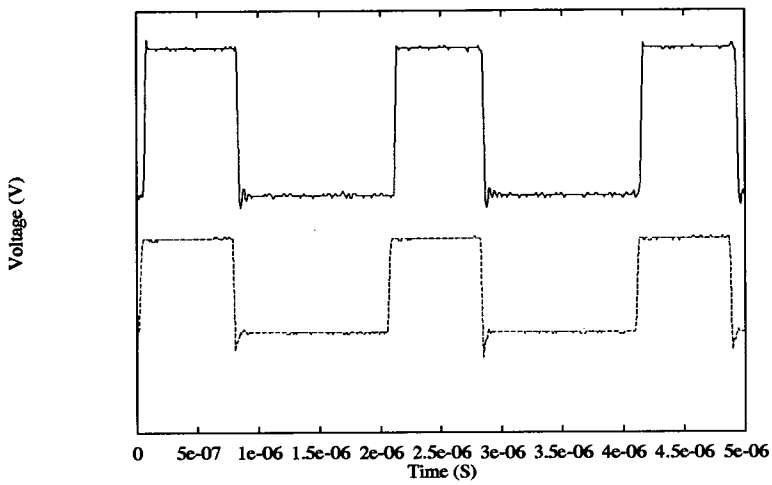


Figure 5.27 Reference Clock and Reference Neuron Phase Locked

5.6.3. EPSILON : Conclusion and Discussion

EPSILON has proved highly successful. All circuits have been demonstrated operational either implicitly or explicitly in the results in the preceding sections. There is an excellent correlation between simulated results and those obtained from functional VLSI. Improvements from the test chips have been made to the functionality of the pulse stream neuron circuit and new untested circuits for pulse width input and output have been successfully added. In addition, the amount of SOLO 1400 generated logic has been reduced to an absolute minimum and all addressing of the synaptic weight array is now implemented using shift register cells pitch matched to the synaptic array. Where deviation exists from simulated results, either experimental procedure or layout problems explain the error. No single error or combination of errors precludes the use of EPSILON for solving neural network problems.

A comparison of multiplier performance for analogue, pulse width and pulse frequency input modes has been performed by Baxter[4]. These produce excellent results for both analogue and pulse width input modes but show a relative decrease in multiplier linearity for the pulse frequency input mode. The results from Test Chip B indicate that this

device is capable of giving good results, ignoring the integrator offset problem, for pulse frequency modulated inputs. Further investigation has shown that there is a problem with the layout of EPSILON which has all 30 neural state inputs to the chip crossing the current set line to the operational amplifiers. This unfortunate layout problem results in transients from the neural state input lines coupling onto the operational amplifier output via the current set line, adversely its performance.

Studies by Baxter[4] indicate that the accuracy obtained by EPSILON is not sufficient to solve neural network problems where the chip is not in the learning loop and accurate mathematical precision is required, for example in the Kohonen network. However, it will be shown in the next Chapter that EPSILON can be used in *chip in loop* learning procedures and has sufficient accuracy and dynamic range to solve *real world* problems.

In order to support EPSILON in solving *real world* problems a more complex and flexible test environment is required. The details of the new test environment are presented in the next Chapter along with system level results.

Chapter 6

EPSILON : System Level Integration and Application

In order to evaluate the capability of EPSILON in implementing neural algorithms a system level environment has been developed that provides support for two EPSILON devices. The system allows the chips to operate in parallel to provide 120 inputs and up to 60 neuron outputs, or in series to implement MLP type structures with a maximum of 120 inputs to the first layer, 30 hidden units and 30 output units. The EPSILON devices may be used in a paged architecture to implement networks of arbitrary size. In addition, feedback networks can be implemented by directly connecting neuron outputs back to neuron inputs.

A vowel recognition problem has been solved using this system, chosen due to the availability of data, local experience of the problem and the *real world* nature of the **analogue data**. A variant of the back propagation training procedure, the Virtual Targets Algorithm[84] has been used to train a multi-layer network to solve the vowel recognition problem. The results from processing this data on EPSILON have been compared to those from software simulation.

6.1. EPSILON : System Level Integration

The system designed to support EPSILON devices provides a flexible environment for the operation of EPSILON in all of its various input and output modes. Two interfaces between a host computer and the system have been provided. The first is a relatively slow bidirectional serial RS-232 link. Commands from the host computer to the system are sent over the serial link and an acknowledgement from the system completes a handshake. The second is a high speed 32 bit bidirectional parallel port with 30 bits used for data transfer and the remaining 2 for a fully interlocked handshake between the host computer and the EPSILON system. The high speed parallel port is used for large data block transfers between the host computer and system and *vice versa.*

The serial link and all system operation is controlled by a Siemens 80C517 microcontroller with associated EPROM program store and RAM. While the microcontroller provides a flexible programming environment it does not have sufficient data processing speed to allow neural learning algorithms to be implemented on it directly. For this reason, all algorithmic learning processing is performed on the host computer and data transferred to and from the EPSILON system.

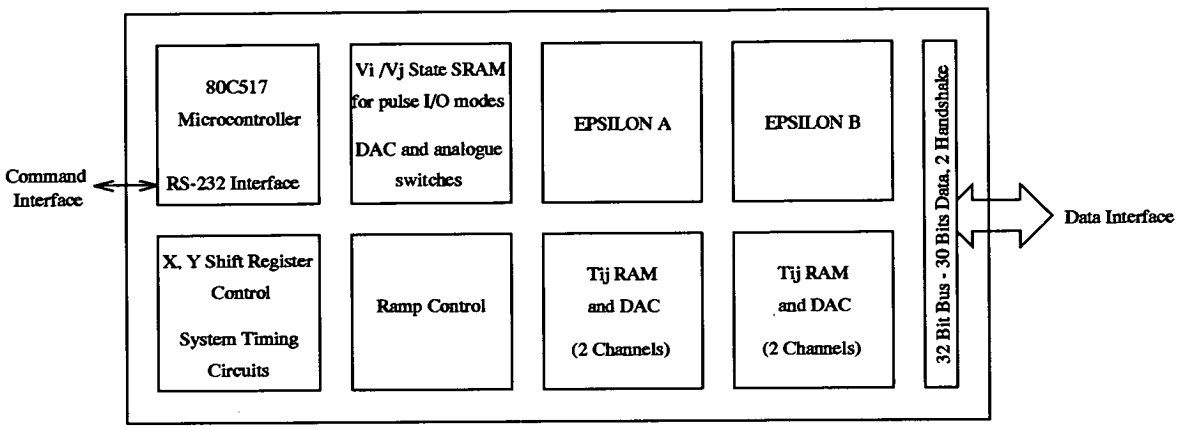


Figure 6.1 EPSILON : System level integration

Figure 6.1 shows all the functional blocks implemented within the system. Digital neural state inputs are either generated from Vi/Vj State RAM as pulse width or pulse frequency modulated signals programmed from the host computer, or from the outputs of EPSILON. Digital output states can be sampled by the Vi/Vj State RAM for subsequent transmission to the host computer for the analysis of pulse widths or pulse frequencies, or the states may form the inputs to EPSILON. Analogue inputs may be generated from a DAC and fed directly to the EPSILON inputs via an analogue multiplexor. The weight refresh circuitry provides continual refresh on two analogue weight channels for each EPSILON chip and may be suspended if required to examine problems of digital noise corrupting the analogue computation on EPSILON. Ramp signal generation is completely programmable from the host computer as the ramp data is stored in static RAM. The gain control for the pulse stream neuron circuits is performed by a programmable digital duty cycle generator implemented on board.

Figure 6.2 shows a simplified representation of the internal data structure of the system for weight and digital neural state generation and capture. The use of tristate outputs on EPSILON allows the implementation of a 30 bit internal bus structure onto which digital (and analogue) inputs and digital outputs can be time multiplexed. The neural state input/output RAM has also been used as a buffer area for transmission of neural synaptic weight data from the host computer to the system over the parallel interface. Data is subsequently transferred from this buffer area to the appropriate weight RAM by the microcontroller over the internal 8 bit data bus.

This diagram illustrates the relative complexity of circuitry, and the resultant chip count required to implement pulse width and pulse stream input/output. In addition, the data bandwidth over the parallel port is relatively high due to the processing of this data by the

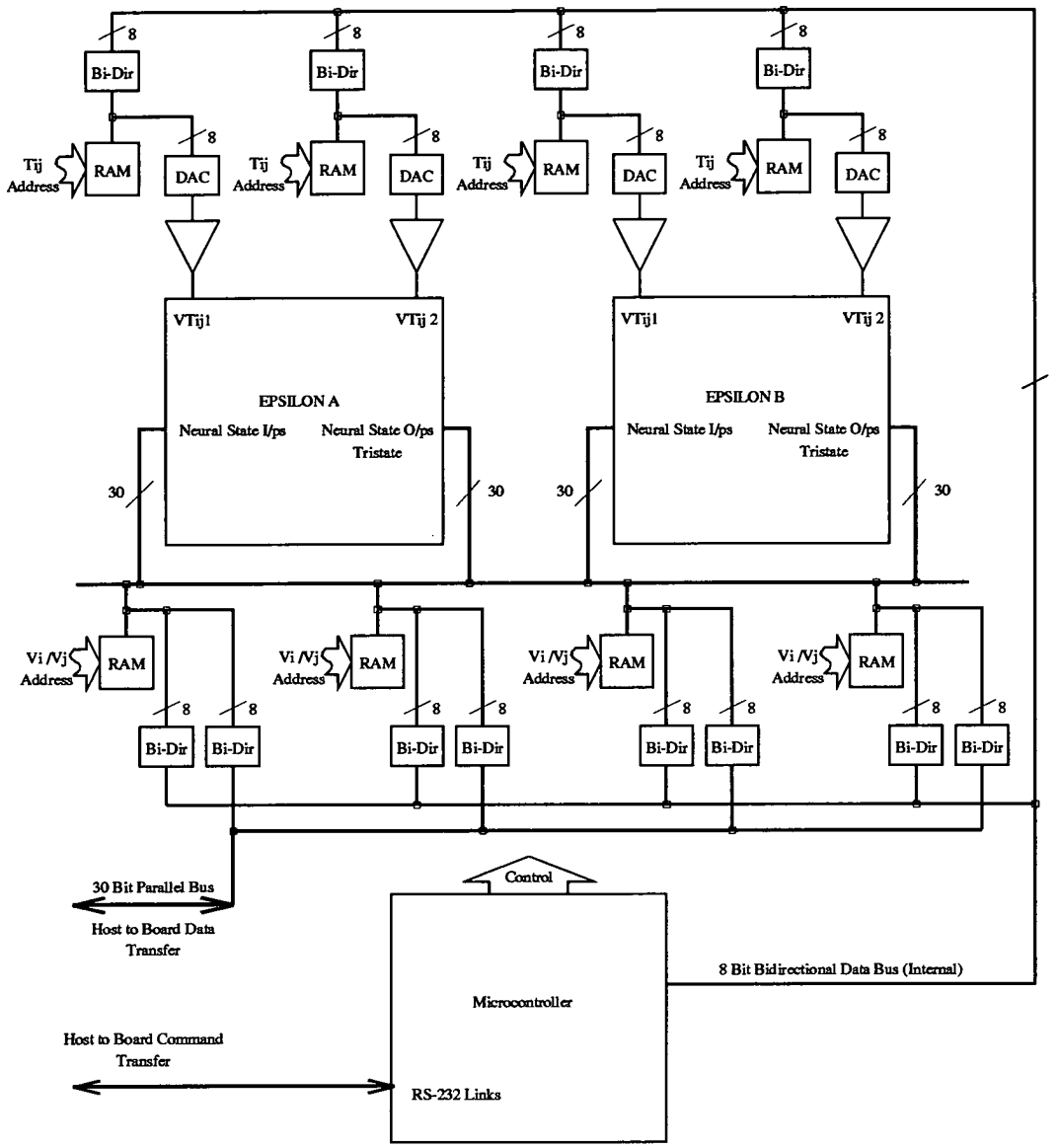


Figure 6.2 EPSILON : System level support circuitry (simplified) Board level data structure. Digital state input/output and weight load mechanisms. Bi-Dir = 8 bit bidirectional tristate buffer.

host computer. This solution therefore is not optimal, however it does use standard readily available components and offers a high degree of flexibility.

6.2. The Oxford/Alvey Vowel Database.

As an example of a *real world* classification task with both training and test sets, the Oxford/Alvey vowel database has been used within the research group [84] to learn to classify vowel sounds from 18 female and 15 male speakers, using a multi-layer feedforward network. The data is a set of analogue outputs from a bank of 54 band-pass filters for 11 different vowel sounds, and 33 speakers. A multi-layer feedforward network architecture with 54 inputs, 27 hidden layer neurons and 11 output neurons has been used with a virtual targets learning strategy[84] to perform training.

The vowel database was a considered choice to test the EPSILON system, primarily because it represented *real world* data. The analogue rather than binary nature of the data is a good test of EPSILON's analogue operating functionality. Furthermore there is considerable experience of the problem of vowel classification within the research group at Edinburgh University.

6.3. The Virtual Targets Training Algorithm.

The virtual targets training algorithm[84] has been designed explicitly to allow the VLSI implementation of learning in multi-layer feedforward networks using analogue techniques. It offers learning speeds and generalisation performance slightly better than standard back-propagation, but is conceptually simpler.

The immunity to analogue inaccuracy is high, in fact high levels of artificially introduced noise have been shown to assist learning[84]. The noise inherent in the analogue circuitry in EPSILON should therefore not degrade performance.

The training procedure is described by the algorithm given in Table 6.1[85].

6.4. Vowel Classification using the Virtual Target Training Algorithm.

The vowel recognition data was used to train a multi-layer network of size 54:27:11 for 2 female speakers on a SUN Workstation using the virtual targets training algorithm.

6.4.1. EPSILON : Off-line Training Considerations.

In order to make the best use of the dynamic range available on EPSILON, the synaptic weight set used for training on the SUN Workstation was initialised to within the range ± 1 and trained with a weight range clipped to within ± 1.5. In order to improve the fault tolerance of the weight set, each weight was corrupted during the forward pass with uniformly distributed random noise of up to 10% of the weight value.

Once learning was satisfactorily completed, in this case when the maximum bit error (mbe) < 0.1, the resultant weight set was downloaded to the host computer connected to

The Virtual Targets Training Algorithm	
1.	**Calculate initial values for the hidden-layer targets**
(a)	Apply input pattern $\{O_{ip}\}$, and read out the states $\{O_{jp}\}$ and $\{O_{kp}\}$ of the hidden and output nodes.
(b)	Assign targets $\{T_{jp}\}$ for the hidden nodes such that $\{T_{jp}\}=\{O_{jp}\}$
2.	**Repeat 1. for all input patterns.**
3.	**Present patterns in random order and allow :**
(a)	weights to evolve according to the following equations : $$\frac{\delta W_{kj}}{\delta t} = \eta_{weights} O_{jp} O'_{kp} \varepsilon_{kp} \quad (1)$$ $$\frac{\delta W_{ji}}{\delta t} = \eta_{weights} O_{ip} O'_{jp} \varepsilon_{jp} \quad (2)$$ where $\eta_{weights}$ is a gain term representing weight learning speed, $\{O_{jp}\}$ and $\{O_{ip}\}$ are the inputs from the previous layer, O'_{kp} and O'_{jp} represent the derivatives of the activation function, ε_{kp} and ε_{jp} are the error terms where $\varepsilon_{kp} = T_{kp} - O_{kp}$ and $\varepsilon_{jp} = T_{jp} - O_{jp}$.
(b)	hidden layer targets to evolve according to the following equation : $$\frac{\delta T_{jp}}{\delta t} = \eta_{targets} \sum_{k=0}^{K} W_{kj} \varepsilon_{kp} \quad (3)$$ where $\eta_{targets}$ is a gain term representing target learning speed, W_{kj} is a weight on the connections between the hidden and output layers, ε_{kp} is the error term where $\varepsilon_{kp} = T_{kp} - O_{kp}$.

Table 6.1 Virtual Targets Training Algorithm.

the EPSILON system.

6.4.2. Mapping the Vowel Classification Problem onto EPSILON.

A single EPSILON device was used in a paged manner to solve the vowel classification problem. The device was used as a 30 input, 30 output array. In practice, 3 of the inputs were used as *system biases* in order to normalise the array for process variations, leaving 27 available to solve the neural network problem. The 54 input layer to hidden layer calculations were therefore paged through the network in 2 separate passes with an additional pass for the hidden layer biases. The hidden layer to output layer calculations were performed in a single pass, with an additional pass for the output layer biases.

Intermediate results were accumulated on the computer where the sigmoid non-linearity was performed.

System biases were used to normalise the performance of the synaptic array so that the result of multiplying a 27 input non-zero state vector by a zero weight array would be a zero output. Due to the performance variations within the synaptic array, the result of this computation on EPSILON will not necessarily produce a zero result. All 30 neural inputs states were fixed and the weight array set to zero weight value. For each neuron output, the *system bias* weight values for that synaptic column were adjusted to give a zero (in the pulse width methodology, 10μs) output. In practice, for successful operation of the *autobias*, each output was required to be in the range 10μs $\pm$ 0.1μs. Any neuron that failed to respond to the *autobias* was ommitted from further neural network evaluation.

While this *autobias* procedure may not be strictly necessary for the successful implementation of the virtual target learning algorithm on the EPSILON system, it allowed intermediate synaptic weight sets from previous EPSILON training cycles to be reloaded from system reset with approximately consistent results.

The weight set calculated off-line on the SUN Workstation was held within the EPSILON host computer environment as a floating point array. Due to the imperfections in the multiplier performance, further training cycles were required with the chip in the learning loop to allow the learning procedure to compensate for the imperfections in EPSILON's multiplier performance. During this further training procedure, the weight array was maintained as a floating point array within the host computer and rounded to 8 bit accuracy for downloading to the EPSILON system for evaluation of the forward pass of the network. All algorithmic training was performed on the host computer with the floating point weight representation.

Further training proceeded until the maximum bit error (mbe) for the 22 training patterns was mbe < 0.3. This less stringent training termination criterion was used to prevent the network from training too rigidly on the training data, and to allow the network to learn within a reasonable time period.

A plot of the mean square error (mse) of the output plotted against training epoch for this problem is shown in Figure 6.3. The arrow (a) denotes a point in the training procedure where training was stopped and restarted from a system reset. The weight array was backed up to hard disc every 25 training epochs and training resumed with the last weight set dumped. At system reset, the autobias procedure performed before training was resumed. The resulting disturbance was compensated for over a relatively small number of learning epochs demonstrating the success of the autobias procedure and learning algorithm in maintaining the general trend of the graph.

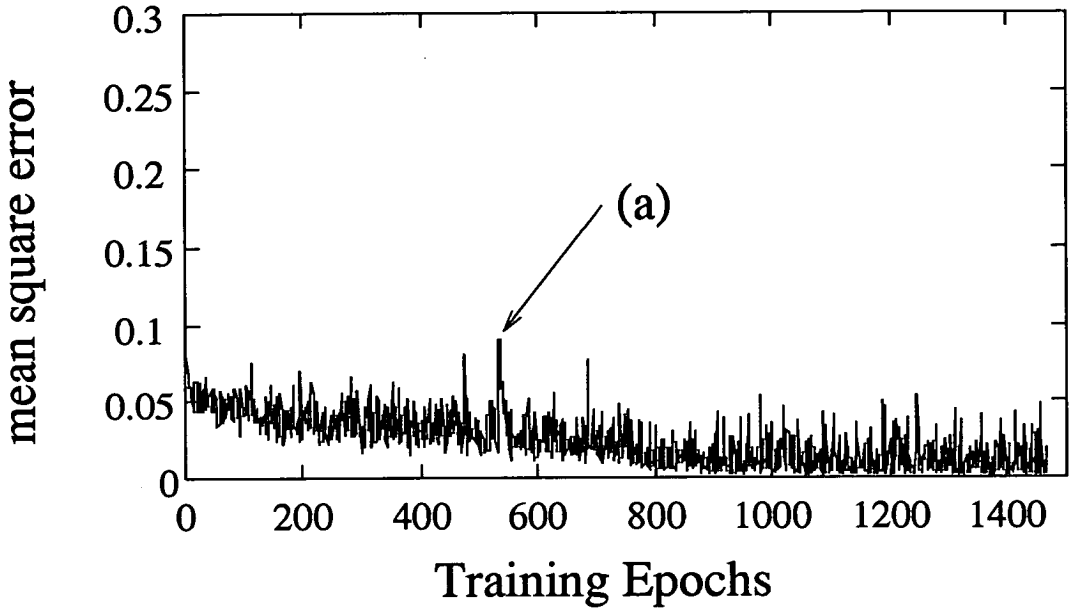


Figure 6.3 EPSILON : Virtual target training on Vowel Data.
Mean square error vs training epoch
(a) denotes a system restart with an intermediate weight set

6.5. Vowel Classification Problem : Results

Once training was successfully achieved, the resultant weight set was used to measure the generalisation performance of the network on the remaining 16 female speakers within the database. Table 6.2 gives the generalisation results from the EPSILON system trained on a single weight set. The maximally responding output neuron was compared against the target response for each of the 11 vowel sounds from the 16 *unseen* female speakers.

EPSILON System : Generalisation Performance **Vowel Data : 16 Female Speakers**
65.34%

Table 6.2 EPSILON System : Vowel data generalisation performance

As a comparison, 20 random weight sets were trained using the virtual targets training algorithm on the same test data on the SUN Workstation. No noise was added during training. The generalisation ability of the network on the 16 unseen Female speakers was calculated for each weight set using the maximally responding output neuron criteria used earlier. The statistics from these results are presented in Table 6.3.

SUN Workstation Simulation Results : Generalisation Performance Vowel Data : 16 Female Speakers 20 Random Weight Sets			
Min.	**Mean**	**Max.**	**Std Dev.**
47.72%	58.21%	67.61%	4.25%

Table 6.3 SUN Workstation Simulation Results.
Vowel data generalisation performance statistics from 20 random weight sets

While not an exhaustive test of the EPSILON system performance, the generalisation results from the EPSILON system indicate that excellent results can be achieved. Indeed, compared to the mean of the simulation results, the generalisation performance of the network has been improved by training with EPSILON induced noise in the forward pass of the network.

6.6. Conclusions and Discussion.

EPSILON has been successfully integrated into a system capable of implementing a range of neural network architectures and demonstrated solving a *real world* problem. The performance of the system on the vowel recognition problem is very encouraging and demonstrates the power of the analogue implementation technology.

The system performance variations measured for pulse width input and output modes and presented in Chapter 5 have not proved an obstacle to successful Virtual Target Training. However, attention to the range of the weight set during training is important due to the limited dynamic range of the synapse. By using an *autobias* technique to compensate for performance variation across chip, a *fully trained* weight set loaded from system reset typically converges with the termination criterion after approximately a further 100 learning Epochs.

While successful implementation of the vowel classification problem on the EPSILON system is an excellent demonstration of the system capability, further experimentation and more comprehensive evaluation remains to be done. At EPSILON chip level the noise levels and repeatability of results should be measured. At the system level, further training and generalisation results will provide a more scientific comparison between EPSILON system performance and those obtained from simulation.

Pulse width modulated input and output signals have been used in the chosen application. In practice, the analogue input mode of EPSILON would allow connection of the system to the 54 filter banks originally used to capture the analogue data for the vowel classification experiments. At the time of writing, the pulse frequency mode of operation has not

been used for neural network experimentation due primarily to the transient problems caused by the operational amplifier layout outlined in Chapter 5, but also to lack of time. The use of pulse frequency coding in implementing network architectures which embody temporal characteristics i.e. feedback and recurrent networks, has yet to be demonstrated on EPSILON.

The system developed around EPSILON to support these experiments has been designed with various constraints and does not represent the optimal environment. The use of RAM to generate and store neural input and output states allows the implementation of a flexible means of generating pulse frequency or pulse width data using standard components. This results, however, in an additional communication bottleneck between the EPSILON system and the host computer and an additional computational burden in generating or evaluating pulse duty cycles or pulse widths. In order to overcome these problems, the generation and decoding of pulse frequency and pulse width signals may be performed using dedicated hardware implemented, for example, using a Field Programmable Gate Array (FPGA)[78]. The data reduction for 8 pulse width and pulse frequency signals transmitted between host and EPSILON system is demonstrated in Table 6.4

Host Computer - EPSILON System Communication Bottleneck : Data Reduction 8 Bit Words for 8 Neural States		
Mode	RAM	FPGA
Pulse Frequency	2000	8
Pulse Width	100	8

Table 6.4 Host Computer - EPSILON System Communication Bottleneck : Data Reduction 8 Bit Words for 8 Neural States. 1% Resolution.

In general, further enhancement would result from the EPSILON system being more closely coupled to the processor performing the training and supervision function[86]. For example, mapping the weight memory and the neural state input output registers into the memory map of this processor would reduce the number of data block moves and communication overheads inherent in the present system. This would require a processor with high speed floating point processing capability, adequate RAM and long term storage, and communication capability to perform the training function. Supervision of this environment may be provided by a remote computer.

Such a system is shown in simplified form in Figure 6.4. Interface to a remote host controller may be provided over a standard interface, the Small Computer Systems Interface

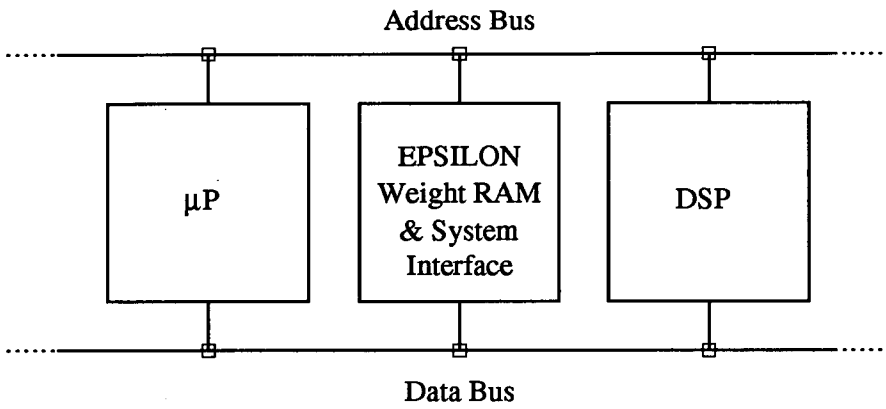


Figure 6.4 Improved EPSILON System
Digital Signal Processor (DSP) Performs learning computations.
Microprocessor (μP) Performs System Supervision.
System memory and interface not shown.

(SCSI), for example[86].

Chapter 7

Summary and Conclusions

7.1. Summary

This thesis has reviewed the wide range of VLSI implementations of neural networks that currently exist. The review concentrates on **working** VLSI implemented in either digital, analogue or mixed signal modes. The variety of implementational styles and techniques reviewed illustrates the vast range of possibilities open to VLSI neural network designers.

A library of cells for the implementation of pulsed neural networks has been developed by the author and other research workers within the Department of Electrical Engineering at the University of Edinburgh. The library includes synapse, neuron, inter-chip communication and associated support circuits. All cells have been implemented and tested in VLSI with results that closely match the HSPICE simulation predictions. Design errors have been made, although not catastrophic, and where deviation from predicted results occur, investigation has found the source of any error. Experience gained early in the development of this cell library has led to circuits that attempt to reduce their performance variation due to the inevitable mismatch of transistors, and that have some form of automatic setup mechanism. These characteristics are seen as essential to the successful development of large analogue neural network chips and multi-chip systems, pulsed or otherwise.

As an example of a large neural network chip developed from the cell library, EPSILON is a flexible neural network building block capable of operating in a variety of input and output modes in a variety of network architectures. As a result of these requirements, both analogue and digital input modes have been implemented allowing EPSILON to be connected to *real world* analogue sensors, pulse frequency or pulse width modulated digital signals. Digital output signals allow cascading of chips or the operation of feedback or recurrent structures.

The demonstration of VLSI pulsed neural networks solving *real world* problems is crucial to their continuing credibility and development. EPSILON has therefore been integrated into a systems environment where it has been demonstrated solving a *real world* vowel classification problem. System level considerations, such as data generation and interpretation has been addressed and the efficient coupling of EPSILON to traditional computing structures has been considered.

7.2. Conclusions.

The main thrust of this thesis is to recommend where, when and how pulsed neural networks, and in particular the cells within the library, would be appropriately applied.

Custom VLSI hardware is generally used for reasons of speed and/or accuracy in simulations of neural networks. This is achieved by designing a custom hardware accelerator to match the computational task. The performance figures given in Chapter 2 where 8 CNAPS chips are capable of 9.67GCPS in solving an MLP network with 1900 inputs, 500 hidden layer units and 12 output units is impressive. The performance of digital VLSI is constantly improving and given the highly automated design process digital neural accelerators should continue to improve.

The accuracy achievable in digital VLSI implementations is also a prerequisite for some neural network algorithms. For example, the Kohonen network[3] requires a high degree of accuracy for successful operation[4].

In digital circuits, the transistor is used as a switch with the result that even very simple functions require a large number of transistors. By striving to exploit the analogue features of transistors, the number of devices required to implement a function can be drastically reduced and much larger computational tasks can be undertaken on chip.

Analogue VLSI is less accurate than digital due to the inherent mismatches between analogue transistors, and may only be used where high precision is not required. Precision of the order of 0.1% is only possible by resorting to special design techniques[87], for example switched capacitor techniques, and is expensive in area. Accuracies of a few percent or tens of percent must be tolerated in order to achieve large high density analogue circuits. While the effect of transistor mismatch can be minimised by good design and by the use of a good analogue process, they do not preclude the use of analogue VLSI for many neural network architectures. As illustrated in Chapter 2, clever use of limited analogue accuracy employed in the ANNA chip combined with a digital interface can yield astounding results. The subthreshold work of Mead also discussed in Chapter 2 illustrates how the problems of large dynamic range in the input signal may be overcome by careful selection of the operating regime of the transistors.

The digital nature of the pulsed signal techniques effectively *switches* analogue circuits in and out of a system, avoiding the problems inherent in purely analogue systems where it is more difficult to maintain all MOSFETs within their correct operating regions. This methodology becomes particularly useful when implementing circuits using a technology optimised for the implementation of digital circuitry. For example, the transistor threshold voltage, crucially important in analogue circuit design, is not critically important in digital implementations. This is evident in the increased variation in threshold voltage between the $2\mu m$ and $1.5\mu m$ Digital CMOS technologies used in the designs in this

thesis.

ES2 Threshold Voltage Variation		
Transistor	2μm	1.5μm
PMOS	0.2V	0.4V
NMOS	0.3V	0.34V

Table 7.1 ES2 Threshold Voltage variation for 2μm and 1.5μm Digital CMOS

The variations in threshold voltage over the process range given in Table 7.1 indicate that although the 1.5μm process results in smaller and correspondingly faster digital circuits, the consequences are that analogue design has to cope with increased performance variation.

The use of digital pulsed signals where analogue information is encoded in the time domain, provides a robust representation of data that is highly insensitive to noise corruption. This not only reduces sources of error within the chip itself, but offers a robust mechanism for communicating neural state information across chip boundaries. The time domain coding of the pulsed signal representation and the use of a relatively slow operational amplifier in the EPSILON system indicate that pulse coded techniques will not necessarily provide the **fastest** solution to a neural network problem. This pulse coded data format is not typically found in conventional computer systems and, as indicated in Chapter 6, dedicated hardware has to be used to generate and decode these signals from computer stored data.

The area where pulsed neural networks will find a competitive edge is in network problems where there is a requirement for a large fan in of analogue input data. Analogue signals can be directly interfaced to pulsed networks (EPSILON, for example) without the requirement for analogue to digital converters (ADCs) required by digital systems. The robust nature of the subsequent processing using pulsed techniques provides higher noise immunity than might be expected from purely analogue techniques.

The performance of the distributed feedback synapse is such that it may be recommended for any network type where *error driven* learning compensates for the performance variation specified. The analogue input mode, using a simple comparator circuit with the external generation of a ramp signal provides interface capability to analogue sensors. The recommendation for feedforward networks is that analogue or pulse width modulated input signals are used, and pulse width representation used for hidden and output layers. The linear nature of the pulse width output neurons facilitates their use in problems where data is paged through a chip, as in the vowel classification problem.

For networks which embody temporal characteristics, for example feedback and recurrent networks, the pulse frequency mode of operation is recommended. While networks of this type have not been evaluated within this thesis, the study of network dynamics using continuous pulse frequency signal representation remains a fertile area for further research.

The inter-chip communication strategy developed within this thesis has not been used on the EPSILON chip. This is due to the architecture of EPSILON where pin out of all neuron circuits has been achieved. Where signal multiplexing of pulse frequency data is required it represents a viable strategy. It is one of the few strategies proposed for inter-chip communication, and the efficiency of the hardware implementation is a direct result of pulse stream coding. If temporal network operation is required, the effects of using this strategy on network dynamics require to be assessed.

While the techniques and circuits presented in this thesis demonstrate the power of pulsed neural network VLSI, further improvements may be achieved. Where the pulsed VLSI device requires significant chip in loop training, the weight load time becomes a significant computational bottleneck. EPSILON, for example, uses a voltage based weight refresh mechanism that requires 3.6ms for weight load. The use of current mode techniques as used in ANNA [28] provide a weight refresh mechanism capable of matching RAM access speeds. This technique may be adapted for use with the distributed feedback synapse to improve dramatically weight load times. Alternatively, more weight channels may be used. Other improvements may include extension of the range of gain characteristics obtainable from the pulse stream neuron, for example.

Further research into the system level operation of EPSILON is desirable to examine issues such as the ability of the network to withstand power supply and temperature variations. Effects such as these have implications for the repeatability of results, the generalisation ability of the network, and the ability of the network to reload previously trained weights sets from power on.

Further refinement of the hardware support circuitry will result in a much smaller and faster system environment for EPSILON. The hardware solution illustrated in Figure 6.4 would allow a conventional commercially available digital signal processor (DSP) board[88] to be attached to the EPSILON system during learning. Once training is complete, the trained weight set could be encoded as EPROM memory and the DSP board removed, resulting in a small portable card containing EPSILON, a microcontroller and FPGA support circuitry with a low bandwidth communication link to a controller.

In conclusion, Table 7.2 provides a summary of where the circuits presented in the cell library may be best used.

The Edinburgh Pulse Stream Cell Library Preferred cell use by network type.				
Neural Network	Synapse	Neurons		Other
		Input	Other	
Multi-Layer Feedforward	Distributed Feedback	Analogue	Pulse Width	None
Feedback or Recurrent (small)	Distributed Feedback	Analogue	Pulse Frequency	None
Feedback or Recurrent (large)	Distributed Feedback	Analogue	Pulse Frequency	Inter-chip Communication Scheme
Kohonen or similar	None	None	None	None

Table 7.2 The Edinburgh Pulse Stream Cell Library : Preferred cell use by network type.

This table, and the work described in the thesis that generated it, satisfies the goal set in Section 1.4 - to define and explore the options afforded by pulse stream neural network VLSI, defining their utility and limitations.

7.3. Further Work in the VLSI Area

(1) Amorphous Silicon Memories : The implementation of fast non-volatile synaptic weight storage using amorphous silicon *programmable resistors* on a conventional CMOS substrate.

(2) On-chip Learning : Implement the virtual target algorithm described in Chapter 6 using pulse stream VLSI techniques.

(3) EPSILON system improvement : Improve the known limitations of EPSILON through a further iteration of silicon and integrate into an optimal hardware environment.

(4) EPSILON applications : Demonstrate neural applications using the refined EPSILON chip and environment.

(5) Noise in Learning : Investigate the effects of noise on neural network learning.

(6) Process Tolerant Design : Investigate circuit techniques for implementing analogue circuits on small geometry digital VLSI.

(7) Process Tolerant Design : Investigate circuit techniques for implementing analogue circuits on small geometry digital VLSI.

(8) Opto-Electronic Neural Networks : Using spatial light modulators for synaptic weight generation and analogue VLSI for subsequent processing.

At the time of writing, the future neural network research interests of the author lie primarily in areas 3 and 4.

References

1. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554 - 2558, April, 1982.
2. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Representations by Back-Propagating Errors", *Nature*, vol. 323, pp. 533-536, 1986.
3. Kohonen, T., *Self-organisation and Associative Memory,* Springer-Verlag, Berlin, 1984.
4. D. J. Baxter, "Process-Tolerant VLSI Neural Networks for Applications in Optimisation", *Ph.D. Thesis, Univ. of Edinburgh*, 1993.
5. A.F. Murray and A.V.W. Smith, "Asynchronous Arithmetic for VLSI Neural Systems", *Electronics Letters*, vol. 23, no. 12, pp. 642-643, June, 1987.
6. A.V.W. Smith, "The Implementation of Neural Networks as CMOS Integrated Circuits", *PhD. Thesis (University of Edinburgh)*, 1988.
7. A.F. Murray and A.V.W. Smith, "A Novel Computational and Signalling Method for VLSI Neural Networks", *European Solid State Circuits Conference*, pp. 19-22, VDE-Verlag, Berlin, 1987.
8. D. Del Corso, F. Gregoretti, C. Pellegrini, and L.M. Reyneri, "An Artificial Neural Network Based on Multiplexed Pulse Streams", *Proc. ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund*, pp. 28-39, June 1990.
9. A. Siggelkow, A.J. Beltman, J.A.G. Nijhuis, and L. Spaanenburg, "Pulse-Density Modulated Neural Networks on a Semi-Custom Gate Forest", *Proc.. ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund (Germany).*, pp. 16-27, June 1990.
10. J. Meador, A. Wu, C. Cole, N. Nintunze, and P. Chintrakulchai, "Programmable Impulse Neural Circuits", *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 101-109, 1990.
11. N. El-Leithy, M. Zaghloul, and R.W. Newcomb, "Implementation of Pulse-Coded Neural Networks", *Proc. 27th Conf. on Decision and Control*, pp. 334-336, 1988.
12. S. Ryckebusch, C. Mead, and J. Bower, "Modelling Small Oscillating Biological Networks in Analog VLSI", *Neural Information Processing Systems (NIPS) Conference*, pp. 384-393, Morgan Kaufmann, December, 1988.
13. J. Tomberg, "Fully Digital Neural Network Implementation Based on Pulse Density Modulation", *IEEE Custom Integrated Circuits Conference, San Diego*, May 1989.

14. Y. Hirai, "A Digital Neuro-chip with Unlimited Connectability for Large Scale Neural Networks", *IJCNN Washington*, pp. 163-169, 1989.

15. P.M. Daniell, W.A.J. Waller, and D.A. Bisset, "An Implementation of Fully Analogue Sum-of-Product Neural Models", *1st IEE Conf. on Artificial Neural Networks*, pp. 52-56, 1989.

16. A.F. Murray, D. Del Corso, and L. Tarassenko, "Pulse-Stream VLSI Neural Networks - Mixing Analog and Digital Techniques", *IEEE Trans. Neural Networks*, pp. 193-204, 1991.

17. I. Aleksander, in *Neural Computing Architectures*, North Oxford Academic Press, 1989.

18. Meridian Parallel Systems, "Powerstack - Expandable Computing Engines", in *Preliminary Flyer*, 41 Carlyle Avenue, Hillington Industrial Estate, Glasgow G52 4XX, February 1993.

19. Krste Asanovic and Nelson Morgan, "Experimental Determination of Precision Requirements for Back-Propagation Training of Artificial Neural Networks", *Proc. 2nd Int. Conf. on Microelectronics for Neural Networks*, pp. 9-15, Oct 16-18 1991.

20. A.F. Murray, A.V.W. Smith, and Z.F. Butler, "Bit - Serial Neural Networks", *Neural Information Processing Systems (NIPS) Conference*, pp. 573-583, 1987.

21. P. B. Denyer and D. Renshaw, in *VLSI Signal Processing : A Bit-Serial Approach*, Addison-Wesley, 1985.

22. Z. F. Butler, "A VLSI Hardware Neural Accelerator using Reduced Precision Arithmetic", *Ph.D. Thesis, Univ. of Edinburgh*, 1990.

23. H. T. Kung, "Why Systolic Architectures?", *Computer*, vol. 15, no. 1, pp. 37-46, January 1982.

24. F. Blayo and P. Hurat, "A VLSI Systolic Array Dedicated to Hopfield Neural Network", in *VLSI for Artificial Intelligence*, ed. J.G. Delgado-Frias and W.R. Moore, pp. 255-264, Kluwer, July 1988.

25. D. Hammerstrom, "A Highly Parallel Digital Architecture for Neural Network Emulation", *Int. Workshop on VLSI for AI and Neural Networks (Oxford)*, 1990.

26. Hal McCartor, "Back Propagation Implementation on the Adaptive Solutions CNAPS Neurocomputer Chip", *Neural Information Processing Systems (NIPS) Conference*, pp. 1028-1031, Morgan Kaufmann, 1991.

27. T. Baker, "Implementation limits for artificial neural networks.", *Master's Thesis Oregon Institute*, 1990.

28. B E. Boser, E. Sackinger, J. Bromley, Y. LeCun, and L. D. Jackel, "An Analog Neural Network Processor with Programmable Topology", *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 2017-2025, Dec. 1991.

29. E. Sackinger, B. E. Boser, and L. D. Jackel, "A Neurocomputer Board Based on the ANNA Neural Network Chip", *Neuarl Information Processing Systems (NIPS) Conference*, no. 4, pp. 773-780, Morgan Kaufmann, 1991.

30. E. Sackinger, B. E. Boser, J. Bromley, Y. LeCun, and L. D. Jackel, "Application of the ANNA Neural Network Chip to High-Speed Character Recognition", *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 498-505, May 1992.

31. J.J. Hopfield and D.W. Tank, "Computing with Neural Circuits: A Model", *Science*, vol. 233, no. 4764, pp. 625 - 633, August, 1986.

32. D. Soo and R. Meyer, "A Four-Quadrant NMOS Analog Multiplier", *IEEE Journal of Solid-State Circuits*, no. 6, Dec. 1982.

33. M. A. Holler, "VLSI Implementations of Learning and Memory Systems : A Review", *Neuarl Information Processing Systems (NIPS) Conference*, no. 3, pp. 993-1000, Morgan Kaufmann, 1990.

34. M. Holler, S. Tam, H. Castro, and R. Benson, "An Electrically Trainable Artificial Neural Network (ETANN) with 10240 "Floating Gate" Synapses", *International Joint Conference on Neural Networks - IJCNN89*, pp. 191-196, June, 1989.

35. H. A. Castro, S. M. Tam, and M. Holler, "Implementation and Performance of an Analog Non-Volatile Neural Network", *IEEE Journal of Analog Integrated Circuits and Signal Processing*.

36. M. L. Mumford, D. K. Andes, and L. R. Kern, "The Mod 2 Neurocomputer System Design", *IEEE Trans. Neural Networks*, vol. 3, no. 3, pp. 423-433, May 1992.

37. C. Mead, in *Analog VLSI and Neural Systems*, Addison-Wesley, 1988.

38. C. Mead, in *Analog VLSI and Neural Systems*, pp. 257-278, Addison-Wesley, 1988.

39. C. Mead, in *Analog VLSI and Neural Systems*, pp. 279-302, Addison-Wesley, 1988.

40. K. A. Boahen and A. G. Andreou, "A Contrast Sensitive Silicon Retina with Reciprocal Synapses", *Neural Information Processing Systems (NIPS) Conference*, pp. 764 - 772, Morgan Kaufmann, 1992.

41. R. G. Benson and T. Delbruck, "Direction Selective Silicon Retina that uses Null Inhibition", *Neural Information Processing Systems (NIPS) Conference*, pp. 756 - 763, Morgan Kaufmann, 1992.

42. Y. Kondo and Y. Sawada, "Functional Abilities of a Stochastic Logic Neural Network", *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 434-443, May 1992.

43. J. E. Tomberg and K. Kaski, "Pulse-Density Modulation Technique in VLSI Implementations of Neural Network Algorithms", *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1277-1286, October 1990.

44. John G. Elias, Hsu-Hsu Chu, and S.M. Meshreki, "A Neuromorphic Impulsive Circuit for Processing Dynamic Signals", *International Conference on Circuits and Systems, San Diego*, pp. 2208-2211, 1992.

45. M. Brownlow, A.F. Murray, and L. Tarassenko, "Results from Pulse Stream Neural Network Devices", *Proc. Int. Workshop on VLSI for Artificial Intelligence and Neural Networks*, pp. A2/1-A2/11, September 1990.

46. B. Linares-Barranco, A. Rodriguez-Vazquez, J.L. Huertas, and E. Sanchez-Sinencio, "CMOS Analog Neural Network Systems Based on Oscillatory Neurons", *International Conference on Circuits and Systems, San Diego*, pp. 2236-2239, 1992.

47. J. Tombs, "Multi-Layer Neural networks and their implementation in Analogue VLSI", *D.Phil Thesis, Univ. of Oxford*, 1992.

48. David Watola, David Gembala, and Jack Meador, "Competitive Learning in Asynchronous Pulse Coded Neural ICs", *International Conference on Circuits and Systems, San Diego*, pp. 2216-2219, 1992.

49. J. G. Elias and B. Chang, "A Genetic Algorithm for Training Networks with Artificial Dendritic Trees", *Proceedings of the International Joint Conference on Neural Networks - IJCNN92*, 1992.

50. A.F. Murray and A.V.W. Smith, "Asynchronous VLSI Neural Networks using Pulse Stream Arithmetic", *IEEE Journal of Solid-State Circuits and Systems*, vol. 23, no. 3, pp. 688-697, 1988.

51. J. Tombs and L. Tarassenko, "A Fast, Novel, Cascadable Design For Multi-Layer Networks", *IEE Second International Conference on Artificial Neural Networks*, pp. 64-68, November 1991.

52. L. Tarassenko, M. Brownlow, G.F. Marshall, and A.F. Murray, "Real-Time Autonomous Robot Navigation Using VLSI Neural Networks", *Neural Information Processing Systems (NIPS) Conference*, pp. 422-428, Morgan Kaufmann, 1991.

53. B.A. Gilbert, "Precise Four-Quadrant Multiplier with Sub-Nanosecond Response", *IEEE Journal of Solid-State Circuits*, vol. SC-3, pp. 365-373, 1968.

54. J. Bailey and D. Hammerstrom, "Why VLSI Implementations of Associative VLCNs Require Connection Multiplexing", *IEEE International Conference on Neural Networks, San Diego, CA, USA.*, vol. 2, pp. 173-180, 1988.

55. M. P. Craven, K. M. Curtis, and B. R. Hayes-Gill, "Frequency Division Multiplexing in an Analogue Neural Network", *Electronics Letters*, vol. 27, no. 11, 23rd May 1991.

56. A. Mortara and E. A. Vittoz, "A Communication Architecture Tailored For Analog VLSI Artificial Neural Networks : Intrinsic Performance and Limitations.", *IEEE Transactions on Neural Networks*, Submitted for Publication September 1992.

57. J.P. Sage, K. Thompson, and R.S. Withers, "An Artificial Neural Network Integrated Circuit Based on MNOS/CCD Principles", *Proc. AIP Conference on Neural Networks for Computing, Snowbird*, pp. 381 - 385, 1986.

58. J.P. Sage and R.S. Withers, "Analog Nonvolatile Memory for Neural Network Implementations", *Proc. Electrochemical Society of America*, 1989. To be published

59. J.P. Sage, R.S. Withers, and K. Thompson, "MNOS/CCD Circuits for Neural Network Implementations", *Proc. Int. Symposium on Circuits and Systems*, pp. 1207-1209, IEEE, May 1989.

60. E. Vittoz, H. Oguey, M.A. Maher, O. Nys, E. Dijkstra, and M. Chevroulet, "Analog Storage of Adjustable Synaptic Weights", *Proc. ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund (Germany).*, pp. 69-79, June 1990.

61. S. Churcher, D.J. Baxter, A. Hamilton, A.F. Murray, and H.M. Reekie, "Towards a Generic Neurocomputing Architecture", *International Conference on Neural Networks (Munich)*, pp. 127-134, 1991.

62. S. Churcher, "VLSI Neural Networks for Computer Vision", *Ph.D. Thesis, Univ. of Edinburgh*, 1993.

63. A.F. Murray, L. Tarassenko, and A.V.W. Smith, "Fully-Programmable Analogue VLSI Devices for the Implementation of Neural Networks", in *VLSI for Artificial Intelligence*, ed. J.G.Delgado-Frias, W.R. Moore, pp. 236-244, Kluwer, 1988.

64. A.F. Murray, L. Tarassenko, and A. Hamilton, "Programmable Analogue Pulse-Firing Neural Networks", *Neural Information Processing Systems (NIPS) Conference*, pp. 671-677, Morgan Kaufmann, 1988.

65. A.F. Murray, A. Hamilton, and L. Tarassenko, "Analog VLSI Neural Networks : Pulse Stream Implementations", in *Journees d'Electronique*, pp. 265-277, Presses Polytechniques Romandes, Lausanne, 1989.

66. A.F. Murray, "Pulse Arithmetic in VLSI Neural Networks", *IEEE MICRO*, vol. 9, no. 6, pp. 64-74, 1989.

67. A. Hamilton, A.F. Murray, S. Churcher, and H.M. Reekie, "Working Analogue Pulse Stream Neural Network Chips", *Proc. Int. Workshop on VLSI for Artificial Intelligence and Neural Networks*, pp. A3/1-A3/9, September 1990.

68. A. Hamilton, A.F. Murray, S. Churcher, and H.M. Reekie, "Working Analogue Pulse Stream Neural Network Chips", in *VLSI for Artificial Intelligence*, ed. J.G.Delgado-Frias, W.R. Moore, pp. 225-233, Plenum (ISBN 0-306-44029-6), 1992.

69. A.F. Murray, D.J. Baxter, H.M. Reekie, S. Churcher, and A. Hamilton, "Advances in the Implementation of Pulse-Stream Neural Networks", *European Conference on Circuit Theory and Design*, vol. II, pp. 422-430, 1991.

70. D.J. Baxter, A.F. Murray, H.M. Reekie, S. Churcher, and A. Hamilton, "Analogue CMOS Techniques for VLSI Neural Networks : Process-Invariant Circuits and Devices", *Proc. IEE Electronics Division Colloquium on Advances in Analogue VLSI*, May 1991.

71. P.B. Denyer and J. Mavor, "MOST Transconductance Multipliers for Array Applications", *IEE Proc. Pt. 1*, vol. 128, no. 3, pp. 81-86, June 1981.

72. Il S. Han and Song B. Park, "Voltage-Controlled Linear Resistors by MOS Transistors and their Application to Active RC Filter MOS Integration", *Proc. IEEE*, pp. 1655-1657, Nov., 1984.

73. A.F. Murray, M. Brownlow, A. Hamilton, Il Song Han, H.M. Reekie, and L. Tarassenko, "Pulse-Firing Neural Chips for Hundreds of Neurons", *Neural Information Processing Systems (NIPS) Conference*, pp. 785-792, Morgan Kaufmann, 1990.

74. S. Churcher, A.F. Murray, and H.M. Reekie, "Pulse-Firing VLSI Neural Circuits for Fast Image Pattern Recognition", *Proc. Int. Workshop on VLSI for Artificial Intelligence and Neural Networks*, pp. G10/1-G10/10, September 1990.

75. D. Jeong, G. Borriello, D.A. Hodges, and R.H. Katz, "Design of PLL-Based Clock Generation Circuits", *IEEE Journal of Solid-State Circuits*, vol. sc-22, no. 2, pp. 255-261, April 1987.

76. H. Ebenhoech, "Make IC digital frequency comparators", *Electron. Des.*, vol. 15, no. 14, pp. 62-64, July 5 1967.

77. R. E. Best, in *Phase-Locked Loops: Theory, Design, and Applications*, New York: McGraw-Hill, 1984.

78. S. J. Arnott, "VLSI Design of a Neural State Interface Chip for Pulsed Neural Networks", *B.Eng E&EE4 Hons. Project Report, Univ. of Edinburgh, Department of Electrical Engineering*, 1993.

79. J.A. Simmons, "Acoustic-Imaging Computations by Echolocating Bats : Unification of Diversely-Represented Stimulus Features into Whole Images", *Neural Information Processing Systems (NIPS) Conference*, pp. 2-9, Morgan Kaufmann, 1989.

80. J. L. McClelland and D. E. Rumelhart, *Explorations in parallel distributed processing a handbook of models, programs, and exercises,* Cambridge, Mass. London MIT Press, 1988.

81. Hebb, D.O., *The Organisation of Behavior,* Wiley, New York, 1949.

82. G. Widrow and M. E. Hoff, "Adaptive switching circuits", *Institue of Radio Engineers, Western Electronic Show and Convention, Convention Record*, vol. 4, pp. 96-104.

83. F. Rosenblatt, "The Perceptron : A Probabilistic Model for Information Storage and Organisation in the Brain", *Psychological Review*, vol. 65, pp. 386-408, 1958.

84. A.F. Murray, "Multi-Layer Perceptron Learning Optimised for On-Chip Implementation - a Noise-Robust System", *Neural Computation*, vol. 4, no. 3, pp. 366-381, 1992.

85. R. J. Woodburn, *Personal Communication*, January, 1993.

86. K. G. Lowe, "VLSI Design of Support Circuitry and System Considerations for the EPSILON Neural Network Chip.", *B.Eng E&EE4 Hons. Project Report, Univ. of Edinburgh, Department of Electrical Engineering*, 1993.

87. E. Vittoz, "Analog VLSI Implementations of Neural Networks", in *Journees d'Electroniques 1989*, Presses Polytechniques Romandes, Lausanne, 1989.

88. Loughborough Sound Images Limited, *DSP Catalogue,* The Technology Centre, Epinal Way, Loughborough, Leicestershire LE11 0QE, U.K., 1992/93.

Appendix 1

Mathematical Derivation of Pulse Stream Neuron Characteristics

This appendix contains the mathematical derivation of the equations used to describe the fixed and electrically adjustable pulse stream neuron circuits described in Chapter 4.

1.1. Fixed Gain Pulse Stream Neuron Transfer Characteristic

The following analysis derives the equations used in Chapter 4 to describe the operation of the pulse stream neuron with fixed gain. Assuming the transistors M3 and M4 in the differential stage of Figure 4.18 are in saturation, their currents can be described by equations A1.1 and A1.2

$$I_{M3} = \frac{\beta_{M3}}{2} (V_{SGM3} - V_T)^2 \tag{A1.1}$$

$$I_{M4} = \frac{\beta_{M4}}{2} (V_{SGM4} - V_T)^2 \tag{A1.2}$$

where V_{SG} and V_T are the source-gate and threshold voltages. The transconductance parameter β is given in terms of physical parameters as,

$$\beta \approx (\mu_0 C_{ox}) \frac{W}{L} \text{ amps/volts}^2 \tag{A1.3}$$

where μ_0 is the surface mobility of the channel for the transistor (cm^2/volt. seconds), and C_{ox} is the capacitance per unit area of the gate oxide (F/cm^2). The variables W and L are the width and length of the device set by the designer. If we define the geometeries of transistors M3 and M4 so that $\beta_{M3} = 2\beta_{M4}$ and substitue β for β_{M4}; the differential input voltage is

$$V_{ID} = V_{SGM3} - V_{SGM4} = \left[\frac{2I_{M3}}{\beta} \right]^{1/2} - \left[\frac{4I_{M4}}{\beta} \right]^{1/2} \tag{A1.4}$$

and

$$I_H = I_{M3} + I_{M4} \tag{A1.5}$$

where it has been assumed that M3 and M4 are matched. Substituting Equation A1.5 into Equation A1.4

$$V_{ID} = \left[\frac{2(I_H - I_{M4})}{\beta}\right]^{1/2} - \left[\frac{4I_{M4}}{\beta}\right]^{1/2} \quad \text{(A1.6)}$$

and therefore

$$V_{ID}^2 = \left[\frac{4I_{M4}}{\beta}\right] + \left[\frac{2(I_H - I_{M4})}{\beta}\right] - 2\left[\frac{8}{\beta^2} I_{M4}(I_H - I_{M4})\right]^{1/2} \quad \text{(A1.7)}$$

Rearranging

$$\beta V_{ID}^2 - 2I_{M4} - 2I_H = 2\left[8I_{M4}\,(I_H - I_{M4})\right]^{1/2} \quad \text{(A1.8)}$$

and squaring both sides.

$$36I_{M4}^2 + I_{M4}\left[8I_H - 4\beta V_{ID}^2 - 32I_H\right] + \beta^2 V_{ID}^4 + 4I_H^2 - 4I_H\beta V_{ID}^2 = 0 \quad \text{(A1.9)}$$

Solving this quadratic equation for I_{M4} allows I_L to be derived. The roots of the quadratic equation may be obtained in the normal manner using Equation A1.10.

$$\frac{-b \pm \left[b^2 - 4ac\right]^{1/2}}{2a} \quad \text{(A1.10)}$$

Where in this case

$$a = 36 \quad \text{(A1.11)}$$

$$b = -24I_H - 4\beta V_{ID}^2 \quad \text{(A1.12)}$$

and

$$c = \beta^2 V_{ID}^4 + 4I_H^2 - 4I_H\beta V_{ID}^2 \quad \text{(A1.13)}$$

Substituting Equations A1.11, A1.12 and A1.13 into Equation A1.10, results in an expression for I_L as given in Equation A1.14.

$$I_L = I_{M4} = \frac{I_H}{3} + \frac{\beta V_{ID}^2}{18} \pm \frac{I_H}{9}\left[\frac{12\beta V_{ID}^2}{I_H} - \frac{2\beta^2 V_{ID}^4}{I_H^2}\right]^{1/2} \quad \text{(A1.14)}$$

The regions in which Equation A1.14 is valid is defined by $V_{ID} < \left[\frac{2I_H}{\beta_x}\right]^{1/2}$ where $\beta_x = \beta_{M4}$ for positive values of V_{ID}, otherwise $\beta_x = \beta_{M3}$. The equation for the duty cycle in terms of I_L and I_H is defined in Equation A1.15.

$$\text{DutyCycle} = \text{PulseWidth} \times \text{PulseFrequency} = \frac{I_L}{I_H + I_L} \quad \text{(A1.15)}$$

The relationship between differential input voltage and duty cycle output has therefore been established by substituting Equation A1.14 into Equation A1.15.

The slope of the transfer characteristic may be derived by calculating

$$\frac{\delta \text{DutyCycle}}{\delta V_{ID}} \left(V_{ID} = 0 \right) \tag{A1.16}$$

Let

$$\text{DutyCycle} = DC = \frac{U}{V} = \frac{I_L}{I_H + I_L} \tag{A1.17}$$

Using the quotient rule

$$\frac{\delta DC}{\delta V_{ID}} = \frac{V \dfrac{\delta U}{\delta V_{ID}} - U \dfrac{\delta V}{\delta V_{ID}}}{V^2} \tag{A1.18}$$

where

$$U = I_L = I_{M4} \tag{A1.19}$$

and

$$V = I_L + I_H = I_{M4} + I_H \tag{A1.20}$$

Therefore

$$\begin{aligned} \frac{\delta U}{\delta V_{ID}} &= \frac{\delta I_L}{\delta V_{ID}} \\ &= \frac{2\beta V_{ID}}{18} \pm \frac{I_H}{9}\left[\frac{12\beta V_{ID}^2}{I_H} - \frac{2\beta^2 V_{ID}^4}{I_H^2}\right]^{-1/2} \frac{1}{2}\left[\frac{24\beta V_{ID}}{I_H} - \frac{8\beta^2 V_{ID}^3}{I_H^2}\right] \end{aligned} \tag{A1.21}$$

Now

$$\frac{\delta V}{\delta V_{ID}} = \frac{\delta I_L}{\delta V_{ID}} + \frac{\delta I_H}{\delta V_{ID}} \tag{A1.22}$$

and

$$\frac{\delta I_H}{\delta V_{ID}} = 0 \tag{A1.23}$$

since I_H is a constant, therefore

$$\frac{\delta U}{\delta V_{ID}} = \frac{\delta V}{\delta V_{ID}} . \tag{A1.24}$$

Substituting Equations A1.19, A1.20 and A1.24 into A1.18

$$\frac{\delta DC}{\delta V_{ID}} = \frac{\left(I_L + I_H \right) \dfrac{\delta U}{\delta V_{ID}} - I_L \dfrac{\delta U}{\delta V_{ID}}}{V^2} = \frac{I_H \dfrac{\delta U}{\delta V_{ID}}}{V^2} \tag{A1.25}$$

For $V_{ID} = 0$

$$V^2 = \frac{16I_H^2}{9} \tag{A1.26}$$

and

$$\frac{\delta U}{\delta V_{ID}} = \frac{4}{3}\left[\frac{\beta I_H}{12}\right] \tag{A1.27}$$

Substituting Equations A1.26 and A1.27 into A1.25

$$\frac{\delta \text{DutyCycle}}{\delta V_{ID}} \left(V_{ID} = 0 \right) = \frac{1}{8}\left[\frac{3\beta}{I_H}\right]^{1/2} \tag{A1.28}$$

The gain of the transfer characteristic is therefore proportional to the square root of β and inversely proportional to the square root of I_H. These parameters can be set by the designer at the design stage.

1.2. Pulse Stream Neuron with Electrically Adjustable Gain

The following analysis derives the equations used in Chapter 4 to describe the operation of the pulse stream neuron with electrically adjustable gain. Assuming the transistors M3 and M4 in the differential stage of Figure 4.21 are in saturation, and assuming the derivation of Equation A1.14 in the previous section, the current through M4 is defined by Equation A1.29.

$$I_{M4} = \frac{I_H + I_G}{3} + \frac{\beta V_{ID}^2}{18} \pm \frac{I_H + I_G}{9}\left[\frac{12\beta V_{ID}^2}{I_H + I_G} - \frac{2\beta^2 V_{ID}^4}{(I_H + I_G)^2}\right]^{1/2} \tag{A1.29}$$

The current through the equivalent transistor in the inner differential stage of Figure 4.20, M4′, is defined by Equation A1.30. Here it has been assumed that $\beta_3' = \beta_4'$ and β' has been substituted for β_4'.

$$I_{M4'} = \frac{I_G}{3} + \frac{\beta' V_{ID}^2}{18} \pm \frac{I_G}{9}\left[\frac{12\beta' V_{ID}^2}{I_G} - \frac{2\beta'^2 V_{ID}^4}{I_G^2}\right]^{1/2} \tag{A1.30}$$

The resultant current I_L is therefore defined as the difference between I_{M4} and $I_{M4'}$ as defined by Equation A1.31.

$$I_L = I_{M4} - I_{M4'} = \frac{I_H}{3} + \frac{(\beta - \beta')V_{ID}^2}{18} \pm \left[\frac{I_H + I_G}{9}\left[\frac{12\beta V_{ID}^2}{I_H + I_G} - \frac{2\beta^2 V_{ID}^4}{(I_H + I_G)^2}\right]^{1/2} - \frac{I_G}{9}\left[\frac{12\beta' V_{ID}^2}{I_G} - \frac{2\beta'^2 V_{ID}^4}{I_G^2}\right]^{1/2}\right] \tag{A1.31}$$

Assuming the duty cycle definition of Equation A1.17, and the quotient rule for differentiation as described in Equation A1.18, then

$$\frac{\delta U}{\delta V_{ID}} = \frac{\delta I_L}{\delta V_{ID}} = \frac{2\beta - \beta' V_{ID}}{18} \pm \left[\frac{I_H + I_G}{9} \left[\frac{12\beta V_{ID}^{\,2}}{(I_H + I_G)} - \frac{2\beta^2 V_{ID}^{\,4}}{(I_H + I_G)^2} \right]^{-1/2} \times \frac{1}{2} \left[\frac{24\beta V_{ID}}{(I_H + I_G)} - \frac{8\beta^2 V_{ID}^{\,3}}{(I_H + I_G)^2} \right] - \frac{I_G}{9} \left[\frac{12\beta' V_{ID}^{\,2}}{I_G} - \frac{2\beta'^2 V_{ID}^{\,4}}{I_G^{\,2}} \right]^{-1/2} \times \frac{1}{2} \left[\frac{24\beta' V_{ID}}{I_G} - \frac{8\beta'^2 V_{ID}^{\,3}}{I_G^{\,2}} \right] \right] \quad \text{(A1.32)}$$

Equations A1.22, A1.23, A1.24 and A1.25 and A1.26 apply here also, resulting in the equation for the gain of the characteristic with $V_{ID} = 0$ given by Equation A1.33.

$$\frac{\delta \text{DutyCycle}}{\delta V_{ID}} (V_{ID} = 0) = \frac{1}{8} \left[\left[\frac{3\beta (I_H + I_G)}{I_H^{\,2}} \right]^{1/2} - \left[\frac{3\beta_i I_G}{I_H^{\,2}} \right]^{1/2} \right] \quad \text{(A1.33)}$$

Appendix 2

Phase Frequency Detector Circuit

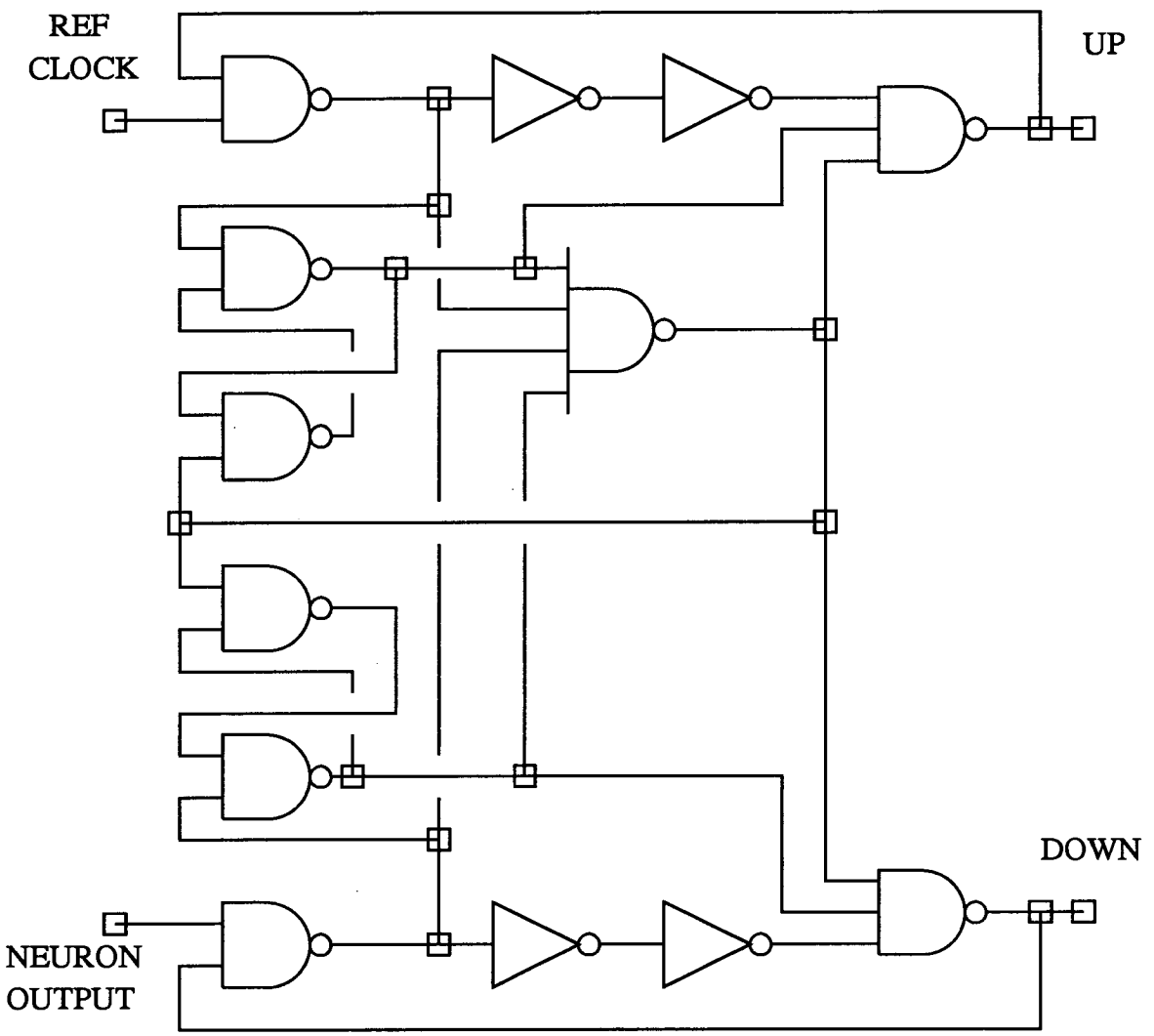


Figure A2.1 Phase Frequency Detector Circuit

Appendix 3

HSPICE Simulation Decks and Net Lists for Pulse Stream Circuits

3.1. A Pulse Stream Synapse using Pulse Width Modulation.

```
* Extracted Synapse Circuit
*-----------------------------------------------------------------
.INCLUDE ../../../../hspice/es2models/ecdm20/l2/typical/nmos
.INCLUDE ../../../../hspice/es2models/ecdm20/l2/typical/pmos
*-----------------------------------------------------------------
.OPTIONS POST NODE NOPAGE GMIN=1.0E-20 GMINDC=1.0E-20
*-----------------------------------------------------------------
*       Buffer
*-----------------------------------------------------------------
M5 7 6 0 0 NMOS L=1.9U W=3.1U AS=19P AD=19P PS=15U PD=15U
M6 7 7 100 100 PMOS L=2.1U W=8.9U AS=45P AD=45P PS=19U PD=19U
*-----------------------------------------------------------------
*       Weight Storage Capacitor
*-----------------------------------------------------------------
M11 0 6 0 100 PMOS L=20.0U W=15.0U AS=75P AD=75P PS=25U PD=25U
M12 0 6 0 0 NMOS L=20.0U W=15.0U AS=75P AD=75P PS=25U PD=25U
*-----------------------------------------------------------------
*       Address Transistor
*-----------------------------------------------------------------
M4 6 5 4 0 NMOS L=2.0U W=3.0U AS=19P AD=19P PS=15U PD=15U
*-----------------------------------------------------------------
*       Inverter and VREF Transistors
*-----------------------------------------------------------------
M1 9 8 7 100 PMOS L=2.1U W=2.9U AS=19P AD=19P PS=15U PD=15U
M2 9 8 14 0 NMOS L=1.9U W=3.1U AS=19P AD=19P PS=15U PD=15U
M3 14 13 0 0 NMOS L=9.9U W=3.1U AS=19P AD=19P PS=15U PD=15U
```

```
*---------------------------------------------------------------
*       Discharge Capacitor
*---------------------------------------------------------------
*---------------------------------------------------------------
*       First Inverter
*---------------------------------------------------------------
M13 10 9 100 100 PMOS L=1.9U W=3.1U AS=19P AD=19P PS=15U PD=15U
M14 10 9 0 0 NMOS L=2.1U W=8.9U AS=25P AD=25P PS=19U PD=19U
*---------------------------------------------------------------
*       Second Inverter
*---------------------------------------------------------------
M15 11 10 100 100 PMOS L=2.1U W=2.9U AS=19P AD=19P PS=15U PD=15U
M16 11 10 0 0 NMOS L=1.9U W=3.1U AS=19P AD=19P PS=15U PD=15U
*---------------------------------------------------------------
*       Pass Transistors
*---------------------------------------------------------------
M7 11 20 21 100 PMOS L=1.9U W=3.1U AS=19P AD=19P PS=15U PD=15U
M8 8 23 24 0 NMOS L=2.1U W=2.9U AS=19P AD=19P PS=15U PD=15U
*---------------------------------------------------------------
*       Output Stage
*---------------------------------------------------------------
M9 100 21 22 100 PMOS L=6.9U W=3.1U AS=19P AD=19P PS=15U PD=15U
M10 22 24 0 0 NMOS L=21.1U W=2.9U AS=19P AD=19P PS=15U PD=15U
M17 0 22 0 0 NMOS L=89.0U W=12.0U
*---------------------------------------------------------------
*       Layout Capacitance
*---------------------------------------------------------------
C15 13 0 11.0F
C17 11 0 15.0F
C18 10 0 14.0F
C19 9 0 29.0F
C20 8 0 16.0F
C21 100 0 39.0F
C22 7 0 23.0F
C23 0 0 97.0F
C24 6 0 17.0F
C25 5 0 3.0F
C26 4 0 5.0F
```

```
C27 22 0 100.0F
*---------------------------------------------------------------
*       Control
*---------------------------------------------------------------
VDD 100 0 DC 5
VREF 13 0 DC 1.5
VCONTROL2 20 0 DC 2.7
VCONTROL1 23 0 DC 2.7
.OPTIONS ITL1=500 ITL2=200 ITL4=100 LVLTIM=1 ITL5=120001 LIMPTS=12001
.OPTIONS PIVTOL=1.0E-25
VADDRESS 5 0 PULSE(0.0 5.0 10N 10N 10N 50N 100000U)
VJ 8 0 PULSE(0.0 5.0 200N 10N 10N 100N 200N)
VTIJ 4 0 PWL(0 1 109N 1 110N 0)
.IC V(22)=2.5
.PRINT TRAN V(8) V(11) V(22)
.TRAN 5N 5U
.END
```

3.2. Pulse Stream Neuron Circuit : Fixed Gain

```
* Pulse Stream Neuron
*** SPICE DECK created from vco.sim, tech=cmos-pw
M1 1 5 4 3 PMOS L=2.0U W=15.0U
M2 7 6 1 3 PMOS L=5.0U W=7.0U
M3 8 5 1 3 PMOS L=2.0U W=15.0U
M4 10 9 8 3 PMOS L=2.0U W=15.0U
M5 12 11 10 3 PMOS L=5.0U W=4.0U
M6 13 12 10 3 PMOS L=10.0U W=4.0U
M7 14 7 1 3 PMOS L=10.0U W=9.0U
M8 6 6 14 3 PMOS L=3.0U W=5.0U
M9 0 15 6 3 PMOS L=3.0U W=20.0U
M10 16 14 1 3 PMOS L=5.0U W=9.0U
M11 15 14 4 3 PMOS L=2.0U W=4.0U
M12 0 6 7 2 NMOS L=5.0U W=3.0U
M13 6 6 14 2 NMOS L=5.0U W=3.0U
M14 1 15 6 2 NMOS L=3.0U W=8.0U
```

```
M15 17 14 15 2 NMOS L=2.0U W=4.0U
M16 14 7 0 2 NMOS L=8.0U W=5.0U
M17 0 13 17 2 NMOS L=2.0U W=10.0U
M18 0 14 16 2 NMOS L=5.0U W=3.0U
M19 0 15 0 2 NMOS L=33.0U W=25.0U
M20 13 13 0 2 NMOS L=2.0U W=10.0U
* C21 18 0 26.0F ROUTING TRACK
C22 16 0 33.0F
C23 0 0 139.0F
C24 13 0 26.0F
C25 15 0 32.0F
C26 14 0 59.0F
C27 12 0 19.0F
C28 11 0 46.0F
C29 10 0 24.0F
C30 9 0 11.0F
C31 8 0 15.0F
C32 7 0 26.0F
C33 6 0 50.0F
C34 5 0 12.0F
C35 4 0 36.0F
C36 1 0 104.0F

M30 1 5 5 1 PMOS L=2U W=15U
M31 5 9 20 1 PMOS L=2U W=15U
M32 20 21 21 1 PMOS L=3U W=3U
M33 21 21 22 0 NMOS L=3U W=3U
M34 22 23 0 0 NMOS L=100U W=3U

VDD 1 0 DC 5V
VNMOS 2 0 DC 0
VPMOS 3 0 DC 5
VGG 9 0 DC 2
VREF 12 0 2.5
VIN 11 0 PULSE (0 2.75 0.05U 0.05U 0.05U 5 10)
VIS 23 0 DC 3.05V
CLD 16 0 1P
```

```
.IC V(15)=0 V(16)=0 V(6)=0 V(7)=5 V(14)=0
.PRINT TRAN V(16)
.OPTIONS POST NODE NOPAGE GMIN=1.0E-20 GMINDC=1.0E-20
+DCON=1
.OPTIONS ITL1=500 ITL2=200 ITL4=100 LVLTIM=1 ITL5=120001
+LIMPTS=12001 PIVTOL=1.0E-25
.INCLUDE ../../hspice/es2models/ecdm20/l2/typical/nmos
.INCLUDE ../../hspice/es2models/ecdm20/l2/typical/pmos
.END
```

3.3. Pulse Stream Neuron Circuit : Electrically Adjustable Gain

```
.INCLUDE ../../../hspice/es2models/ecpd15/l6/typical/nmos
.INCLUDE ../../../hspice/es2models/ecpd15/l6/typical/pmos
M50 1 11 11 1 PMOS W=10U L=5U
M51 1 9 9 1 PMOS W=10U L=5U
*** SPICE DECK created from vtvco.sim, tech=cmos-pw
M1 6 5 4 3 PMOS L=5.0U W=7.2U
M2 8 7 6 3 PMOS L=10.0U W=7.2U
M3 1 9 6 3 PMOS L=5.0U W=10.0U
M4 10 9 1 3 PMOS L=5.0U W=10.0U
M5 1 11 10 3 PMOS L=5.0U W=10.0U
M6 12 11 1 3 PMOS L=5.0U W=10.0U
M7 14 13 12 3 PMOS L=5.0U W=10.0U
M8 15 14 1 3 PMOS L=15.0U W=12.0U
M9 16 14 15 3 PMOS L=15.0U W=12.0U
M10 10 5 17 3 PMOS L=5.0U W=10.0U
M11 18 7 10 3 PMOS L=10.0U W=10.0U
M12 1 16 13 3 PMOS L=5.0U W=12.0U
M13 19 13 1 3 PMOS L=5.0U W=12.0U
M14 0 16 15 3 PMOS L=4.0U W=14.8U
M15 0 4 4 2 NMOS L=5.0U W=10.0U
M16 17 4 0 2 NMOS L=5.0U W=10.0U
M17 0 17 17 2 NMOS L=5.0U W=10.0U
M18 0 8 8 2 NMOS L=5.0U W=10.0U
M19 18 8 0 2 NMOS L=5.0U W=10.0U
M20 0 18 18 2 NMOS L=5.0U W=10.0U
```

```
M21 20 18 0 2 NMOS L=5.0U W=10.0U
M22 14 13 20 2 NMOS L=5.0U W=10.0U
M23 0 14 0 2 NMOS L=13.0U W=16.8U
M24 0 14 21 2 NMOS L=14.6U W=4.0U
M25 21 14 16 2 NMOS L=14.6U W=4.0U
M26 21 16 1 2 NMOS L=4.0U W=4.8U
M27 0 16 13 2 NMOS L=5.0U W=4.0U
M28 0 13 19 2 NMOS L=5.0U W=4.0U
C29 1 6 5.8F
C30 14 0 11.4F
C31 21 0 9.8F
C32 20 0 5.5F
C33 0 0 105.1F
C34 19 0 27.9F
C35 18 0 33.4F
C36 17 0 27.6F
C37 16 0 38.5F
C38 15 0 43.5F
C39 12 0 18.5F
C40 10 0 42.5F
C41 8 0 28.9F
C42 6 0 35.5F
C43 4 0 27.7F
C44 7 0 16.0F
C45 5 0 17.9F
C46 13 0 47.5F
C47 11 0 15.6F
C48 9 0 13.9F
C49 14 0 84.5F
C50 1 0 126.0F
GIPWC 11 0 50 0 3U
GIGC 9 0 51 0 3U
VGI2 51 0 DC 1
VGI 50 0 DC 1
VDD 1 0 5V
VPMOS 3 0 DC 5V
VNMOS 2 0 DC 0V
VREF 7 0 DC 2.5
```

```
*********************************
*                               *
*  sweep vxi from 3.5 to 1.5    *
*       with LOW gain           *
*                               *
*********************************
VXI 5 0 DC 2.10
CLOAD 19 0 0.5P
.IC V(14)=1

.TRAN 5N 160U
.PRINT TRAN V(19) V(14)
.OPTIONS POST NODE NOPAGE GMINDC=1E-20
.OPTIONS INGOLD=2
+DCON=1
.END
```

3.4. Phase Frequency Detector

```
* Phase Frequency Detector for PLL
*** SPICE DECK created from pfd.sim, tech=cmos-pw
M1 5 4 1 3 PMOS L=2.0U W=4.0U
M2 5 6 1 3 PMOS L=2.0U W=4.0U
M3 8 7 1 3 PMOS L=2.0U W=4.0U
M4 8 9 1 3 PMOS L=2.0U W=4.0U
M5 11 10 1 3 PMOS L=2.0U W=4.0U
M6 11 8 1 3 PMOS L=2.0U W=4.0U
M7 10 11 1 3 PMOS L=2.0U W=4.0U
M8 10 12 1 3 PMOS L=2.0U W=4.0U
M9 14 13 1 3 PMOS L=2.0U W=4.0U
M10 14 5 1 3 PMOS L=2.0U W=4.0U
M11 13 14 1 3 PMOS L=2.0U W=4.0U
M12 13 12 1 3 PMOS L=2.0U W=4.0U
M13 15 5 1 3 PMOS L=2.0U W=4.0U
M14 16 15 1 3 PMOS L=2.0U W=4.0U
```

M15 12 8 1 3 PMOS L=2.0U W=4.0U
M16 12 14 1 3 PMOS L=2.0U W=4.0U
M17 12 5 1 3 PMOS L=2.0U W=4.0U
M18 12 11 1 3 PMOS L=2.0U W=4.0U
M19 17 8 1 3 PMOS L=2.0U W=4.0U
M20 18 17 1 3 PMOS L=2.0U W=4.0U
M21 9 18 1 3 PMOS L=2.0U W=4.0U
M22 9 12 1 3 PMOS L=2.0U W=4.0U
M23 9 11 1 3 PMOS L=2.0U W=4.0U
M24 6 16 1 3 PMOS L=2.0U W=4.0U
M25 6 12 1 3 PMOS L=2.0U W=4.0U
M26 6 14 1 3 PMOS L=2.0U W=4.0U
M27 19 4 0 2 NMOS L=3.0U W=4.0U
M28 5 6 19 2 NMOS L=3.0U W=4.0U
M29 20 7 0 2 NMOS L=3.0U W=4.0U
M30 8 9 20 2 NMOS L=3.0U W=4.0U
M31 21 10 0 2 NMOS L=3.0U W=4.0U
M32 11 8 21 2 NMOS L=3.0U W=4.0U
M33 22 11 0 2 NMOS L=3.0U W=4.0U
M34 10 12 22 2 NMOS L=3.0U W=4.0U
M35 23 13 0 2 NMOS L=3.0U W=4.0U
M36 14 5 23 2 NMOS L=3.0U W=4.0U
M37 24 14 0 2 NMOS L=3.0U W=4.0U
M38 13 12 24 2 NMOS L=3.0U W=4.0U
M39 0 5 15 2 NMOS L=3.0U W=4.0U
M40 0 15 16 2 NMOS L=3.0U W=4.0U
M41 25 8 12 2 NMOS L=3.0U W=4.0U
M42 26 14 25 2 NMOS L=3.0U W=4.0U
M43 27 5 26 2 NMOS L=3.0U W=4.0U
M44 0 11 27 2 NMOS L=3.0U W=4.0U
M45 0 8 17 2 NMOS L=3.0U W=4.0U
M46 0 17 18 2 NMOS L=3.0U W=4.0U
M47 28 18 9 2 NMOS L=3.0U W=4.0U
M48 29 12 28 2 NMOS L=3.0U W=4.0U
M49 0 11 29 2 NMOS L=3.0U W=4.0U
M50 30 16 6 2 NMOS L=3.0U W=4.0U
M51 31 12 30 2 NMOS L=3.0U W=4.0U
M52 0 14 31 2 NMOS L=3.0U W=4.0U

```
C53 0 12 14.9F
C54 0 14 9.0F
C55 0 9 5.6F
C56 0 11 10.0F
C57 0 5 11.3F
C58 4 14 35.6F
C59 6 0 5.5F
C60 0 8 11.7F
C61 1 12 5.0F
C62 0 0 76.0F
C63 16 0 18.6F
C64 18 0 16.9F
C65 17 0 14.9F
C66 15 0 15.6F
C67 14 0 70.0F
C68 5 0 51.5F
C69 13 0 24.2F
C70 12 0 82.2F
C71 11 0 54.1F
C72 8 0 61.4F
C73 10 0 24.2F
C74 9 0 42.8F
C75 7 0 6.0F
C76 6 0 42.2F
C77 4 0 19.3F
C78 1 0 218.7F

VDD 1 0 DC 5
VNMOS 2 0 DC 0
VPMOS 3 0 DC 5
VRF 7 0 PULSE(0 5 1.2U 0N 0N 1U 2U)
VCO 4 0 PULSE(0 5 0.2U 0N 0N 1.2U 2.4U)
.IC V(9)=5 V(6)=5 V(5)=5 V(8)=5 V(11)=0 V(14)=0 V(12)=5
.TRAN 2N 5U
.PRINT TRAN V(9) V(10)
.OPTIONS POST NODE NOPAGE GMIN=1.0E-20 GMINDC=1.0E-20 INGOLD=2
+DCON=1
.OPTIONS ITL1=500 ITL2=200 ITL4=100 LVLTIM=1 ITL5=120001
```

```
+LIMPTS=12001 PIVTOL=1.0E-25
.INCLUDE ../../../hspice/es2models/ecpd15/l6/typical/nmos
.INCLUDE ../../../hspice/es2models/ecpd15/l6/typical/pmos
.END
```

3.5. EPSILON VI_H and VI_G Current Set Circuit

```
FEEDBACK CIRCUITS FOR PULSE WIDTH AND GAIN CONTROL
.INCLUDE ../../../hspice/es2models/ecpd15/l6/typical/nmos
.INCLUDE ../../../hspice/es2models/ecpd15/l6/typical/pmos
*** SPICE DECK created from iref.sim, tech=cmos-pw
M1 5 4 1 3 PMOS L=2.0U W=4.0U
M2 5 6 1 3 PMOS L=2.0U W=4.0U
M3 8 7 1 3 PMOS L=2.0U W=4.0U
M4 8 9 1 3 PMOS L=2.0U W=4.0U
M5 11 10 1 3 PMOS L=2.0U W=4.0U
M6 11 8 1 3 PMOS L=2.0U W=4.0U
M7 10 11 1 3 PMOS L=2.0U W=4.0U
M8 10 12 1 3 PMOS L=2.0U W=4.0U
M9 14 13 1 3 PMOS L=2.0U W=4.0U
M10 14 5 1 3 PMOS L=2.0U W=4.0U
M11 13 14 1 3 PMOS L=2.0U W=4.0U
M12 13 12 1 3 PMOS L=2.0U W=4.0U
M13 15 5 1 3 PMOS L=2.0U W=4.0U
M14 16 15 1 3 PMOS L=2.0U W=4.0U
M15 12 8 1 3 PMOS L=2.0U W=4.0U
M16 12 14 1 3 PMOS L=2.0U W=4.0U
M17 12 5 1 3 PMOS L=2.0U W=4.0U
M18 12 11 1 3 PMOS L=2.0U W=4.0U
M19 17 8 1 3 PMOS L=2.0U W=4.0U
M20 18 17 1 3 PMOS L=2.0U W=4.0U
M21 9 18 1 3 PMOS L=2.0U W=4.0U
M22 9 12 1 3 PMOS L=2.0U W=4.0U
M23 9 11 1 3 PMOS L=2.0U W=4.0U
M24 6 16 1 3 PMOS L=2.0U W=4.0U
```

M25 6 12 1 3 PMOS L=2.0U W=4.0U
M26 6 14 1 3 PMOS L=2.0U W=4.0U
M27 19 4 0 2 NMOS L=3.0U W=4.0U
M28 5 6 19 2 NMOS L=3.0U W=4.0U
M29 20 7 0 2 NMOS L=3.0U W=4.0U
M30 8 9 20 2 NMOS L=3.0U W=4.0U
M31 21 10 0 2 NMOS L=3.0U W=4.0U
M32 11 8 21 2 NMOS L=3.0U W=4.0U
M33 22 11 0 2 NMOS L=3.0U W=4.0U
M34 10 12 22 2 NMOS L=3.0U W=4.0U
M35 23 13 0 2 NMOS L=3.0U W=4.0U
M36 14 5 23 2 NMOS L=3.0U W=4.0U
M37 24 14 0 2 NMOS L=3.0U W=4.0U
M38 13 12 24 2 NMOS L=3.0U W=4.0U
M39 0 5 15 2 NMOS L=3.0U W=4.0U
M40 0 15 16 2 NMOS L=3.0U W=4.0U
M41 25 8 12 2 NMOS L=3.0U W=4.0U
M42 26 14 25 2 NMOS L=3.0U W=4.0U
M43 27 5 26 2 NMOS L=3.0U W=4.0U
M44 0 11 27 2 NMOS L=3.0U W=4.0U
M45 0 8 17 2 NMOS L=3.0U W=4.0U
M46 0 17 18 2 NMOS L=3.0U W=4.0U
M47 28 18 9 2 NMOS L=3.0U W=4.0U
M48 29 12 28 2 NMOS L=3.0U W=4.0U
M49 0 11 29 2 NMOS L=3.0U W=4.0U
M50 30 16 6 2 NMOS L=3.0U W=4.0U
M51 31 12 30 2 NMOS L=3.0U W=4.0U
M52 0 14 31 2 NMOS L=3.0U W=4.0U
M53 33 1 32 3 PMOS L=5.0U W=7.2U
M54 35 34 33 3 PMOS L=10.0U W=7.2U
M55 1 36 33 3 PMOS L=5.0U W=10.0U
M56 37 36 1 3 PMOS L=5.0U W=10.0U
M57 1 38 37 3 PMOS L=5.0U W=10.0U
M58 39 38 1 3 PMOS L=5.0U W=10.0U
M59 41 40 39 3 PMOS L=5.0U W=10.0U
M60 42 41 1 3 PMOS L=15.0U W=12.0U
M61 43 41 42 3 PMOS L=15.0U W=12.0U
M62 37 1 44 3 PMOS L=5.0U W=10.0U

M63 45 34 37 3 PMOS L=10.0U W=10.0U
M64 1 43 40 3 PMOS L=5.0U W=12.0U
M65 4 40 1 3 PMOS L=5.0U W=12.0U
M66 0 43 42 3 PMOS L=4.0U W=14.8U
M67 0 32 32 2 NMOS L=5.0U W=10.0U
M68 44 32 0 2 NMOS L=5.0U W=10.0U
M69 0 44 44 2 NMOS L=5.0U W=10.0U
M70 0 35 35 2 NMOS L=5.0U W=10.0U
M71 45 35 0 2 NMOS L=5.0U W=10.0U
M72 0 45 45 2 NMOS L=5.0U W=10.0U
M73 46 45 0 2 NMOS L=5.0U W=10.0U
M74 41 40 46 2 NMOS L=5.0U W=10.0U
M75 0 41 0 2 NMOS L=13.0U W=16.8U
M76 0 41 47 2 NMOS L=14.6U W=4.0U
M77 47 41 43 2 NMOS L=14.6U W=4.0U
M78 47 43 1 2 NMOS L=4.0U W=4.8U
M79 0 43 40 2 NMOS L=5.0U W=4.0U
M80 0 40 4 2 NMOS L=5.0U W=4.0U
M81 49 48 1 3 PMOS L=2.0U W=4.0U
M82 49 50 1 3 PMOS L=2.0U W=4.0U
M83 52 51 1 3 PMOS L=2.0U W=4.0U
M84 52 53 1 3 PMOS L=2.0U W=4.0U
M85 55 54 1 3 PMOS L=2.0U W=4.0U
M86 55 52 1 3 PMOS L=2.0U W=4.0U
M87 54 55 1 3 PMOS L=2.0U W=4.0U
M88 54 56 1 3 PMOS L=2.0U W=4.0U
M89 58 57 1 3 PMOS L=2.0U W=4.0U
M90 58 49 1 3 PMOS L=2.0U W=4.0U
M91 57 58 1 3 PMOS L=2.0U W=4.0U
M92 57 56 1 3 PMOS L=2.0U W=4.0U
M93 59 49 1 3 PMOS L=2.0U W=4.0U
M94 60 59 1 3 PMOS L=2.0U W=4.0U
M95 56 52 1 3 PMOS L=2.0U W=4.0U
M96 56 58 1 3 PMOS L=2.0U W=4.0U
M97 56 49 1 3 PMOS L=2.0U W=4.0U
M98 56 55 1 3 PMOS L=2.0U W=4.0U
M99 61 52 1 3 PMOS L=2.0U W=4.0U
M100 62 61 1 3 PMOS L=2.0U W=4.0U

M101 53 62 1 3 PMOS L=2.0U W=4.0U
M102 53 56 1 3 PMOS L=2.0U W=4.0U
M103 53 55 1 3 PMOS L=2.0U W=4.0U
M104 50 60 1 3 PMOS L=2.0U W=4.0U
M105 50 56 1 3 PMOS L=2.0U W=4.0U
M106 50 58 1 3 PMOS L=2.0U W=4.0U
M107 63 48 0 2 NMOS L=3.0U W=4.0U
M108 49 50 63 2 NMOS L=3.0U W=4.0U
M109 64 51 0 2 NMOS L=3.0U W=4.0U
M110 52 53 64 2 NMOS L=3.0U W=4.0U
M111 65 54 0 2 NMOS L=3.0U W=4.0U
M112 55 52 65 2 NMOS L=3.0U W=4.0U
M113 66 55 0 2 NMOS L=3.0U W=4.0U
M114 54 56 66 2 NMOS L=3.0U W=4.0U
M115 67 57 0 2 NMOS L=3.0U W=4.0U
M116 58 49 67 2 NMOS L=3.0U W=4.0U
M117 68 58 0 2 NMOS L=3.0U W=4.0U
M118 57 56 68 2 NMOS L=3.0U W=4.0U
M119 0 49 59 2 NMOS L=3.0U W=4.0U
M120 0 59 60 2 NMOS L=3.0U W=4.0U
M121 69 52 56 2 NMOS L=3.0U W=4.0U
M122 70 58 69 2 NMOS L=3.0U W=4.0U
M123 71 49 70 2 NMOS L=3.0U W=4.0U
M124 0 55 71 2 NMOS L=3.0U W=4.0U
M125 0 52 61 2 NMOS L=3.0U W=4.0U
M126 0 61 62 2 NMOS L=3.0U W=4.0U
M127 72 62 53 2 NMOS L=3.0U W=4.0U
M128 73 56 72 2 NMOS L=3.0U W=4.0U
M129 0 55 73 2 NMOS L=3.0U W=4.0U
M130 74 60 50 2 NMOS L=3.0U W=4.0U
M131 75 56 74 2 NMOS L=3.0U W=4.0U
M132 0 58 75 2 NMOS L=3.0U W=4.0U
M133 78 77 76 3 PMOS L=5.0U W=7.2U
M134 79 34 78 3 PMOS L=10.0U W=7.2U
M135 1 36 78 3 PMOS L=5.0U W=10.0U
M136 80 36 1 3 PMOS L=5.0U W=10.0U
M137 1 38 80 3 PMOS L=5.0U W=10.0U
M138 81 38 1 3 PMOS L=5.0U W=10.0U

M139 83 82 81 3 PMOS L=5.0U W=10.0U
M140 84 83 1 3 PMOS L=15.0U W=12.0U
M141 85 83 84 3 PMOS L=15.0U W=12.0U
M142 80 77 86 3 PMOS L=5.0U W=10.0U
M143 87 34 80 3 PMOS L=10.0U W=10.0U
M144 1 85 82 3 PMOS L=5.0U W=12.0U
M145 48 82 1 3 PMOS L=5.0U W=12.0U
M146 0 85 84 3 PMOS L=4.0U W=14.8U
M147 0 76 76 2 NMOS L=5.0U W=10.0U
M148 86 76 0 2 NMOS L=5.0U W=10.0U
M149 0 86 86 2 NMOS L=5.0U W=10.0U
M150 0 79 79 2 NMOS L=5.0U W=10.0U
M151 87 79 0 2 NMOS L=5.0U W=10.0U
M152 0 87 87 2 NMOS L=5.0U W=10.0U
M153 88 87 0 2 NMOS L=5.0U W=10.0U
M154 83 82 88 2 NMOS L=5.0U W=10.0U
M155 0 83 0 2 NMOS L=13.0U W=16.8U
M156 0 83 89 2 NMOS L=14.6U W=4.0U
M157 89 83 85 2 NMOS L=14.6U W=4.0U
M158 89 85 1 2 NMOS L=4.0U W=4.8U
M159 0 85 82 2 NMOS L=5.0U W=4.0U
M160 0 82 48 2 NMOS L=5.0U W=4.0U
M161 36 36 1 3 PMOS L=5.0U W=10.0U
M162 90 90 1 3 PMOS L=5.0U W=10.0U
M163 91 53 1 3 PMOS L=5.0U W=5.2U
M164 92 50 1 3 PMOS L=5.0U W=8.0U
M165 94 93 36 2 NMOS L=25.0U W=4.8U
M166 94 95 90 2 NMOS L=25.0U W=4.8U
M167 0 96 94 2 NMOS L=5.0U W=10.0U
M168 0 92 91 2 NMOS L=5.0U W=5.2U
M169 0 50 92 2 NMOS L=5.0U W=4.0U
M170 38 38 1 3 PMOS L=5.0U W=10.0U
M171 97 97 1 3 PMOS L=5.0U W=10.0U
M172 98 9 1 3 PMOS L=5.0U W=5.2U
M173 99 6 1 3 PMOS L=5.0U W=8.0U
M174 101 100 38 2 NMOS L=25.0U W=4.8U
M175 101 95 97 2 NMOS L=25.0U W=4.8U
M176 0 96 101 2 NMOS L=5.0U W=10.0U

M177 0 96 96 2 NMOS L=5.0U W=10.0U
M178 0 99 98 2 NMOS L=5.0U W=5.2U
M179 0 6 99 2 NMOS L=5.0U W=4.0U
C180 14 0 9.0F
C181 0 77 6.2F
C182 33 1 5.8F
C183 41 0 11.4F
C184 0 52 11.7F
C185 12 1 5.0F
C186 1 36 7.2F
C187 4 0 6.8F
C188 1 77 6.4F
C189 9 0 8.6F
C190 1 78 5.8F
C191 11 0 10.0F
C192 5 0 11.3F
C193 0 83 11.4F
C194 4 14 35.6F
C195 58 48 35.6F
C196 0 50 8.6F
C197 0 56 14.9F
C198 0 58 9.0F
C199 91 95 6.4F
C200 0 95 11.4F
C201 0 98 8.2F
C202 8 0 11.7F
C203 56 1 5.0F
C204 0 48 6.8F
C205 0 53 8.6F
C206 0 55 10.0F
C207 0 91 11.3F
C208 0 49 11.3F
C209 0 93 6.2F
C210 0 96 10.8F
C211 1 38 8.4F
C212 6 0 8.6F
C213 1 34 6.2F
C214 12 0 14.9F

C215 101 0 12.6F
C216 100 0 12.6F
C217 99 0 24.6F
C218 98 0 22.5F
C219 97 0 24.3F
C220 96 0 36.7F
C221 94 0 12.7F
C222 95 0 57.9F
C223 93 0 25.6F
C224 92 0 24.6F
C225 91 0 38.2F
C226 90 0 24.3F
C227 89 0 9.8F
C228 88 0 5.5F
C229 87 0 33.4F
C230 86 0 27.6F
C231 85 0 38.5F
C232 84 0 43.5F
C233 81 0 18.5F
C234 80 0 42.5F
C235 79 0 28.9F
C236 78 0 35.5F
C237 76 0 27.7F
C238 34 0 59.0F
C239 77 0 33.8F
C240 82 0 47.5F
C241 38 0 84.0F
C242 36 0 81.2F
C243 83 0 84.5F
C244 0 0 506.9F
C245 60 0 18.6F
C246 62 0 16.9F
C247 61 0 14.9F
C248 59 0 15.6F
C249 58 0 70.0F
C250 49 0 51.5F
C251 57 0 24.2F
C252 56 0 82.2F

C253 55 0 54.1F
C254 52 0 61.4F
C255 54 0 24.2F
C256 53 0 52.4F
C257 51 0 7.4F
C258 50 0 55.6F
C259 48 0 56.0F
C260 1 0 921.1F
C261 47 0 9.8F
C262 46 0 5.5F
C263 45 0 33.4F
C264 44 0 27.6F
C265 43 0 38.5F
C266 42 0 43.5F
C267 39 0 18.5F
C268 37 0 42.5F
C269 35 0 28.9F
C270 33 0 35.5F
C271 32 0 27.7F
C272 40 0 47.5F
C273 41 0 84.5F
C274 16 0 18.6F
C275 18 0 16.9F
C276 17 0 14.9F
C277 15 0 15.6F
C278 14 0 70.0F
C279 5 0 51.5F
C280 13 0 24.2F
C281 12 0 82.2F
C282 11 0 54.1F
C283 8 0 61.4F
C284 10 0 24.2F
C285 9 0 50.3F
C286 7 0 7.4F
C287 6 0 53.8F
C288 4 0 56.7F
VDD 1 0 DC 5
VPMOS 3 0 DC 5

```
VNMOS 2 0 DC 0
V2_5 95 0 DC 2.5
VREF 34 0 DC 2.5
VROP 100 0 DC 2.5
VROG 93 0 DC 2.5
VVIG 77 0 DC 2.6
IPG 1 96 DC 6U
VRFP 7 0 PULSE(0 5 1.2U 0N 0N 1U 2U)
VRFG 51 0 PULSE(0 5 0.2U 0N 0N 1.2U 2.4U)

.IC V(5)=5 V(8)=5 V(11)=0 V(12)=5 V(14)=0
.IC V(49)=5 V(52)=5 V(55)=0 V(56)=5 V(58)=0
.IC V(41)=1 V(83)=1
.TRAN 5N 5U
.PRINT TRAN V(4) V(48)
.OPTIONS POST NODE NOPAGE GMINDC=1E-20
.OPTIONS INGOLD=2
+DCON=1
.END
```

3.6. Pulse Stream Regeneration - Pulse Width Control Circuit

```
** SPICE file created for circuit iset
** Technology: ecpd1.5/1
**
** NODE: 0 = GND
** NODE: 1 = Vdd
** NODE: 100 = Vdd
M0 100 101 101 1 PMOS L=4.0U W=6.0U
M1 102 101 100 1 PMOS L=4.0U W=6.0U
M2 100 103 103 1 PMOS L=2.0U W=3.0U
M3 103 104 104 1 PMOS L=2.0U W=3.0U
M4 100 101 102 1 PMOS L=4.0U W=6.0U
M5 101 101 100 1 PMOS L=4.0U W=6.0U
M6 100 102 105 1 PMOS L=4.0U W=32.0U
M7 102 106 107 0 NMOS L=4.0U W=20.0U
```

```
M8 107 108 101 0 NMOS L=4.0U W=20.0U
M9 109 104 105 0 NMOS L=4.0U W=12.0U
M10 107 108 101 0 NMOS L=4.0U W=20.0U
M11 102 106 107 0 NMOS L=4.0U W=20.0U
M12 104 104 109 0 NMOS L=4.0U W=16.0U
M13 109 104 107 0 NMOS L=4.0U W=16.0U
M14 108 110 109 0 NMOS L=2.0U W=4.0U
M15 100 111 112 1 PMOS L=5.0U W=3.0U
M16 111 111 100 1 PMOS L=5.0U W=3.0U
M17 100 111 112 1 PMOS L=5.0U W=3.0U
M18 111 111 100 1 PMOS L=5.0U W=3.0U
M19 100 111 112 1 PMOS L=5.0U W=3.0U
M20 111 111 100 1 PMOS L=5.0U W=3.0U
M21 100 111 112 1 PMOS L=5.0U W=3.0U
M22 111 111 100 1 PMOS L=5.0U W=3.0U
M23 100 111 112 1 PMOS L=5.0U W=3.0U
M24 111 111 100 1 PMOS L=5.0U W=3.0U
M25 113 114 114 0 NMOS L=2.0U W=4.0U
M26 100 111 112 1 PMOS L=5.0U W=3.0U
M27 111 111 100 1 PMOS L=5.0U W=3.0U
M28 100 111 112 1 PMOS L=5.0U W=3.0U
M29 115 116 108 1 PMOS L=2.0U W=3.0U
M30 112 117 115 1 PMOS L=5.0U W=3.0U
M31 111 111 100 1 PMOS L=5.0U W=3.0U
M32 100 111 112 1 PMOS L=5.0U W=3.0U
M33 111 111 100 1 PMOS L=5.0U W=3.0U
M34 118 117 111 1 PMOS L=5.0U W=3.0U
M35 114 114 118 1 PMOS L=2.0U W=3.0U
M36 100 111 112 1 PMOS L=5.0U W=3.0U
M37 111 111 100 1 PMOS L=5.0U W=3.0U
M38 100 111 112 1 PMOS L=5.0U W=3.0U
M39 111 111 100 1 PMOS L=5.0U W=3.0U
M40 113 119 109 0 NMOS L=120.0U W=3.0U
M41 120 121 105 1 PMOS L=2.0U W=3.0U
M42 119 122 120 1 PMOS L=2.0U W=3.0U
M43 100 123 121 1 PMOS L=2.0U W=3.0U
M44 100 110 122 1 PMOS L=2.0U W=3.0U
M45 120 123 105 0 NMOS L=2.0U W=3.0U
```

```
M46 119 110 120 0 NMOS L=2.0U W=3.0U
M47 121 123 109 0 NMOS L=2.0U W=3.0U
M48 122 110 109 0 NMOS L=2.0U W=3.0U
M49 109 108 109 0 NMOS L=14.0U W=28.0U
M50 109 120 109 0 NMOS L=5.0U W=5.0U
* Capacitor
M51 109 119 109 0 NMOS L=100.0U W=50.0U
C0 124 109 75F
C1 125 0 387F
C2 123 0 315F
C3 120 0 523F
C4 121 0 379F
C5 122 0 464F
C6 119 0 7925F
C7 118 0 74F
C8 115 0 74F
C9 117 0 708F
C10 116 0 319F
C11 114 0 436F
C12 112 0 2376F
C13 111 0 3345F
C14 110 0 581F
C15 113 0 244F
C16 126 0 90F
C17 124 0 97F
C18 127 0 109F
C19 107 0 1726F
C20 106 0 631F
C21 109 0 6709F
C22 108 0 1153F
C23 105 0 2132F
C24 104 0 825F
C25 103 0 244F
C26 102 0 1557F
C27 101 0 1682F
C28 100 0 5896F
```

3.7. Pulse Stream Regeneration - Time-out Mechanism

```
** SPICE file created for circuit timeout
** Technology: ecpd1.5/1
**
** NODE: 0 = GND
** NODE: 1 = Vdd
** NODE: 100 = Vdd
M0 100 101 101 1 PMOS L=4.0U W=6.0U
M1 102 101 100 1 PMOS L=4.0U W=6.0U
M2 100 103 103 1 PMOS L=2.0U W=3.0U
M3 103 104 104 1 PMOS L=2.0U W=3.0U
M4 100 101 102 1 PMOS L=4.0U W=6.0U
M5 101 101 100 1 PMOS L=4.0U W=6.0U
M6 100 102 105 1 PMOS L=4.0U W=32.0U
M7 102 106 107 0 NMOS L=4.0U W=20.0U
M8 107 108 101 0 NMOS L=4.0U W=20.0U
M9 109 104 105 0 NMOS L=4.0U W=12.0U
M10 107 108 101 0 NMOS L=4.0U W=20.0U
M11 102 106 107 0 NMOS L=4.0U W=20.0U
M12 104 104 109 0 NMOS L=4.0U W=16.0U
M13 109 104 107 0 NMOS L=4.0U W=16.0U
M14 100 110 110 1 PMOS L=4.0U W=6.0U
M15 111 110 100 1 PMOS L=4.0U W=6.0U
M16 100 112 112 1 PMOS L=2.0U W=3.0U
M17 112 113 113 1 PMOS L=2.0U W=3.0U
M18 100 110 111 1 PMOS L=4.0U W=6.0U
M19 110 110 100 1 PMOS L=4.0U W=6.0U
M20 100 111 114 1 PMOS L=4.0U W=32.0U
M21 111 115 116 0 NMOS L=4.0U W=20.0U
M22 116 108 110 0 NMOS L=4.0U W=20.0U
M23 109 113 114 0 NMOS L=4.0U W=12.0U
M24 116 108 110 0 NMOS L=4.0U W=20.0U
M25 111 115 116 0 NMOS L=4.0U W=20.0U
M26 113 113 109 0 NMOS L=4.0U W=16.0U
M27 109 113 116 0 NMOS L=4.0U W=16.0U
```

```
M28 109 115 109 0 NMOS L=99.0U W=37.0U
M29 109 106 109 0 NMOS L=99.0U W=37.0U
M30 117 118 100 1 PMOS L=5.0U W=3.0U
M31 119 120 117 1 PMOS L=5.0U W=3.0U
M32 100 121 106 1 PMOS L=2.0U W=5.0U
M33 100 122 115 1 PMOS L=2.0U W=5.0U
M34 106 121 123 0 NMOS L=3.0U W=3.0U
M35 115 122 124 0 NMOS L=3.0U W=3.0U
M36 119 119 125 0 NMOS L=3.0U W=3.0U
M37 123 119 126 0 NMOS L=3.0U W=3.0U
M38 124 119 127 0 NMOS L=3.0U W=3.0U
M39 125 125 109 0 NMOS L=3.0U W=3.0U
M40 126 125 109 0 NMOS L=3.0U W=3.0U
M41 127 125 109 0 NMOS L=3.0U W=3.0U
C1 127 0 63F
C2 126 0 63F
C3 125 0 222F
C4 124 0 24F
C5 123 0 24F
C6 119 0 570F
C7 117 0 149F
C8 120 0 857F
C9 118 0 857F
C10 116 0 1726F
C11 115 0 6029F
C12 114 0 1900F
C13 113 0 825F
C14 112 0 244F
C15 111 0 1557F
C16 110 0 1682F
C17 121 0 580F
C18 122 0 540F
C19 107 0 1726F
C20 106 0 5981F
C21 109 0 6273F
C22 108 0 1225F
C23 105 0 1711F
C24 104 0 825F
```

C25 103 0 244F
C26 102 0 1557F
C27 101 0 1682F
C28 100 0 7249F

3.8. Pulse Stream Regeneration Circuit

** SPICE file created for circuit vco
** Technology: ecpd1.5/1
**
** NODE: 0 = GND
** NODE: 1 = Vdd
M0 100 101 101 1 PMOS L=4.0U W=6.0U
M1 102 101 100 1 PMOS L=4.0U W=6.0U
M2 100 103 103 1 PMOS L=2.0U W=3.0U
M3 103 104 104 1 PMOS L=2.0U W=3.0U
M4 100 101 102 1 PMOS L=4.0U W=6.0U
M5 101 101 100 1 PMOS L=4.0U W=6.0U
M6 100 102 105 1 PMOS L=4.0U W=32.0U
M7 102 106 107 0 NMOS L=4.0U W=20.0U
M8 107 108 101 0 NMOS L=4.0U W=20.0U
M9 109 104 105 0 NMOS L=4.0U W=12.0U
M10 107 108 101 0 NMOS L=4.0U W=20.0U
M11 102 106 107 0 NMOS L=4.0U W=20.0U
M12 104 104 109 0 NMOS L=4.0U W=16.0U
M13 109 104 107 0 NMOS L=4.0U W=16.0U
M14 100 110 110 1 PMOS L=4.0U W=6.0U
M15 111 110 100 1 PMOS L=4.0U W=6.0U
M16 100 112 112 1 PMOS L=2.0U W=3.0U
M17 112 113 113 1 PMOS L=2.0U W=3.0U
M18 100 110 111 1 PMOS L=4.0U W=6.0U
M19 110 110 100 1 PMOS L=4.0U W=6.0U
M20 100 111 114 1 PMOS L=4.0U W=32.0U
M21 111 115 116 0 NMOS L=4.0U W=20.0U
M22 116 117 110 0 NMOS L=4.0U W=20.0U

```
M23 109 113 114 0 NMOS L=4.0U W=12.0U
M24 116 117 110 0 NMOS L=4.0U W=20.0U
M25 111 115 116 0 NMOS L=4.0U W=20.0U
M26 113 113 109 0 NMOS L=4.0U W=16.0U
M27 109 113 116 0 NMOS L=4.0U W=16.0U
M28 100 105 118 1 PMOS L=4.0U W=8.0U
M29 100 119 120 1 PMOS L=4.0U W=8.0U
M30 118 121 122 1 PMOS L=4.0U W=8.0U
M31 122 105 109 0 NMOS L=5.0U W=5.0U
M32 122 121 109 0 NMOS L=5.0U W=5.0U
M33 100 114 123 1 PMOS L=4.0U W=8.0U
M34 100 124 125 1 PMOS L=4.0U W=7.0U
M35 100 125 126 1 PMOS L=4.0U W=8.0U
M36 127 120 122 1 PMOS L=4.0U W=8.0U
M37 100 119 127 1 PMOS L=4.0U W=8.0U
M38 128 119 100 1 PMOS L=4.0U W=8.0U
M39 121 120 128 1 PMOS L=4.0U W=8.0U
M40 123 120 124 1 PMOS L=4.0U W=8.0U
M41 126 122 121 1 PMOS L=4.0U W=8.0U
M42 120 119 109 0 NMOS L=4.0U W=4.0U
M43 124 114 109 0 NMOS L=5.0U W=5.0U
M44 124 120 109 0 NMOS L=5.0U W=5.0U
M45 125 124 109 0 NMOS L=4.0U W=4.0U
M46 121 125 109 0 NMOS L=5.0U W=5.0U
M47 121 122 109 0 NMOS L=5.0U W=5.0U
M48 127 119 122 0 NMOS L=4.0U W=4.0U
M49 121 119 128 0 NMOS L=4.0U W=4.0U
M50 129 130 100 1 PMOS L=5.0U W=3.0U
M51 129 130 100 1 PMOS L=5.0U W=3.0U
M52 129 130 100 1 PMOS L=5.0U W=3.0U
M53 129 130 100 1 PMOS L=5.0U W=3.0U
M54 129 130 100 1 PMOS L=5.0U W=3.0U
M55 129 130 100 1 PMOS L=5.0U W=3.0U
M56 129 130 100 1 PMOS L=5.0U W=3.0U
M57 129 130 100 1 PMOS L=5.0U W=3.0U
M58 131 132 129 1 PMOS L=5.0U W=3.0U
M59 115 127 131 1 PMOS L=2.0U W=3.0U
M60 129 130 100 1 PMOS L=5.0U W=3.0U
```

M61 129 130 100 1 PMOS L=5.0U W=3.0U
M62 109 115 109 0 NMOS L=14.0U W=28.0U
M63 109 127 115 0 NMOS L=2.0U W=4.0U
M64 106 128 133 0 NMOS L=2.0U W=4.0U
M65 134 130 100 1 PMOS L=5.0U W=3.0U
M66 135 132 134 1 PMOS L=5.0U W=3.0U
M67 106 128 135 1 PMOS L=2.0U W=3.0U
M68 133 136 108 0 NMOS L=2.0U W=4.0U
M69 137 130 100 1 PMOS L=5.0U W=3.0U
M70 138 132 137 1 PMOS L=5.0U W=3.0U
M71 108 139 138 1 PMOS L=2.0U W=3.0U
M72 109 106 109 0 NMOS L=99.0U W=37.0U
M73 109 108 109 0 NMOS L=99.0U W=37.0U
M74 100 139 140 1 PMOS L=4.0U W=7.0U
M75 100 140 141 1 PMOS L=4.0U W=8.0U
M76 141 136 119 1 PMOS L=4.0U W=8.0U
M77 140 139 109 0 NMOS L=4.0U W=4.0U
M78 119 140 109 0 NMOS L=5.0U W=5.0U
M79 109 136 119 0 NMOS L=5.0U W=5.0U
C0 113 105 32F
C1 100 128 43F
C2 117 108 23F
C3 114 109 75F
C4 115 109 180F
C5 108 128 31F
C6 105 127 46F
C7 106 127 23F
C8 117 110 27F
C9 114 111 27F
C10 127 133 35F
C11 136 128 12F
C12 109 119 88F
C13 110 116 46F
C14 115 111 25F
C15 117 113 23F
C16 109 121 105F
C17 141 0 198F
C18 140 0 686F

C19 138 0 74F
C20 137 0 149F
C21 135 0 74F
C22 134 0 149F
C23 133 0 469F
C24 131 0 74F
C25 132 0 878F
C26 129 0 2376F
C27 130 0 695F
C28 126 0 198F
C29 123 0 198F
C30 125 0 733F
C31 124 0 840F
C32 122 0 1577F
C33 121 0 1632F
C34 120 0 1299F
C35 118 0 198F
C36 119 0 2827F
C37 127 0 1196F
C38 128 0 1035F
C39 116 0 1726F
C40 115 0 786F
C41 117 0 847F
C42 114 0 2106F
C43 113 0 825F
C44 112 0 244F
C45 111 0 1557F
C46 110 0 1682F
C47 139 0 566F
C48 136 0 657F
C49 143 0 109F
C50 107 0 1726F
C51 106 0 1132F
C52 109 0 15471F
C53 108 0 1570F
C54 105 0 2448F
C55 104 0 825F
C56 103 0 244F

C57 102 0 1557F
C58 101 0 1682F
C59 100 0 12241F
** NODE: 100 = Vdd
** NODE: 109 = GND
** NODE: 0 = GND!
** NODE: 1 = Vdd!

Appendix 4

VLSI Layout

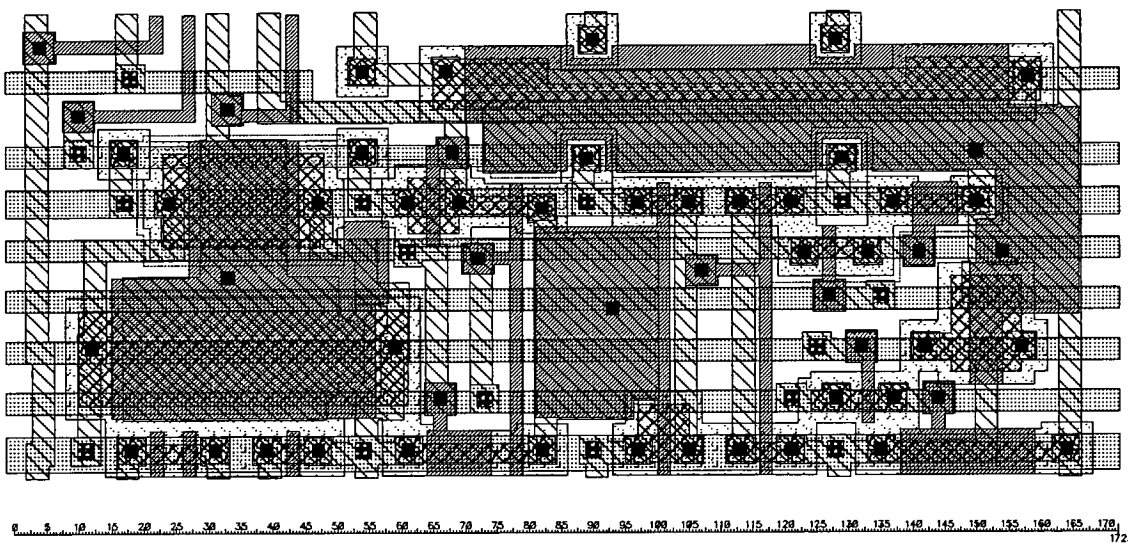


Figure A3.1 Pulse Stream Synapse using Pulse Width Modulation
Layout (2μm CMOS)

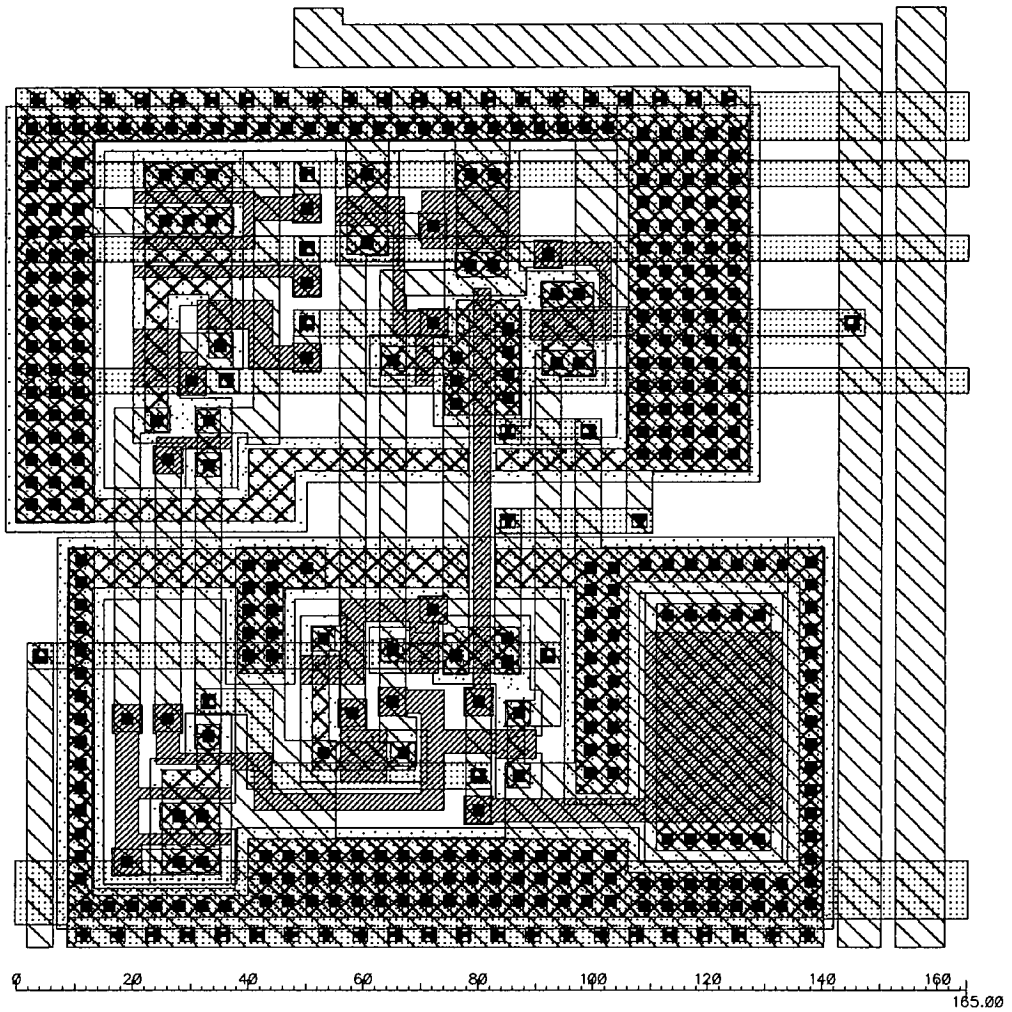


Figure A3.2 Pulse Stream Neuron Circuit (Fixed Gain)
Layout (2μm CMOS)

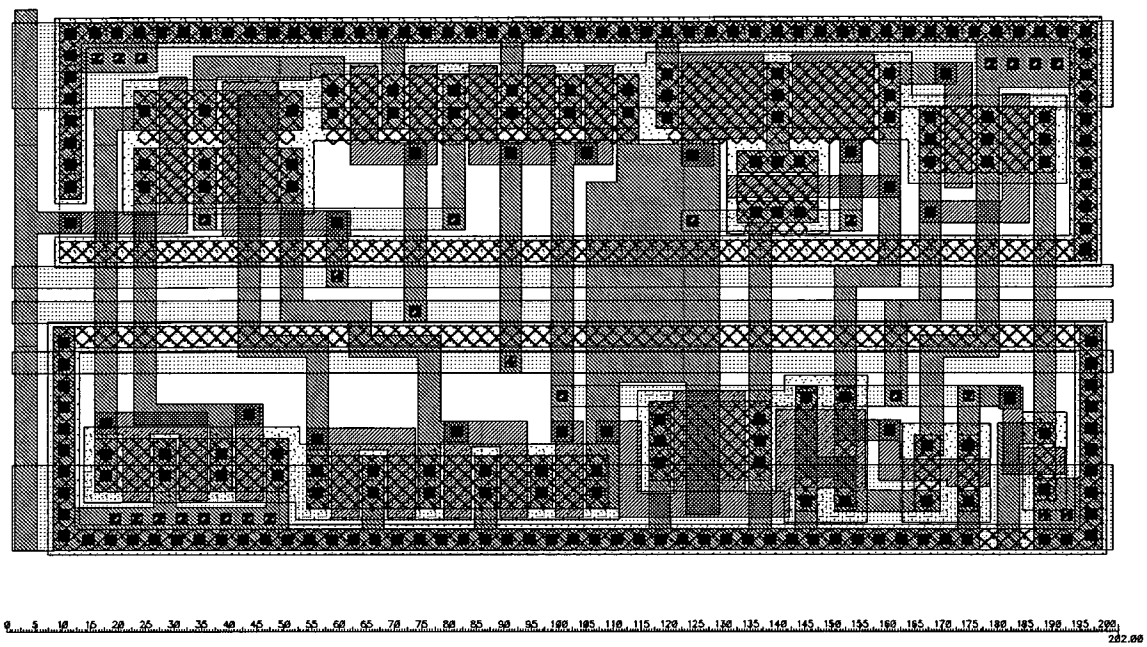


Figure A3.3 Pulse Stream Neuron Circuit (Electrically Adjustable Gain)
Layout (1.5μm CMOS)

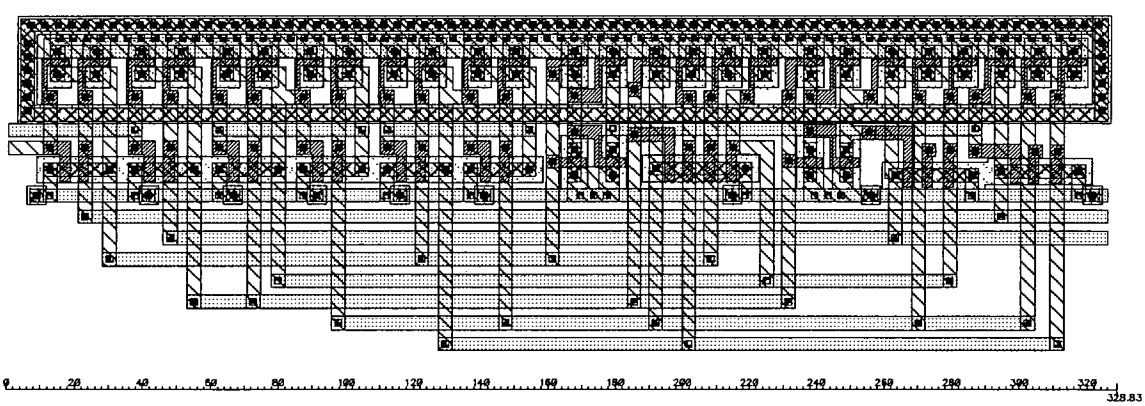


Figure A3.4 Phase Frequency Detector
Layout (1.5μm CMOS)

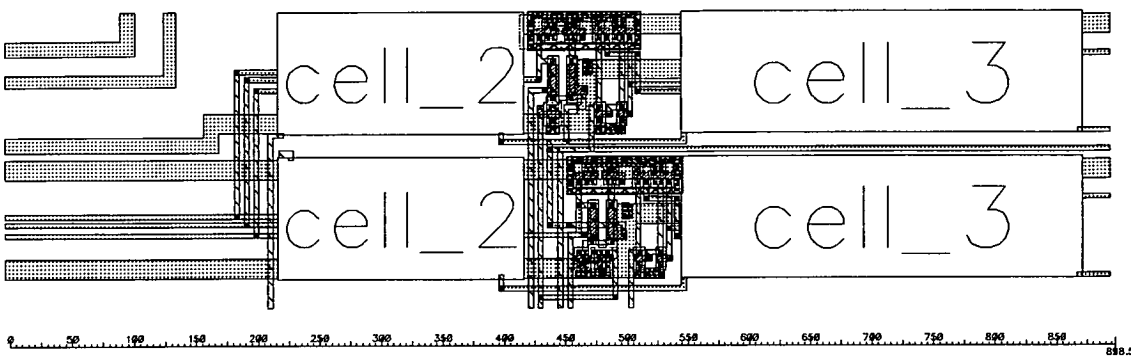


Figure A3.5 EPSILON VI_H and VI_G Current Set Circuit
Cell 2 : Pulse Stream Neuron Circuit (Electrically Adjustable Gain)
Cell 3 : Phase Frequency Detector
Layout (1.5μm CMOS)

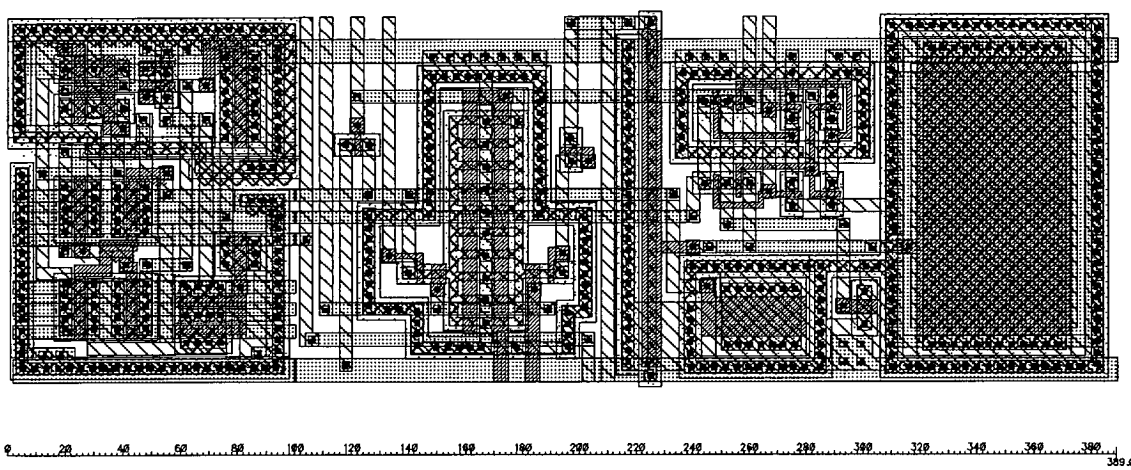


Figure A3.6 Pulse Stream Regeneration - Pulse Width Control Circuit
Layout (1.5μm CMOS)

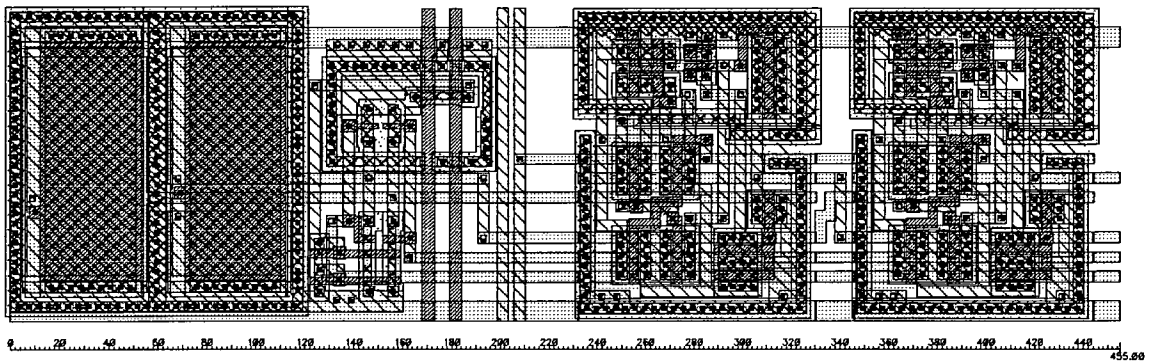


Figure A3.7 Time-out Mechanism
Layout (1.5μm CMOS)

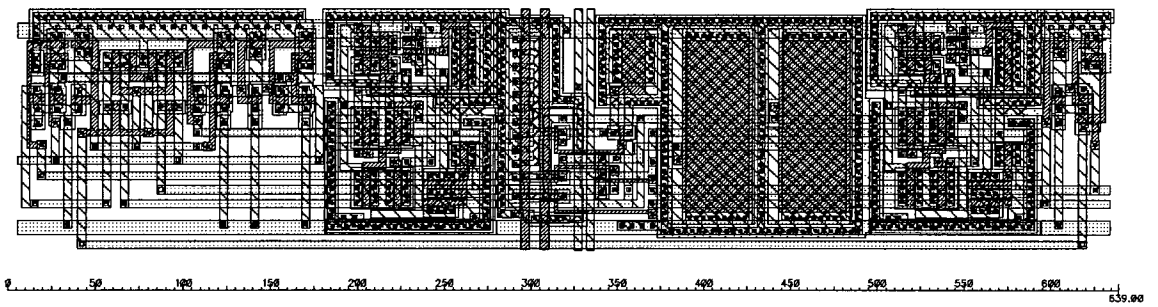


Figure A3.8 Pulse Stream Regeneration Circuit
Layout (1.5μm CMOS)

Appendix 5

Published Papers

The following is a list of papers published by the author during the course of this PhD. Apart from the penultimate paper in this list, these papers have not been reproduced here for reasons of space, but are available in the appropriate conference proceedings, books and journal articles.

A.F. Murray, L. Tarassenko and **A. Hamilton** "Programmable Analogue Pulse-Firing Neural Networks" *Advances in Neural Information Processing Systems* Morgan Kaufmann, pp. 671-677 1988.

A.F. Murray, **A. Hamilton**, H.M. Reekie and L. Tarassenko "Pulse - Stream Arithmetic in Programmable Neural Networks" *Int. Symposium on Circuits and Systems* , pp. 1210-1212 Portland, Oregon 1989.

A.F. Murray, **A. Hamilton** and L. Tarassenko "Analog VLSI Neural Networks : Pulse Stream Implementations" *Journees d'Electronique* Presses Polytechniques Romandes, pp 265-277, Lausanne, 1989

A.F. Murray, M. Brownlow, **A. Hamilton**, Il Song Han H.M. Reekie and L. Tarassenko "Pulse-Firing Neural Chips for Hundreds of Neurons"
Advances in Neural Information Processing Systems Morgan Kaufmann, pp. 785-792, 1990.

A. F. Murray, D. Baxter, Z. Butler, S. Churcher, **A. Hamilton**, H. M. Reekie and L . Tarassenko "Innovations in Pulse Stream Neural VLSI : Arithmetic and Communications" *IEEE Workshop on Microelectronics for Neural Networks*, Dortmund, pp. 8-15, 1990.

A. Hamilton, A.F. Murray, S. Churcher and H.M. Reekie "Working Analogue Pulse Stream Neural Network Chips" *Proc. Int. Workshop on VLSI for Artificial Intelligence and Neural Networks* pp. A3/1-A3/9, University of Oxford, September 1990.

A. Hamilton, A.F. Murray, S. Churcher and H.M. Reekie "Working Analogue Pulse Stream Neural Network Chips" *VLSI for Artificial Intelligence*, Edited by J.G.Delgado-Frias, W.R. Moore, pp. 225-233, Plenum (ISBN 0-306-44029-6), 1992.

A. F. Murray, L. Tarassenko, H. M. Reekie, **A. Hamilton**, M. Brownlow, D. Baxter and S. Churcher "Pulsed Silicon Neural Nets - Following the Biological Leader" in *Introduction to VLSI Design of Neural Networks*, U. Ramacher, Kluwer, pp. 10 3-123, 1991.

D. Baxter, A. F. Murray, H. M. Reekie, S. Churcher and **A. Hamilton** "Analogue CMOS Techniques for VLSI Neural Networks : Process-Invariant Circuits and Devices" *Proc. IEE Electronics Division Colloquium on Advances in Analogue VLSI*, May 1991.

D. Baxter, A. F. Murray, H. M. Reekie, S. Churcher and **A. Hamilton** "Advances in the Implementation of Pulse-Stream Neural Networks" *European Conference on Circuit Theory and Design 91*, Vol II, pp. 422-430 September 1991. (Invited Paper).

S. Churcher, D. J. Baxter, **A. Hamilton**, A. F. Murray and H. M. Reekie "Towards a Generic Neurocomputing Architecture" *International Conference on Neural Networks*, Munich, October 1991.

A. Hamilton, A. F. Murray, D. J. Baxter, S. Churcher, H. M. Reekie and L. Tarassenko "Integrated Pulse-Stream Neural Networks - Results, Issues and Pointers" *IEEE Trans. Neural Networks*, Vol 3, No. 3, May 1992.

S. Churcher, D. J. Baxter, **A. Hamilton**, A. F. Murray and H. M. Reekie "Generic Analog Neural Computation - The EPSILON Chip" *Advances in Neural Information Processing Systems 5*, 1993.

Integrated Pulse Stream Neural Networks: Results, Issues, and Pointers

Alister Hamilton, Alan F. Murray, Donald J. Baxter, Stephen Churcher, H. Martin Reekie, *Member, IEEE*, and Lionel Tarassenko

Abstract—**This paper reports new results from working analog VLSI implementations of two different pulse stream neural network forms. The circuits are rendered relatively invariant to processing variations, and the problem of cascadability of synapses to form large systems is addressed. A strategy for interchip communication of large numbers of neural states has been implemented in silicon and results are presented. The circuits demonstrated confront many of the issues that blight massively parallel analog systems, and offer solutions.**

I. Introduction

While there is still some divergence of opinion regarding the merits and demerits of analog VLSI as a vehicle for neural integration, the analog option has many adherents; see, for example, [1]–[9]. Analog circuit techniques allow the efficient realization of the arithmetic functions that are a prerequisite for the implementation of neural networks. Functions such as addition and multiplication [10], [11] map elegantly into small analog circuits, and the compactness introduced allows for massively parallel arrays of such operators. The limitations of analog circuit technology are, however, well known. Analog circuits are noise-prone and susceptible to the vagaries of process variation, and analog signals are difficult to distribute and cascade. An adaptive learning procedure allows some of the imperfections inherent in analog VLSI to be dealt with automatically, as their effects make themselves felt *when the chip is in the learning loop*. When the hardware is *not* in the learning loop, uniformity must be introduced more actively, by including self-adjusting mechanisms that remove (or at least reduce) the effects of, for instance, threshold variation across a silicon die. We have designed systems that have a high degree of uniformity by employing novel circuits and feedback signals to compensate for processing variations across the chip. In addition, we have studied methods for communicating large numbers of effectively analog signals across a multiplexed data link, using time as an analog information axis. The communication scheme has already been outlined in an earlier "tutorial" paper [12], which set out to explain the options within a pulse stream framework without presenting detailed results. This paper presents results from a working VLSI prototype, combining the merits of digital signaling (robustness, ease of regeneration) with the compactness and dynamic range of analog circuitry, providing concrete evidence of successful working circuits.

The remaining sections of this paper review the pulse stream method and present a series of circuits and results from VLSI devices that illustrate the success of the technique, and its effectiveness in combating the problems that beset massively parallel analog systems.

Finally, we draw some general conclusions regarding the silicon and other results presented, to point the way forward for pulse stream and other fundamentally analog VLSI implementations.

Manuscript received July 23, 1991; revised October 9, 1991. This work was supported by the U.K. Science and Engineering Research Council (SERC) and by the EEC (ESPRIT *NERVES*). Support was also provided by Thorn-EMI and British Aerospace through CASE studentships to D. Baxter and S. Churcher.

A. Hamilton, A.F. Murray, D.J. Baxter, S. Churcher, and H.M. Reekie are with the Department of Electrical Engineering, University of Edinburgh, Edinburgh, EH9 3JL, Scotland, U.K.

L. Tarassenko is with the Department of Engineering Science University of Oxford, Oxford, OXI 3PJ, U.K.

IEEE Log Number 9105641.

II. Pulse Stream Signaling

This section draws out the fundamentals of pulse stream signaling and systems. This material has been covered elsewhere, but is included here in the interests of completeness. Neurobiological systems are known to operate, at least for the greater part, on such a principle, and communications systems have used PAM (pulse amplitude modulation), PWM (pulse width modulation), and PCM (pulse code modulation) for data transmission for some time.

Pulse stream encoding was first used and reported in the context of neural integration in 1987 [13], [14], and has since been used by a number of other groups (see, for example, [15]–[22]). The underlying rationale is simple:

- Analog computation is attractive in neural VLSI, for reasons of compactness, potential speed, asynchronousness, and lack of quantization effects.
- Analog signals are far from robust against noise and interference, are susceptible to process variations between devices, and are not robust against the rigors of interchip communication.
- Digital silicon processing is more readily available than analog.
- Digital signals are robust, easily transmitted and regenerated, and fast.
- Digital multiplication is area- and power-hungry.

These considerations all encourage a hybrid approach, seeking to blend the merits of both digital and analog technology. The pulse stream technique uses *digital* signals to carry information and control *analog* circuitry, while storing further *analog* information on the time axis, as will be described below. A number of possible techniques exist for coding a neural state $0 < S_i < 1$ onto a pulsed waveform V_i with

1045–9227/92/0300–xxxx$03.00

frequency ν_i, amplitude A_i, and pulse width δ_i [12]. Of these, three are relevant to the systems described in this paper.

A. Pulse Amplitude Modulation (PAM)

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

B. Pulse Width Modulation (PWM)

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

C. Pulse Frequency Modulation (PFM)

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

In summary, pulsed techniques can code information across several pulses or within a single pulse. The former enjoys an advantage in terms of accuracy, while the second sacrifices accuracy for increased bandwidth. To date, we have made no attempt to either preserve or utilize pulse phase information. It is well known that phase information enriches the complexity of neurobiological computation [23]. While we anticipate using such techniques in due course, we have thus far concentrated on developing chips that do not use phase information, while developing circuits that will ultimately be amenable to its use. The ensuing sections describe analog VLSI devices that use a combination of the above techniques, moving between the analog and digital domains as appropriate, to optimize the robustness, compactness, and speed of the associated network chips.

III. Integrated Pulse Stream Neural Networks

This section reports results from two distinct pulse stream neural networks implemented on a single test chip using European Silicon Structures' (ES2) 2 μm CMOS technology.

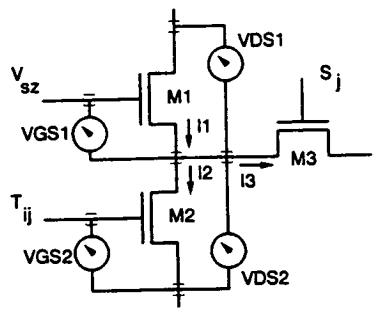


Fig. 1. Transconductance multiplier circuit.

In addition a scheme for communicating large numbers of neural states across a multiplexed data link is demonstrated.

Both networks comprise an array of synapses that implement the multiplication of an incoming pulse stream by a synaptic weight. Synaptic weights are stored dynamically as charge on a capacitor at each synaptic site and are refreshed from off-chip RAM. A neuron at the foot of each synaptic column converts the accumulated activity into an output pulse stream. Neither dynamic storage nor RAM backup memory is seen as a viable long-term solution to the synapse memory problem, and we are working towards alternative approaches, as indicated in Section IV.

A. Pulse Stream Synapse 1: Transconductance Multiplier Circuit

The heart of this synapse design, shown in Fig. 1, is the multiplier formed by transistors M1, M2, and M3. Transistors M1 and M2 output a current proportional to the weight voltage, T_{ij}, which is then pulsed by the switch transistor, M3, controlled by the incoming neural state, S_j. The resulting output current, integrated over a period of time, represents the multiplication of T_{ij} by S_j.

The operation of the multiplier can be explained with reference to the MOSFET transistor characteristic equations. The equation of interest here is that for the drain–source current, I_{DS}, for a MOSFET in the *linear* or *triode* region:

$$I_{DS} = \frac{\mu C_{ox} W}{L}\left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right]. \qquad (1)$$

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

This expression for I_{DS} contains a useful product term: $(\mu C_{ox} W)/(L) \times V_{GS} \times V_{DS}$; however, it also contains two other unwanted terms in $V_{DS} \times V_T$ and V_{DS}^2. To *remove* these nonlinear terms a second identical MOSFET, M1, is employed, as shown in Fig. 1. Assuming $V_{DS1} = V_{DS2}$ the output current is defined by

$$I_3 = \mu C_{ox} \frac{W_1}{L_1}(V_{GS1} - V_{GS2})V_{DS1} \qquad (2)$$

where V_{GS2} represents the weight voltage and V_{GS1} represents the zero weight voltage.

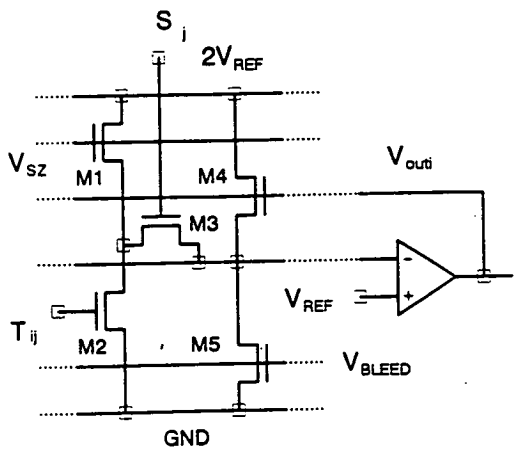


Fig. 2. Pulse stream synapse 1: transconductance multiplier.

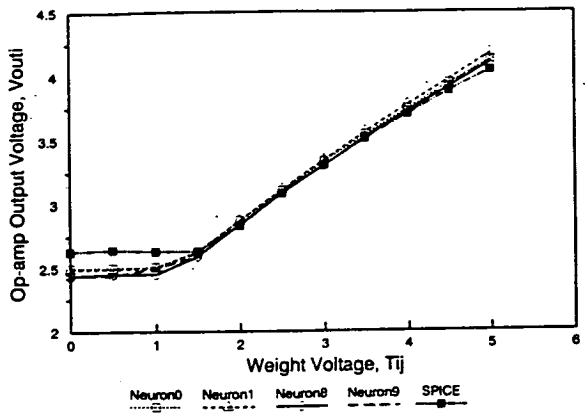


Fig. 3. Pulse stream synapse: comparison of predesign SPICE simulation results with postfabrication measurements.

This is a fairly well known *transconductance multiplier* circuit. It was reported initially for use in signal processing chips such as filters [11]. In our implementation we have used this circuit as a voltage-controlled current source by clamping V_{DS2} to a constant value; therefore I_3 is linearly dependent on V_{GS2}, which represents the synaptic weight. The output of the circuit M1/M2/M3 is therefore a stream of current pulses whose magnitude is proportional to T_{ij} and whose frequency is proportional to S_j.

To minimize variations in performance of this circuit across chip, additional buffer transistors, M4 and M5, operating in their linear region have been added as shown in Fig. 2. The operational amplifier at the foot of each postsynaptic column provides a feedback signal, V_{outi}, that controls the current in all the buffer stages in that column of synapses so that it balances the current being sourced or sunk by the multipliers. Thus the buffer stage plus the feedback operational amplifier is functionally equivalent to a standard operational amplifier current-to-voltage converter, where the resistor in the feedback loop has been replaced by transistor M4. Transistor M5 is controlled by reference voltage V_{BLEED} and allows the buffer stage to both source and sink current. The value of V_{BLEED} determines the voltage about which V_{outi} varies. In order to achieve good matching, transistors M4 and M5 should be placed physically close to the multiplier transistors.

An on-chip feedback loop incorporating a transconductance multiplier with a reference zero weight voltage at the weight input is used to determine the value of V_{SZ} automatically. This mechanism compensates for process variations between chips.

The output voltage of the operational amplifier represents a "snapshot" of all the weights switched in at a particular moment in time. This output voltage is then integrated over time to form the neural activity.

As shown in Fig. 3, which compares predesign simulation results with postfabrication measurements, the predicted results from SPICE simulations correspond closely to the measured results from fabricated circuits. These results also confirm the linear relationship between the synaptic weight voltage and the output voltage of the operational amplifier. The output voltage of the operational amplifier is plotted against the weight voltage T_{ij} with transistor M3 switched on. The maximum variation in operational amplifier output voltage over the linear weight range as a fraction of the total operational amplifier output range was found to be 6%. This result is very encouraging, and compares favorably with the 20% to 50% variation in performance quoted for simple current mirrors fabricated using the MOSIS digital process [1], [24], [25]. The synapse circuit occupies an area of 130 μm × 165 μm including the weight storage capacitor and addressing circuitry. The measurements were taken from a 10 × 10 array of synapses.

B. Voltage Integrator Circuit

As the output voltage of the operational amplifier represents a "snapshot" of all the weights switched in at a particular moment in time, the operational amplifier needs to be followed by a voltage integrator, to obtain the aggregated neural activity. This integrator is composed of a differential stage and cascode current mirrors (Fig. 4). The current I_{EXT}, through these current mirrors is determined off-chip to minimize the effects of process variation on the integrator's output current range. The output current from the integrator is directly proportional to the difference between the signals V_{outi} and the reference voltage V_{OZ}. As the integrator capacitor has been implemented as an NMOS transistor, any variations in the gain of the differential stage are tracked by the variations in the integration capacitance. Thus the rate of change of voltage will remain the same over all process variations. The resultant integrator occupies an area of 165 μm × 200 μm.

C. Pulse Stream Neuron 1: Voltage-Controlled Oscillator

The neuron is a voltage-controlled oscillator (VCO) that fires out pulses of constant width at a period determined by its neural activity input. The output state of the neuron can therefore be described as a *duty cycle*, defined as the percentage of the period the neuron output is in a high state. The transfer characteristic of the neuron is such that the duty

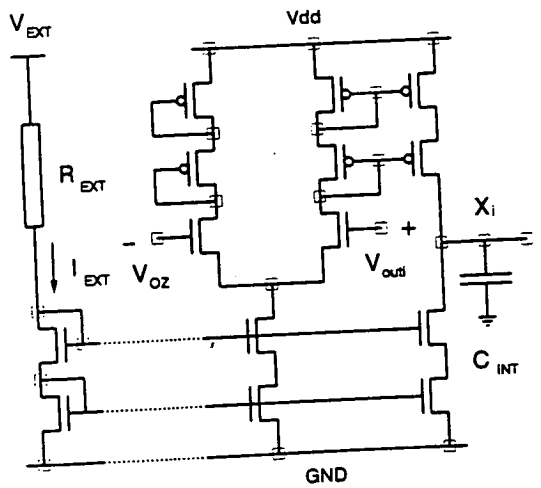


Fig. 4. Voltage integrator circuit.

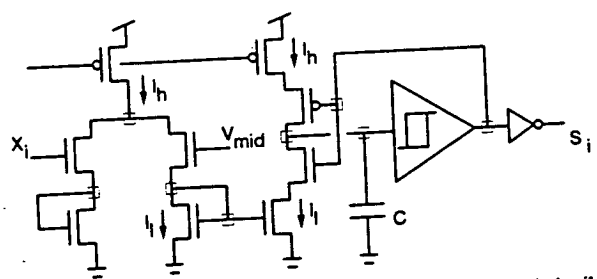


Fig. 5. Pulse stream neuron 1: voltage-controlled oscillator circuit details.

cycle varies sigmoidally from 0% to 50% with input activity voltage.

A charge-pump phase-locked loop [26] has been used to obtain an accurate phase relationship between an off-chip reference clock and an internal reference VCO. In practice the reference VCO is set for a duty cycle of 50% and the feedback technique sets I_h in Fig. 5, and hence the neuron output pulse width, to be identical to that of the reference clock. Current mirrors are then used to "copy" I_h for use in all other VCO's. It should be noted that while the off-chip reference clock and internal reference VCO are locked together, all other VCO's on chip are asynchronous to it and to each other.

A simplified circuit diagram of the VCO is shown in Fig. 5. The space between pulses can be controlled by varying I_l. In order to achieve the 0% to 50% duty cycle variation, I_l should vary between 0 and I_h. This has been achieved using a transconductance amplifier with a tail current I_h, whose transfer characteristic is naturally sigmoidal in character. This technique preserves the ratio between I_h and I_l despite variations in actual values arising from process variations. As a result, measured variations in duty cycle are less than the raw process variations across chip. By careful choice of transistor sizes in this stage, the resulting input voltage to duty cycle characteristic can be made sigmoidal.

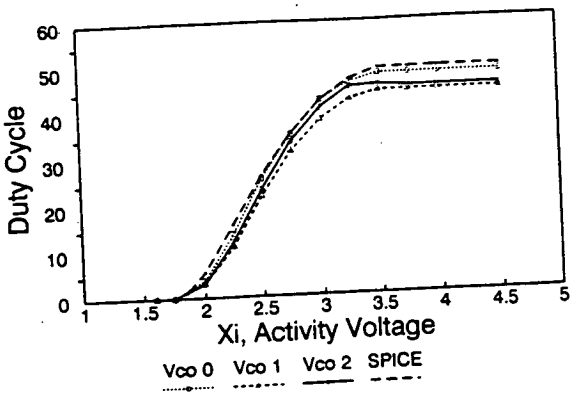


Fig. 6. Pulse stream neuron 1: comparison of SPICE simulation results with postfabrication measurements.

A comparison of the simulated characteristic and the measured characteristic of three neurons is shown in Fig. 6. The midpoint of the sigmoid characteristic has been set by the voltage V_{mid}. The mismatch between neuron characteristics is most pronounced at the extremes of the activity range. This is due to imprecise current mirroring. Variations of between 10% and 15% in the performance of current mirrors across chip have been measured.

The neuron circuit is small in area, occupying an area of only 165 μm $\times$ 165 μm. A significant cause for concern, that physically adjacent neurons might lock to each other because of mechanisms such as coupling through power supply rails, proved to be unfounded. This did not happen in practice, even when ideal conditions for lockup were created.

D. Integrated Pulse Stream Network 1

The operation of a complete section of the network is illustrated in the oscilloscope photograph of Fig. 7. The top trace shows a pulse stream arriving at the synapse inputs. The synaptic weights have been set to be fully excitatory. The integrator output, shown in the second trace, increases linearly with time.

It is interesting to note at this stage that the "activity update process" associated with this form is distinct from that of other analog neural forms. Conventional networks express the neural activity x_i as the result of the following synaptic summation:

$$x_i(t) = \left[\sum_{j=0}^{j=n-1} T_{ij} S_j(t) \right]. \tag{3}$$

In the pulse stream network presented here, the result of each pulse arriving is to either add or subtract a small package of charge to the integration capacitor, and thus increment or decrement x_i. The update process algorithm for the voltage x_i with respect to a single pulse is therefore

$$x_i(t + \Delta t) \approx x_i(t) + \delta \times \left[\sum_{j=0}^{j=n-1} T_{ij} S_j(t) \right] \tag{4}$$

where δ is controlled by the characteristics of the voltage integrator. As the integrator output, and therefore the neuron

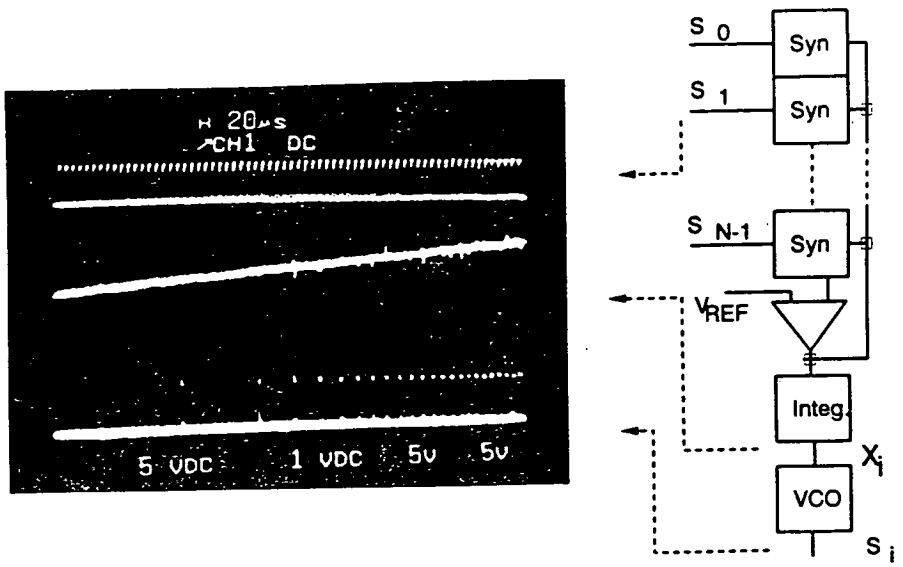


Fig. 7. Integrated pulse stream network 1: block diagram and oscilloscope traces. Top trace: pulse stream arriving at synapse inputs. Second trace: integrator output representing neural activity increasing linearly with time. Third trace: neuron output switching on with increased activity.

activity, increase, the neuron switches on, outputting a pulse stream shown in the third oscilloscope trace.

The speed of operation of this network is limited by the performance of the operational amplifier used in the synapse feedback circuit. Owing to the limited bandwidth of the operational amplifier design employed here, the input pulse width has been limited to 1 μs.

E. Self-Depleting Neural System

The neuron underlying this system is the most biologically inspired of those we have developed. It effectively depletes its own reserves of activity after each pulse, in a manner analogous to its biological exemplar. This is intrinsically interesting, and results in an efficient and elegant neuron design. It also prepares the way for more direct use of phase information as discussed in subsection II-C, preserving as it does the time of occurrence of individual pulses, rather than integrating to extract only frequency information. Other groups [17] have investigated this technique more fully, driven by a stronger biological impetus.

The neuron in this design integrates current and outputs a train of pulses of fixed pulse width, whose frequency depends on the magnitude of the input current. To first order at least, this is analogous to the process found in biological neurons. For each received impulse, the synapse either adds or subtracts packets of charge to an integration capacitor. Impulses arriving at synapses with an excitatory weight add charge, while those arriving at a synapse with an inhibitory weight remove charge. When enough charge has accumulated on the integration capacitor, the neuron fires, generating a single pulse of fixed width. The process of neuronal firing removes all charge stored on the integration capacitor (we refer to this process as self-depletion). Therefore, in order for subsequent pulses to be generated by the neuron, further stimulus is required at the synapses feeding that neuron.

The self-depleting property of the neuron has interesting consequences for network dynamics. The neuron has no long-term memory of previous stimulation as this is effectively lost after each neural pulse. The update process for the activity voltage in (4) differs in that the term $x_i(t)$ is reset to zero after each neural pulse.

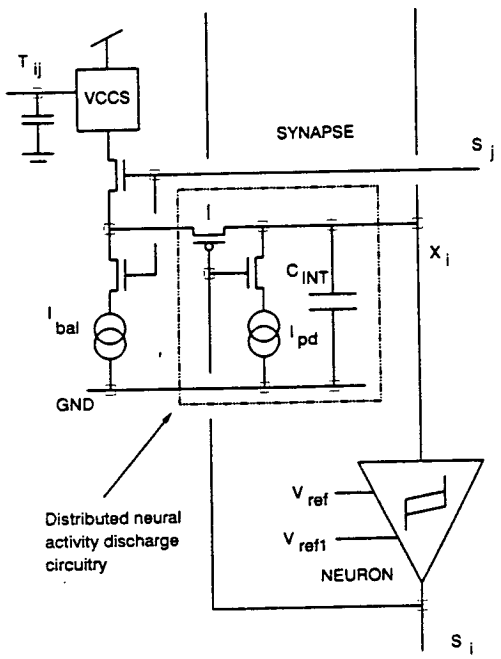


Fig. 8. Self-depleting network: synapse and neuron circuit details.

F. Pulse Stream Synapse 2: Gated Current Source Circuit

A schematic diagram of the synapse and neuron is shown in Fig. 8. The synaptic weight, T_{ij}, stored dynamically as a voltage at each synapse, controls a voltage-controlled current source (VCCS). The current source comprises a transistor in its linear region, which has its drain–source current set by T_{ij}; this current is subsequently mirrored onto the output of the VCCS. A pulse, S_j, arriving at the synapse input gates this current onto the activity capacitor while simultaneously removing a fixed charge via a balance current. The result of this operation can yield either a net increase in charge on the activity capacitor (representing excitation) or a net decreased (representing inhibition).

The synapse circuit occupies an area of 130 μm × 140 μm including the weight storage capacitor and addressing circuitry. The activity capacitor is distributed through a column of synapses, making the synapse design 100% cascadable.

G. Pulse Stream Neuron 2: Self-Depleting Form

Charge from the synapses flows onto the distributed activity capacitor, causing the activity voltage to rise. When the voltage reaches a reference level, V_{ref}, the neuron fires a pulse and simultaneously discharges the activity capacitor to a second lower reference voltage, V_{ref1} (the neuron has depleted itself). The output pulse width from the neuron is determined by the rate of discharge of the activity capacitor. Further excitation causes the voltage on the activity capacitor to rise and produce a further pulse.

In common with the synapse layout design and in order to ease cascadability, the self-depletion circuit is distributed across synapse sites. The remainder of the neuron circuit occupies an area of 100 μm × 140 μm.

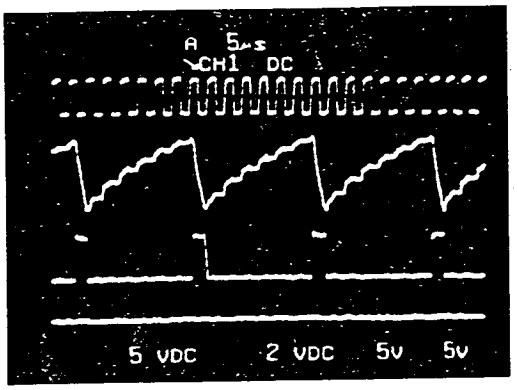


Fig. 9. Integrated pulse stream network 2: oscilloscope traces. Top trace: pulse stream arriving at synapse inputs. Second trace: activity increasing as each pulse arrives. Third trace: neuron output pulses.

H. Integrated Pulse Stream Network 2

Fig. 9 illustrates the operation of the self-depleting network in silicon. The top trace shows a pulse stream arriving at the input to an excitatory synapse. The second trace shows the activity increasing in steps as each pulse arrives. Activity increases until it reaches the triggering threshold of the neuron circuit and then an output pulse, shown in the lower trace, is produced. The process repeats for subsequent output pulses. While the input pulse width to this system has been set to 1 μs, the system can be operated with pulse widths down to 0.1 μs.

The measured characteristics of the self-depleting network are compared against those obtained from SPICE simulations in Fig. 10. The relationship between the synaptic weight voltage and output neuron frequency for a fixed input is approximately linear over the required voltage range.

I. A Novel, Self-Timed Communications Scheme

While analog techniques have facilitated compact and accurate circuits for synapse and neuron, we use digital signals to communicate neural states around the chip. The robust nature of digital signals make them ideal for communicating neural information over chip boundaries. Clearly, if we wish to implement large neural systems on chip, then it is impractical to allocate a single pin to each neural input or output. Here we demonstrate a self-timed communication scheme that multiplexes neural state information onto or off chip asynchronously.

In pulse-stream neural systems the state of a neuron is encoded in its pulse firing rate. A "snapshot" of this information, however, is also contained in the space between individual pulses and it is this information that we use to communicate across chip boundaries. The use of an asynchronous handshake between the device transmitting neural states and the receiving device allows the communication to proceed as soon as possible, without resorting to the use of fixed time slots.

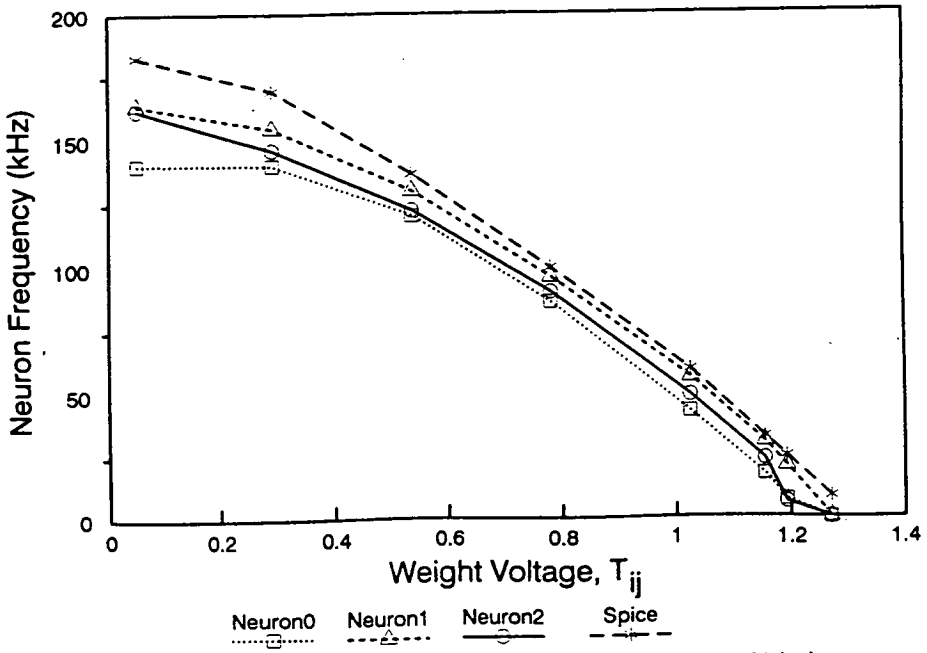


Fig. 10. Integrated pulse stream network 2: comparison of SPICE simulation results with postfabrication measurements.

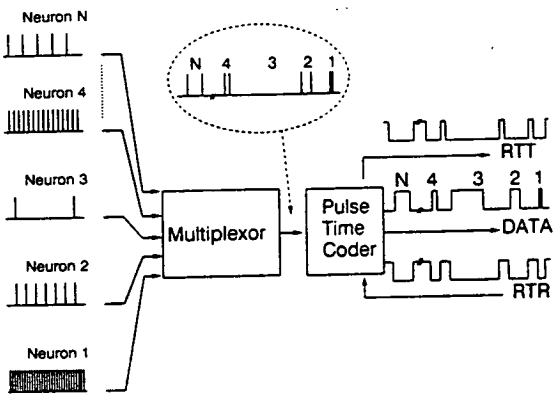


Fig. 11. Self-timed communications scheme: block diagram.

The details of the communication scheme are shown in Fig. 11. Each neuron in turn has its state information encoded onto a single digital signal line as a pulse whose length represents the period between individual output pulses from the current neuron. A pair of handshake lines, referred to here as RTT (request to transmit) and RTR (ready to receive), control the flow of data between the transmitting and receiving devices. In practice, algorithmic state machines have been used to implement these control functions.

The operation of the data transfer mechanism is shown in parts (a) and (b) of Fig. 12, where the output of neuron 1 is shown in the top trace, RTT in the second, RTR in the third and the DATA signal in the final trace. In Fig. 12(a) both the transmitting and the receiving state machine are clocked at 2 MHz and the data transfer for the neuron in question is completed in approximately 12 μs (for an example input duty cycle of approximately 15%). Fig. 12(b) shows the transmitting state machine communicating to an IBM PC, which is relatively slow, and the data transfer operation now takes approximately 45 μs. Communication therefore proceeds at the speed of the slowest device, as in any self-timed system.

The transmitter can send information to other chips, where the pulse width information is decoded and used to regenerate the original pulse stream signal. Since no addressing information is sent with the data, the only provision for correct operation is that the transmitting device should have the same number of neurons transmitting information as the receiving device. The links between different devices are defined by direct hardware connections at board level. Thus different networks can be realized by using different interconnection strategies. As demonstrated in Fig. 12(b), the transmitter is also capable of transmitting information to a host computer via conventional parallel interface ports, using the same communication circuitry and protocols as developed for interchip communication.

IV. Discussion

This paper presents results from totally integrated pulse stream neural network chips. Novel analog circuit forms have been implemented on a digital process and their correct operation demonstrated. Our design methodology has actively reduced the effects of process variation that result in poor transistor matching, yielding more uniformly matched circuit elements. The small size of the circuit elements will allow large pulse stream chips to be fabricated.

These achievements allow the implementation of large neural systems that require a multichip architecture, for example, multilayer networks with each layer mapped onto a separate

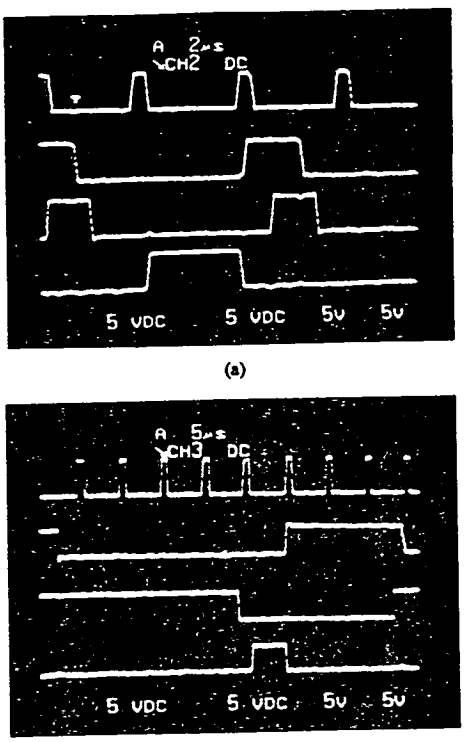


Fig. 12. Self-timed communications scheme: oscilloscope traces. Top trace: neuron output. Second trace: request to transmit (RTT). Third trace: ready to receive (RTR). Fourth trace: encoded DATA signal. (a) Transfer between two chips. (b) Transfer between chip and IBM PC.

chip. To this end we have designed a large pulse stream demonstrator chip using the first of the two systems outlined in this paper. This consists of an array of 120 × 30 synapses with 30 on-chip neurons. This chip has been implemented using ES2's 1.5 μm CMOS digital process and occupies an area of 10 mm by 9.5 mm. The estimated worst-case power consumption for this device is 350 mW, which is well within the safe limits for a chip of this size.

The intended applications for these chips are in the areas of pattern recognition and optimization. Over the last few years a number of other designs have been fabricated [27]–[29]. Indeed, pulse stream neural networks have been applied successfully to the problem of robot localization [12], [30] as part of a real-time autonomous robot navigation system based on neural network modules where a small array of synapses were integrated onto a single chip. They also formed part of a simple isolated word-recognition module [30].

While pulse frequency modulation clearly has many advantages, other pulse signaling techniques can yield faster computation times. To this end, we are experimenting with pulse width modulation techniques. The problems associated with high transient power demand arising from the synchronism of signal edges alluded to earlier have been addressed and solutions found. In these circuits the computation

$$\sum_{j=0}^{j=n-1} T_{ij} S_j(t)$$

is calculated with every pulse. Also under investigation is a continuous time system based upon the transconductance multiplier principle outlined earlier.

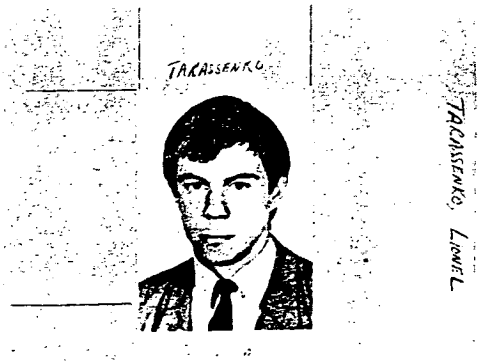
We have therefore demonstrated small, efficient synapse and neuron circuits and addressed the issues of process variations, cascadability, and intercommunication, all of vital importance to neural VLSI. In each of these areas we have made significant advances toward generic solutions. In the area of weight storage and appropriate learning strategies for VLSI, however, there is still much to be done. We are developing a novel amorphous-silicon device for nonvolatile analog weight storage [31] and learning schemes amenable to VLSI on-chip implementation [32], [33]. In both areas, the advantages of pulse stream techniques are again becoming evident, and it is our intention to continue in this direction.

REFERENCES

[1] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1988.
[2] Y. Le Cun *et al.*, "Optical character recognition and neural-net chips," in *Proc. Int. Neural Network Conf.—INNC-90* (Paris), 1990, pp. 651–655.
[3] J. P. Sage, R. S. Withers, and K. Thompson, "MNOS/CCD circuits for neural network implementations," in *Proc. Int. Symp. Circuits Syst.*, May 1989, pp. 1207–1209.
[4] J. Hopfield, "The effectiveness of analog, neural net, hardware," *Network*, vol. 1, no. 1, pp. 27–40, 1990.
[5] E. Vittoz *et al.*, "Analog storage of adjustable synaptic weights," in *Proc. ITG/IEEE Workshop Microelectronics for Neural Networks* (Dortmund, Germany), June 1990, pp. 69–79.
[6] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network (ETANN) with 10240 'floating gate' synapses," in *Proc. Int. Joint Conf. Neural Networks—IJCNN89*, June 1989, pp. 191–196.
[7] H. Card and W. Moore, "Implementation of plasticity in MOS synapses," in *Proc. 1st IEE Conf. Artificial Neural Networks*, 1989, pp. 33–36.
[8] L. A. Akers, M. R. Walker, D. K. Ferry, and R. O. Grondin, "A limited interconnect, highly layered synthetic neural architecture," in *VLSI for Artificial Intelligence*, W. R. Moore, Ed., Norwell, MA: Kluwer, July 1988, pp. 218–226.
[9] A. F. Murray, "Silicon implementations of neural networks," *Proc. Inst. Elec. Eng.*, pt. F, vol. 138, no. 1, pp. 3–12, 1991.
[10] B. A. Gilbert, "Precise four-quadrant multiplier with sub-nanosecond response," *IEEE J. Solid-State Circuits*, vol. SC-3, pp. 365–373, 1968.
[11] P. B. Denyer and J. Mavor, "MOST transconductance multipliers for array applications," *Proc. Inst. Elec. Eng.*, pt. 1, vol. 128, no. 3, pp. 81–86, June 1981.
[12] A. F. Murray, D. Del Corso, and L. Tarassenko, "Pulse-stream VLSI neural networks—Mixing analog and digital techniques," *IEEE Trans. Neural Networks*, vol. 2, pp. 193–204, Mar. 1991.
[13] A. F. Murray and A. V. W. Smith, "Asynchronous arithmetic for VLSI neural systems," *Electron. Lett.*, vol. 23, no. 12, pp. 642–643, June 1987.
[14] A. F. Murray and A. V. W. Smith, "A novel computational and signalling method for VLSI neural networks," in *Proc. European Solid State Circuits Conf.*, 1987, pp. 19–22.
[15] D. Del Corso, F. Gregoretti, C. Pellegrini, and L. M. Reyneir, "An artificial neural network based on multiplexed pulse streams," in *Proc. ITG/IEEE Workshop Microelectronics for Neural Networks* (Dortmund), June 1990, pp. 28–39.
[16] A. Sigglekow, A. J. Beltman, J. A. G. Nijhuis, and L. Spaanenburg, "Pulse-density modulated neural networks on a semi-custom gate forest," in *Proc. ITG/IEEE Workshop Microelectronics for Neural Networks* (Dortmund, Germany), June 1990, pp. 16–27.
[17] J. Meador, A. Wu, C. Cole, N. Nintunze, and P. Chintrakulchai, "Programmable impulse neural circuits," *IEEE Trans. Neural Networks*, vol. 2, pp. 101–109, Jan. 1991.

[18] N. El-Leithy, M. Zaghloul, and R. W. Newcomb, "Implementation of pulse-coded neural networks," in *Proc. 27th Conf. Decision and Control*, 1988, pp. 334–336.
[19] S. Ryckebusch, C. Mead, and J. Bower, "Modelling small oscillating biological networks in analog VLSI," in *Proc. Neural Inform. Processing Syst. Conf.*, Dec. 1988, pp. 384–393.
[20] J. Tomberg, "Fully digital neural network implementation based on pulse density modulation," in *Proc. IEEE Custom Integrated Circuits Conf.* (San Diego), May 1989.
[21] Y. Haria, "A digital neuro-chip with unlimited connectability for large scale neural networks," in *Proc. IJCNN* (Washington), 1989, pp. 163–169.
[22] P. M. Daniell, W. A. J. Waller, and D. A. Bisset, "An implementation of fully analogue sum-of-product neural models," in *Proc. 1st IEE Conf. Artificial Neural Networks*, 1989, pp. 52–56.
[23] J. A. Simmons, "Acoustic-imaging computations by echolocating bats: Unification of diversely-represented stimulus features into whole images," in *Proc. Neural Inform. Processing Systems Conf.*, 1989, pp. 2–9.
[24] M. A. Sivilotti, M. R. Emerling, and C. Mead, "VLSI architectures for implementation of neural networks," in *Proc. AIP Conf. Neural Networks for Computing* (Snowbird), 1986, pp. 408–413.
[25] E. Vittoz, "Analog VLSI implementations of neural networks," in *Journees d'Electroniques 1989*. Lausanne: Processesses Polytechniques Romandes, 1989.
[26] D. Jeong, G. Borriello, D. A. Hodges, and R. H. Katz, "Design of PLL-based clock generation circuits," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 255–261, Apr. 1987.
[27] A. F. Murray, L. Tarassenko, and A. Hamilton, "Programmable analogue pulse-firing neural networks," in *Proc. Neural Inform. Processing Systems (NIPS) Conf.*, 1988, pp. 671–677.
[28] A. F. Murray, A. Hamilton, H. M. Reekie, and L. Tarassenko, "Pulse-stream arithmetic in programmable neural networks," in *Proc. Int. Symp. Circuits Syst.* (Portland, OR), 1989, pp. 1210–1212.
[29] A. F. Murray *et al.*, "Pulse-firing neural chips for hundreds of neurons," in *Proc. Neural Inform. Processing Systems (NIPS) Conf.*, 1990, pp. 785–792.
[30] L. Tarassenko, M. Brownlow, and A. F. Murray, "VLSI neural networks for autonomous robot navigation," in *Proc. Int. Neural Network Conf.—INNC-90* (Paris), vol. 1, 1990, pp. 213–216.
[31] M. J. Rose *et al.*, "Amorphous silicon analogue memory devices," *J. Non-Cryst. Sol.*, vol. 115, pp. 168–170, 1989.
[32] A. F. Murray, "Analog VLSI and multi-layer perceptrons—Accuracy, noise and on-chip learning," presented at Int. Conf. Neural Networks, Munich, Germany, 1991.
[33] A. F. Murray, "Analogue noise-enhanced learning in neural network circuits," *Electron. Lett.*, vol. 2, no. 17, pp. 1546–1548, 1991.

Alister Hamilton is a Research Fellow in the Department of Electrical Engineering at the University of Edinburgh, where he is working toward the Ph.D. He has a B.Sc. in communication and electronic engineering and an M.Sc. in digital techniques. His research interest have been in image processing with the U.K. Medical Research Council and more recently in the implementation of neural networks in VLSI.



Alan F. Murray was born in 1953 in Edinburgh, where he also went to school. He has a wife and two children. In 1975 he received a BSc Hons (1st Class) in physics from the University of Edinburgh, and a Ph.D. in solid-state physics in 1978.

Since then he has worked for three years as a research physicist (two in Canada) and for three years as an integrated circuit design engineer. Since 1984 he has been a lecturer in Electrical Engineering at Edinburgh University. VLSI modeling of neural networks has been his primary research interest since 1985. In 1986, he developed the unique "pulse stream" method for neural integration, which has been implemented and extended over the past five years in close collaboration with Lionel Tarassenko. Dr. Murray has over 90 publications, including an undergraduate textbook.

Donald J. Baxter received the B.Sc. degree in electronic and electrical engineering from the University of Edinburgh, where he is currently working toward the Ph.D. degree. His main research areas cover the development of process-invariant analog VLSI CMOS circuits and the use of neural networks for solving optimization problems. Other activities include rugby and golf.

Stephen Churcher graduated in 1989 from the University of Edinburgh with a B.Sc. (first class honors) in electronics and electrical engineering. Since then he has been studying for a Ph.D. at the same institution under the supervision of Alan Murray, with technical support and sponsorship from British Aerospace. His research activities center on the design of analog CMOS VLSI neural networks for computer vision tasks, with particular emphasis on region classification in natural scenes.

Mr. Churcher is the author of ten publications in this field and is an associate member of the Institution of Electrical Engineers. When not working, he enjoys technical rock/ice climbing, white water kayaking, ski-mountaineering, and badminton.

H. Martin Reekie (M'83) was born on July 10, 1956. He received a B.Sc. degree in mathematics and statistics and a Ph.D. degree in electronics, both from Edinburgh University, in 1978 and 1981, respectively.

After short periods as a Research Assistant at Edinburgh University and with Siemens in Austria, he became a lecturer at Edinburgh University. His research interests are in analog circuit design and neural networks.