Hardware Neural Systems for Applications: A Pulsed Analog Approach

Geoffrey Bruce Jackson

Thesis submitted for the degree of
Doctor of Philosophy
The University of Edinburgh
February 1996

Abstract

In recent years the considerable interest in the biologically inspired computational paradigm of artificial neural networks has led to a drive to realise these structures as VLSI hardware. At Edinburgh University research has focused on pulse stream neural networks in which neural states are encoded in the time domain as a stream of digital pulses. To date research centred on developing analog CMOS circuits to implement neural functions. In this thesis these circuits are developed and higher, system level issues addressed in order to produce a neural network system suited to use in real-world applications.

To discover the key requirements for use in real-world applications, examples of application based hardware systems are reviewed, as is the field of pulse stream neural networks. These requirements led to the design of a VLSI chip, EPSILON II; a pulse stream neural chip optimised for use on the boundary of the analog domain of the real-world and the digital domain of conventional computing. The EPSILON processor card (EPC) places this chip in a system level framework that oversees chip operation and provides interfaces to analog signals, a standard digital bus and other EPCs. The system level approach taken provides a versatile platform for prototyping applications while operating with minimal host supervision.

To demonstrate the versatility of this approach several applications were developed that utilised this hardware. Foremost amongst these was an autonomous mobile robot that utilises the analog nature of the hardware to provide a direct interface to real-world sensors. Also presented are a series of experiments investigating back-propagation learning on a variety of MLP problems. This study reveals the limits and practicalities of training hardware neural networks, in particular the effects of limited weight dynamic range were found to be of primary importance.

From this work conclusions are drawn as to the effectiveness and future development of hardware neural computation; specifically the ability to interface to the analog domain and the issues involved in interfacing to conventional computing devices are highlighted.

Declaration

I declare that this thesis has been completed by myself and that, except where indicated to the contrary, the research documented is entirely my own.

Acknowledgements

I would like to express my thanks to the following people for their invaluable assistance during the course of my PhD.

- My supervisors, Alan Murray and Alister Hamilton, for their advice and assistance.
- The members of the Integrated Systems Group, in particular Andrew Holmes, Robin Woodburn, Andrew Murray and Pete Edwards.
- The Association of Commonwealth Universities and The British Council, in particular Martin Sexton, Jo Bundred and Graham Megennis, for the Commonwealth Scholarship that made it all possible.
- Alison Cherry, Ann Duncan and Lynne Munro for keeping me sane.
- Jey Ngole and his colleagues for access to the ATM data.
- Graham Cairns and his colleagues for access to data for the 1-of-N classification problems.
- Ulrich Nehmzow for initial discussions on instinct-rule robots and the loan of Alder for early experimentation.
- My Aussie visitors; Dushan, Jacqui B & M, Liz, Matt, Bec, Dave and Nic, for welcome distraction and cultural refreshment.
- Last but by no means least Roger and Barbara, my parents, for their continual encouragement and support throughout my entire education.

Contents

List of Figures

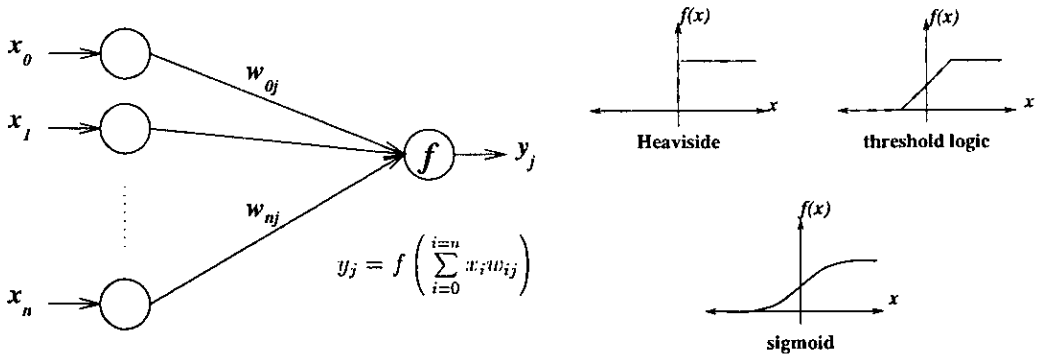
List of Tables

Chapter 1
Introduction

This chapter discusses the basic concepts of neural networks and the motivation for their implementation in hardware. It moves on to introduce the work of this thesis; developing hardware neural networks in such a way as to facilitate their use in applications. The chapter concludes with a formal statement of the aim of the thesis and an overview of the thesis structure.

1.1 Background and Motivation

It is self-evident that biological systems routinely carry out computational tasks, such as vision and speech, that are beyond the scope of present artificial systems. Fields of science, such as neurobiology and neurophysiology, evolved to investigate the structure and mechanics of biological nervous systems. These fields revealed that the processing techniques employed by biological systems consist of massively parallel and highly interconnected networks of relatively simple processing elements – neurons. Neural networks are a computational paradigm inspired by these processing techniques observed in biological systems. The field can be traced back to the early 1940's when McCulloch and Pitts first proposed a model for the biological neuron[77]. The development of a mathematical model, and the later advent of serial computers capable of simulating networks of neurons, formed the backbone of neural network development.



(a) Neuron Model **(b) Various Thresholding Functions**

Figure 1–1: McCulloch-Pitts Model of a Neuron

Figure 1–1(a) demonstrates the basic McCulloch-Pitts model of a neuron where inputs x_i are weighted by a variable weight w_{ij}, summed together then undergo a thresholding function f, to produce an output y_j. The thresholding function takes a variety of forms dependent on the neural model and learning algorithm employed, some examples are shown in figure 1–1(b).

The early 1960's saw methods introduced to adapt the weights of the McCulloch-Pitts neuron to perform useful functions:

- Rosenblatt introduced the Perceptron[99], a neural structure with a heaviside non-linearity. Training of the Perceptron was based on Hebb's Law [48] which relates the biological principle that a synaptic junction is reinforced if repeatedly excited.

- Widrow and Hoff with the ADALINE[1] introduced the concept of adapting weights proportional to an error term with their learning rule known as the Widrow–Hoff delta rule. This structure was able to perform such tasks as adaptive filtering and equalisation[116].

Interest in the field was diminished temporarily when Minsky and Papert showed that the single layered neural structures of the day could only solve linearly separable problems[80]. The field remained virtually dormant until the 1980's when Rumelhart, Hinton and Williams[100] demonstrated the gradient descent *back-propagation* algorithm for training the Multi-layer Perceptron (MLP), a structure consisting of multiple layers of neurons where outputs of the preceding layer feed the next. The ability of this architecture to form arbitrary mappings between input and output re-kindled interest in neural networks.

Since this time other network structures and training methods have emerged such as Kohonen's self-organising feature maps [62], Radial Basis Functions[10] and Adaptive Resonance Theory (ART)[20].

A study of recent literature shows examples of neural networks providing solutions, often superior to conventional methods, to problems such as:

- Image analysis, for example optical character recognition[66,94], medical image analysis[74, 101].

- Optimisation and control, for example the travelling salesman problem[8], the inverted pendulum problem[6,107], job scheduling[28] and robotic control and navigation[105, 95].

- Speech recognition or analysis systems[59,75].

- Classification, for example classification of sleep patterns [104], aircraft identity from radar signals [21] or cardiac arrhythmia [58].

[1] ADAptive LInear NEurons

It is these types of successes that provide ample motivation for further development of the neural network field.

1.2 Hardware and Implementation

The fact that neural networks are parallel architectures implies a large computational overhead when they are implemented on a conventional serial machine. For this reason much research has been conducted into developing dedicated VLSI hardware to assist in neural computation. Two major techniques are available to do this: digital and analog.

Digital technology has produced examples of neural architectures such as the CNAPS[2] and HANNIBAL[87] chips. The advantages of digital technology are:

- High speed operation.
- Conventional technology.
- Predictable precision.
- High noise immunity.

The disadvantages are that:

- The technology is suited to a predominately serial architecture. Large bus sizes are difficult to distribute in a parallel manner.
- Operational blocks such as multipliers require considerable silicon area and consume significant power.
- No direct interface to the *real-world*, thus A/D conversion is required to interface to real-world data.

Analog technology provides the possibility of compact, low power multiplier circuits, parallel operation and direct analog interfacing. These reasons make analog implementation the preferred technology for the applications area studied in this thesis. A variety of techniques have been used for analog neural network implementation, for example:

- Fully analog designs using Gilbert multipliers and EEPROM weight storage[51].
- Designs based around multiplying digital-to-analog converters[27].
- Designs using MOS sub-threshold techniques[7,112].

- Designs using CCD structures[22].
- Switched capacitor implementations[98,110].
- Designs encoding neural states as pulses (pulse stream)[43,84].
- Designs emulating biological structures such as the retina[71,78] and cochlea[72].

Yet despite this wealth of hardware research few practical applications have emerged into general use. Going back to 1988 a DARPA[2] study of neural networks revealed that of the 77 neural network applications investigated only 4 had resulted in field tested systems[115]. Furthermore, none of these used dedicated neural network hardware. The situation at present has progressed little. Is this a failure to address the issues of blending conventional computing technology with neural networks to produce practical solutions? It is the author's belief that this is a major factor. This thesis takes the pulse stream methodology, an implementation scheme whereby analog neural states are encoded in the time domain as a digital pulse stream, and develops this methodology with the aim of integration into conventional systems for use in applications.

1.3 Applications and Practicalities of Neural Networks.

Defining the type of applications to be studied is important as neural network technology must compete with more conventional digital techniques in solving real-world problems. Neural networks must concentrate on areas where their advantages:

- Parallelism.
- Speed.
- Analog nature.
- Adaptability.

outweigh their disadvantages:

- The inability to interrogate a solution fully.
- Unproven technology.
- Specialist nature.

[2]**D**efence **A**dvanced **R**esearch **P**rojects **A**gency

Such an area is the boundary of the real, analog world and digital processing, such as pre-processing/interpretation of analog sensor data or control problems involving real-time analog sensor signals. Here a modest neural network can act as an *intelligent analog-to-digital converter* presenting preprocessed information to its host. The key requirements of this technological development are that a device must:

- Work directly with analog signals.
- Accept digital information from its host.
- Act autonomously from, and interface simply to, the host system.
- Provide a moderate size network to process data.
- Have the potential for a highly integrated, low cost solution.

Examining the pulse stream methodology employed in this thesis, its areas of strength can be summarised as:

- Analog or digital inputs.
- Digital compatible outputs.
- Compact, low power.
- Cheap CMOS implementation.
- Modest size.
- Scalable and cascadable design.

The pulse stream methodology has many of the requirements for the type of applications use described earlier, such as the advantage of working on the boundaries of the analog and digital domains. That it lacks some of the other requirements, as do all implementation methodologies, calls for the necessity of addressing these areas at system level. The EPSILON chip, the predecessor of this work, demonstrated this as the lack of several of the requirements listed above made its use in practical systems difficult. This thesis evolves strategies and hardware to address these requirements to produce a pulse stream neural network implementation that operates on this interface between the analog real-world and the digital domain of conventional computing. It then investigates the practicalities of such an approach through development of simple applications.

1.4 Aims of the Project

Having introduced the field to which it contributes, the aim of this project can be summarised as:

> **Aim: To study the practical ramifications and issues involved in the development of pulsed analog hardware neural networks for use in real-world applications.**

This study was approached via a four pronged strategy:

1. A study of existing hardware neural network applications was made, examining their strengths and weaknesses.

2. A development of existing pulse stream VLSI was undertaken to provide a base for system level development.

3. Major issues raised by the study resulted in system level development to satisfy the requirements of applications use. Issues particularly focussed on were; interfaces to analog and digital domains and autonomous operation.

4. Finally experiments are conducted on demonstration applications to verify and test the system and further investigate the practical ramifications of the issues raised.

The basic premise on which this work depends is:

> Neural networks should be applied at a system level and act on the boundary between information domains; specifically the analog "real-world" and digital systems. This can be achieved with networks of modest size imbedded in a system architecture that allows efficient data interaction with the real world and conventional digital processors.

The approach is summarised in figure 1–2 where the pulse stream based neural system that was developed provides an interface between the analog input domain and the digital domain of conventional computing.

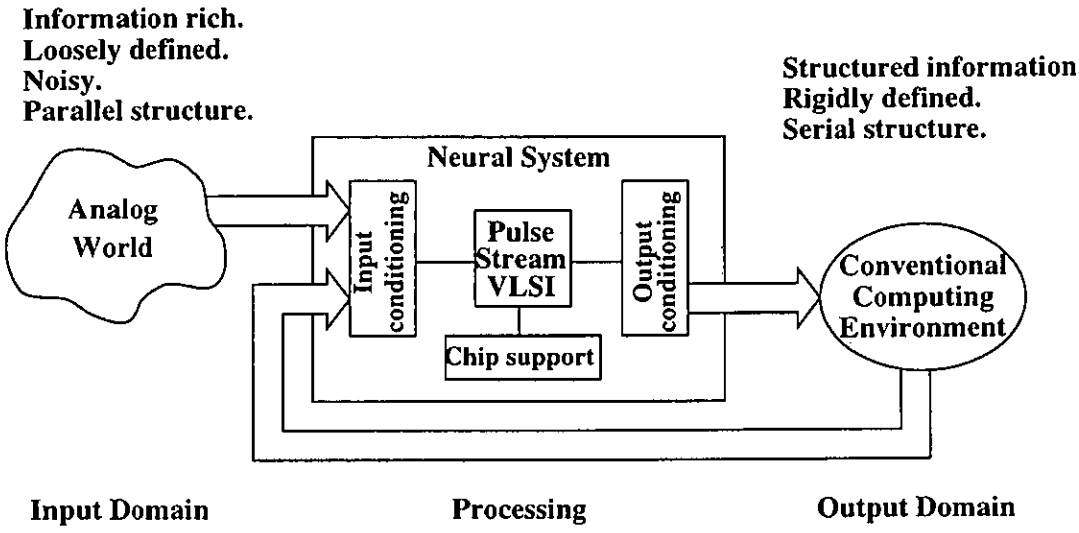


Figure 1–2: Project Overview.

1.5 Thesis Outline

Part I contains the principal literature review sections of the thesis. Chapter 2 investigates the hardware implementation of neural networks using pulse stream techniques, the methodology used in the hardware design of the project.

Chapter 3 examines the use of hardware neural networks in an applications environment by reviewing some of the most successful of recent applications directed implementations.

In Chapter 4 issues raised by the review are discussed and the design methodology presented for the hardware specification.

Part II of the thesis details the design of the hardware constructed; Chapter 5 presents the work done at a VLSI design level for this thesis – the EPSILON II chip. The design of the chip is presented along with characterisation results. Chapter 6 is devoted to the system level development of the EPSILON processor card (EPC). This work embeds the EPSILON II chip in system with interfaces to analog data, other EPSILON II chips and a conventional digital bus.

Part III discusses the use of the EPSILON processor card system for applications development. Chapter 7 configures two EPCs as a multi-layer perceptron. This is used to investigate the practicalities of training the hardware network in the presence of analog hardware non-idealities. A variety of problems were used in this investigation:

1. An artificial character recognition problem which allowed a graded problem complexity to aid initial investigation and model development.

2. A more complex real-world problem of a link admission controller for ATM communications networks. This problem was used to study the performance of the hardware as a function approximator.

3. The final set of experiments examined the performance of the EPC in solving three real-world 1-of-N classification problems.

Chapter 8 presents an autonomous mobile robot named Kryton which utilises the analog input capabilities of the EPC to map analog sensor input to motor control outputs.

Finally Chapter 9 discusses the issues raised by the thesis and draws conclusions as to the success of the work.

1.6 Areas of Contribution

The work of this thesis focuses on a study of the issues and practicalities of placing pulse stream neural networks into an applications context. As part of this study, design and fabrication of a VLSI chip was undertaken to meet the requirements revealed for the intended use on boundary of the analog and digital domains. The VLSI section utilises proven circuit structures for synaptic and neural functions, however improvements have been made over previous use of these circuits through architecture changes and judicious circuit and layout modifications. This is described in Chapter 5.

The key result of this thesis was the system level philosophy whereby the needs of neural hardware were addressed in the context of applications. The research revealed the necessity to embed neural VLSI in a system framework that is capable of interfacing to larger digital systems along with the input domain of the analog real-world.

The thesis concludes with a series of demonstration applications. These were designed to test and evaluate the hardware and study the issues involved with practical application of the hardware.

Chapter 7 studied issues involved in training the hardware under practical conditions. The principal issues raised here were the effects that limited dynamic range in the hardware weight set had on network performance and problem solving ability. Chapter 8 demonstrated the hardware in a situation closely matched to the perceived primary applications area: a direct analog interface to real-world data with neural processing, interacting with a higher-level digital system. From the success of this the validity of the philosophy developed was confirmed.

1.7 Summary

This chapter has introduced the field of neural network study and discussed the motivation of hardware implementation. It concludes that applications best served by hardware neural network technology lie on the interface between the analog and digital domains and that hardware development should reflect this. Consistent with this the aim of the thesis was presented and structure of the thesis summarised.

Part I

Hardware Neural Networks: Concepts and Issues

Chapter 2

Pulse Stream Neural Computation

2.1 Introduction

This chapter introduces the principles of pulse stream neural computation as a hardware implementation methodology. It examines the development of the field and reviews the strengths of different methods when compared to other techniques. Following this the EPSILON design will be discussed as this forms the basis of the chip-level design of this thesis.

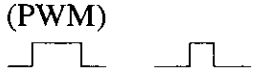
2.2 Background and Implementation Issues

Drawing inspiration from the pulse based nature of biological systems together with the engineering practicalities of utilising cheap and available digital VLSI processes, the pulse-stream methodology has evolved rapidly since its inception by Murray and Smith in 1987[85]. Pulse-stream techniques are characterised by their encoding (modulation) of neural states (or occasionally weights) as pulses. There are several techniques used for this:

Pulse frequency modulation (PFM)

Here the neural state is encoded as the frequency of the pulse stream. It may be done by varying the duty cycle of the signal (as shown to the left), or by keeping a constant duty cycle. Modulation of this form is normally achieved with the use of a voltage controlled oscillator (VCO).

Pulse Width Modulation (PWM)



In this scheme the neural state is encoded in the width (on-time) of a pulse. It is generally a synchronous scheme where pulses are guaranteed to be present in a fixed maximum time interval.

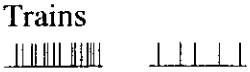
Pulse Density Modulation (PDM)

The value of the PDM signal is defined by the relation between the number of high pulses (N_+) and low pulses (N_-). For example for the coding

$$S_i = \frac{(N_+ - N_-)}{(N_+ + N_-)}$$

when $N_+ = N_-$, $S_i = 0$ and the scheme encodes values between $+1$ and -1.

Stochastic Pulse Trains



A stochastic pulse train encodes the value of S_i in the density of stochastic (that is randomly distributed) pulses. This scheme is used mainly in digital pulse stream implementations.

Encoding neural states as pulses offers several advantages when compared to the use of analog voltages or currents:

- Information is encoded in the time domain and signal levels are digital. This enables easy regeneration and distribution of signals by conventional digital methods, such as invertor chains. This form of communication is thus much less susceptible to noise than analog voltages or currents, providing an efficient method of communication between neural chips.

- Combining analog techniques on-chip with these digital signals, compact analog multipliers can be constructed. This combines the advantages of compact analog computation with the ease of digital signal distribution.

- Although conversion (demodulation) is necessary to interface pulse-stream neural chips to digital hosts; this can be done with conventional digital techniques without the need for A/D converters. This can make demands at system level easier to realise.

However there are also disadvantages in using pulse streams:

- Switching noise introduced by high frequency digital signals can couple with sensitive analog circuitry.

- In all but the PDM scheme multiplication is generally restricted to two quadrants. Four-quadrant multiplication is considerably more difficult to implement.

- To interface to analog signals a pulse modulator is needed.

Different implementations address these strengths and weaknesses differently. Pulse stream implementations fall into three broad categories depending upon the method used to perform synaptic multiplication, these are:

1. Digital implementations.

2. Switched-capacitor implementations.

3. Analog implementations.

The remainder of this section will present examples of these implementation techniques and discuss their advantages and disadvantages before moving on to discuss the EPSILON chip and the pulse stream methodology employed on it.

2.2.1 Digital Implementations

Figure 2–1 shows a scheme for digital pulse stream implementation. Here the weight $S_{T_{ij}}$

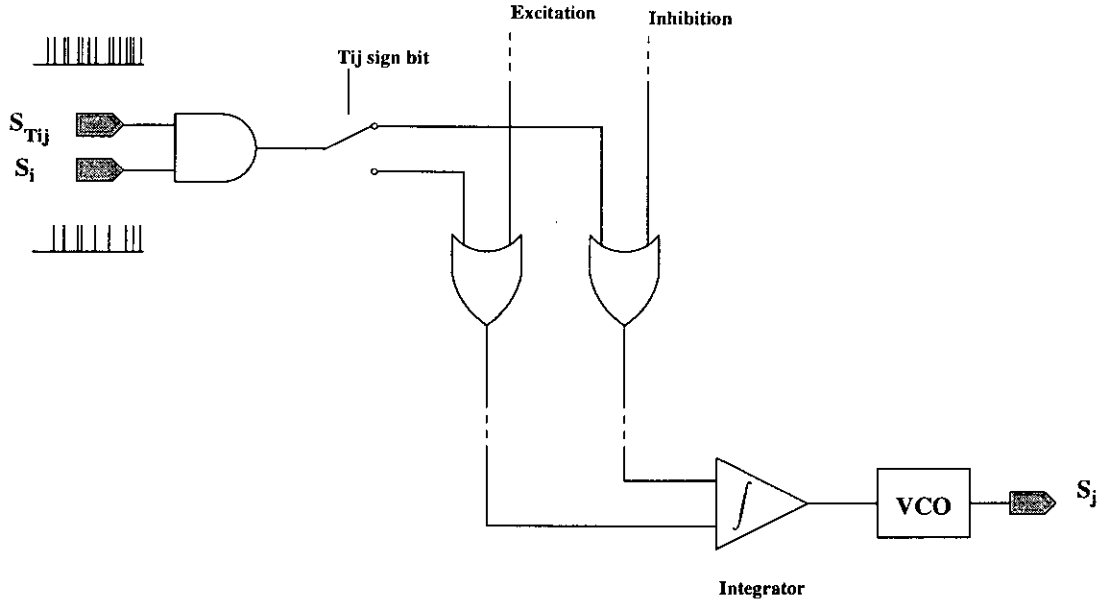


Figure 2–1: Digital Pulse Stream Scheme.

along with the input S_i are encoded as a stochastic or asynchronous PFM pulse stream. If the weight and the input pulse stream have a probability of a pulse being present such that:

$$P(A) = S_{T_{ij}} \text{ and } P(B) = S_i$$

and if $P(A)$ and $P(B)$ are statistically independent, then:

$$P(A \text{ AND } B) = P(A).P(B) = S_{T_{ij}} S_i$$

Thus a simple AND gate provides multiplication. Summation is similarly achieved with an OR gate and bipolar weights can be implemented with a sign bit switching the result to separate excitatory and inhibitory summation lines. Examples of this type of scheme can be found in [32,86,109].

Another digital example from Murray and Smith[86] uses local digital storage and global chopping clocks to represent the weight value as shown in figure 2–2. Here the input is gated for a fraction of the time T proportional to the weight magnitude with the aid of the global chopping clocks.

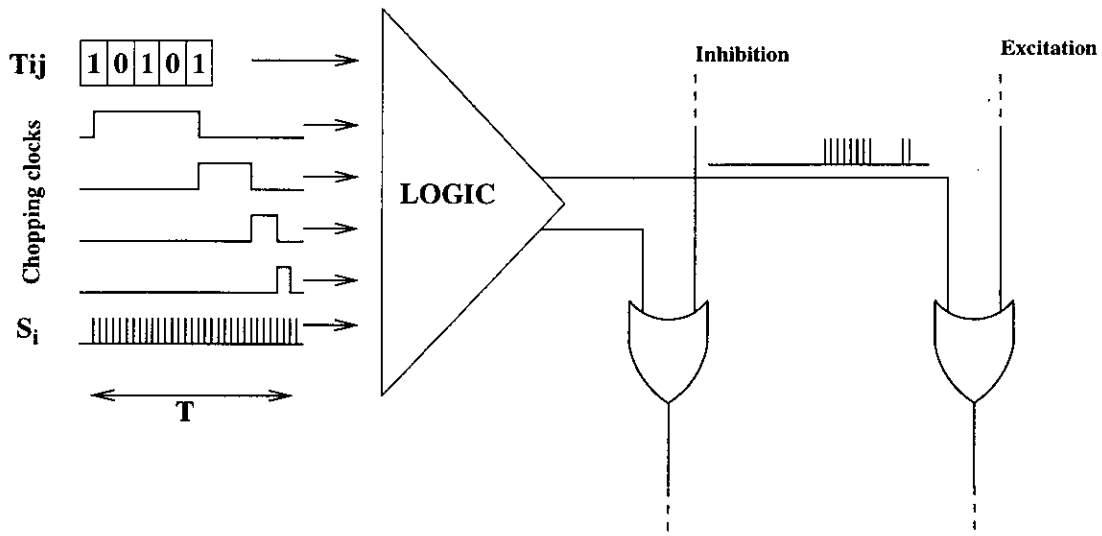


Figure 2–2: Digital Pulse Stream Scheme.

All these digital implementations suffer from inaccuracies introduced by coding noise due to *collisions* of pulses: that is when two pulses arrive at a summing OR gate at the same time an error is introduced. This is minimised by using a sparse coding of signals and a long integration time. Also despite the use of compact AND and OR gates to carry out the synaptic multiplication the synapse area is large due to the presence of digital RAM to store the weight values.

2.2.2 Switched Capacitor Implementations

Switched capacitor circuits are based on transferring packets of charge stored on a capacitor. The idea of using switched capacitor techniques for neural networks was first proposed by Tsividis in 1989 [110], others have developed implementations such as Brownlow *et al*[14,15] and Jackson[60]. Figure 2–3 shows the basic idea: The weight is stored as a voltage, $V_{T_{ij}}$ and for each period of the input pulse stream S_i a packet of charge proportional to $(V_{T_{ij}} - V_{\text{ref}})$ is transferred to the summation line. Thus if S_i is pulse-frequency modulated; multiplication of $T_{ij}.S_i$ is achieved. The advantage of this implementation is that the transfer function is determined by the capacitor ratio $\frac{C_i}{C_{\text{int}}}$ which is well defined and process tolerant.

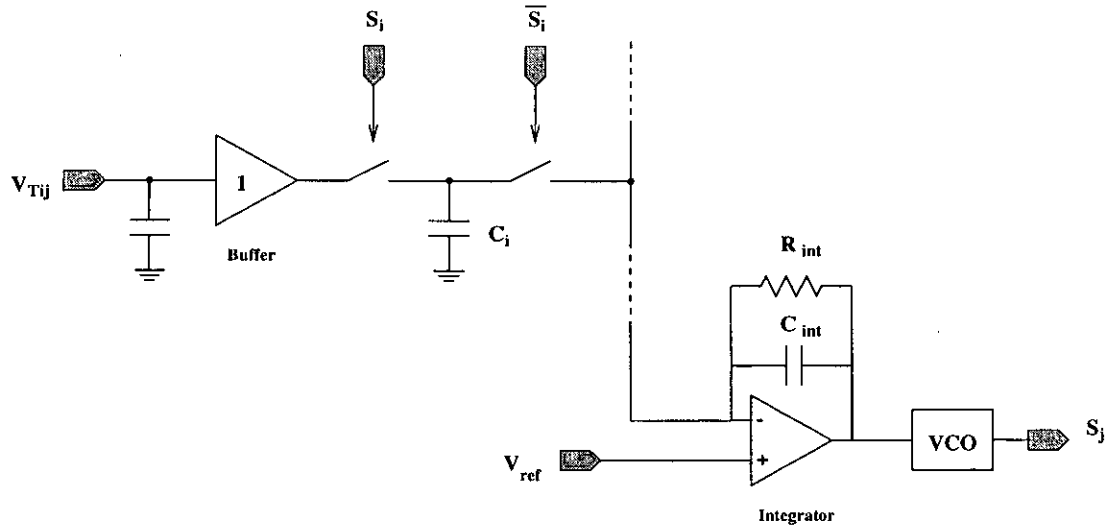


Figure 2–3: Switched Capacitor Pulse Stream Scheme.

2.2.3 Analog Implementation

The basic technique of most analog pulse stream implementations is summarised in figure 2–4. Here a current source, I_{wt}, proportional to the weight, T_{ij}, is gated by S_i which may be pulse-width or pulse-frequency modulated or stochastic. Summation is achieved *free* by virtue of Kirchoff's current law. There are several examples of such schemes [97,96,108] including the EPSILON methodology, the topic of the next section. These implementations have the potential for compact synapse cells but unlike the switched capacitor implementations are not inherently process tolerant. The challenge of design is to use analog circuit design techniques to produce process tolerant designs.

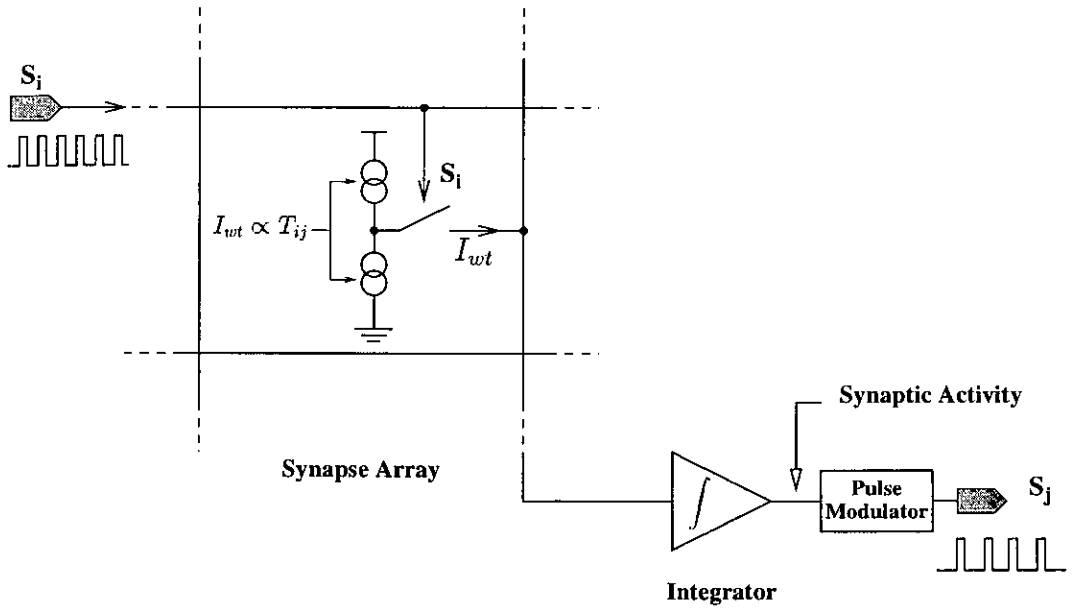


Figure 2–4: Analog Pulse Stream Scheme.

2.3 EPSILON Pulse Stream Neural Computation

The philosophy behind the development of the EPSILON[1] chip was to design a set of process invariant cells for pulse stream neural computation[23,45]. The analog approach was chosen over digital or switched capacitor techniques because:

- Digital implementations are large due to digital weight storage needed and can be inaccurate due to coding collisions, a problem that increases with network size.
- Switched capacitor circuits are process tolerant due to the reliance only on capacitor ratios but are not scalable without redesign of integrator. Switched capacitor techniques are also only suited to pulse-frequency modulation schemes.
- Analog techniques offered the possibility of compact design, process tolerance with design effort and operation under both pulse-frequency and pulse-width modulation schemes.

This section presents the EPSILON cells and the EPSILON chip a 120 input, 32 neuron pulse stream neural network chip. As shown in figure 2–4 the primary building blocks of an artificial neural network are the synapse and neuron. These will now be considered in turn for the EPSILON design.

[1] **E**dinburgh **P**ulse **S**tream **I**mplemenation of a **L**earning **O**riented **N**etwork.

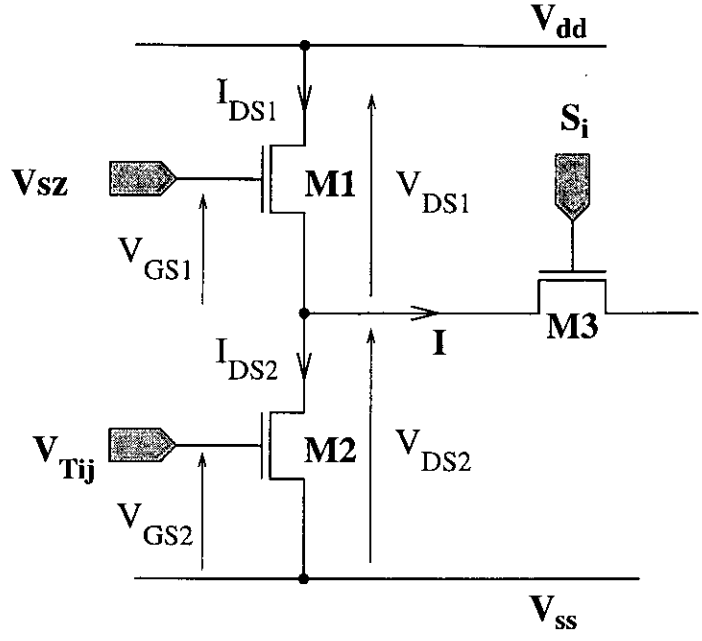


Figure 2–5: Transconductance Multiplier.

2.3.1 Distributed Feedback Synapse

The distributed feedback synapse forms the basis of EPSILON and also EPSILON II, the chip fabricated in conjunction with this thesis. To analyse the performance of EPSILON II an understanding of the building blocks of EPSILON is required. This section presents the EPSILON distributed feedback synapse which consists of:

- Synaptic transconductance multipliers.
- A voltage integrator, which integrates the voltage output of the synapse array to produce the net activity.
- and a bias generation scheme.

Synaptic Transconductance Multiplier

The basis of the EPSILON synapse design is the transconductance multiplier shown in figure 2–5. This circuit was first proposed for use in filtering applications [29]. The circuit operates the MOSFET transistors in their linear region where the characteristic drain-source current is given by:

$$I_{DS} = \beta \left[(V_{GS} - V_T) V_{DS} - \frac{{V_{DS}}^2}{2} \right] \tag{2.1}$$

where $\beta = \frac{\mu_0 C_{ox} W}{L}$, C_{OX} is the oxide capacitance/area, μ_0 the surface carrier mobility, W and L the transistor width and length respectively. By clamping $V_{DS1} = V_{DS2}$ and ensuring M1 and M2 are well matched, the non-linear terms in equation 2.1 can be cancelled, that is:

$$I = I_{DS1} - I_{DS2} = \beta (V_{GS1} - V_{GS2}) V_{DS1} \tag{2.2}$$

The above analysis ignores the effect of substrate bias (*body effect*), that is the threshold voltage, V_T, of M1 and M2 will be different due to unequal bulk-source voltages. Including this effect by defining:

$$\Delta V_T = V_{T_{M2}} - V_{T_{M1}} \tag{2.3}$$

the transconductance expression becomes:

$$I = I_{DS1} - I_{DS2} = \beta (V_{GS1} - V_{GS2} + \Delta V_T) V_{DS1} \tag{2.4}$$

This expression is proportional to the weight voltage (V_{GS2}) as wanted, however it is also dependent upon β, a process dependent parameter that will vary across the chip. To make the synapse more process tolerant the scheme of figure 2–6 was developed. Here a second

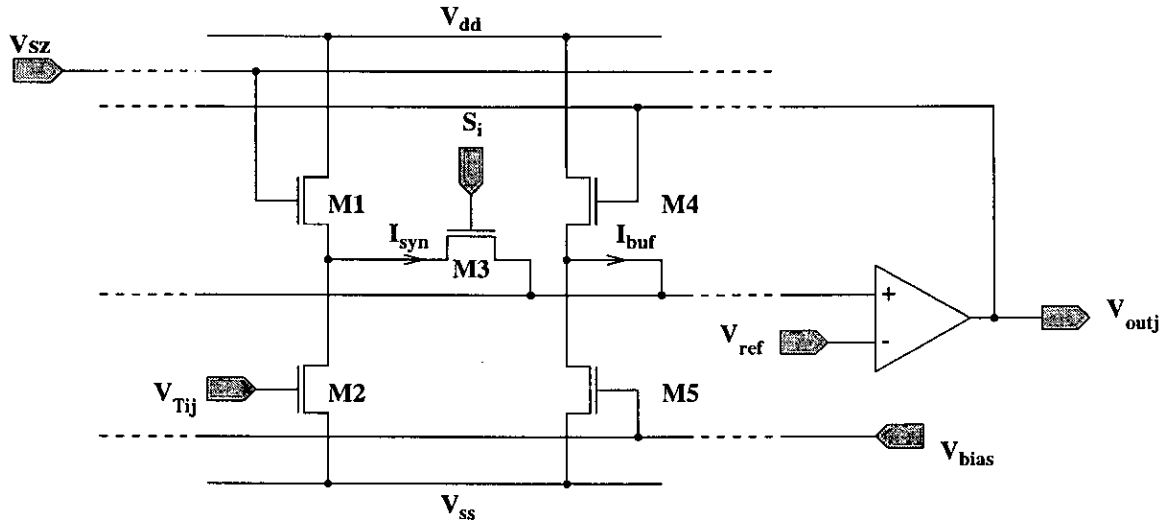


Figure 2–6: EPSILON Distributed Feedback Synapse.

transconductance stage (M4,M5) has been added as a buffer and is placed in a feedback loop with an operational amplifier located at the foot of each synaptic column. This feedback loop ensures that the current sourced from the synapse transconductance stages (I_{syn}), equals the current sunk by the buffer stage (I_{buf}). Solving this equation ($I_{\text{syn}} + I_{\text{buf}} = 0$), an expression for the output voltage, V_{out_j} can be found:

$$V_{\text{out}_j} = \frac{1}{N} \frac{\beta_{\text{syn}}}{\beta_{\text{buf}}} \sum_{j=0}^{N-1} \left(V_{T_{ij}} - (V_{\text{sz}} - V_{\text{ref}} - \Delta V_T) \right) + V_{\text{bias}} + V_{\text{ref}} + \Delta V_T \tag{2.5}$$

where N is the number of synapses, $\beta_{\text{syn}} = \beta_{M1} = \beta_{M2}$ and $\beta_{\text{buf}} = \beta_{M4} = \beta_{M5}$. Note that for the EPSILON synapse $V_{dd} = 1.5V$, $V_{ss} = 0.5V$ and $V_{\text{ref}} = 1.0V$.

Distributing the buffer stages throughout the synapse array ensures close matching between M1, M2, M4 and M5, thus $\frac{\beta_{\text{syn}}}{\beta_{\text{buf}}}$ simplifies to a ratio of transistor W/L ratios – a process invariant quantity, more details of this can be found in Baxter[9].

Voltage Integrator

The voltage output V_{out_j} represents the sum of instantaneous synaptic activity. To produce the net synaptic activity, this voltage is integrated over time by the voltage integrator of figure 2–7. Here the voltage is converted to a current by a differential transconductance amplifier. This

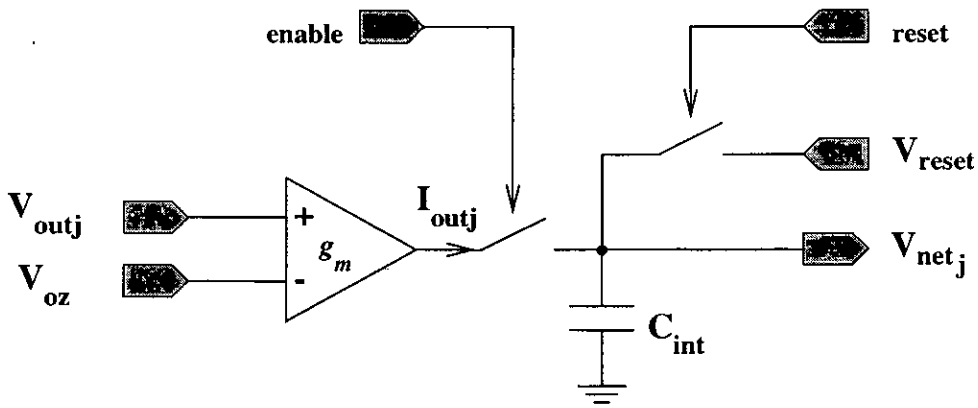


Figure 2–7: EPSILON Voltage Integrator.

current is integrated on the capacitor C_{int} to produce the net synaptic activity, V_{net_j}. Switches to **enable** the integration and **reset** the initial value of the output are also provided.

Two global references, V_{oz} and V_{sz}, determine the operating point of the synapse column. Setting these as:

$$V_{\text{sz}} = V_{Tij_z} + V_{\text{ref}} + \Delta V_T \tag{2.6}$$

$$V_{\text{oz}} = V_{\text{bias}} + V_{\text{ref}} + \Delta V_T \tag{2.7}$$

Gives an expression for the net activity as:

$$\begin{aligned} V_{\text{net}_j} &= \frac{1}{C_{\text{int}}} \int_{t_1}^{t_2} I_{\text{out}_j} dt \\ &= \frac{1}{C_{\text{int}}} \int_{t_1}^{t_2} g_m \left(V_{\text{out}_j} - V_{\text{oz}} \right) dt \\ &= (t_2 - t_1) \frac{1}{C_{\text{int}}} \frac{g_m}{N} \frac{\beta_{\text{syn}}}{\beta_{\text{buf}}} \sum_{j=0}^{N-1} \left[(V_{Tij} - V_{Tij_z}) \times DC_{S_j} \right] \end{aligned} \tag{2.8}$$

$$= k \sum_{j=0}^{N-1} T_{ij} S_j \tag{2.9}$$

where V_{Tij_z} is the weight voltage representing a zero value weight, g_m is the gain of the transconductance amplifier and DC_{S_j} is the duty cycle of the input pulse stream which encodes the input value. Thus V_{net_j} is the sum of weights, $T_{ij} = (V_{Tij} - V_{Tij_z})$ times inputs, $S_j = DC_{S_j}$.

Bias Generation

The distributed feedback synapse minimises the operational reliance on process parameters, however the two global references, V_{oz} and V_{sz}, are dependent on ΔV_T, itself a process dependent parameter. Thus the value required of these references will vary from chip to chip. To remove this problem an on-chip generation scheme was devised as shown in figure 2–8. A dummy

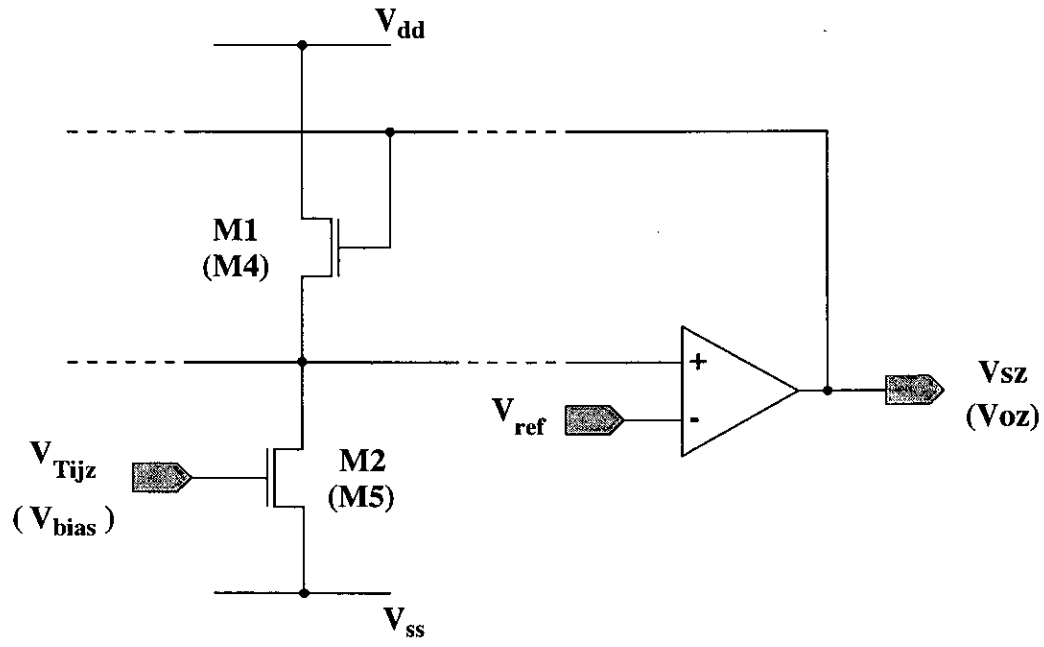


Figure 2–8: EPSILON Bias Generation Scheme.

column of synapses is used to provide transistors closely matched to those in the synaptic array. Solving the characteristic equations of these circuits the reference values generated are those of equations 2.6 and 2.7.

2.3.2 EPSILON Neurons

The second principal building block is the neuron. The distributed feedback synapse column produces a net output activity voltage, V_{net_j}, which is converted by the output neurons to a pulse

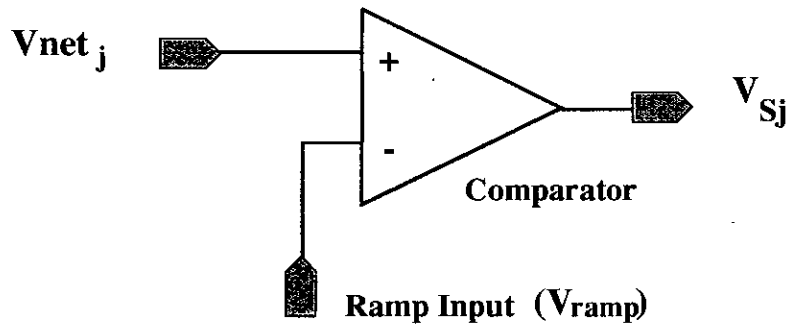


Figure 2–9: EPSILON PWM Neuron.

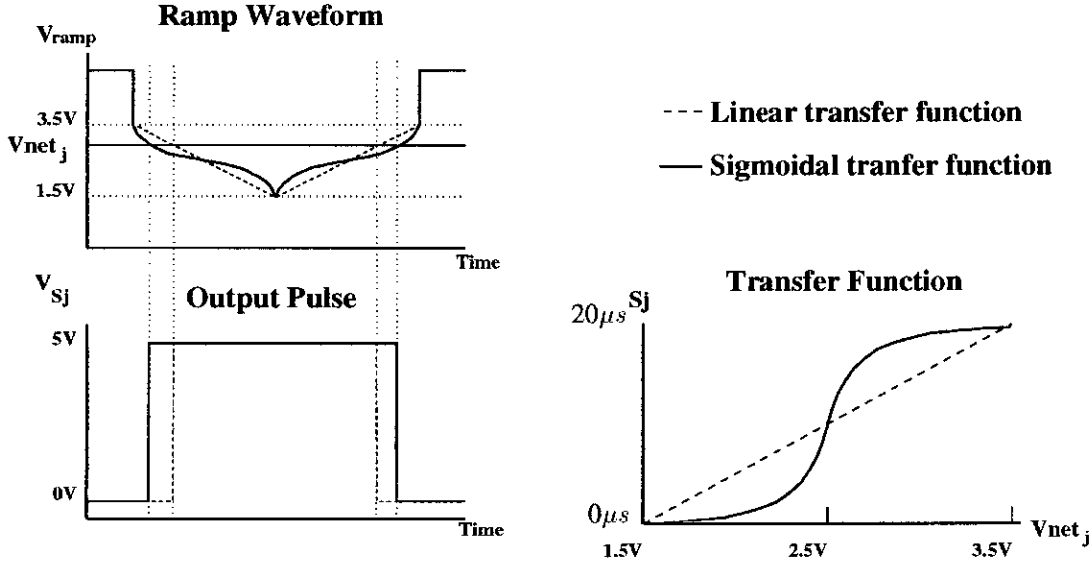


Figure 2–10: EPSILON PWM Neuron Ramps and Transfer Functions.

modulated signal. Two pulse modulation schemes were included on EPSILON; a pulse-width modulation scheme for use when fast computation time is important and a pulse-frequency scheme to be used when asynchronous computation is the prime consideration.

Pulse-Width Neuron

The pulse-width modulating (PWM) neuron is used to modulate both analog inputs to the network and to produce pulse-width outputs. The neuron itself consists of a conventional comparator which compares net synaptic activity (or the analog input in the case of an input neuron) to a reference ramp waveform (figure 2–9). It is this reference ramp waveform that determines the shape of the neuron transfer function. Shown in figure 2–10 are ramp waveforms for a linear (blue) and sigmoidal (red) transfer function. The ramp waveforms used are *double-sided* to minimise switching noise which would otherwise be increased by all neurons switching

together if the ramp was single-sided. The comparator is a process invariant device and the ramp wavèform is produced off-chip from values stored in digital RAM, this makes the pulse-width neuron largely invariant to process variations. This flexibility and invariance is achieved with a trade-off to added complexity off-chip in terms of RAM and DAC circuitry for ramp generation.

Pulse-Frequency Neuron

The basic principle of the operation of the pulse-frequency neuron design is shown in figure 2–11. The high time of the output pulse is constant and determined by the current I_H charging

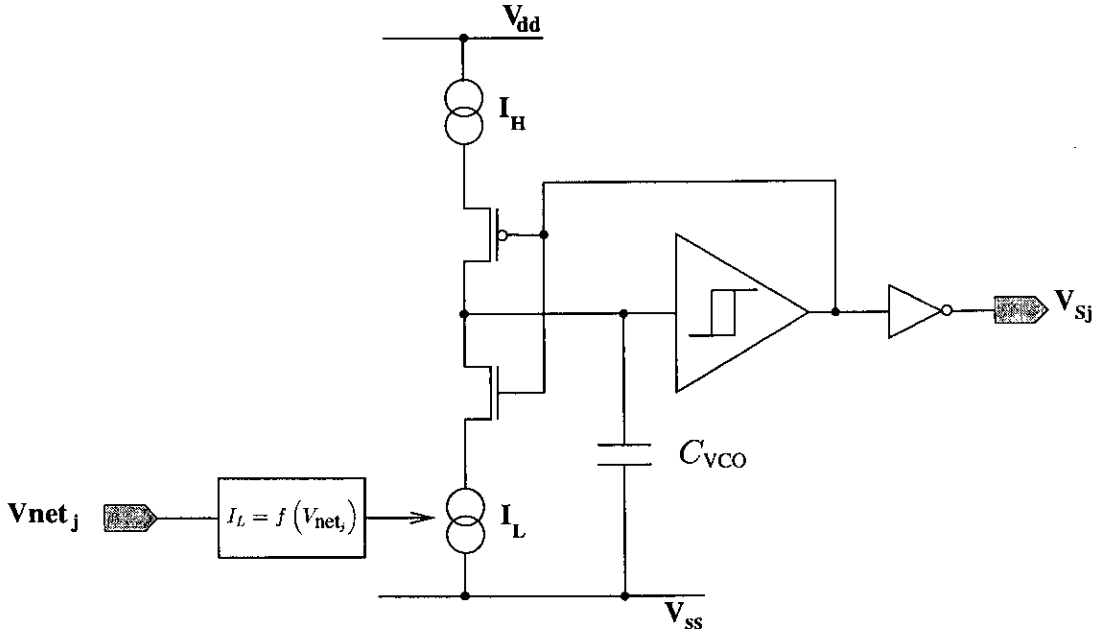


Figure 2–11: Basic Principles of EPSILON PFM Neuron.

the capacitor C_{vco}. The low time of a pulse is determined by the voltage controlled current sink, I_L discharging C_{vco}. This current sink is formed by a differential stage to give a sigmoidal transfer function with respect to the net synaptic activity, V_{net_j}, with a maximum current equal to I_H. This in effect varies the duty cycle of the resulting pulse stream between 0% ($I_L = 0$) and 50% ($I_L = I_H$). The actual voltage controlled current sink is considerably more complex than this as it incorporates techniques to vary the sigmoid gain or *temperature*, for more details on this see Hamilton [44]. Also to promote process tolerance a reference generation scheme using phase-locked-loops (PLL) is used to set the maximum current I_H and the reference to vary the sigmoid gain[44].

2.4 EPSILON Chip Results

Table 2–1 summarises the features and specifications of the EPSILON chip. All modes of

EPSILON Chip Specifications	
No. of state input pins	30
No. of actual state inputs	120, MUX'd in banks of 30
No. of State outputs	30 Directly pinned out
Input mode programmability	All analog/All digital
Input Modes	analog, PW or PF
Output modes	PW or PF
No. of synapses	3600
No. of weight load channels	2
Weight load time	$3.6ms$
Weight storage	Dynamic
Maximum speed (cps)	$360Mcps$
Technology	$1.5\mu m$ CMOS
Die size	$9.5mm \times 10.1mm$
Packaging	144 pin PGA
Maximum power dissipation	$350mW$

Table 2–1. Comparison of EPSILON Chip Specifications

operation performed satisfactorily, though design imperfections resulted in a degradation from expected results. The most serious problem was in power supply distribution to the synapse array which led to offsets in the synapse characteristic[9]. This can be seen in figure 2–12 which shows the synaptic multiplication characteristic in the PWM output mode. The characteristic in the PFM output mode shows similar behaviour (figure 2–13) though this mode experienced more noise variations due to the many edges inherent in a pulse-frequency modulated signal coupling with analog references. These problems are discussed further in Chapter 5 when the development of the VLSI component of this thesis is presented.

2.4.1 FENICS – EPSILON at System Level

Despite the shortcomings present in the EPSILON device, the performance was still sufficient to warrant development of a system level test bed to use the chip for neural computation. This system, named FENICS[2], performed support functions to enable a neural network to be

[2]FENICS: Fast Electronic Neural Information Computing System

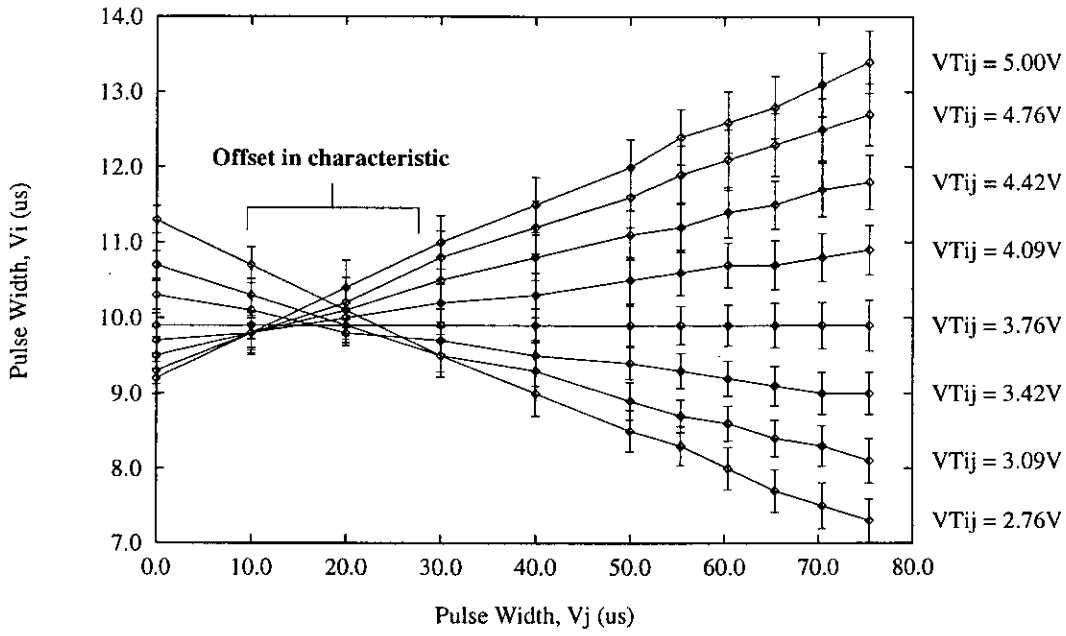


Figure 2–12: EPSILON PWM Characterisation Results.

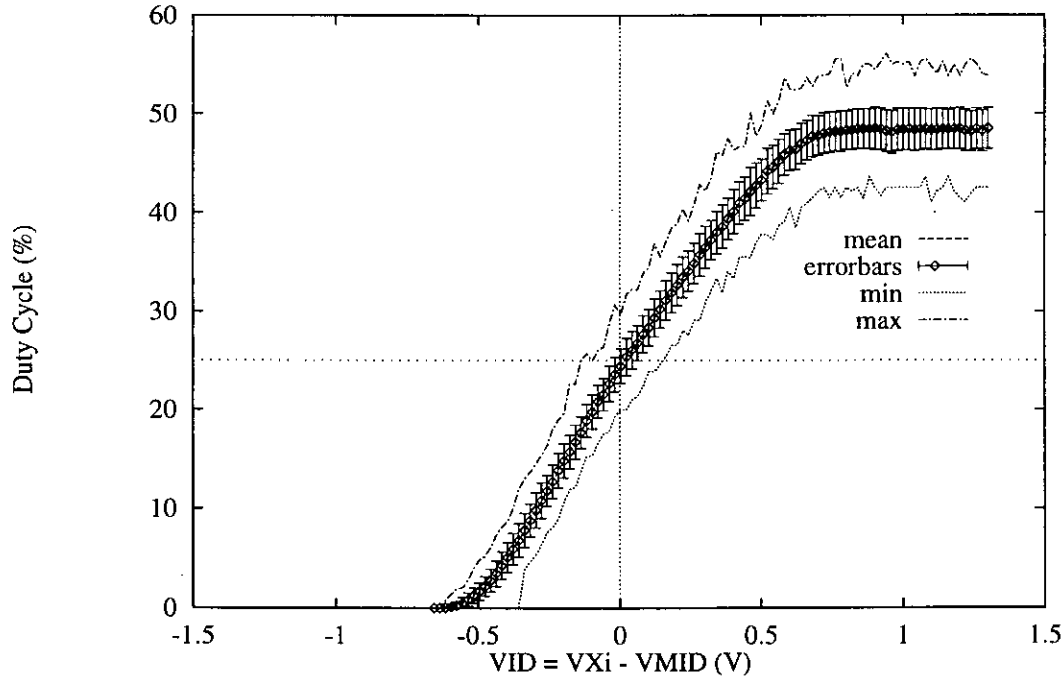


Figure 2–13: EPSILON PFM Characterisation Results.

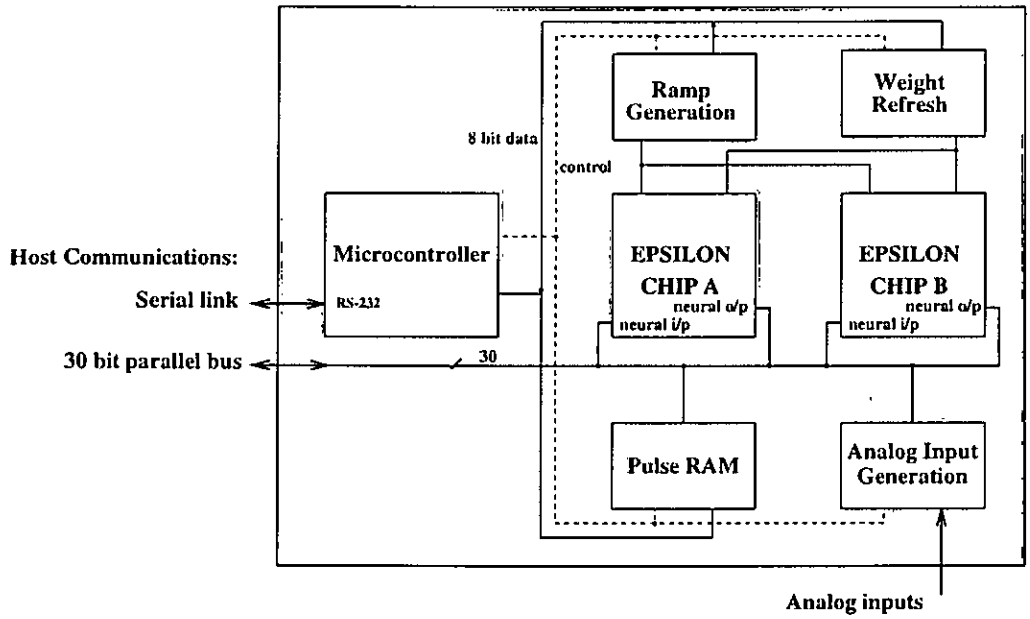


Figure 2–14: FENICS: EPSILON System Level Board.

implemented using the EPSILON chip. The general system architecture is shown in figure 2–14. The *pulse RAM* is a bank of memory used to store pulse streams, either sampled from EPSILON outputs for later processing, or generated by the system to apply as inputs to EPSILON. Sub-systems for ramp generation and weight refresh were also included. Control of the system is carried out by a microcontroller which transfers data around the system using an 8 bit data bus. Communication with the host system is over the microcontroller's serial link for commands and a 30 bit parallel bus for data. The overall system performance was restricted by the slow speed of the microcontroller and the serial nature of generation and processing of pulse streams over the 8 bit microcontroller bus. These issues are further discussed in Chapter 6.

Three test problems were implemented on the FENICS system:

1. An image classification task to label scenes as *roads* or *not roads* was performed on a 45:12:2 MLP trained with the back-propagation algorithm[24].

2. A speech processing task involving classification of 11 vowel sounds from a database of 33 speakers. This was performed on a 54:27:11 MLP trained using the virtual targets algorithm[44].

3. The travelling salesman problem (TSP) using a Kohonen network[9].

For the first two problems the hardware network was trained chip-in-loop (CIL), a technique whereby inaccuracies present in the hardware can be partially compensated for by training with these non-idealities present. The Kohonen network was trained off-line in software and the weight set downloaded and used on hardware.

Problem	Hardware results	Software Results	
	% Correct (% Std. Dev.)		
Image classification[24]	63.57 (4.86)	67.56 (8.33)	
Vowel classification[44]	65.34 (N/A)	58.21 (4.25)	
Kohonen TSP[9]	Correct Solution?		Variation between hardware and software
5 city	pass	pass	$-16.2\% \rightarrow 0\%$
9 city	pass	pass	$-5.4\% \rightarrow 6.6\%$
10 city	fail	pass	

Table 2–2. Summary of Problems Implemented on EPSILON.

Table 2–2 shows the results of these experiments. It demonstrates the potential performance of EPSILON style pulse stream processing for CIL training as results from hardware compare well with software networks. For the case of the Kohonen TSP; hardware non-idealities caused the network to fail for problem greater than 10 cities. The large variations seen between software and hardware networks indicates that weights evolved for an ideal software network are not suitable for use on hardware.

2.5 Summary

This chapter has introduced pulse stream neural computation. It has shown that pulse stream techniques offer advantages for system integration in that network outputs are digital signals that can be buffered, transferred and processed with digital technology rather than requiring an A/D conversion overhead. The EPSILON distributed feedback synapse was examined as a process tolerant synapse design with the flexibility of operation with either PWM or PFM pulse streams. The PWM output neuron scheme was shown as a very accurate and flexible synchronous modulation method and the results of problems implemented on EPSILON showed promise of effective solutions in a chip-in-loop training situation.

Chapter 3

Neural Networks for Practical Applications

3.1 Introduction

The aim of this thesis is to develop pulse stream neural hardware suitable for implementing solutions to real world applications. To illustrate some of the issues associated with this objective, this chapter focuses on other hardware neural network implementations.

To gauge the need for specialist neural hardware, a brief outline of hardware implementation using conventional technology is first discussed. Following this, to focus on application issues, four implementations have been selected for study. While not an exhaustive examination of the available implementations, these represent the most successful of applications–oriented hardware implementations. The chapter is structured around case studies of these four hardware neural network implementations:

1. The Adaptive Solutions **CNAPS** system.
2. The Intel **ETANN** chip, a generic neural processor chip.
3. The **Synaptics Inc** cheque reader system.
4. The **Kakadu** project in which a hardware neural network was developed for detection of cardiac arrythmia.

The objective of this review is to determine how the examples above interact with larger systems, in order to provide a focus three areas are addressed for each case study:

Target application: The target application is discussed to define the context in which the example was developed.

Neural structure: The neural structure defines in what way neural network interacts with larger systems. Three factors define this structure:

1. Synaptic multiplication determines the input characteristics (e.g. analog/digital single-ended/differential) along with internal data representation of the neural network.
2. The weight storage technique used determines how the neural network is adapted; this is done most often under control of an outside system.
3. The neuron structure determines the output characteristics of the neural network.

The choice and design of these three elements have a strong influence on such metrics as speed, area and power consumption of the network.

System Performance: The performance of the neural architecture in use gives an indication of the success of the implementation. Key factors here are:

- System support required for chip operation.
- Training methods suitable for network use.
- Success of the implementation for the target application.

After presentation of the four case studies the issues raised by the comparative methodologies are discussed. It is these that play an important role in Chapter 4 which deals with the specification of the hardware content of this thesis.

3.2 Using Conventional Technology to Implement Neural Networks

To assess the need for dedicated neural hardware, this section outlines the effectiveness of how conventional *off the shelf* hardware can be used to implement neural networks. The obvious technology to accomplish this is DSP[1] technology as a neural network requires fast numerical operations to implement the multiply and accumulate functions of the synapses. For example, the TMS320 series can perform such multiply–accumulates at a rate of 40MHz, this is equivalent to neural performance of 0.040 billion connections per second. As will be seen in the following case studies, this is at least an order of magnitude less than all but one of the implementations reviewed. The throughput of such a hardware implementation would be dependent on network size as the parallel neural structure is being implemented on a serial processing device. Though this lower throughput in itself is perhaps justification for dedicated hardware development, further justification derives from the nature of the application areas the neural technology targets. That is, the defined application area lies on the boundary of the analog and digital domains. To interface a DSP to analog signals a large A/D hardware overhead is required.

[1] Digital Signal Processing

3.3 Case Study: Adaptive Solution's CNAPS

The Adaptive Solutions CNAPS[2] chip-set is a fully digital parallel processing system designed primarily for neural network implementation. The system is based on two custom chips:

1. The CNAPS chip, an array of parallel processors.
2. The CNAPS Sequencer Chip (CSC) that controls one or more CNAPS chips.

Together the processor and sequencer form a single-instruction, multiple-data (SIMD) computer where each processor from the CNAPS chip(s) executes the same instruction on multiple data. This maps neatly to the neural network structure of each neuron (processor) performing multiply and accumulate operations in parallel (multiple) streams of input data.

3.3.1 CNAPS Structure

The powerhouse of the CNAPS system is the parallel processor chip the CNAPS-1064 which has an array of 64 integer processors called **Processor Nodes** (PN's). Figure 3–1 shows how

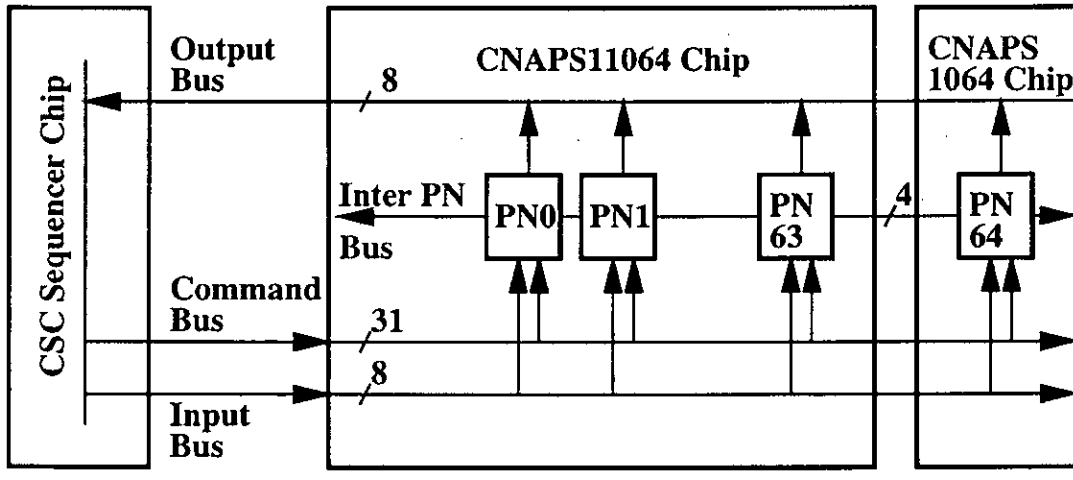


Figure 3–1: A CNAPS System Consisting of CSC Controller and Multiple CNAPS Chip

CNAPS chips connect under control of the CSC chip to form a parallel system[82]. Each CNAPS chip has 64 PN's. The structure of individual PN's is shown in figure 3–2. Each PN has a multiplier (up to 16 bit x 16 bit) a 32 bit adder/accumulator, a logic/shifter block, a 32 word register and a 12 bit address unit accessing 4K Bytes of local (e.g. weight) memory. The data representation is fixed-point, two's complement arithmetic. This generally has 16 bit

[2]Co-processing Node Architecture for Parallel Systems.

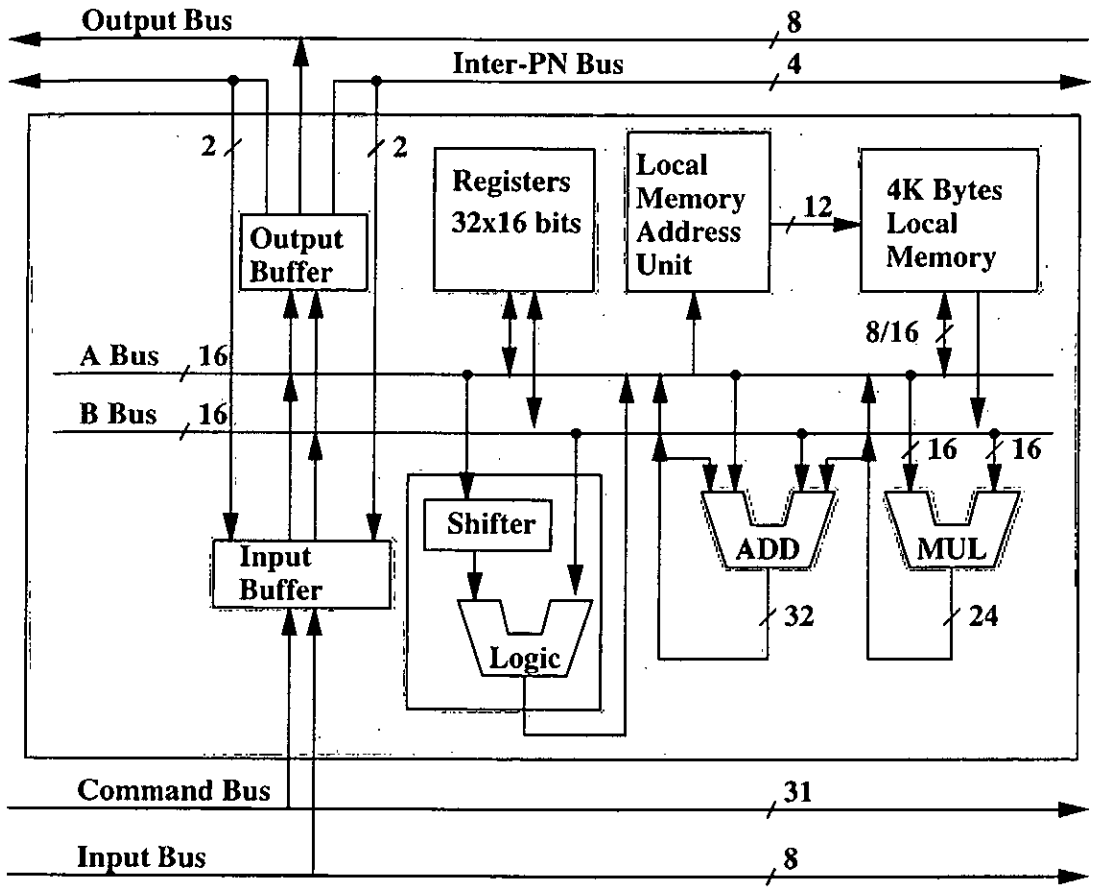


Figure 3–2: Internal Block Diagram if a CNAPS Processor Node.

internal resolution with extra *head-room* given to multiply/accumulates. Two broadcast busses supply data (8 bit) and instructions (32 bit) to all PNs on the chip. Output busses are also 8 bits and PNs place data onto it via one of several arbitration schemes. Most operations take only one clock cycle, exceptions are 16 bit I/O and high resolution multiplications which take two. PNs communicate with their nearest neighbours via a 4 bit bus used amongst other things for arbitration and winner-take-all evaluations[3,2,46].

3.3.2 CNAPS System Performance

Adaptive Solutions manufacture CNAPS boards for PCs and VMEbus as well as a dedicated server system. Extensive software is also available including a dedicated 'C' compiler, binary libraries, an assembler and applications development packages. At board level, the VMEbus board for example, allows the up to 8 CNAPS chips (512 PNs) and the addition of an application specific mezzanine board for custom I/O. Neural network algorithms such as backpropagation have been demonstrated on CNAPS systems to be up to 69×10^3 times faster during learning and 38×10^3 times faster in feedforward mode than a SUN 3 workstation[76].

There are several reported applications running on CNAPS systems[42,47,54,79,75,82]. Gruber *et al*[42] describe a neural network classifier that is used as a trigger in a physics particle collision experiment. Here VMEbus based CNAPS computers take input data from a $20MHz$ bus carrying data from various detectors involved in the experiment. The CNAPS system was able to process this data in the required $20\mu s$.

Holt *et al*[54] describe a speech recognition system for automated telephone operator systems. Here the CNAPS computer performs other parallel operations such as Fourier analysis as well as a neural network classifier with 128 PNs to achieve classification in $1100ms$.

Matsuura *et al*[75] describe a speech recognition system based on phoneme extraction. The system utilises a CNAPS processor to do spectral feature extraction followed by phoneme recognition using a 135:220:23 MLP trained by back-propagation then word recognition by DTW matching. It achieves a 97% recognition rate on test set data for a vocabulary of 100 words.

3.3.3 CNAPS Summary

The CNAPS system offers neural network processing within a conventional digital systems environment. It is an expandable system where extra chips can be easily added to increase network size. Each processing node stores multiple weights locally and acts as a column of synapses. The network architecture is also variable, but for sparsely connected networks, due to the nature of the SIMD architecture, efficiency of processor use decreases substantially[47]. The CNAPS architecture offers little possibility of a highly integrated solution but rather is used as a parallel accelerator. As CNAPS has no fixed architecture it is not limited to any particular neural network architecture; it can also be used to implement other functions that benefit from parallel operation such as Fourier analysis[54] and image processing[2].

3.4 Case Study: Intel ETANN

Intel's **E**lectrically **T**rainable **A**rtificial **N**eural **N**etwork was the first serious commercial attempt at providing hardware for generic neural computation. As such there is no specific target application associated with its design. The chip utilises analog EEPROM cells for weight storage and fully analog representations for input and output. The chip has two 80x64 synapse arrays consisting of 64 inputs plus 16 bias synapses fully connected to 64 outputs. One array takes inputs from 64 external analog inputs while the other is a feedback network fed from the analog network outputs. This feedback array can also be used as a second layer.

3.4.1 Neural Structure

The basic ETANN neural structure is shown in figure 3–3. The design utilises fully differential

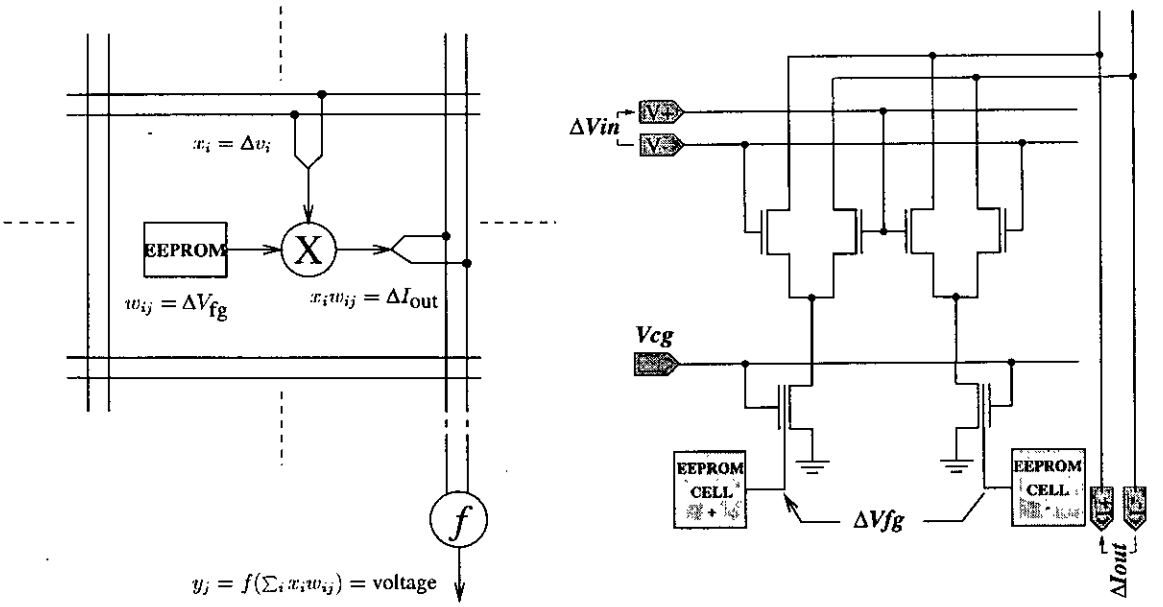


Figure 3–3: ETANN Neural Structure.

signals to enhance noise immunity and temperature invariance. A NMOS Gilbert-multiplier produces an output current ΔI_{out} proportional to a multiplication between the weight stored on the floating gates ΔV_{fg} and the input signal ΔV_{in}. The stored weight is changed by adding or removing electrons from the floating gate by Fowler-Nordheim tunnelling between the gates and diffusion. The resolution of this weight is dependent on how accurately this floating gate voltage can be set and how well the weight is retained. Results from [51] show that for long term ($\approx$ 15 year) storage 4 bits of resolution is possible. Using "bake–train" techniques, whereby

the chip is baked at high temperatures after training to promote the relaxation of the weights then re-trained[103], up to 7 bits of long term resolution is possible.

Figure 3–4 shows the block diagram of the Intel ETANN chip highlighting the input, output and weight storage channels along with the references and controls needed by these functions. Input and output to the chip is in the form of individually pinned, single ended analog voltages.

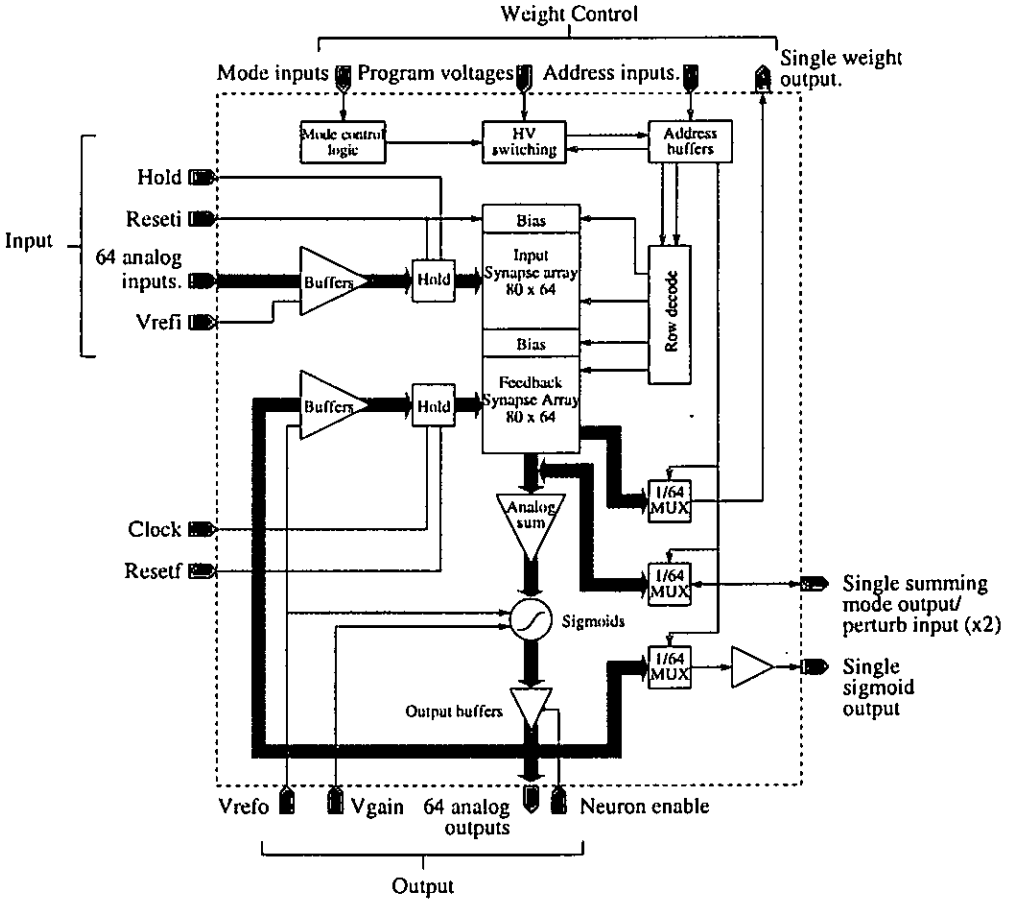


Figure 3–4: Block diagram of the Intel 80170NX ETANN

Internally, representation is differential with respect to the reference voltages V_{refi} and V_{refo}, for input and output respectively. The input synapse array takes inputs from the 64 analog voltage inputs. The network architecture can be varied by using the feedback array of synapses in one of three ways:

1. Network output can be fed-back under control of the **clock** signal allowing implementation of Hopfield networks.

2. Two layer operation is possible: The first layer is evaluated with the feedback layer disabled and the input array active. The outputs are then sampled and held while the feedback array evaluates the second layer with the input array disabled.

3. Up To 128 inputs to a single layer are possible by disabling the neurons via the **neuron disable** pin and the 64 output pins can be used as an additional 64 inputs.

Weight Adaptation

The ETANN chip is addressed by 14 address lines to access all 10240 weights. The present state of the addressed EEPROM cell can be monitored via the **single weight output** pin of figure 3–4. The host processor must decide on the pulse-width and heights necessary to change the weight to the new desired value. These will be in the range of 12–20V and $10\mu s$–$1ms$. After a weight change pulse is applied the weight can be sampled again and an iterative process used to fine tune the weight. Obviously this can entail a lengthy process if learning is over thousands of epochs. Thus, normal procedure would be to train a software simulation, download the weights and *tune* the weights with the ETANN chip in loop.

3.4.2 System Performance

Several papers describe use of the ETANN chip in systems [50,61,69,102,103]. One that demonstrates potential difficulties (or inappropriate use) of ETANN is from the Naval Air Warfare Centre in California[61]. Here the goal was to embed ETANN into a digital system for real-time use; primarily for local area processing on 2-dimensional images. To achieve this, 128 channels of D/A conversion and 64 channels of A/D conversion are needed. It was found that with this overhead it was very hard to run the system at ETANN's full speed and it presented a very large hardware overhead. An analog communications bus was ruled out due to low drive capability of the ETANN output buffers ($0.375mA$) and unavailability of suitable analog memories for storing data[61].

The application of Lindsey *et al*[69] in drift chamber tracking is more appropriate. Here analog sensor readings from a drift chamber experiment are fed directly to ETANN. Training was carried out chip-in-loop (CIL) using Intel's PC based ETANN board and development system.

Tam *et al*[102] demonstrate the multi-chip ability of ETANN using an analog bus. They used the *ETANN Multi-chip prototyping Board (EMB)* from Intel, along with supporting software which develops a set of weights for ETANN off-line then trains in loop to adjust weights. To gauge the complexity of training ETANN, an 8 ETANN system took over 12 hours to perform 4 CIL training epochs!

3.4.3 ETANN Summary

ETANN provides a solution to the vexing problem of on-chip non-volatile weight memory by using analog EEPROM. It does this at a cost of a long and computationally intensive training

time. The fully analog I/O preserves the parallel nature of the network and is ideal for problems with analog data. However for digital data or interfacing to digital systems a high overhead must be met in A/D and D/A conversion. Commercially the ETANN chip was unsuccessful; Intel's neural networks group has now been disbanded and the chip withdrawn from sale.

3.5 Case Study: Synaptics Corporation OCR Cheque Reader

The Synaptics Corporation has been developing neural network systems for commercial use for nearly a decade. One product that illustrates technical success, though not commercial success, is their OCR[3] system for hand swiped cheques. Table 3–1 shows the basic specifications of the required system[34].

Table 3–1. OCR system for hand-swiped cheques

FONT:	E13B
SPEED:	$\leq$ 1000 characters/second
ACCURACY:	$\geq$ 99.995% correct classification
COMPLEXITY:	Custom chip plus micro-controller.
MANUFACTURING VOLUME:	$\approx$100,000 units/year.
MANUFACTURING COST:	< \$175 per system.

An engineering solution to this problem is quite difficult due to the presence of many free parameters, such as:

- Unknown X & Y position of character.
- Unknown and variable velocity of cheque.
- Unknown and variable reflectivity of cheque (brightness).
- Unknown and variable ink density of cheque (contrast).
- Presence of corrupt or damaged characters.
- High input data rate, 23 megapixels/sec.

3.5.1 Neural Structure

System Architecture

The basic architecture of the system is shown in figure 3–5 consists of a neural network chip (I1000 chip) which incorporates a photo-sensor array for direct optical input and a digital

[3]Optical Character Recognition

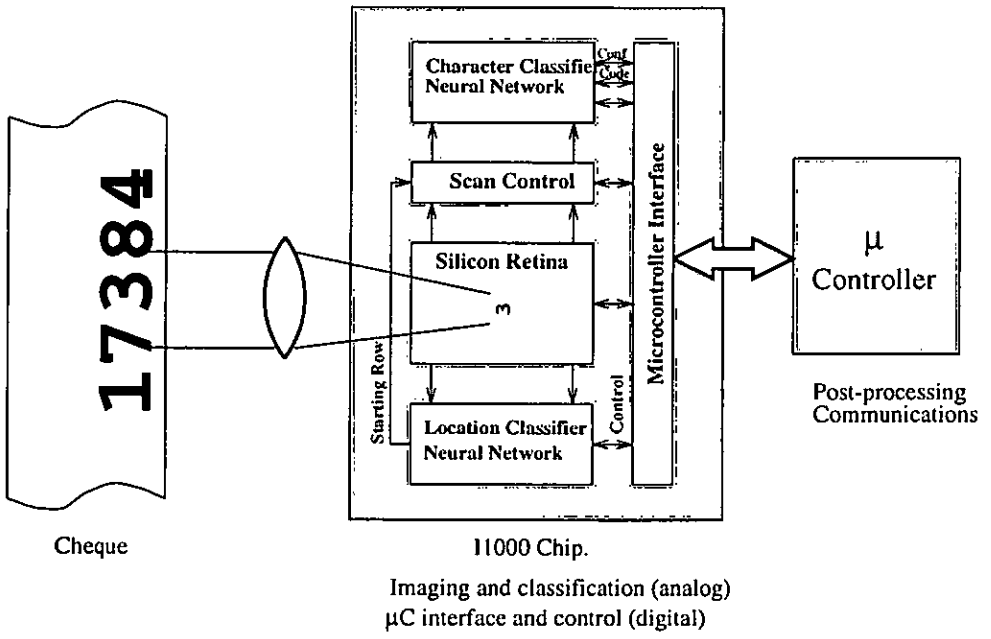


Figure 3–5: Architecture of Synaptics OCR cheque reader system

interface to a micro-controller for output. The internal architecture of the neural network chip is shown in figure 3–6. One neural network is used to locate the character to be classified which triggers the scan control section to present the character to a second classification network. A third neural architecture performs a winner-take-all (WTA) function along with a confidence measure.

Circuitry to convert the network outputs into signals compatible with the micro–controller bus is also included on the chip.

I1000 Circuits

The I1000 was custom designed to provide a solution to the single fixed problem defined in table 3–1, this allowed the use of fixed weights in the network as adaptability was not required. The weights are encoded as the width-to-length ratios of the synaptic connection transistors shown in figure 3–7(a).

The winner-take-all circuit of figure 3–7(b) consists of a series of commoned current conveyors[7,65,112] and operates on the following principle: The bias voltage V_{bias} produces a reference current, I_{bias}. All the lower NMOS transistors are controlled by a common gate voltage V_c. At equilibrium, in the case where one input current is significantly higher than all the others (I_{max}), V_c will stabilise such that the lower transistor conducts I_{max} in saturation. As all other currents are less than I_{max}, all other lower transistors must leave saturation, drastically reducing their output voltages V_n and shutting off the upper NMOS transistors. Consequently virtually all of I_{bias} is sourced from the upper NMOS transistor of the winning input. Thus

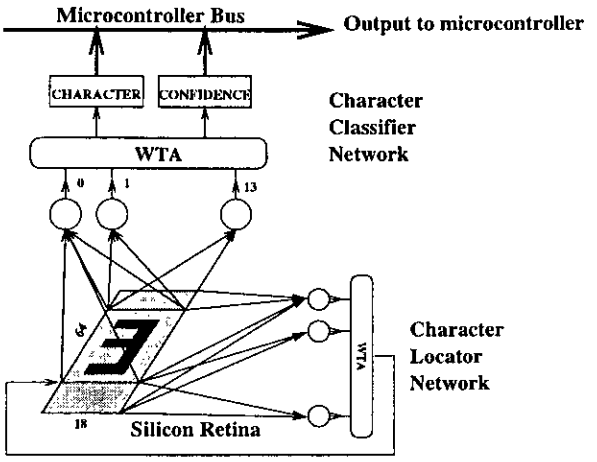


Figure 3–6: Neural network configuration on I1000 chip.

the output voltage V_{max} associated with I_{max} is much greater than all other outputs which are approximately zero. Also, for operation in the sub-threshold region, V_{max} logarithmically encodes I_{max}[65].

If two inputs are approximately equal (say $I_0 \approx I_1$), more comparison of V_c and the output voltages produces a confidence measure for the decision – if they are very close, a low confidence is implied.

Figure 3–7(c) shows a pixel of the silicon retina. This circuit, derived from Carver Mead's work at Caltech[78], has a very fast response time to cope with the up to 40 thousand frames per second input rate. The logarithmic characteristic of a MOSFET in sub-threshold is useful for compression of the input and an adaptive element is added to account for device mismatch and variations in optics, illumination and temperature[34].

3.5.2 System Performance and Success

Despite the technical success of the Synaptics OCR the product did not meet with commercial success. According to Faggin[33], the director of Synaptics, reduction of the unit cost after the contract was finalised made the product uneconomical. Other Synaptics projects have also had considerable technical success but little commercial success, such as address locator systems for the US postal service. The current product which the company is hoping will find commercial success is a touch pad *mouse* for portable computers which uses neural techniques to convert C/V[4] characteristics of a touch pad into mouse movement signals.

This lack of success shows the difficulty in bringing neural products to the marketplace,

[4]Capacitance/voltage characteristic

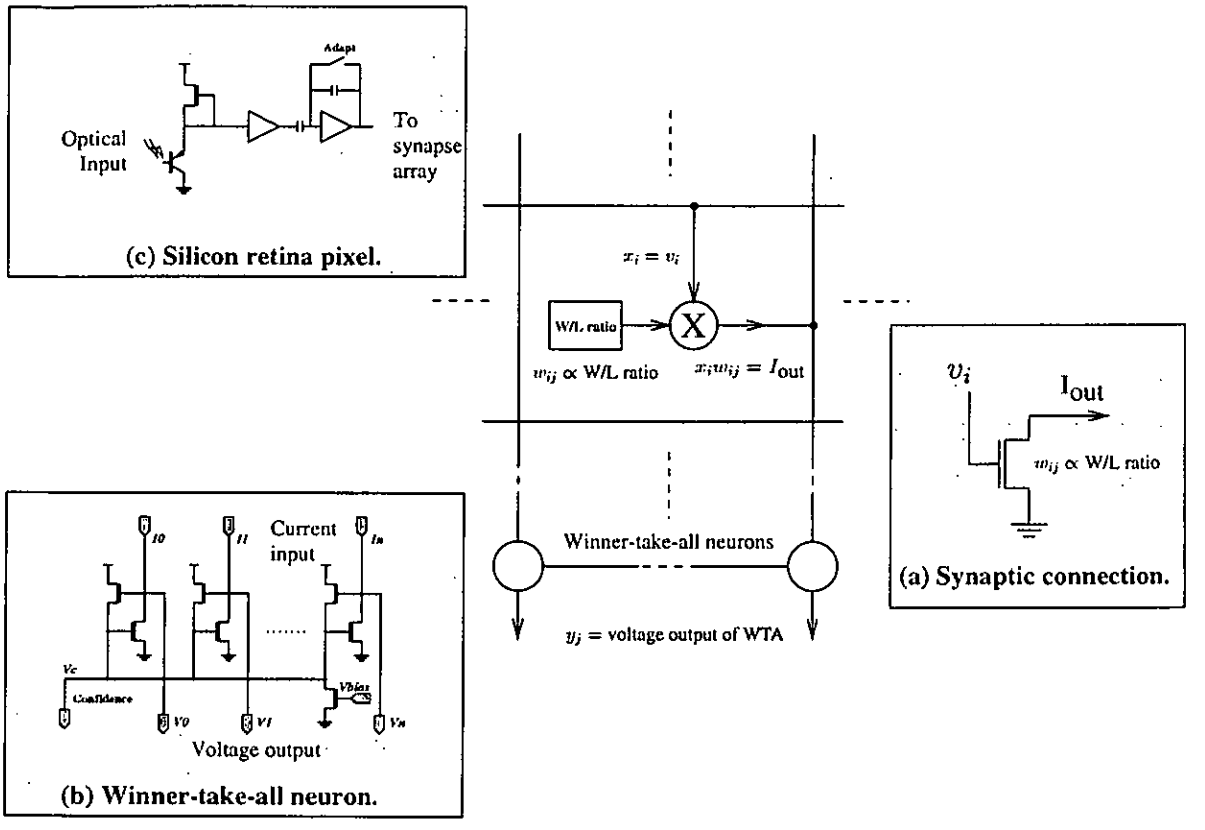


Figure 3–7: Circuit Blocks of OCR system

for despite high technical quality and even cost competitiveness, proven technology is usually adopted in preference to unproven neural technology.

3.5.3 Synaptics Summary

The Synaptics I1000 demonstrates that neural network hardware can be custom designed for applications solutions. The static nature of the problem allows for a fixed weight solution leading to very compact synapse layout. The chip also incorporates direct optical input and bus compatible digital output and control leading to a highly integrated solution with high throughput.

3.6 Case Study: Kakadu - A low power neural network for tachycardia detection

The Kakadu design was developed by the SEDAL group at Sydney University, Australia in order to classify intracardiac electrogram (ICEG) waveforms in the context of implantable cardiac defibrillators. The need to ensure long battery life and the limited size of implantable defibrillators means there are strict power and area requirements on the chip. A very low power analog/digital hybrid CMOS design, operating in the sub-threshold region, was evolved to solve this problem.

Classification of ICEG involves monitoring a time varying signal that represents the functioning of the heart. The goal is to detect a dangerous situation (ventricular tachycardia) and trigger the defibrillator to correct this. Conventional digital time series analysis techniques are too power and area intensive for an implantable device and the present solution which classifies tachycardia on timing information alone fails to classify all situations correctly.

3.6.1 Neural Structure

The basic neural structure of the Kakadu chips is shown in figure 3–8. In this design synaptic multiplication is performed between the differential input voltage $\Delta v_i = (V_+ - V_-)$ and the current I_{DAC} which represents the weight.

$$\Delta I_{\text{out}} = I_{out+} - I_{out-} = \begin{cases} +I_{DAC} \tanh\left(\frac{\kappa(V_+ - V_-)}{2}\right) & \text{if B5=1} \\ -I_{DAC} \tanh\left(\frac{\kappa(V_+ - V_-)}{2}\right) & \text{if B5=0} \end{cases} \tag{3.1}$$

The current I_{DAC} is generated by the binary weighted currents *I1–I4* that originate from a current reference block on-chip. Six bits of local storage in the form of static flip-flops hold the weight in a digital form, *B5* being a sign bit switching the output current. The differential output currents I_{out+} and I_{out-} are commoned with other synapses and fed to a resistive neuron. Network non-linearity is distributed across the proceeding layer of synapses by virtue of the tanh non-linearity of the synapse transfer function equation 3.1. Several circuits to implement the resistive neuron have been fabricated and tested[26] culminating in a design utilising common-mode feedback to provide the necessary range for large fan-in networks[27].

The latest chip of the family includes a bucket brigade device (BBD) to sample and present the continuous time ICEG to the network. It also includes a variable gain neuron and winner-take-all circuitry as well as the necessary reference generation and decoding circuitry[27].

Power consumption of the chip is kept extremely low by cycling the current references of

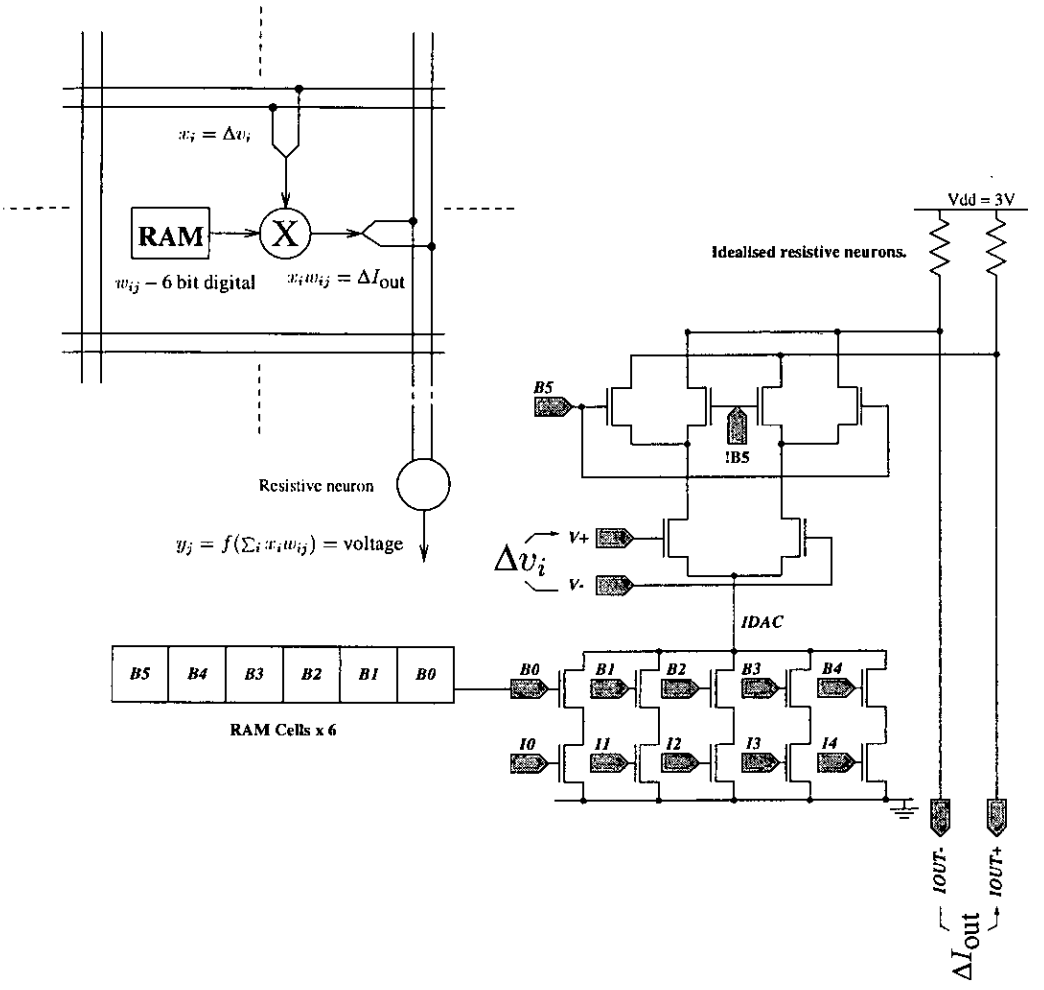


Figure 3–8: Kakadu synapse design

the bias circuitry achieving a total chip power dissipation of $186nW$ from a $3V$ supply for a nominal heart (i.e. classification) rate of 120bpm.

3.6.2 System Performance of Kakadu

A range of benchmarking test problems, such as XOR, four bit parity and a simple character recognition were successfully trained on Kakadu test chips[67,68] demonstrating its viability as a general neural network architecture. Several learning techniques were used to train Kakadu including modified backpropagation and weight perturbation. The most successful method was the combined search algorithm (CSA). The CSA uses the twin minimisation strategy of a modified weight perturbation combined with a random search[67,117].

The functional blocks of the final chip are highlighted in the floorplan of figure 3–9. Note that with the BBD input device the ICEG waveform is sampled directly into the chip. The winner-

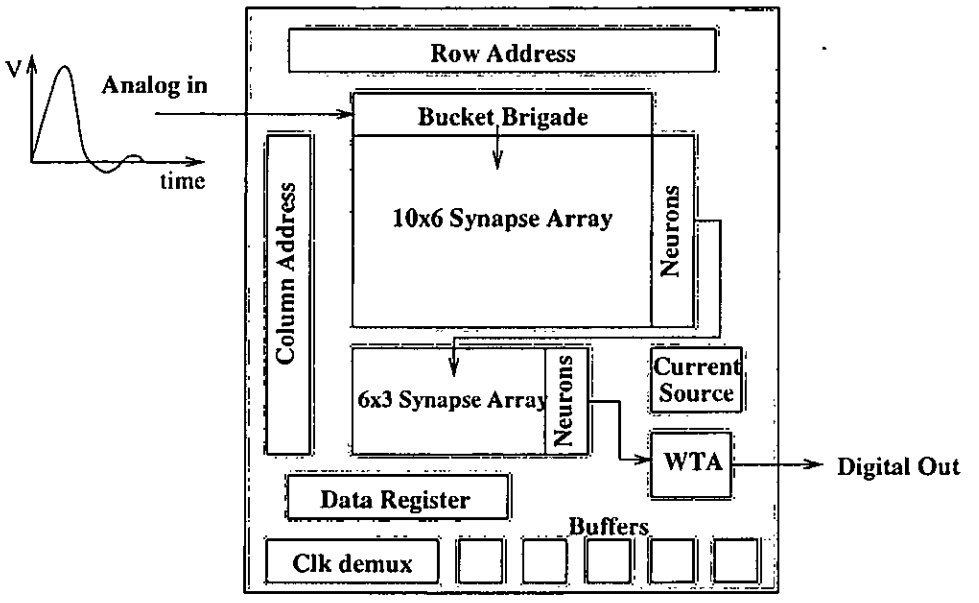


Figure 3–9: Floorplan of the Kakadu chip.

take-all network produces a digital output which allows direct interfacing to the defibrillator. This chip has been trained on the individual morphology data of seven patients. In six cases the chip was able to correctly classify dangerous tachycardia in the test set. Interestingly, the seventh patient, in which the network was not successful, proved to have a morphology of ventricular tachycardia that even a human expert had trouble distinguishing[27].

3.6.3 Kakadu Summary

Kakadu demonstrates a medium sized network (9 neurons and 78 synapses) successfully solving a rather complex classification problem. The low power, fully integrated design approach has led to a hybrid scheme of digital weight storage and analog state representation. With the addition of the BBD input structure, continuous time analog input is fed directly to the chip. Output from the WTA circuitry is essentially digital.

3.7 Discussion

The four examples described in the previous sections were chosen to illustrate neural networks designed for real-world applications. Table 3–2 summarises salient features for each implementation.

Included also are the figures for EPSILON, reviewed in Chapter 2, the starting point for hardware developed in this thesis. These case studies have been chosen to be representative,

	Synaptics OCR	**ETANN**	**Kakadu**	**CNAPS**	**EPSILON**
Connections per second $\times 10^9$	≈ 1	2.5	0.001[5]	1.2	0.36
Neurons	not known	64	9	64 processor nodes	30
Synaptic connections	$\approx$ 20,000	10,240	78		3,600
Weight storage	fixed by transistor geometry	Analog EEPROM	Digital	Digital	Dynamically refreshed capacitor
Technology and Size	$1.6\mu m$ CMOS 5x4.6mm	$1\mu m$ CMOS EEPROM 11.6x7.6mm	$1.2\mu m$ CMOS 2.2x2.2mm	$0.8\mu m$ CMOS 26.2x27.5mm	$1.5\mu m$ CMOS 9.5x10.1mm
Weight Resolution	not known	$\leq$ 7.5 bits	6 bits	8 or 16 bit fixed point	8 bit
Synapse Density (per mm^2)	$\approx$800	500	83.5	11.26	100
Power Consumption	10mW	1W	200nW	7W	350mW

Table 3–2. Summary of features of chips reviewed

though the best of, other designs presented in the literature. Interpreting the data from table 3–2 some generalisations on implementation of hardware neural networks can be extrapolated.

3.7.1 Neural Structure

Analog v Digital

Taking the CNAPS system as being representative of digital architectures we can see that it offers a high degree of flexibility in terms of network architecture as well as the ability of implementing learning algorithms. It also interfaces directly to conventional digital systems but requires extra sub-systems to interface to analog signals. In terms of power and silicon usage it is the least efficient of the cases studied.

[5]Design is optimised for a low power consumption, for higher bias, thus faster settling times, greater speed can be achieved.

Analog/Hybrid Architectures

The analog/hybrid architectures all have lower power, greater chip density and, in general, lower resolution than their digital counterparts. Resolution of multiply–accumulates is often hard to judge and dependent upon the noise floor of internal or even support circuitry[41]. Other factors delineating various designs include weight storage and interface ability.

Weight Storage

The four most common forms of weight storage are represented in the designs of table 3–2. The geometrically fixed weight storage of the Synaptics design is a specialised case where we can achieve very high space and power efficiencies at the expense of losing the adaptability often sought in neural solutions. Other fixed weight schemes include resistive arrays of binary valued amorphous silicon resistors[40,56], resistors in thin film technology[16], or use of fixed ratio capacitors[25].

EEPROM offers analog storage in a slightly more expensive process than standard CMOS. It is compact and essentially non-volatile but comes at the expense of slow programming. Other EEPROM schemes have been demonstrated apart from ETANN such as the compact two transistor cell of Kramer *et al*[63].

Kakadu and EPSILON both use digital techniques for primary weight storage. Kakadu incorporates this on-chip while EPSILON uses external memory to dynamically refresh on-chip capacitors.

It is fair to say that neural implementations are still waiting for technology to provide a better solution to analog weight storage. Some possibilities under development such as SONOS devices which offer lower programming voltages than EEPROM in smaller cells[114]. Another technology that shows promise is amorphous silicon (a-Si:H) structures. These devices may offer much faster programming times in a very compact size[53].

3.7.2 System Performance

Interface Considerations

Other differences between the neural network architectures of the four case studies lie in how they interface to both their target problems and systems. A dedicated solution such as the Synaptics I1000 shows the most efficient situation. Here input is highly parallel in the form of direct optical input. Output is a micro-controller compatible digital bus.

Kakadu also has an efficient analog front end where an input signal is captured in real-time by a BBD obviating the need for expensive (in terms of area and power) A/D conversion.

For generic usage, I/O is not so readily defined. Two issues are important here: Firstly the interfacing to target problems such as real-time analog signals or digital data and output interfacing to conventional digital systems. The second issue concerns cascadability where it is desirable for network outputs to be compatible with inputs.

Section 3.4.2 presented several examples of interface schemes to ETANN. It demonstrated that using analog busses an efficient interface could be developed and the massive D/A overhead needed to interface input to a digital bus. All ETANN applications required A/D conversion to interface output to a digital system as would be expected.

EPSILON offers analog or pulse modulated input (but not a mixture). Advantages of pulse modulated schemes is the digital signal levels obviate the need for A/D and D/A voltage conversion. Magnitude of signals is encoded in time so digital processing is still needed to convert values to and from integer values. If done by the host system this can be computationally expensive.

Suitable Applications

In looking at the problems that all the architectures reviewed have been applied to, definite trends emerge: Most are interfaced to real-world analog data. Even in most digital cases data is generated from real-world sources (such as speech) through A/D conversion perhaps followed by preprocessing (such as DFT analysis).

The most successful applications provide smooth interfaces between the neural component and input and output domains. The I1000 and Kakadu are good examples of this. All applications reviewed process data and pass it on to some conventional digital system for use and/or further processing. These factors tend to suggest that suitable applications for neural solutions are ones requiring processing of analog data and an interface to conventional digital computing.

3.8 Summary

The case studies presented in this chapter demonstrate the non-trivial nature of designing neural network hardware for applications usage. It was seen that analog/hybrid architectures offer compact, low power, high speed solutions when compared to digital implementations. Interface considerations were shown to be of primary importance as principal factor in the success of an architecture was how easily and efficiently it interfaced to target problems and systems.

Other researchers such as Vittoz[113] also expound this view that compact, low power circuits coupled with ease of interfacing to real-world data make analog techniques the principal choice for parallel neural computation.

The case studies also show neural networks are rarely used in a *stand-alone* fashion, but rather perform as part of an overall system, often pre-processing analog data.

Chapter 4

A System Specification for Hardware Neural Network Development

4.1 Introduction

This chapter presents the general specification for the hardware to be designed in this thesis. This specification was arrived at by considering the issues raised in the last chapter as well as experience gained from working with the original EPSILON chip. The specification was further focused by an examination of potential applications. Finally these are combined to outline a system specification that is realised in the subsequent hardware design work of the thesis.

4.2 Issues Raised by Case Studies

In the previous chapter some recent examples of neural network hardware were reviewed. The Kakadu chip and Synaptics cheque reader are examples of very application specific devices and although they have proved successful in providing solutions to their tasks, have not met with commercial success. The ETANN system had the goal of providing generic neural computation support, however in searching the literature for proof of its effectiveness one finds few examples of use to justify such a claim[1].

The CNAPS system perhaps comes closest to generic neural computation support. It achieves this with the cost of large silicon area and high power consumption. Its interface capabilities to digital systems is transparent but to analog/real-world data it requires large hardware overheads. Another disadvantage of this type of architecture with respect to applications is that there is no possibility of a highly integrated solution.

To aid the specification, the following generalisations are drawn from the case studies:

[1] In a search through the BIDS ISI database from 1985 to 1995 only 10 publications, of which 3 were from Intel claimed use of the system.

- Large generic neural processors such as ETANN fail to interface effectively to problems they might be used to solve. This lack of interface makes their advantages over conventional digital signal processing techniques or software implemented neural networks negligible; especially as their performance is static while the performance of software solutions increases with digital processor speed. This means their desirability is limited as little advantage is gained by providing a hardware solution.

- The modest size networks of the Kakadu and Synaptics class give an effective and efficient solution to a fixed problem. They have the advantage of interfacing closely to their goal tasks.

- Specialised hardware (such as the Synaptics cheque reader) fill a narrow niche that must be chosen carefully if the application is to be successful. Other considerations than pure technical merit may cause the implementation to be considered non-ideal.

- Successful implementations often appear in the area where we are dealing with real world (analog) data.

- Neural networks are most attractive where the possibility of a highly integrated solution is of great benefit. For example low power, single chip, small physical size for the Kakadu chip and mass production for the Synaptics cheque reader.

- Examples have been shown of problems where modest size networks offer good solutions for non-trivial problems.

- Analog based computation offers significant power and silicon density advantages over comparable digital based architectures.

4.3 Search for Suitable Applications

As the primary aim of this thesis is to use hardware neural networks in applications, the scope was not to limit the design to a particular application but keep the system generic so that it would be useful for prototyping a variety of applications. The danger of designing neural networks for the sake of themselves is that, while contributing significantly to our understanding of hardware issues, they often become unusable in applications, EPSILON and ETANN support this.

Consequently, throughout the project, a vigorous search was conducted for suitable applications to demonstrate the hardware and ensure the hardware was useable and useful.

Several possible applications were researched and successful demonstrators are presented later in the thesis. Defining a variety of target applications also assisted and focused a system specification, the applications considered were: an autonomous mobile robot, a character

recognition system, a link admission controller and inverted pendulum controller. These are discussed in the remainder of this section.

Autonomous mobile robot

This application, one that was followed to completion involves a controller for an autonomous mobile robot and is presented in Chapter 8. The control scheme is based on a software exemplar developed by Nehmzow[88]. The neural network hardware requirements for this controller are modest: a single layer network with four output neurons and 10's of inputs. Inputs consist of analog sensors along with generated digital data. Outputs must be available to a microprocessor for further processing. Advantages of a hardware solution to this problem are real-time operation and processing directly analog sensor signals on the limited power budget of an autonomous vehicle.

Simple character recognition

In preliminary work to gain familiarity with the original EPSILON/FENICS system, a demonstrator of a simple digit recognition MLP was implemented. While this problem is purely artificial it served to extract valuable lessons at chip and system level from the previous EPSILON work. The problem used purely digital I/O and the major lesson learnt was that data bottle-necks in transferring this digital data limited system speed and that these were all rooted at system rather than chip level. This problem is useful for testing the system and as a comparison to the original EPSILON system.

ATM routing

This function approximation and classification problem stems from work by Nordström and Gällmo *et al* [92]. It uses a MLP to approximate the probability of data packet loss in an ISDN network to facilitate the decision whether to accept new connections. A hardware solution is advantageous for this problem as this function needs to be carried out in real time in each network router.

Inverted pendulum controller

This was another real-time control problem considered. It again involves direct analog sensor input along with digital sensor and historical data. A substantial amount of time was spent in prototyping this problem but the mechanical difficulties of the system prevented completion of a fully working demonstrator.

4.4 System Specification

The generalisations made in Section 4.2, the experience extracted from the case studies of Chapter 3 and the lessons gained from the original EPSILON device all merge to create the following list of axioms around which the system specification evolves:

1. The system must have the ability to deal with analog inputs.
2. The system must have the ability to deal with digital data.
3. The system must interface with conventional serial machines.
4. The system can be of modest network size yet large enough to solve useful problems.
5. The system must be able operate independently without external control.
6. The system must be cascadable.
7. The system must have the potential for a highly integrated solution.

The remainder of the section expands these points.

1. Analog inputs

The most suitable niche for neural network technology is in interpreting or preprocessing real world data. This data is, in general, inherently analog in nature. Thus to avoid an unnecessary external hardware overhead, any system must be able to accept analog data inputs.

2. Digital inputs

If the system is to interface to a conventional computer, the network must be able to accept data generated by its host. Also, the most effective means of data storage available to us is digital so historical information is most likely to come in this form.

3. Interface considerations

Consider the neural network as a data preprocessor; the data must be freely available to the next stage of computation. It is generally envisaged that the neural network will be a slave device of a serial based host machine, thus a digital bus interface common to other I/O peripherals on

the system seems sensible. The danger with highly parallel neural technology is that system performance can be degraded by data-flow bottle necks.

4. Size

Examples of networks of modest size (10's of neuron as against 100's or more) were seen in Chapter 3 performing useful tasks. Conversely, larger neural network chips become unwieldy in terms of smoothly integrating with a digital system. It is these two factors along with a restricted silicon budget for this work that the specification calls for a modest size networks instead of "massive" one.

5. Autonomy

As an effective sub-system of an overall system a neural network processor must demand as little computational overhead from its host as possible. Ideally the only functions a host should provide are overall control signals and an interface to I/O data. To achieve this a neural network must provide its own support functions such as weight storage and I/O control.

6. Cascadability

As the goal of this work is to provide hardware support for prototyping neural network applications, it is highly likely that for some problems, network size will be greater than the physical network on chip. For this reason it is important that chips can be cascaded to increase network size or depth. The major requirement needed to meet this specification is compatible input and output representations.

7. Integration

Chapter 3 demonstrated that many applications demand a highly integrated solution for success. While not proposing to produce a fully integrated system designed for a specific application, design choices leading to possibilities of further integration should be favoured.

4.5 Summary

This chapter has drawn together the research conducted through literature survey, the first hand experience and knowledge of the EPSILON system and the requisite needs of possible applications to produce a coarse specification of the hardware to be developed. In the next two chapters these specifications will be expanded and implemented in the design of the chip level and system level hardware which underpins the work of this thesis.

Part II

Hardware Development

Chapter 5
The EPSILON II Chip

5.1 Introduction

In previous chapters we have introduced pulse stream methodology and circuits, examined how and where to apply neural network hardware to applications and evolved a specification on how that may be achieved. This chapter presents the VLSI hardware that forms the foundation for later system level work. It will start by reviewing the original EPSILON design and detailing improvements to the device. Following this the specifications of Chapter 4 will be addressed and changes outlined to satisfy these. VLSI improvements occur either at a circuit level or architectural level; these improvements are presented next. The final section will present characterisation results from the chip, assess the success of the design and offers solutions to the problems encountered.

5.2 EPSILON Background

The EPSILON chip represented a significant achievement in the evolution of pulse stream neural networks. It realised a network of sufficient size for use on real world problems and was demonstrated performing such tasks as vowel recognition with MLPs[44], image classification[24] and optimisation with Kohonen networks[9]. It did however have several shortcomings that interfered with predicted operation, these are presented below along with the solutions implemented:

1. **Multiplier Error:** Synaptic multiplication exhibited an anomaly in the characteristic close to and at zero input (see figure 2–12). This was attributed by Baxter in [9] to a power supply distribution problem arising from the non-zero sheet resistance of the long metal power tracks that supplied the synaptic array.

Solution: Minimise power track length from pad to core and make distribution into core symmetrical. Thicken power tracks through core.

2. **Neuron Offset:** Device matching problems caused a random offset that affected neurons by producing variation in their zero reference level. The extent of this affect was hard

to quantify due to the distortion of the neuron characteristic by the power distribution problem.

Solution: For practical use of EPSILON a number of synapses were used as bias to alleviate this effect reducing the total number of synapses available to the network. For EPSILON II additional autobias synapses were added to remove neuron zero offsets. This is achieved by adjusting the autobias synapse weights to produce a zero neuron output for zero input.

3. **Ineffective On-chip Biases:** Feedback bias generation circuitry (dummy synapses) failed to produce the correct values of bias for operation.

Solution: With the power distribution problem it was difficult to assess the extent of this problem; however bias array was located on the edge of the chip. To cancel edge effects and get a better match to operational synapses dummy synapses should be placed in the middle of the array.

4. **Injected Digital Noise:** Shift registers for the weight refresh operation caused large spikes on the supply rails injecting noise into the synaptic array.

Solution: Design shift registers with smaller transistors to draw less transient current and isolate power supplies.

5. **Coupling of Pulses and Analog References:** Routing problems on the chip led to excessive coupling between digital pulse modulated signals and sensitive analog references.

Solution: Develop chip floorplan and architecture to minimise signal path length of analog references and isolate digital signals form analog ones.

6. **Incorrect Control Signal:** Phase locked loop (PLL) circuitry for pulse frequency neurons needed an extra inversion of signal for correct operation.

Solution: Add an extra inversion to PLL.

The details of these solutions is presented in the architecture and circuit design sections of this chapter.

5.3 System Level Requirements

A number of the system level requirements outlined in Chapter 4 have a bearing on chip level design. This section describes the areas that were addressed in the design:

1. Mixed signal inputs.
2. Recovery of analog inputs.
3. Minimising control signals.
4. Access to neuron activity voltages.

5.3.1 Mixed Signal Inputs

The need to deal with both analog and digital inputs has been specified. EPSILON could be configured to have **all** analog inputs **or** all pulse stream inputs. It has been noted that many problems in the target application area lie on the boundary of real-world and digital systems and require a mixed signal approach whereby analog and digital data is fused in neural network processing. For this reason it was decided to re-design the input neuron structure of EPSILON II such that each input could be individually configured as an analog or pulse modulated signal.

5.3.2 Analog Recovery

Section 4.4 specified a high level of autonomy and integration. One requirement for operation as an analog interface is a knowledge of inputs states as this is required for many learning schemes, for example backpropagation. This is obviously non-trivial in the case of analog inputs as an extensive overhead of A/D conversion would be needed to extract this information. To overcome this the facility of recovering analog inputs as a linearly pulse-width modulated output was also designed into the chip. This is relatively trivial as input neurons carry out this function to present data to the synaptic array. All that was required was to route this signal to output pads.

5.3.3 Control Rationalisation

As highly parallel designs such as neural networks are essentially pad limited, a rationalisation of control signals was undertaken to integrate control logic for the weight refresh operation, mode selection and chip control.

5.3.4 Activity Preset

Some potential applications examined used Hopfield networks[70], a fully fed-back architecture where inputs are imposed at the output and the network allowed to settle in a stable state. To allow this a facility to preset individual neuron activity voltages was added.

5.4 EPSILON II Specifications

EPSILON Chip Specifications		
Major Design Changes		
	EPSILON	EPSILON II
No. of state input pins	30	32
No. of actual state inputs	120, MUX'd in banks of 30	32
No. of State outputs	30 Directly pinned out	32 Directly pinned out
Input mode programmability	All analog/All digital	Bit programmable
Digital recovery of analog inputs	No	Yes - PW modulated
Additional *autobias* synapses	None	4 per output neuron
Programmable activity voltage	No	Yes
Number of control signals	11	6
Secondary Specifications		
No. of synapses	3600	1024
No. of weight load channels	2	1
Weight load time	$3.6ms$	$2.3ms$
Weight storage	Dynamic	Dynamic
Maximum speed (cps)	$360Mcps$	$102.4Mcps$
Maximum input sampling rate	50kHz	50kHz
Technology	$1.5\mu m$ CMOS	$1.5\mu m$ CMOS
Die size	$9.5mm \times 10.1mm$	$6.9mm \times 7mm$
Maximum power dissipation	$350mW$	$320mW$

Table 5–1. Comparison of EPSILON and EPSILON II Specifications

The previous sections outlined the majority of the circuit changes required to meet the specifications of EPSILON II. What remains is the choice of network architecture. The silicon area available for the design was $49mm^2$ which formed the major constraint on the design. It was decided that for efficient interfacing to digital systems I/O dimensions should be a

multiple of eight. This, together with the area available, led to a 32x32[1] network architecture. Table 5–1 summarises the specifications of the EPSILON II device and highlights the major changes between it and its predecessor EPSILON. The details of how these specifications were implemented is the topic of the next two sections; the first presents architectural details while the second presents circuit level design.

5.5 EPSILON II Architecture

Results from Baxter[9] and studies undertaken by the author point to many of the EPSILON shortcomings being attributed to layout level architecture. This section examines the architecture of the original EPSILON chip then presents the architecture of the EPSILON II chip discussing the improvements made.

Figure 5–1 shows the general architecture of the EPSILON device and highlights signal flow through the chip. The following points can be derived from this figure:

- Routing of analog references is very long and crosses all pulse stream inputs or outputs contributing greatly to unwanted signal coupling.
- Power routing is long and asymmetric.
- Many routing paths are unnecessarily lengthened by poor pad placements. For instance analog references are padded onto the chip from the opposite side from which they are used.
- Bias generation dummy synapses are on the edge of the array: For more representative operation under conditions of a systematic variation in threshold voltage across the die, these should be in a central location.
- Some digital control signals have long signal paths that cross analog references promoting coupling to these sensitive analog signals.
- Routing of synaptic input signals (green) proceeds to the top left hand corner before being distributed back down the chip to input neurons. This forms unnecessarily long signal paths (these can be a mixture of analog or digital signals) and leads to coupling with analog references (this was demonstrated in Hamilton [44]).
- Neuron outputs (dark blue) originate from the bottom of the core and are routed to the right hand edge pads. Again signal paths are long and this leads to coupling with analog references.

[1] When stating network size in this manner we are referring to *input dimension* × *output dimension.*

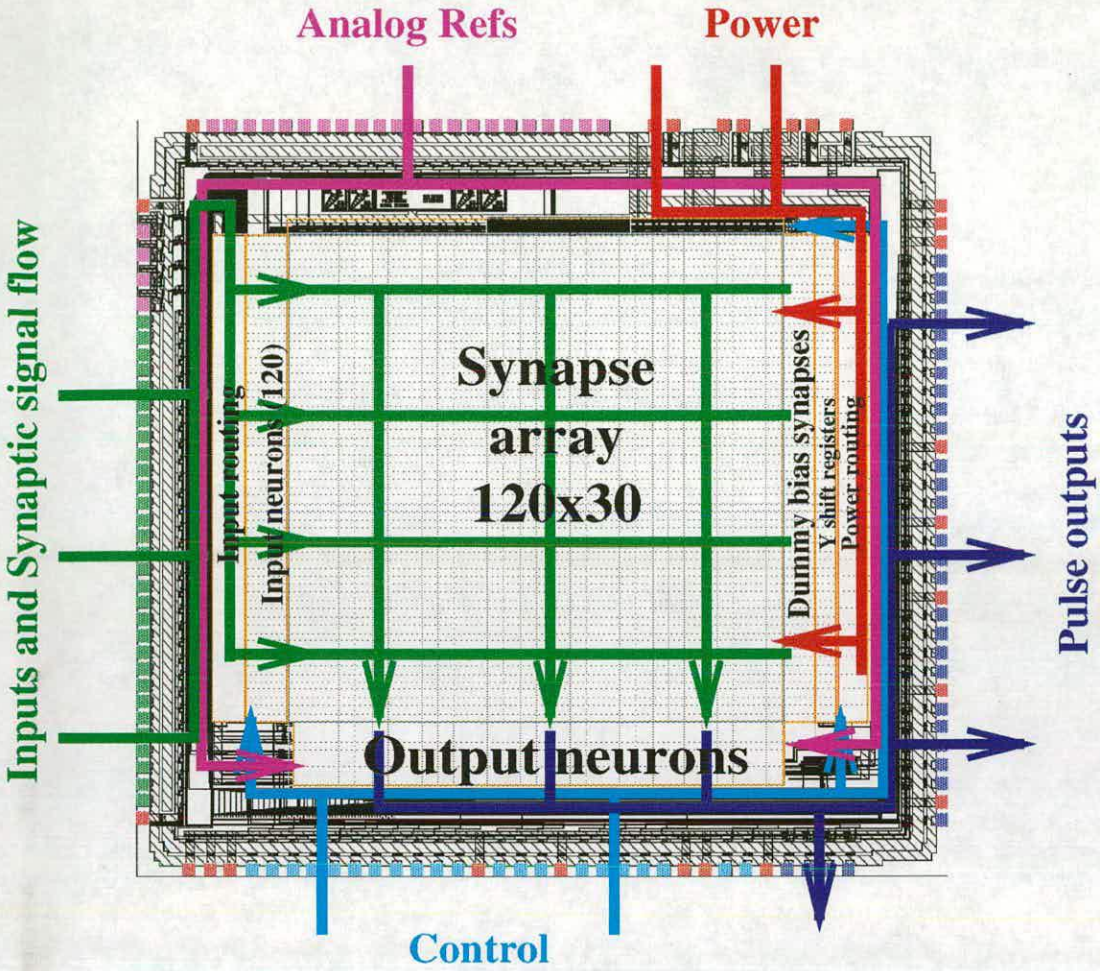


Figure 5–1: Signal flow through EPSILON

Figure 5–2 shows how these architectural points were addressed in the design of EPSILON II.

- The 32x32 synaptic matrix is split into four 32x8 arrays and placed symmetrically about the chip centre.

- Input neurons, routing, as well as input pads are all along the left hand side of the chip minimising signal paths from pad to input neurons.

- Dummy bias synapses are located in a vertical column through the centre of the chip to obtain a better match to synapse array.

- Output neurons are located and padded out, sixteen each, on the top and bottom edges of the chip. This minimises routing distance and isolates pulse outputs from the noise sensitive analog references.

- Analog references for the synaptic arrays along with synapse power supplies all originate along the right-hand side of the chip well away from and digital signal routing.

- 'X' shift registers run horizontally through the chip centre while 'Y' registers run vertically along the right-hand edge along with power routing. Power to these is isolated from the analog supplies.

These architectural modifications are designed to reduce or alleviate the non-idealities found in EPSILON as outlined in Section 5.2.

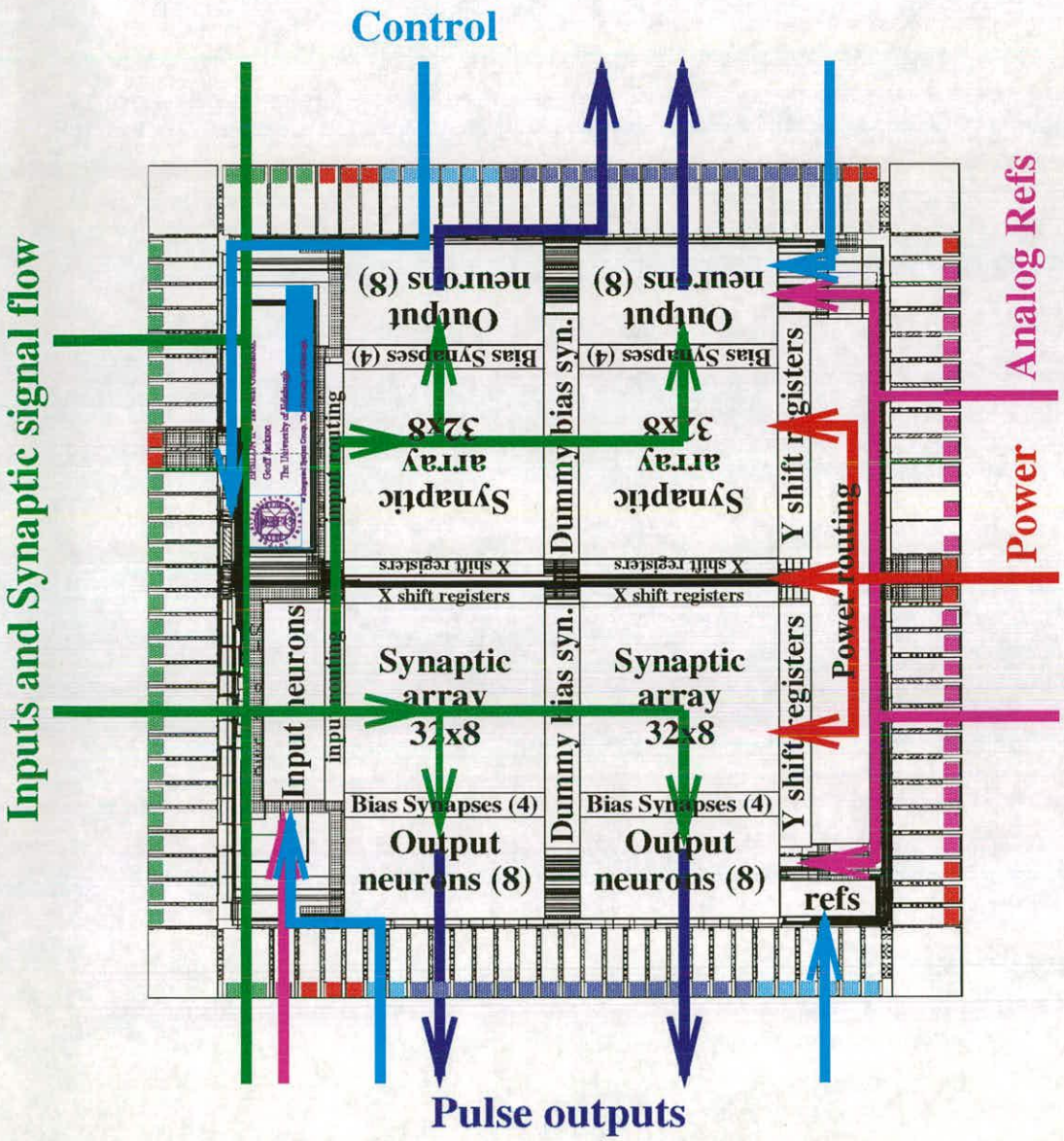


Figure 5–2: Signal flow through EPSILON II

5.6 EPSILON II Circuits

This section presents the circuit changes made to address the remaining problems of Section 5.2 and the system requirements of Section 5.3. These are presented as functional blocks:

1. Input neuron.
2. Synapse.
3. Output neuron.
4. Shift register.
5. Control logic.

5.6.1 Input Neuron

The specification for the input neuron is for the cell to be programmable as either an analog or pulse modulated input. To accomplish this the circuit if Figure 5–3 is used. The SRAM cell

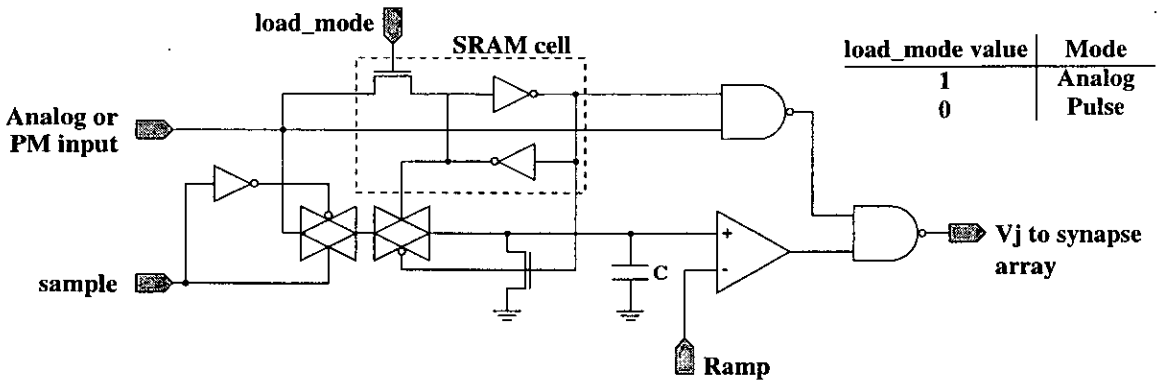


Figure 5–3: EPSILON II Input Neuron Circuit

of cross-coupled invertors is loaded with the value of the associated synaptic input when the **load_mode** signal is high. The invertors are scaled such that this can be done through a single NMOS transistor (circuit details are given in Section A.2).

In analog mode ('1' loaded) the signal from the synaptic input pad is sampled onto the capacitor **C** when the **sample** signal is high. This value is held on **C** when **sample** goes low and a linear dual-sided ramp is applied as explained in Section 2.3.2 to produce a pulse-width modulated output to the synaptic array.

The layout of the cell is pitch-matched for the synapse array, implying that two input neurons occupy a cell $100\mu m$ in height. The layout is shown in Appendix A, figure A–3 highlighting the various circuit components. NAND gates are scaled to provide the necessary drive to the synapse array.

5.6.2 Synapse

The synapse used was basically that in the EPSILON device. The only changes made were to the horizontal metal 1 routing to provide thicker power rails and the addition of two vertical bus lines to route input pulse-widths to output pads for the analog recovery mode. These two lines are utilised by an "overlay" cell containing a pass transistor which is placed on the diagonal axis through the array. Figure A–4 in Appendix A shows the layout of this cell.

5.6.3 Output Neuron

Basic circuitry is again similar to that of EPSILON and includes the feedback operational amplifier for the synapse array[9], pulse-width modulating neuron[24] and pulse frequency modulating neuron[44]. A cell for control of the activity capacitor allows it to be reset to a mid-point or accessed and programmed similarly to a weight refresh operation. A new cell to multiplex the output modes was also designed. This cell selects the neuron output as either pulse-width mode, pulse-frequency mode or the pulse-width modulated analog recovery mode. Layout of the output neuron in included in Appendix A, Figure A–5.

5.6.4 Shift Register

The operation of the raster-scan weight refresh requires shift register cells pitch matched to the X and Y dimensions of the synapse array. As previously mentioned, the shift register in EPSILON caused large power supply transients and so re-design was undertaken to prevent this. Simulation and layout were performed by a Napier University undergraduate project student, Alan Clark, under supervision of the author and Dr Alister Hamilton. Layout for these cells can be found in Section A.6.

5.6.5 Control Rationalisation

The pad-limited nature of an I/O dense chip such as EPSILON II meant that pad usage had to be kept to a minimum. Coupled with this was the desire to keep external support circuitry to a minimum, this led to design of logic to minimise the number control signals. The EPSILON chip contained eight pads for control of the weight refresh system. This was reduced to four pads on EPSILON II by designing the refresh control logic of Figure 5–4. External signals

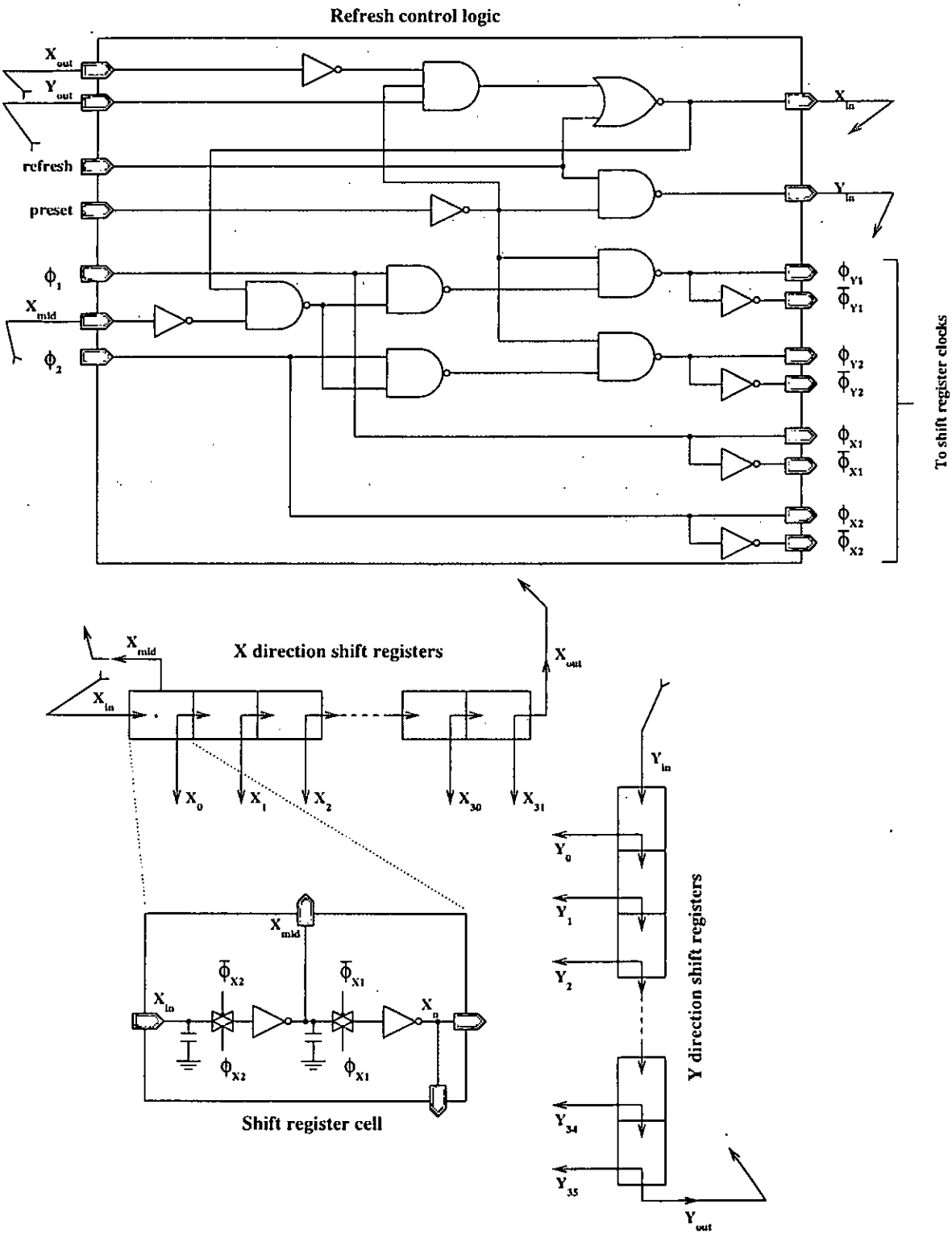


Figure 5–4: EPSILON II Shift Register Control Logic

are now reduced to a two phase clock, ϕ_1 and ϕ_2, which clock the X direction shift registers, a **refresh** and **preset** control signal. The application of the refresh signal starts a refresh cycle coincident with ϕ_1. The **preset** signal is used to reset the X and Y registers and along with **refresh** is used to start a preset operation on the activity capacitors.

5.7 EPSILON II Characterisation Results

This section performs basic characterisation tests on the EPSILON II device to gauge the success of the design changes. Several series of tests are described in in this section:

1. Initial verification tests.
2. Characterisation of the pulse-width neuron.
3. Testing of reference generation circuitry.
4. Neuron variation for zero input response: a test to examine the spread of offsets in the neuron characteristics.
5. Tests to judge the performance of the autobiasing scheme in removing the above offsets.
6. Characterisation of the synaptic multiplication.
7. An investigation into an anomaly in neuron zero response.

5.7.1 Initial Testing

The EPSILON II chip was fabricated using European Silicon Structures $1.5\mu m$ double metal CMOS process. A photomicrograph of the chip is shown in figure 5–5. After fabrication a series of tests were performed on the chip to verify that it functioned correctly. These tests were performed using the FENICS board designed for the EPSILON chip[44] by constructing a prototype board containing EPSILON II to plug into the EPSILON chip socket. Tests done included:

- Testing of weight download system.
- Simple characterisation runs to test synapse operation.
- Testing of PWM and PFM neurons to confirm correct operation.

These tests proved successful so the design of the EPSILON processor card (EPC) described in Chapter 6 was undertaken. The results presented in this section were taken using of the EPC.

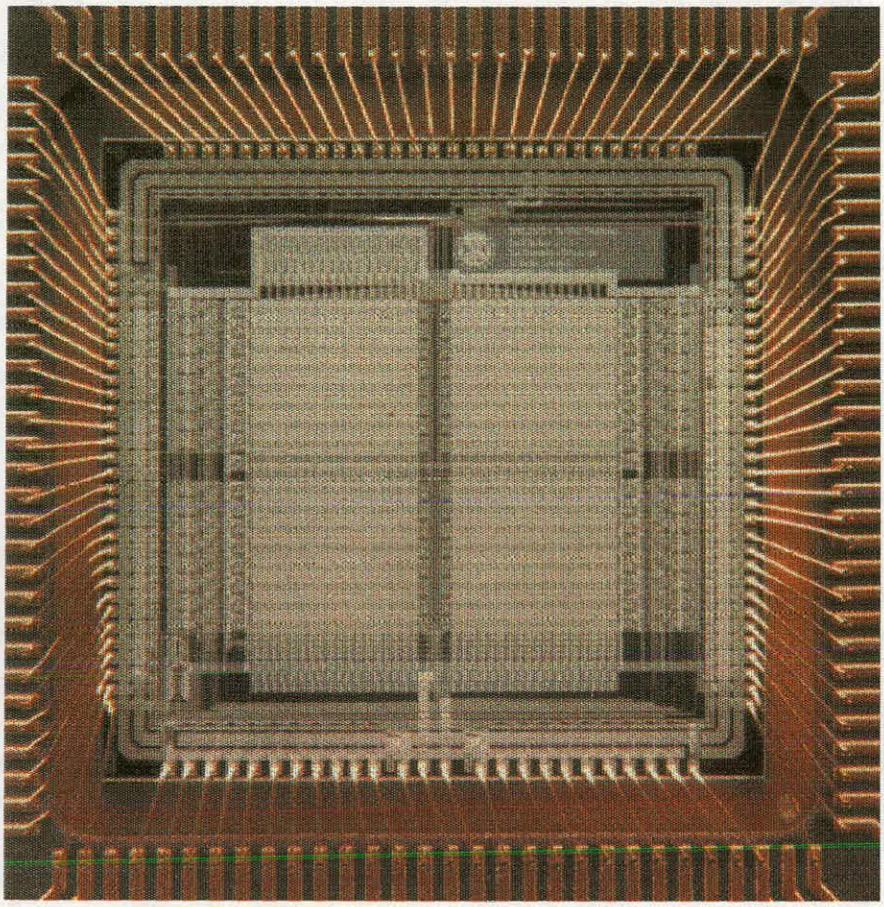


Figure 5–5: EPSILON II Chip Photograph

5.7.2 Pulse-Width Neuron Characterisation

The output mode of operation used throughout this thesis is the pulse-width modulating mode as it offers substantial flexibility in control of the neuron transfer function. To test this mode and verify its operation the reset voltage (shown in figure 2–7) was swept across the input range, centred on 2.5V, and the pulse-width of the output pulse measured. The circuit proved linear with a maximum of 1 bit offset to the 8 bit resolution of the measuring system (the EPC); confirming the linearity of the ramp generation circuitry and the robust nature of the pulse-width comparator neuron.

5.7.3 Reference Bias Setup

The synapse circuitry requires two global reference voltages that set the synapse characteristic zero or mid-points. The first of these is V_{sz} which determines the zero weight voltage (see figure 2–6). The second reference V_{oz} sets the voltage integrator mid-point such that for zero synaptic current the integrator output produces a nominal $10\mu s$ output pulse-width. Both these

references have feedback generation circuitry as explained in Section 2.3.1. To test these references all synaptic weights were set to zero value ($T_{ij} = 3.75V$) and the neuron output pulse-width measured with a linear ramp for:

1. A zero input ($S_i = 0\mu s$) pulse-width (zero input response).

2. A $S_i = 10\mu s$ input pulse-width (zero weight response).

For correct operation both these inputs should yield the nominal midpoint of $S_j = 10\mu s$ as net synaptic activity should be zero in both cases.

Unfortunately, despite the architectural changes made in EPSILON II, these circuits failed to generate suitable references for correct operation. This implies that the power supply distribution problem and edge effects (Section 5.2) of EPSILON was not wholly responsible for malfunction of these reference generators.

To investigate these references they were set manually by adjusting V_{oz} to get a nominal $S_j = 10\mu s$ output for a $S_i = 0\mu s$ input pulse then adjusting V_{sz} to get a nominal $S_j = 10\mu s$ output for a $S_i = 10\mu s$ input pulse. Results from these tests indicate that a variation of 10mV in V_{oz} led to a large ($\approx 2\mu s$) change in neuron output[2]. This indicates that the circuitry is very sensitive to this reference – the effects of this is discussed further in Section 5.8. For the remainder of the experiments presented here V_{sz} and V_{oz} are set manually.

5.7.4 Zero Input Response Variation

With V_{sz} and V_{oz} set to produce an average $S_i = 10\mu s$ output, it is possible to examine the spread of neuron offsets by measuring their zero input response. This offset is due to the mismatches between the distributed feedback amplifiers and voltage integrators for each neuron.

The variation of the zero input response for the seventeen operational chips was measured and the results summarised in figure 5–6. The figure is a histogram showing the frequency of the zero output response for 5,100 measurements (5,100 - 17 chips × 32 neurons × 10 repetitions). As the graph shows, there is a wide spread of zero input response values, most lying in the range of $5\mu s$–$15\mu s$. This large spread indicates a poor matching between neurons. To confirm that this is not due to an architectural problem as on EPSILON, where power supply variations caused large variations, in figure 5–7 each neuron is plotted separately for the seventeen working chips. In this graph average values of the zero input response along with the standard deviation is shown. From this it can be seen that there is no gross trend according to geographical position on the chip as was the case with EPSILON. This implies that while the architectural changes made were successful in alleviating problems in EPSILON, these problems masked other non-idealities now apparent in the spread of zero input response. The two circuit blocks

[2] 10mV was the smallest increment measurable with available equipment.

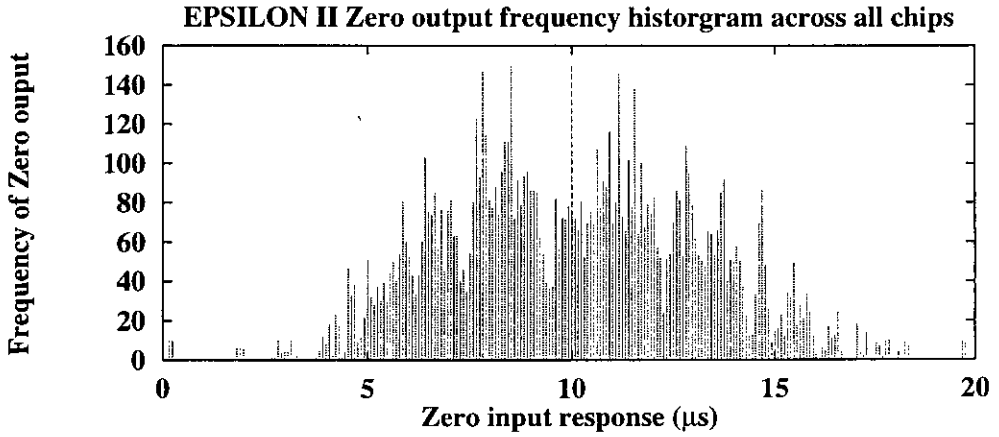


Figure 5–6: EPSILON II Characterisation – Zero State Histogram.

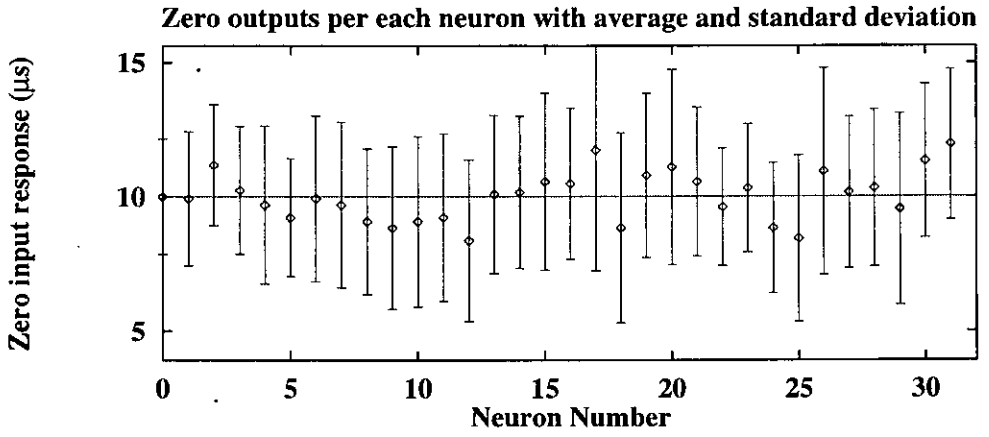


Figure 5–7: EPSILON II Characterisation – Zero State of Neurons.

responsible for this characteristic are the distributed feedback amplifier(DFA) and the voltage integrator. Their contributions to these non-idealities are discussed in Section 5.8. The next section determines how effectively these offsets can be removed by using the autobias synapses included on EPSILON II.

5.7.5 Autobias Characterisation

That offsets (due to component mis-matches) in neuron zero input response would be present was a known problem; for this reason autobias synapses were included as a way of removing these. This section details the results of the tests performed using the autobias synapses to determine their ability to reduce zero input offsets. This reduction is achieved by adjusting the weights of the autobias synapses iteratively until the neuron zero input response is the desired $10\mu s$. Figure 5–8 shows this process in operation whereby all neurons characteristics are shifted

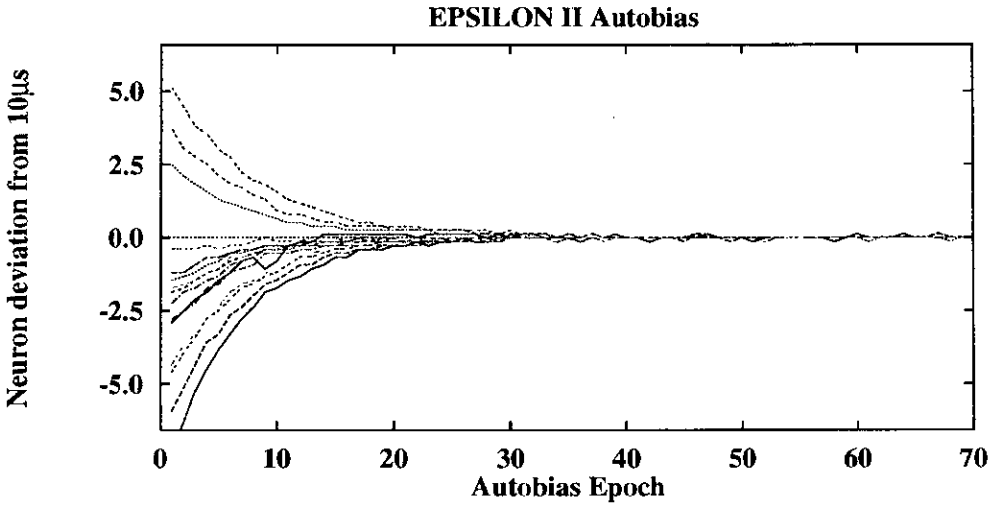


Figure 5–8: EPSILON II Autobias Settling Response.

to the defined zero point. The autobias neurons can shift the characteristic by approximately $\pm 6\mu s$, which means that some neurons lie outside this range as seen in figure 5–6. These neurons can either be used as they will now lie closer to the ideal zero point, or not used decreasing the number of neurons available.

Table A–5 gives a summary of the twenty chips received from fabrication, their functionality, and how many neurons lie out of autobias range. The majority of chips have less than four neurons out of autobias range. In Chapter 7 it is shown that learning with the chip in loop can accommodate such hardware non-idealities. The aim of removing offsets as far as possible is to achieve close matching between chips and to allow the possibility of off-line evolution of weights sets in software. This section has shown that the addition of autobias synapses to EPSILON II removes all but a few of the zero input offsets associated with neurons.

5.7.6 Synapse/Neuron Characterisation

This section examines the multiplication characteristics of the synapses. Due to the nature of the chip it is not possible to characterise individual synapses. Instead, columns of synapses are characterised by applying identical weights and inputs to all synapses and reading output from a linear pulse-width modulating neuron. Figure 5–9 shows the characteristic as input pulses are swept from $S_i = 0 \rightarrow 20\mu s$ for various weight voltages. Similarly, figure 5–10 shows the characteristic as weight voltage is swept from $T_{ij} = 2.5 \rightarrow 5.0V$ for various input pulse-widths. Both these graphs plot the average of 30 run samples for a single column of synapses, in this case column 1 on chip 2.

From both these sweeps it can be seen that the EPSILON II synapse achieves good linearity. To further examine the spread of these values figure 5–11 and figure 5–12 show error bars

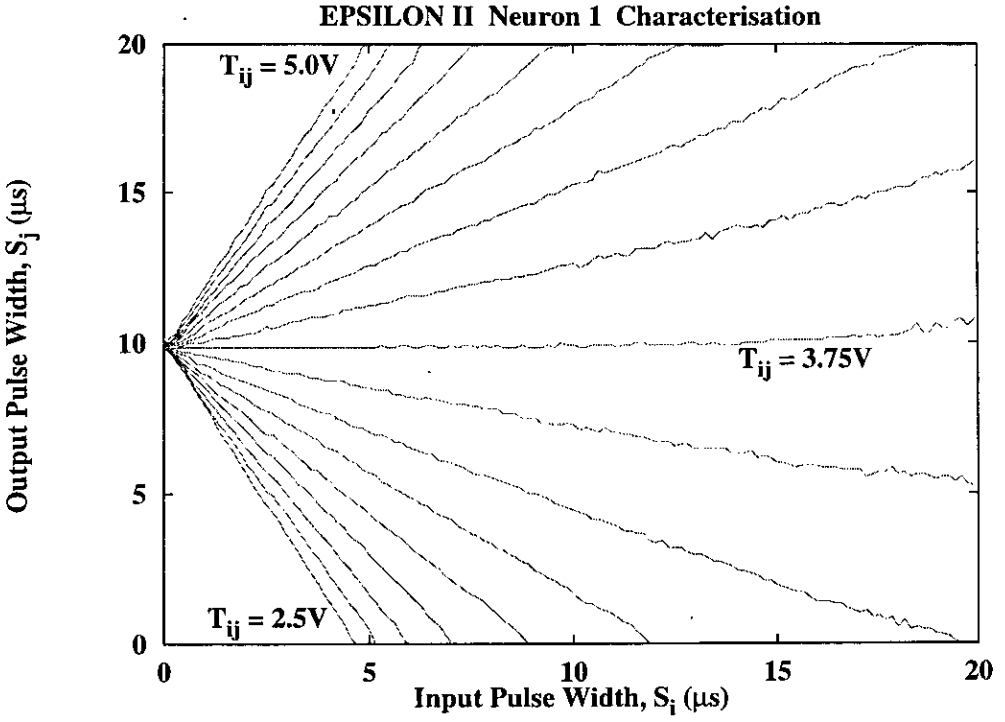


Figure 5–9: EPSILON II Characterisation – Neuron Average Response.

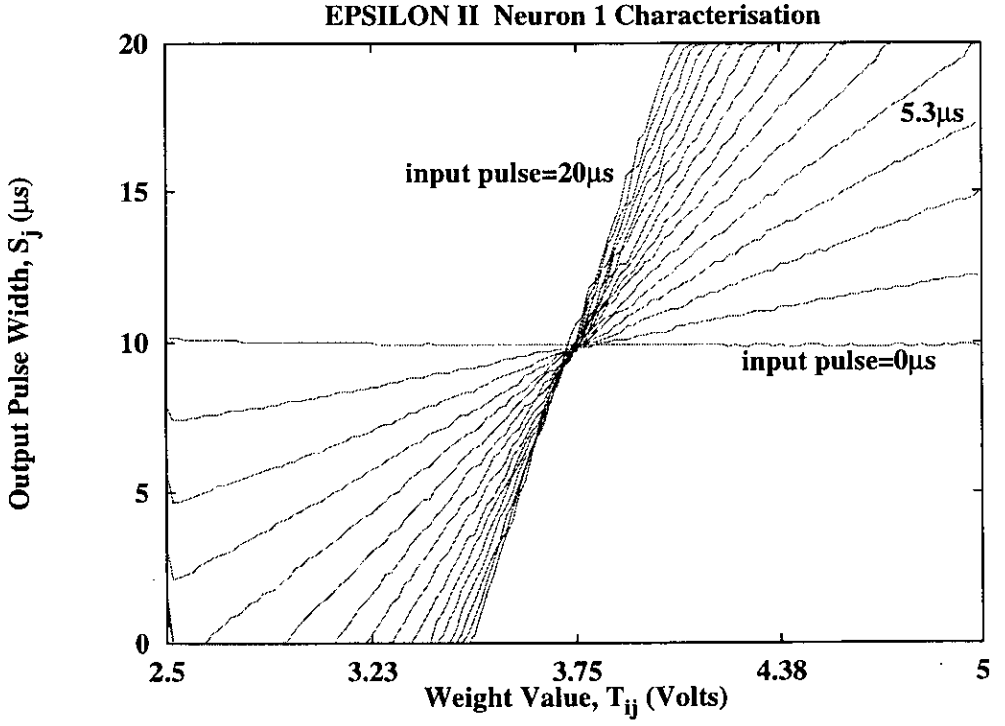


Figure 5–10: EPSILON II Characterisation – Weight Response.

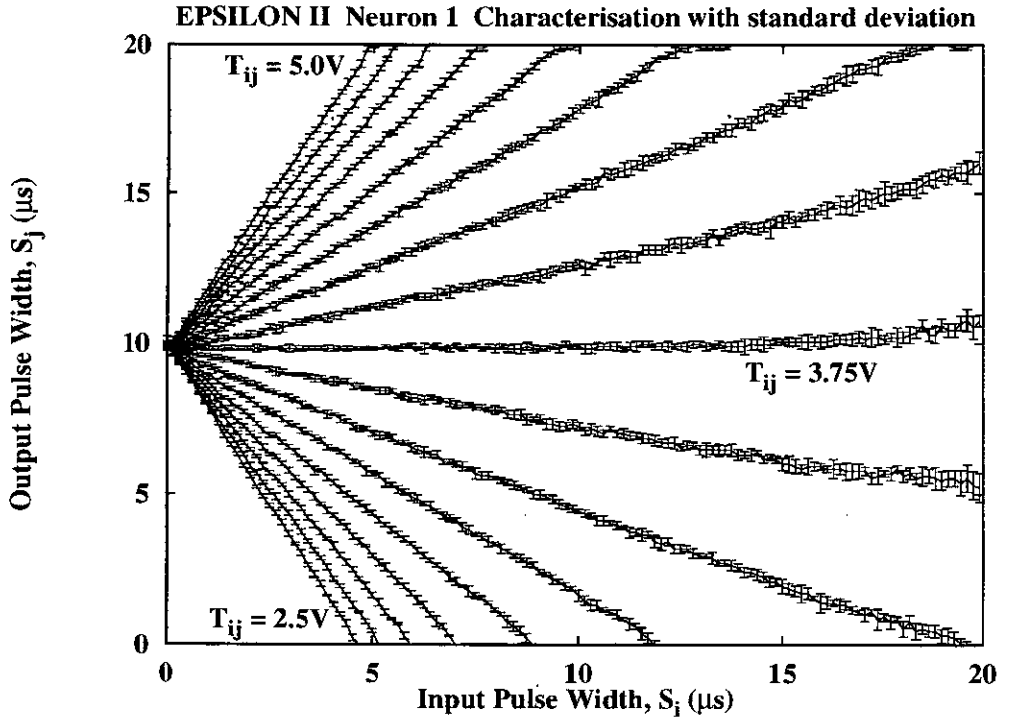


Figure 5–11: EPSILON II Characterisation – Response with Standard Deviation.

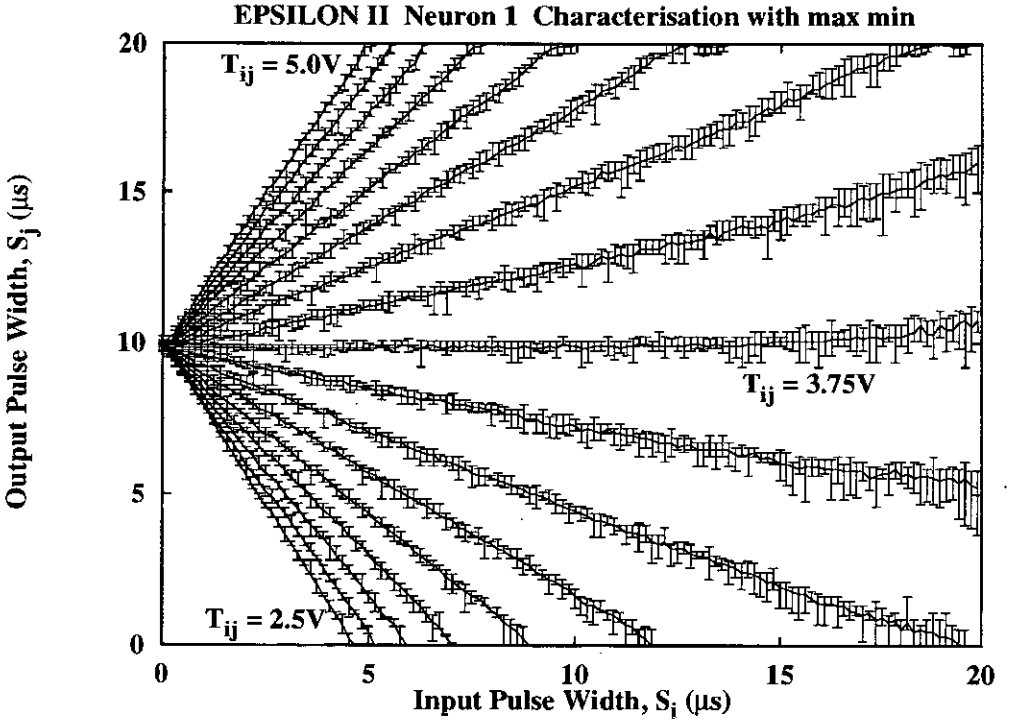


Figure 5–12: EPSILON II Characterisation – Response with Maximum and Minimum.

of the standard deviation and maximum and minimum values respectively of these sweeps. These results show the spread of values becomes progressively greater with increasing input pulse-width. This is as expected as the spread is primarily due to noise inherent in the synapses, which is only fed through when an input is active.

An additional point to note from these graphs is the tight zero point; that is all sweeps converge to a single point at the $10\mu s$ output pulse. This is in contrast to EPSILON where power supply problems caused great distortion in this region (see figure 2–12). This implies that the power supply distribution problem of EPSILON has been solved. In similar characterisations performed on EPSILON a maximum (minimum) absolute standard deviation of $857ns$ ($517ns$) was recorded[9]. The maximum (minimum) absolute standard deviation for the characterisation of figure 5–11 is $517ns$ ($17ns$). This tighter standard deviation represents a significant improvement in the noise response indicating that isolation of analog references has been successful in reducing noise coupling.

The trace for a zero weight value (i.e. $T_{ij} = 3.75V$) is not perfectly horizontal. This varies from synapse column to column as the value of the global zero reference V_{sz} needed to anull the zero weight current varies with process parameters and device mismatch. This is not considered a major problem and in Chapter 7 it is shown that chip-in-loop learning can accommodate such variations.

5.7.7 Neuron 0 Anomaly

In the course of testing the chip an anomaly in the characteristics of neuron 0 (bottom left-hand corner of figure 5–2) was revealed. Investigation proved that response of neuron 0 was being affected by the weight values present on neuron 31 (top right-hand corner of figure 5–2). This implied some coupling of the weights on neuron 0 with those on neuron 31. It was discovered that the delay induced by an extra logic gate, along with the long signal path from weight refresh logic to the Y-shift registers, produced a delay in the clocking of the Y-shift register. Thus in the transition between addressing weight $T_{31,n}$ and $T_{0,n+1}$ an overlap caused by the delay path momentarily addresses $T_{0,n}$ affecting its stored value. Though this does not fully overwrite the column 0 synaptic weights, in practice column (neuron) 0 was not used in experiments.

The solution to this problem for future fabrications is to simply add a dummy X-shift register after column 31 to allow the Y-register to change rows without inadvertently addressing any synaptic weights.

5.8 EPSILON II – Unresolved Issues

Two major unresolved issues remain in the performance of EPSILON II circuitry:

1. The failure of bias generation circuitry to produce the correct values of V_{oz} and V_{sz} for chip operation.

2. The large spread in offsets of neuron characteristics that was observed as quantified in the zero input response measurements.

It is believed that these two non-idealities are linked and this section discusses possible causes, concludes what is the major contributing cause and offers a possible solution.

5.8.1 Investigating Neuron Offset

It is relatively easy to narrow down candidate blocks to investigate neuron offset characteristics:

- Synapse transconductance pairs are eliminated as they are isolated from the circuit for the zero input response measurements.

- The pulse-width comparator was shown in Section 5.7.2 to have minimal offset.

The remaining two candidates are:

- The voltage integrator.

- The distributed feedback amplifier.

Voltage Integrator

The voltage integrator is simply a differential transconductance device which subtracts the net synaptic activity, V_{out_j} from the output zero reference, V_{oz} and converts this to a current, I_{out_j} which is integrated in the integration capacitor, C_{int} (see figure 5–13). The transconductance stage consists of a differential stage followed by a current mirror. As with any CMOS differential stage mismatch between "identical" MOS devices, along with bias errors from mirror matching, will produce an input offset. Allen and Holberg[4] quote this as being typically in the $5mV$ – $20mV$ range.

SPICE simulations show that the predicted transconductance, g_m of the stage is:

$$g_m = 1.1\mu S \tag{5.1}$$

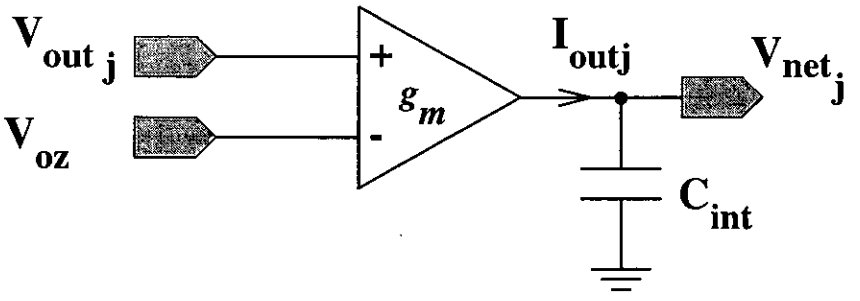


Figure 5–13: EPSILON II Voltage Integrator.

Thus for a worst case input offset of $\pm 20mV$:

$$\Delta I_{\text{out}_j} = \pm 22nA \tag{5.2}$$

This current charges the integration capacitor $C_{\text{int}} = 3pF$ giving a voltage change in the $20\mu s$ integration period of:

$$\Delta V_{\text{net}_j} = \frac{I_{\text{out}_j} t}{C_{\text{int}}} = \pm 147mV \tag{5.3}$$

with the linear ramp used in the characterisation experiments having a slope of $0.1V/\mu s$ this produces a zero input response spread of

$$S_{j_{zero}} = 10 \pm 1.47\mu s \tag{5.4}$$

This figure is too small to account for the spread encountered thus the voltage integrator is not the major contributing factor to the spread.

Distributed Feedback Amplifier

The effect of mismatch in the distributed feedback amplifier is more difficult to calculate. Expansion of the transistor equations while modelling V_{off} as shown in figure 5–14 leads to a complex expression. Instead a SPICE simulation was used to investigate the effects of input offset voltage on V_{out_j} and I_{out_j}. The simulation used extracted circuit values and included the synapse, feedback amplifier and voltage integrator. To investigate the effect that an offset in the distributed feedback amplifier would have on the zero input response, the offset voltage was swept $\pm 20mV$ measuring V_{out_j} and I_{out_j}. Results of this are shown in figure 5–15(a). Performing similar calculations to the previous section:

$$\Delta I_{\text{out}_j} = -118nA \rightarrow +112nA$$

$$\Delta V_{\text{net}_j} = \frac{I_{\text{out}_j} t}{C_{\text{int}}} = -786mV \rightarrow 746mV$$

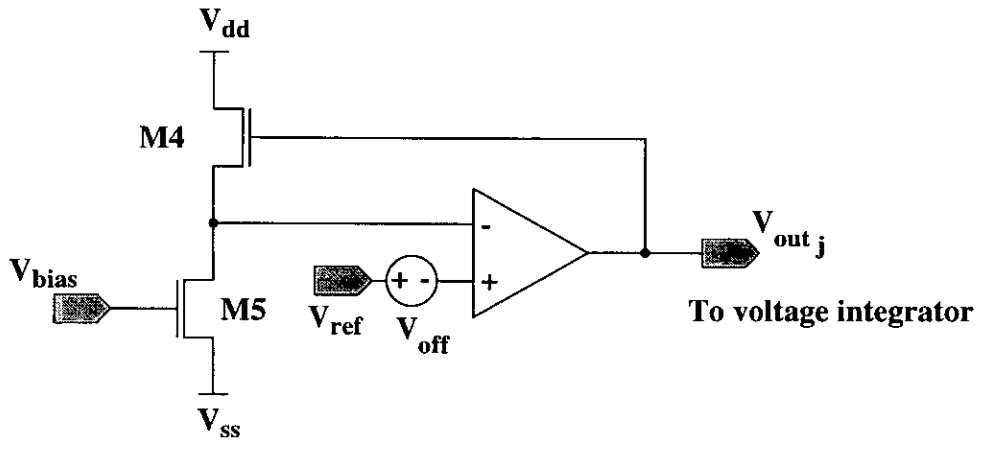


Figure 5–14: EPSILON II Distributed Feedback Amplifier Offset Experiment.

$$S_{j_{zero}} = 2.14\text{–}17.46\mu S \tag{5.5}$$

This spread is slightly larger than the observed spread in zero input response, thus demonstrating a typical input voltage offset in the distributed feedback amplifier is sufficient to produce the observed spread in zero input response values.

To investigate the effect of input offset voltage on the synaptic multiplication characteristic the synapse was included in the simulation and weight voltage on the synapse was swept for $V_{\text{off}} = \pm 20mV$. The results of this simulation are shown in figure 5–15(b). The black horizontal line in the centre of the response corresponds to the zero input response results. That the points at either end of this line, corresponding to $V_{\text{off}} = \pm 20mV$, do not coincide with the zero weight response marked on the characteristics shows a difference between zero input response and zero weight response as observed in the experimental results. The different slopes on the plots also show that input offset voltage has an effect on synaptic multiplier gain.

The feedback bias generation circuitry uses identical circuit blocks of a transconductance pair and distributed feedback amplifier to generate V_{sz} and V_{oz}, it is no surprise therefore that these do not produce appropriate values. To balance the integrator:

$$V_{\text{oz}} = V_{\text{bias}} + V_{\text{ref}} + V_{\text{off}} + \Delta V_t + V(\Delta V_{\text{DS}}) \tag{5.6}$$

where ΔV_t is the difference in threshold voltage between M1 and M2 due to substrate bias and $V(\Delta V_{\text{DS}})$ is a function of unequal drain-source voltages between M1 and M2 and is zero for $V_{\text{off}} = 0$. Thus any input offset will affect both the V_{oz} value required to balance any particular integrator and also the V_{oz} reference produced by the feedback bias generation circuit.

To balance the synaptic transconductance multiplier:

$$V_{\text{sz}} = V_{\text{Tijz}} + V_{\text{ref}} + V_{\text{off}} + \Delta V_t + V(\Delta V_{\text{DS}}) \tag{5.7}$$

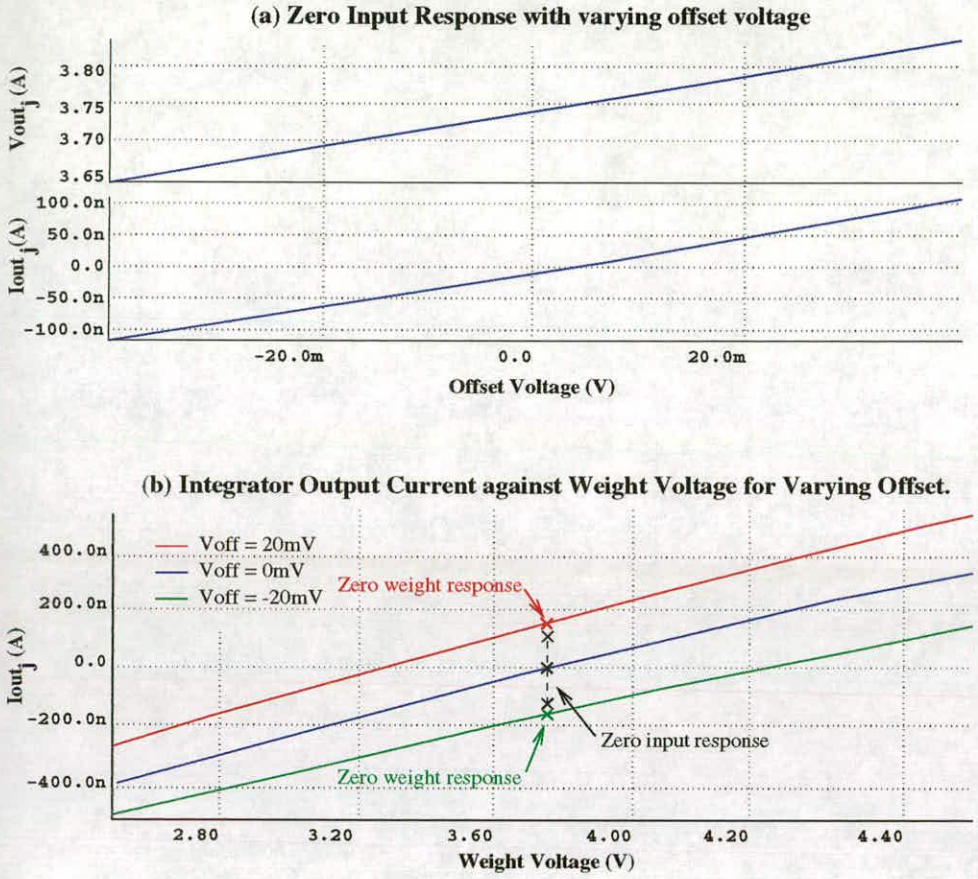


Figure 5–15: Effects of Op-amp offset on EPSILON II Response.

This again shows a dependence on input offset voltage of the DFA. The error introduced by the unequal drain-source voltages also effects the gain of the transconductance pair as seen in the varying slopes of characteristics in figure 5–15(b).

Thus the effects of moderate input offset voltage can account for a portion of all the major types of non-idealities observed in the EPSILON II response; zero input response offset, zero weight response offset, gain variation and bias generation failure.

5.8.2 Removing Input Offset

The previous section showed that input voltage offsets on the distributed feedback amplifier have large effects on EPSILON II response. It is therefore important that a method is developed to minimise these offsets.

With the pulse-width modulation scheme a calculation is performed in three steps:

1. Inputs are applied and synaptic activity integrated.

2. Output ramp is applied and pulse-width is generated.

3. Integration capacitor is reset for next operation.

In steps 2 and 3 the distributed feedback amplifier is unused and an autozeroing operation can be implemented. One such scheme is shown in figure 5–16. Here during ϕ_1 the offset voltage is sampled and stored on C_{az}, this could take place during the integration reset operation.

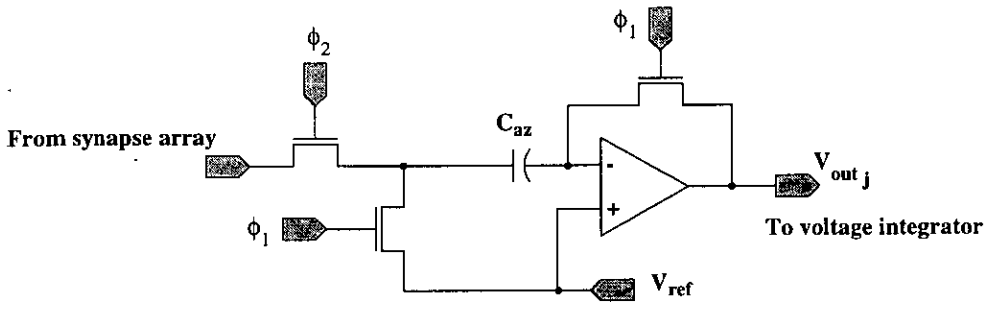


Figure 5–16: Autozeroing Scheme for Distributed Feedback Amplifier.

During ϕ_2 the amplifier is used as normal but the charge stored on C_{az} effectively cancels the input offset voltage. To use this scheme the amplifier must be stable in unity gain negative feedback, simulations on the amplifier used in EPSILON II confirm that it is. Using this type of autozeroing offset can be significantly reduced, it will not however be completely removed as effects such as charge injection from the MOS switches and decay on C_{az} will prevent ideal operation[39].

Implementation of such a scheme holds promise of greatly improving EPSILON II matching and feedback bias generation.

5.9 Summary

This chapter has presented the EPSILON II chip, its design, testing and characterisation. The architecture of the chip was developed to overcome problems encountered in the previous generation EPSILON chip such as power supply distribution problems, excessive signal coupling and poor performance of automatic bias generation. Other changes were incorporated to make the new device more amenable to system level integration for use in applications such as an analog recovery mode, programmable inputs and control rationalisation.

In comparing characterisation results to the previous generation EPSILON chip significant improvements were noted; specifically the synaptic multiplication characteristic is now linear around zero input and noise on output is less due to less coupling of analog references with digital signals.

A significant spread in neuron offset for a zero input response was observed. Autobias synapses were able to remove this offset in most cases. The key contributing factor to this neuron offset was identified as due to input offset voltage in the distributed feedback amplifier. This was also implicated in the observed incorrect operation of reference bias generation circuitry and imperfections in the zero weight characteristic. A solution to minimising this in future chip designs was also presented.

Chapter 6
The EPSILON Processor Card

6.1 Introduction

The previous chapter presented the EPSILON II chip. This device in itself however is insufficient to fill the system specification of Chapter 4; it requires additional circuitry to provide such support operations as weight refresh, ramp and reference generation and I/O management. This calls for a sensible system level framework. In this chapter such a solution is presented, the EPSILON Processor Card (EPC). The EPC is a peripheral device designed to interface to a standard digital bus and provide data channels to the analog world and the pulse stream domain of EPSILON II.

This chapter begins by presenting the necessary system level functions needed to support the EPSILON II chip. Following this, the architecture and design of the EPC is presented leading to a discussion of the use of the EPC.

6.2 System Level Considerations

The EPC operates as a peripheral in a digital environment. It must also fulfil functions of pulse stream communications between EPSILON II chips and control direct analog signal access to the chip. Along with this I/O management role, the EPC must also provide support functions necessary to operation of the chip. A summary of the functions implemented on the EPC along with data and control flow between them is shown in figure 6–1. These functions, which are discussed in this section, are:

1. Weight refresh of on chip dynamic storage.
2. Ramp generation for pulse-width modulating neurons.
3. Pulse conversion of inputs and outputs.
4. Analog reference generation.
5. Analog signal interface to access analog input data.

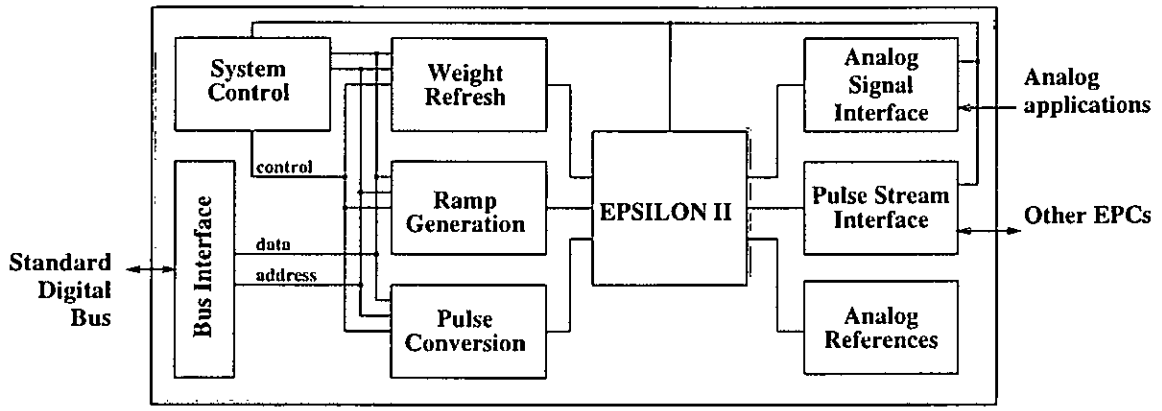


Figure 6–1: Block Diagram of EPC Operations.

6. Pulse stream interface to allow cascading of boards via pulse stream communication between EPSILON II chips.

7. Bus interface to map the EPC onto a standard digital bus system.

8. System control to oversee operation of functional blocks and EPSILON II chip.

6.2.1 Weight Refresh

As EPSILON II uses dynamic weight storage, the weights are stored on the EPC in digital RAM which are used to periodically refresh on chip weights via D/A conversion. There is no provision on EPSILON II for on-chip learning, thus weights must also be accessible to an external processing device for modification. The functions necessary to carry out weight refresh are shown in Figure 6–2. The functional block is under control of the **Refresh Control** block which generates RAM addresses, control signals to the weight RAM and DAC, arbitrates access to the weight RAM for bus requests and generates the timing signals to drive the on-chip refresh circuitry of Section 5.6.5. The logical implementation of this block is included in Section B.3 and figure C–4.

6.2.2 Ramp Generation

The pulse-width modulation (PWM) employed on EPSILON II requires analog ramp signals to operate. These ramps may be linear or some arbitrary function. Figure 6–3 shows a functional diagram to implement such a function. The ramp waveform is stored in RAM; to generate the ramp this data is fed sequentially to the DAC for conversion to an analog waveform. The ramp control block arbitrates bus requests to modify data in the ramp memory and generates the address sequence and control signals to memory and DAC to produce the ramp. In

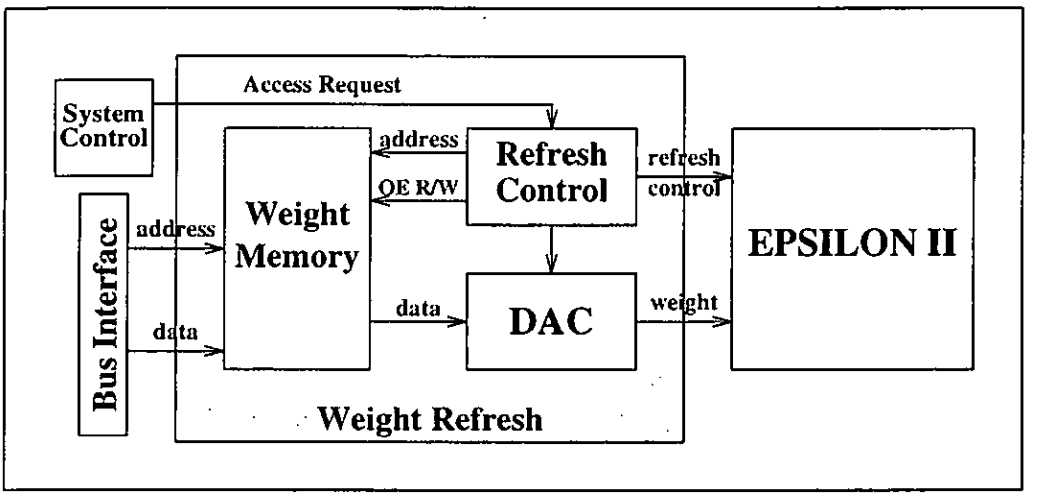


Figure 6–2: Block Diagram of Weight Refresh Operation.

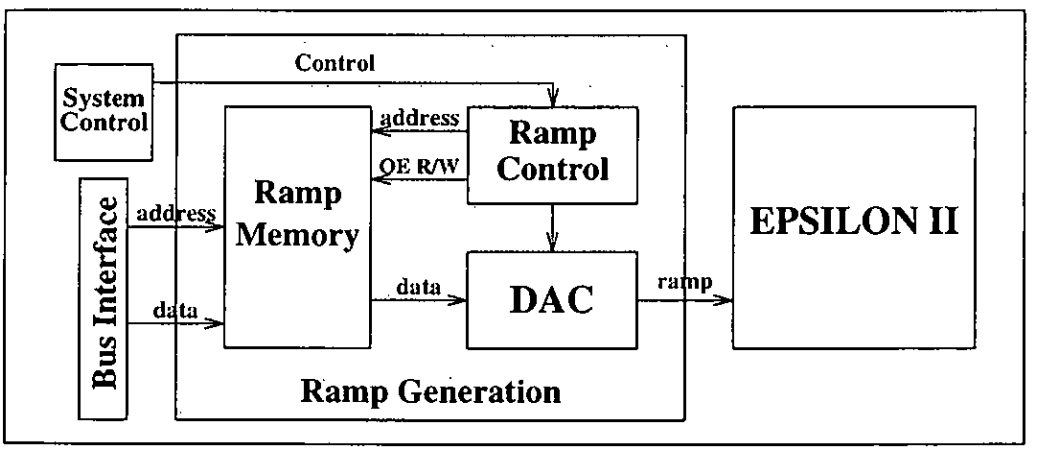


Figure 6–3: Block Diagram of Ramp Generation.

practice, generation logic is part of the pulse conversion as ramp generation occurs along with pulse conversion. The logical implementation of this function is included in Section B.2 and figure C–5.

6.2.3 Pulse Conversion

Data representation at the interface of EPSILON II is in the form of pulse streams. Data representation of digital systems is binary words of set resolution. To provide bus compatible I/O, data conversion between these data representations are necessary.

Binary to pulse-width conversion

To enable data generated by digital systems to be processed by EPSILON II a conversion between binary words and pulse-widths is required. Figure 6–4 shows such a scheme. This scheme is analogous to the pulse-width neuron on EPSILON II: a linear reference ramp (signal

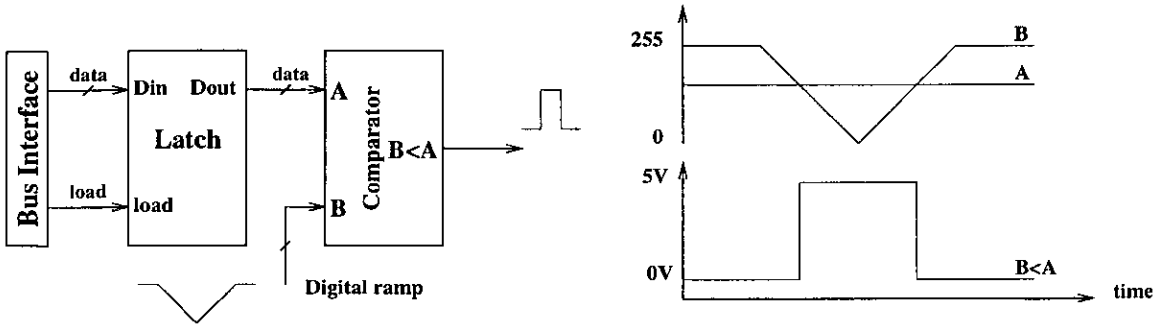


Figure 6–4: Block Diagram Binary to Pulse Width Conversion.

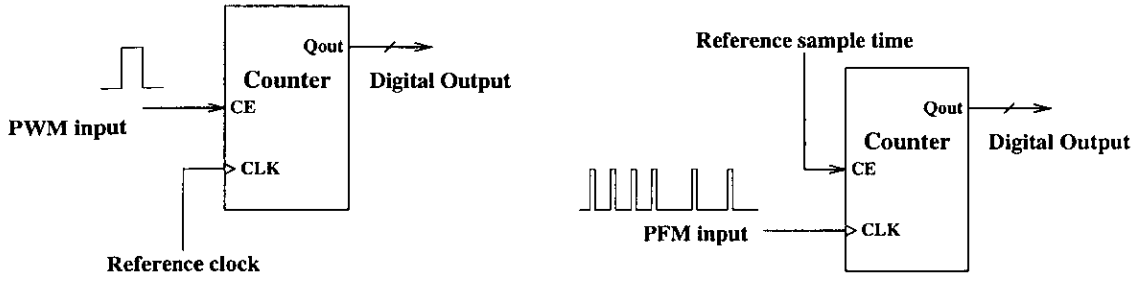
B) is generated and the data (A) is compared to this. The result (B<A) is a pulse-width proportional to the magnitude of A. The same ramp (B) can be fed to a DAC to generate the ramp used in the input neurons that converts analog inputs EPSILON II to pulse-widths. The logical implementation of the binary to pulse-width conversion is included in Section B.2.1.

Pulse-width to binary conversion

To utilise the results from EPSILON II, outputs need to be converted from pulse streams to binary words. Already available from the ramp generation block is a clock signal and the sequence of addresses which scans the ramp memory to produce the output pulse-width neuron's ramp. One scheme to calculate output pulse-width would be to use the rising and falling edges of the pulse-width to latch ramp memory address values. The pulse-width is then calculated by subtracting these two values. The disadvantage of this scheme is that any noise or spikes on the pulse-width signal can severely distort the result. An alternative scheme not as susceptible to noise is shown in figure 6–5-a. Here the pulse-width is used to enable a counter clocked by the reference clock. The counter output is thus proportional to the pulse-width. For the EPC a word length of 8 bits was chosen as the resolution. Thus for a $20\mu s$ ramp a reference clock frequency of 12MHz is used. Logical implementation of the pulse-width to binary conversion is included in Section B.2.2.

Pulse frequency to binary conversion

If EPSILON II is operated in pulse-frequency output mode a different conversion is needed. One such scheme is shown in figure 6–5-b. Here a counter is used to count the incoming pulses



(a) Pulse Width to Binary Conversion **(b) Pulse Frequency to Binary Conversion**

Figure 6–5: Block Diagram Pulse Stream to Binary Conversion.

for a sample time t_{ref} to produce a digital output. The pulse-frequency output of EPSILON II is a pulse stream of fixed high time ($1\mu s$) with a duty cycle varying from zero to 50%. Thus to convert to a resolution of 8 bits (256 levels):

$$\begin{aligned} t_{\text{ref}} &= \text{Max. number steps} \times \text{period of max. output} \\ &= 256 \times \text{high time} \times \frac{100}{\text{max. duty cycle}} \\ &= 256 \times 2\mu s = 512\mu s \end{aligned} \tag{6.1}$$

6.2.4 Analog References

For the purposes of experimentation, most references on EPSILON II were pinned directly off-chip rather than tied to a global reference on-chip. Thus a total of 5 current references and 4 voltage references must be supplied to the chip. Also the failure of the feedback bias generation circuitry for V_{oz} and V_{sz} implies these voltages must be supplied as well. Along with these references the non-standard synapse power supplies of 0.5V and 1.5V are produced locally. The details of producing and setting up these references and supplies are given in figure C–6 and Section C.4.

6.2.5 Analog Signal Interface

The number of analog neural inputs to the EPC is variable and application dependent. Analog input to EPSILON II is via the same 32 inputs as pulse modulated input. On the EPC, these 32 inputs are also commoned with the 32 outputs (which can take a high impedance state) to form a *pulse bus*. The analog input to the EPC must be able to be isolated from this bus. Thus analog inputs enter the bus through analog switches. These are configurable by a software controlled *input mode mask* which maps which inputs are to be analog (mode 1) and which pulse modulated (mode 0). Figure 6–6 shows this arrangement. The **Mask register** stores the configuration of the inputs. To initialise the EPSILON II chip, **load_mode** is brought high

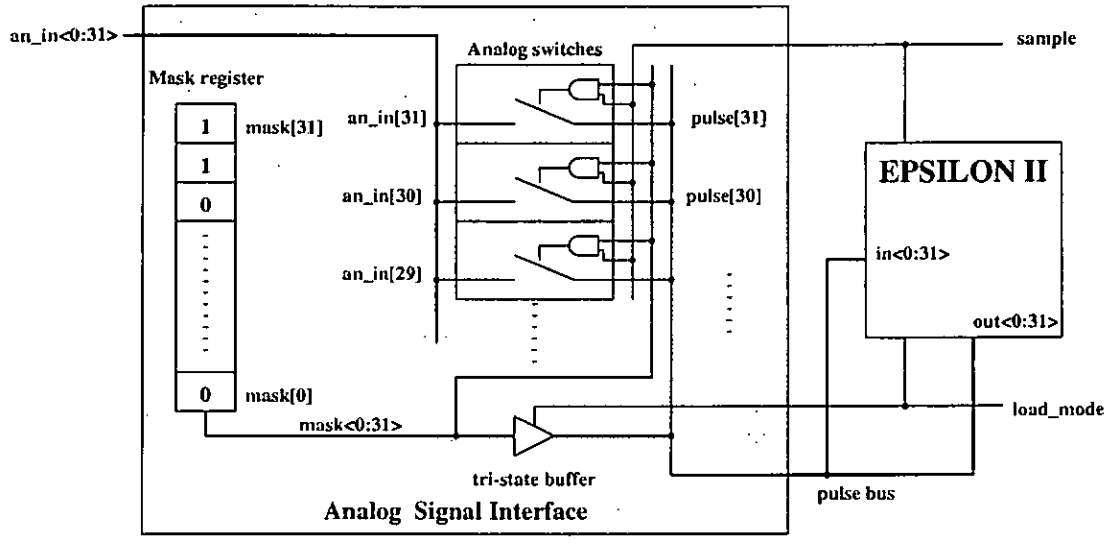


Figure 6–6: Block Diagram of Analog Signal Interface.

which loads the configuration into the the latches of the EPSILON II input neurons. Analog input data is sampled when **sample** is high and held when low. In practice, inputs on the EPC are configured as groups of four to reduce the chip count on the board.

6.2.6 Pulse Stream Interface

The pulse stream interface addresses the system specification of cascadability. The pulse stream interface allows the internal **pulse bus** of the EPC to be extended to other EPCs. To do this a dedicated bus for neural pulse streams was designed with control signals to allow communication and synchronisation between boards. The scheme is summarised in figure 6–7. The **System Control** block configures the board to accept inputs from the neural bus, or place outputs on the bus, by controlling the bi-directional tri-state buffer. A system controller placing data on the bus indicates it is doing so via a control signal. A controller waiting for inputs uses this to start an EPSILON II processing cycle.

6.2.7 Bus Interface

The system specification calls for an interface to a standard digital bus. Some suitable bus standards include the PC-bus, VME-bus and STE bus. As one target application, specifically the instinct-rule robot, requires a stand-alone processor the STE bus was chosen for the following reasons:

- A wide range of inexpensive processor cards and other peripheral cards were available.

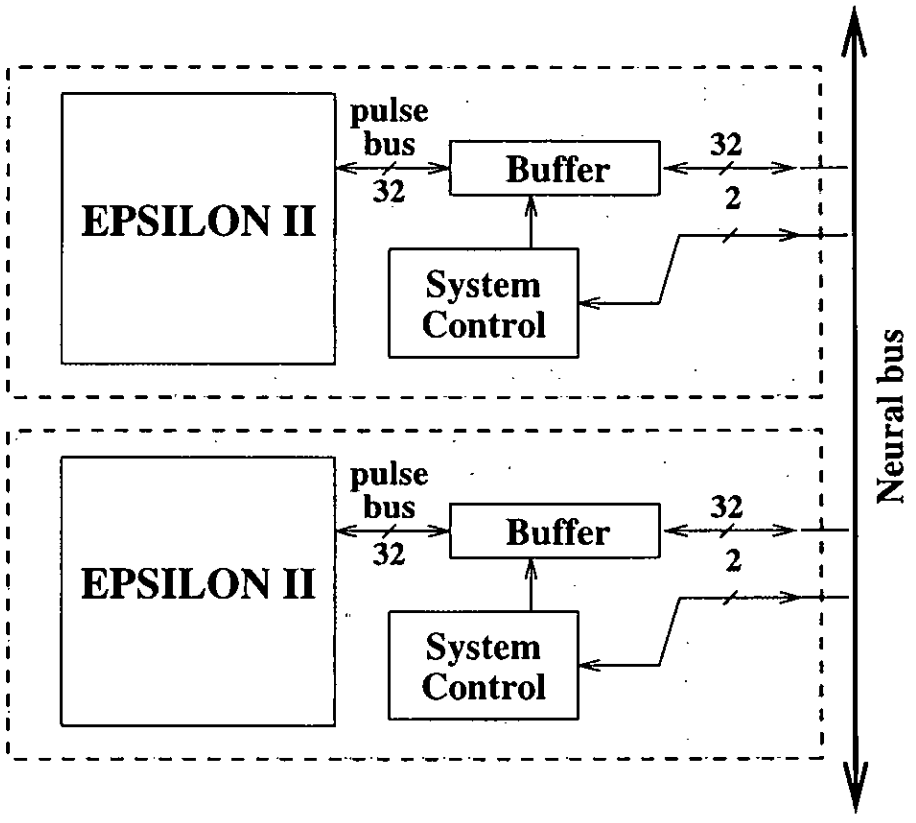


Figure 6–7: Block Diagram of Neural Bus Scheme.

- Compact size of *Eurocard* standard size boards was ideal for the robot application.
- Asynchronous 8 bit data bus was sufficient for the essentially 8 bit resolution data of EPSILON II.

In order to interface to this bus the EPC must:

- Decode addresses.
- Interpret STE control signals such as address & data strobes and read/write control lines.
- Provide a *data acknowledge* (DTACK) signal to indicate completion of a bus cycle.

Schematics of hardware to do this are included in Section B.4 and figure C–11. For more details on the STE bus architecture see Mitchell [81].

6.2.8 System Control

The **System Control** block addresses the system specification of autonomy. It holds overall control of the previously described functional blocks as well as providing the control signals for the EPSILON II chip. It synchronises and sequences all the operations carried out on the EPC and will be discussed further in Section 6.5.3.

6.3 EPC Architecture

In the previous section the functional blocks of the EPC were presented. What remains to be discussed is how these blocks were implemented in hardware. To gain an insight into the design decisions let us return to the FENICS system designed to support EPSILON. This system employed a combination of a micro-controller and standard 74 Series logic to perform similar functions to the EPC, however it suffered from several major drawbacks. Firstly there were major data bottle-necks in the system due principally to 30 bit data (the output dimension of EPSILON) being processed over 8 bit buses. Transferring data over lower dimension busses adds an extra overhead that on the EPC is minimised by transferring data over the internal 32 bit **pulse bus**.

Pulse conversions on FENICS were intended to be carried out by the micro-controller. This serial processing device, with a relatively slow clock speed, proved too slow and conversion was found to be quicker by up-loading raw sampled pulses to a PC for processing. To alleviate this type of problem, the hardware for processing pulse conversions was designed. However if this was to be implemented with standard logic packages, the chip count on the board would be enormous. Along with this, other functions such as weight refresh and ramp generation, which were also performed on FENICS required several chips to realise.

To prevent an unacceptable chip count a FPGA[1] was used to implement much of the digital logic and processing. Figure 6–8 shows the distribution of functions on the EPC highlighting those implemented on the FPGA. The FPGA chosen was a re-programmable device to allow the EPC to be customised to various applications. For instance, if pulse conversion is required for all inputs and outputs, then the size of the FPGA is insufficient to implement all pulse conversion circuitry in parallel. In this case pulses are stored or sampled in a a block of RAM 32 bits wide (the pulse RAM of figure 6–8) and processed after an EPSILON II cycle has taken place, in the case of outputs, or before and loaded into RAM in the case of inputs. This is the most generalised case of the EPC and is what is presented in Appendix B. For an application such as the instinct-rule robot, not all output neurons are used: in this case only four – thus pulse conversion can be implemented in parallel on the FPGA. Similar optimisations

[1] Field Programmable Gate Array

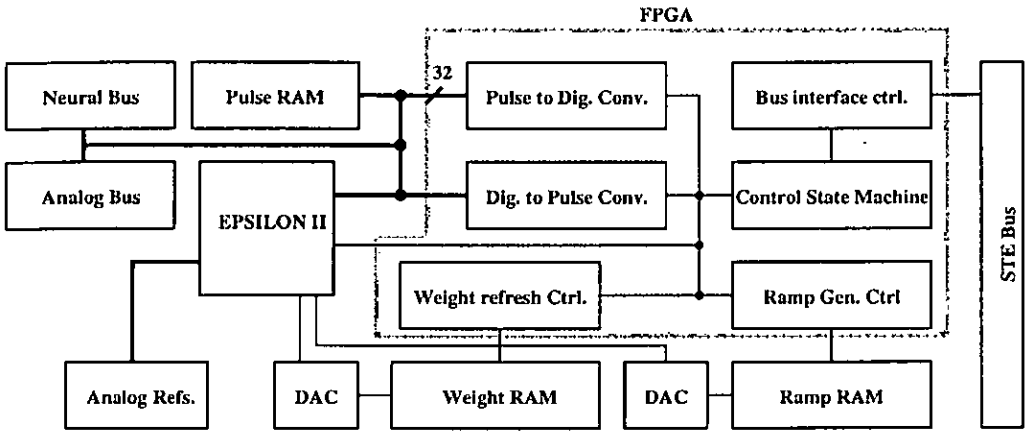
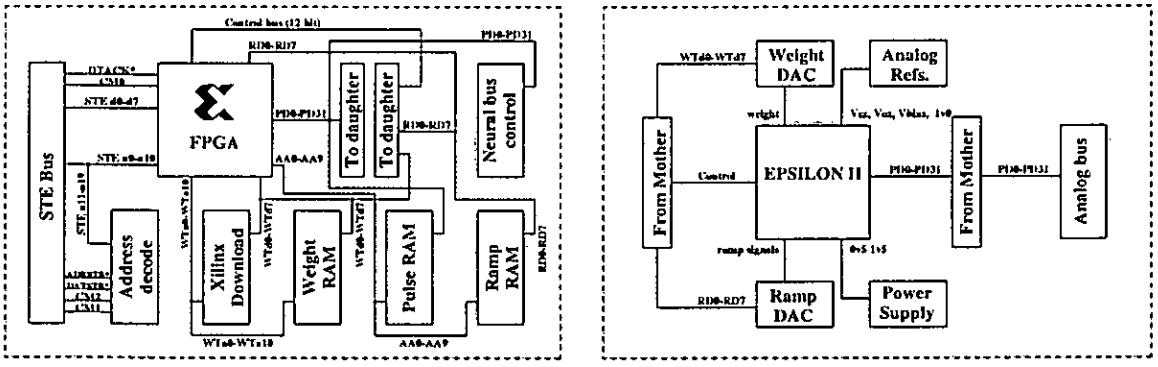


Figure 6–8: EPC Block Diagram.

can be implemented if many inputs are analog, reducing the demand on binary-to-pulse-width conversion, and allowing processing to be done in a fast, parallel manner if possible.

6.4 EPC Design

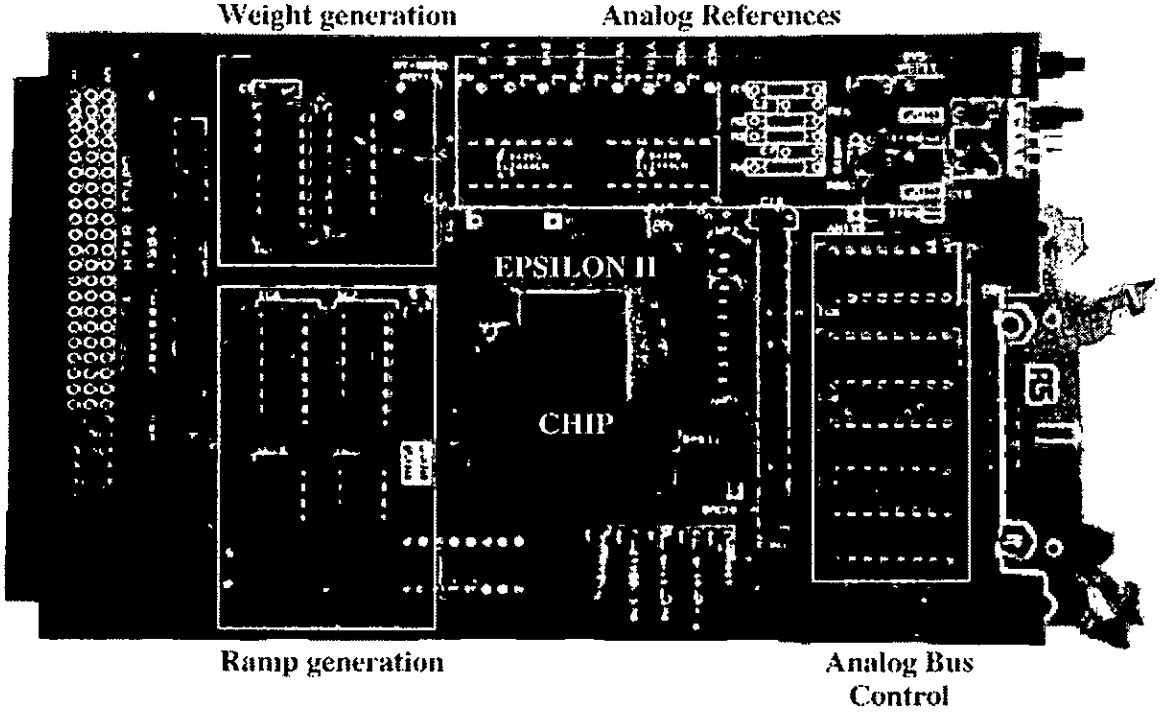
The EPC was designed to conform to the STE bus standard and was laid out on *Eurocard* size boards. To achieve this and to provide good noise isolation between analog and digital functions a twin board approach was taken. A mother-board carrying out digital functions and containing the FPGA is mated to daughter-board via an 80-way bus (see figure 6–9–a). The daughter-board contains the EPSILON II device and all D/A circuitry, analog references and



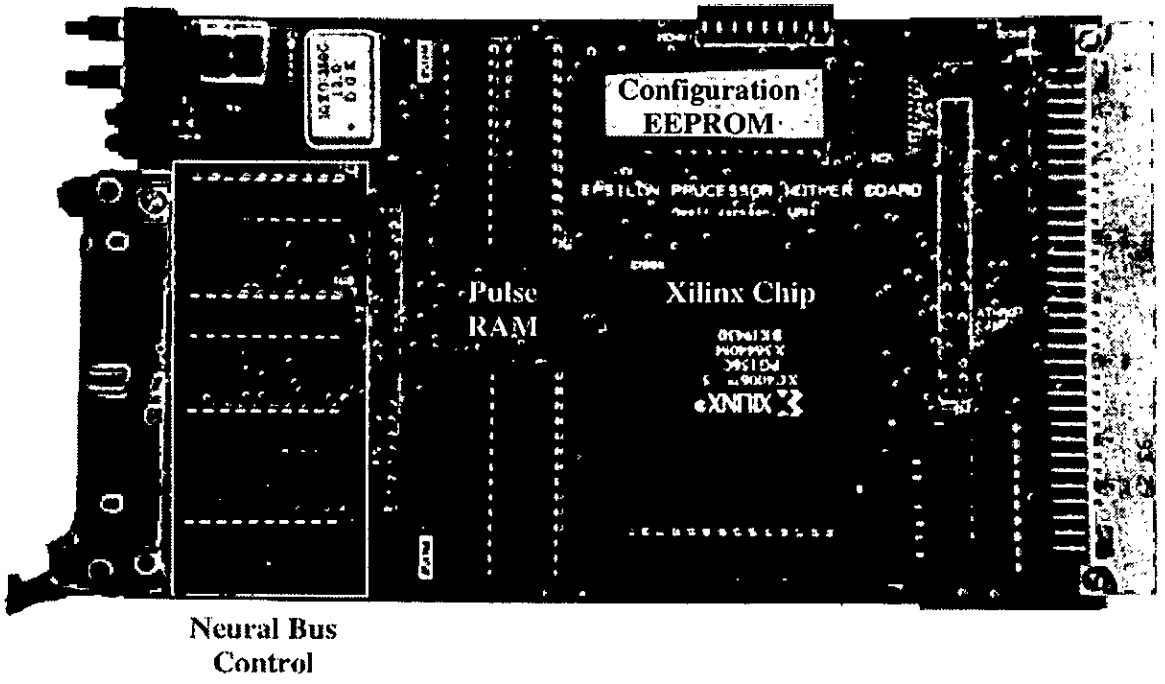
a) EPC Mother-Board. b) EPC Daughter-Board.

Figure 6–9: EPSILON Processor Card Boards.

power supplies (see figure 6–9–b). A photograph of the EPC is shown in figure 6–10 with the key components highlighted.



EPC Daughter Board



EPC Mother Board

Figure 6–10: Photograph of the EPC.

Details of the board layout and the set-up procedure for the board can be found in Appendix C.

6.5 Using the EPC

This section describes the practicalities of using the EPC as part of a system. Firstly the issues of data representation are discussed and a set of state variables is defined for the different data representations used in the EPC. Next the I/O and control of the EPC is presented from the point of view of the host system. Finally with this information available the system control block of the EPC is explained.

6.5.1 Data Representation

Data communication between the EPC and its host is via the 8 bit STE bus. Data communication to and from the EPSILON II chip is in the form of analog voltages or pulse modulated waveforms. The EPC carries out data conversions between these two domains. To avoid confusion, table 6–1 defines state variables for the various data types and summarises the conversion undertaken on the EPC. These variables are used throughout the remainder of the thesis such

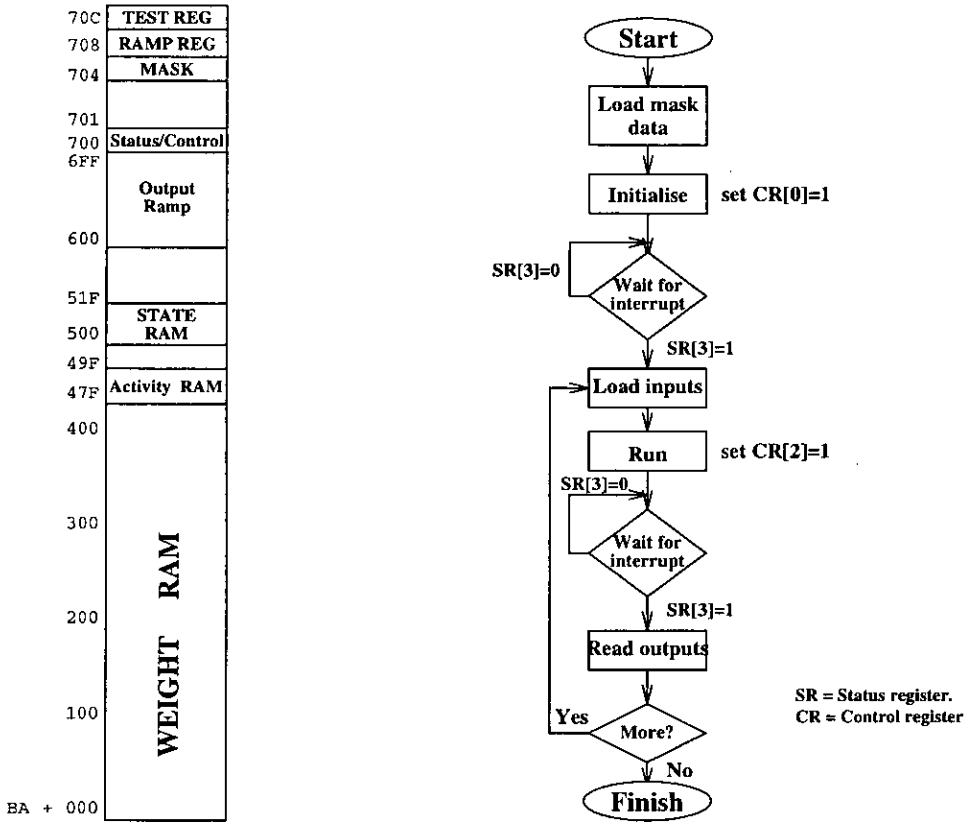
Table 6–1. Data Representations in EPC and EPSILON II

	EPC	EPSILON II	Conversion
inputs	$X_i \in [0, 0\text{xFF}]$	$S_i \in [0, 20\mu s]$ pulse-width	$X_i =$ 8-bit PW conversion
outputs	$Y_j \in [0, 0\text{xFF}]$	$S_j \in [0, 20\mu s]$ pulse-width	$Y_j =$ 8-bit PW conversion
weights	$W_{ij} \in [0, 0\text{xFF}]$	$T_{ij} \in [2.5, 5V]$ voltage	$T_{ij} =$ 8-bit A/D conversion

that the data type is implicit in the name. The state variables for EPSILON II (S_i, S_j & T_{ij}) have already been used in Chapter 5. These are mapped by the EPC into 8 bit integers (X_i, X_j & W_{ij}) for use by the host system.

6.5.2 Memory Mapped I/O

The I/O of the EPC is memory mapped onto the STE bus. The location of the EPC in memory is determined by setting jumpers on the board to define the base address of the device. The memory map of the EPC relative to this base address is shown in figure 6–11–a. The principal



a) Memory map of EPC. **b) Flow chart for EPC operation.**

Figure 6–11: EPC Memory Map and Operational Flow.

memory blocks are:

- Weight memory – write only.
- Neural state memory – State inputs (X_i) are written to this block and state outputs (Y_j) are read upon completion of processing.
- Control register (on write) and status register (on read).
- Mask register - 4 bytes which hold the mask that determines whether an input is analog or pulse modulated.
- Ramp memory – 256 values that define the shape of the pulse-width output neuron's ramp.

Control of the EPC is via the control register which determines what function the EPC is to perform. The status register is read to determine the current status of the device. The functions of individual bits are given in figure 6–12. Bits 0–4 of the control register are common to all

BIT	Status Register	Control Register
0	Init. in process	Initialise
1	Refresh Paused	Recover
2	Reset Button	Run
3	Interrupt	Acknowledge
4	Inputs changed	Refresh Pause
5	BIN-TO-PW	Motor 1
6	Running Net	Motor 2
7	PW-TO-BIN	Motor 3

Figure 6–12: EPC Status and Control Registers.

FPGA customisations while bits 5–7 are user definable for particular applications; in this case motor control signal for the mobile robot presented in Chapter 8. Setting one of bits 0–2 sets in motion the corresponding EPC function. When this is complete an interrupt is flagged either on bit 3 of the status register or on a user selectable interrupt line of the STE bus. This interrupt is acknowledged by setting bit 3 of the control register.

A flow chart example of running the EPC is shown in figure 6–11–b. These are the operations that a host performs in using the EPC, namely:

- Load the mask register that configures inputs as analog or pulse stream.
- **Initialise** (CR[0]) is then set to perform an initialisation function that downloads this mask to the EPSILON II chip.
- When this function is complete the **interrupt** bit (SR[3]) goes high. This is acknowledged by the host by setting the **acknowledge** bit (CR[3]) high. Further functions can be carried out once **interrupt** goes low.
- To start a network run inputs are loaded to the board then **run** (CR[2]) is set high to begin cycle. Cycle is finished by the same interrupt and acknowledge procedure as above.
- Outputs are now available and can be read from the state memory.

6.5.3 System Control

The core digital processing of the EPC is accomplished in the Xilinx FPGA. The *firmware* code to initialise this device is loaded at power-up from an on board EEPROM or during development via a special Xilinx serial cable. The FPGA design performs the functions outlined in Section 6.2 such as weight refresh, pulse conversion, STE interface, bus control and system control. The system control consists of a master state machine which sequences and provides control signals to the functional blocks within the EPC. An example of this state machine (as it can be customised for various applications) is shown in figure 6–13, while logical implementation can be found in Section B.5. Each state of the state machine triggers an EPC function:

WAIT	Idle state – waits for change in control register.
INITIALISE	Initialises the EPSILON II chip by down-loading the mask register to the EPSILON II chip input neurons.
B-TO-PW	Binary-to-pulse-width conversion – Triggers the binary-to-pulse-width conversion function and waits for the BTPfin signal from that block.
SAMPLE	Sample and hold analog inputs.
FIRE INPUT	Apply input ramp and any pulse modulated inputs to EPSILON II.
FIRE OUTPUT	Apply output ramp to EPSILON II and capture output pulses.
PW-TO-BIN	Convert pulse-width modulated outputs to binary numbers. These are stored in the state RAM for the host to read.
INTERRUPT	Generate an interrupt to signal processing finished. This interrupt is cleared by host via setting the **acknowledge** bit of the control register (CR[3]) high.

This approach of allowing customisation of internal logic gives the EPC great flexibility. For instance in Chapter 7 two EPCs are cascaded. In this case the system control state machine is modified to start a cycle on receipt of control signals over the neural bus.

6.6 Summary

In this chapter the *EPSILON Processor Card* was presented. It constitutes a platform whereby access to the EPSILON II chip is made transparent to a host device. All support functions necessary to the operation of the chip are carried out independently of the host and the use of FPGA technology allows customisation of internal functions for different applications. The EPC communicates with external data via three channels:

1. Its host via a STE bus.

2. Analog data via an analog input bus.

3. Other EPCs via a neural pulse stream bus.

This structure means the EPC can communicate effectively in an applications environment rather than decreasing the host system performance by transferring large quantities of unprocessed data across the system bus.

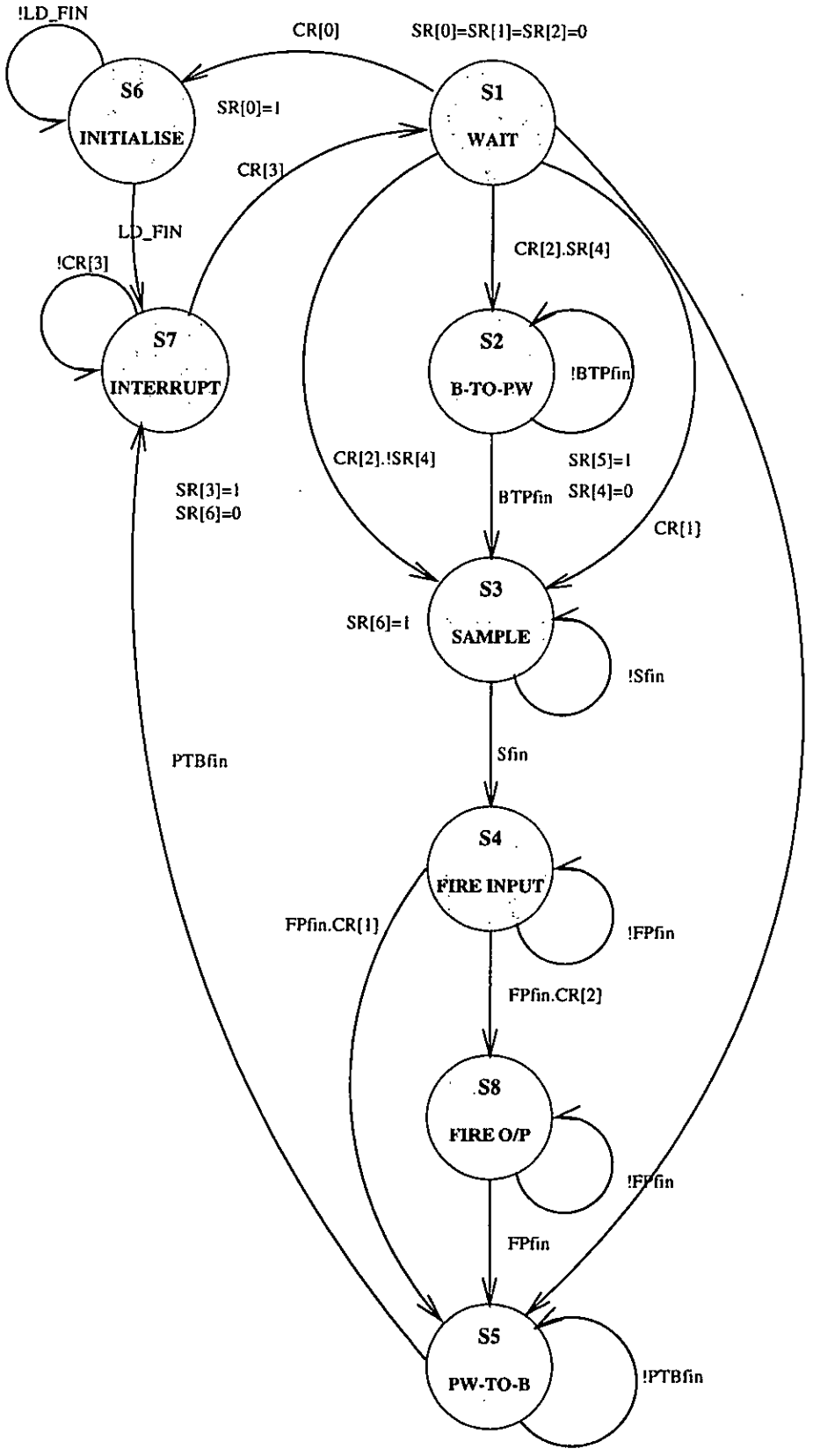


Figure 6–13: EPC Control State Machine.

Part III

Applications Development

Chapter 7

Hardware Issues in Back-Propagation Training

7.1 Introduction

To use the EPC in applications a network structure must be imposed on the hardware and a training algorithm used to develop a weight set for a solution. In this, the first of the applications development chapters, the use of the EPC as a multi-layer perceptron (MLP) is explored. The MLP is the most widely used of network structures; the most common training algorithm used to train MLPs is the back-propagation training algorithm. The non-idealities present in analog hardware raises issues concerning the performance of the network as well as the ability to effectively train it. In this chapter the back-propagation algorithm is used to train the hardware forward pass network and the empirical effects of hardware non-idealities associated with this are investigated. This investigation explores the practicalities and limitations of using the EPC as a hardware neural network.

7.2 Experimental Approach

The emphasis of this chapter is on the practicalities of training the EPC chip-in-loop rather than a theoretical study of precision and accuracy issues. For this reason, rather than implementing simulations to model the effects of individual hardware non-idealities, the approach is from a system level. To perform comparisons, each application of the EPC is compared against a floating point software model. For most experiments a second software model is used to model the effects of limited precision and dynamic range in the weight set. The results from these simulations are compared to actual hardware results to gauge the limitations of practical hardware.

In the first section of this chapter the hardware configuration is described and a software model is developed to form a baseline for comparison – this entails determining the data transformation necessary for communication and matching between hardware and software. Following this techniques of maximising dynamic range and accuracy are outlined. The

hardware non-idealities that effect network performance are introduced and the results of work by others investigating these effects are reviewed.

The remainder of the chapter presents the experimental work to investigate the practicalities of using the EPC. The first problem used is an artificial character recognition task. While not truly representative of the problems likely to be found in the defined applications area of the EPC, it allows an approach of graded complexity to probe hardware performance. This study reveals practical aspects of hardware use particularly in relation to weight dynamic range and choice of learning parameters.

The lessons learnt in this investigation are next applied to more difficult real-world problems:

- Link admission control in an ATM (Asynchronous Transfer Mode) network.
- A speaker identification problem.
- A medical data classification problem.
- A region classification problem.

The ATM problem investigates the EPC as a function approximator while the others involve 1-of-N classification problems. These problems are more representative of the types of tasks found in the real-world. They have been chosen to provide a spread of difficulties from the easier speaker identification problem to the difficult ATM and medical data problems, to the very difficult region classification problem. The three 1-of-N classification problems were also used by Cairns of Oxford University[18] in a study on the effects of analog precision in learning, with the speaker identification problem being implemented on a pulse stream hardware network. This allows comparisons to be made between the work of this thesis and that of Cairns.

7.3 The EPC as a Multi-layer Perceptron

Training a multi-layer perceptron (MLP) with back-propagation consists of two distinct operations: firstly a forward pass which propagates network inputs through one or more hidden layers to the output layer. This operation is implemented on the EPC hardware. The second operation occurs only during the training phase and compares the output results to target values producing an error term which is propagated to previous layers to determine weight updates. This error calculation and back-propagation is done in software forming a chip-in-loop system.

Data representation in hardware is implicit in various physical quantities such as voltages and currents. These are all bounded quantities and subject to quantization when generated by digital means via the EPC. The data in software can be represented as a high precision 32 bit floating point number. As such various transformations and models are needed to interchange

data and simulate the hardware network. The remainder of this section describes the hardware system and the transformations of data between the hardware and software domain.

7.3.1 Hardware Configuration

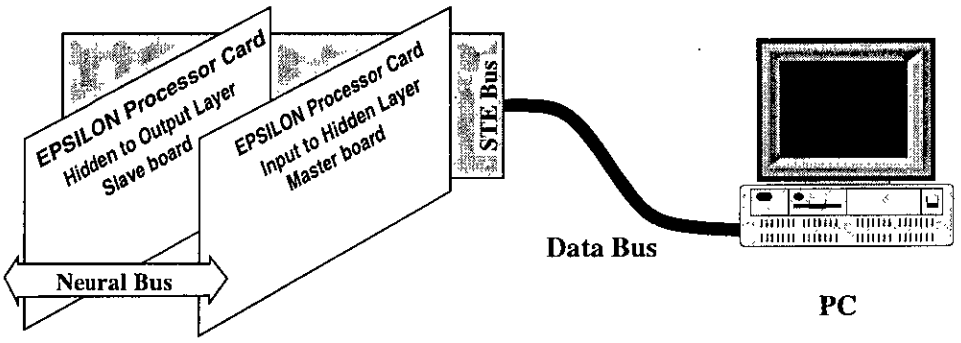


Figure 7–1: Two EPC System for the Character Recognition Problem

To configure the EPC as a multi-layer perceptron (MLP) a system consisting of two EPCs with a common digital bus and neural pulse bus is used as shown in figure 7–1. One EPC is configured as a *master* board and processes the input-to-hidden layer of the network while the other is configured as a *slave* and processes the *master's* outputs as inputs, thus implementing the hidden-to-output layer. The EPCs are under the control of the PC via a 40-way I/O card mimicking the STE bus.

7.3.2 Data Transformation

Table 7–1. Data Representations in EPC and PC

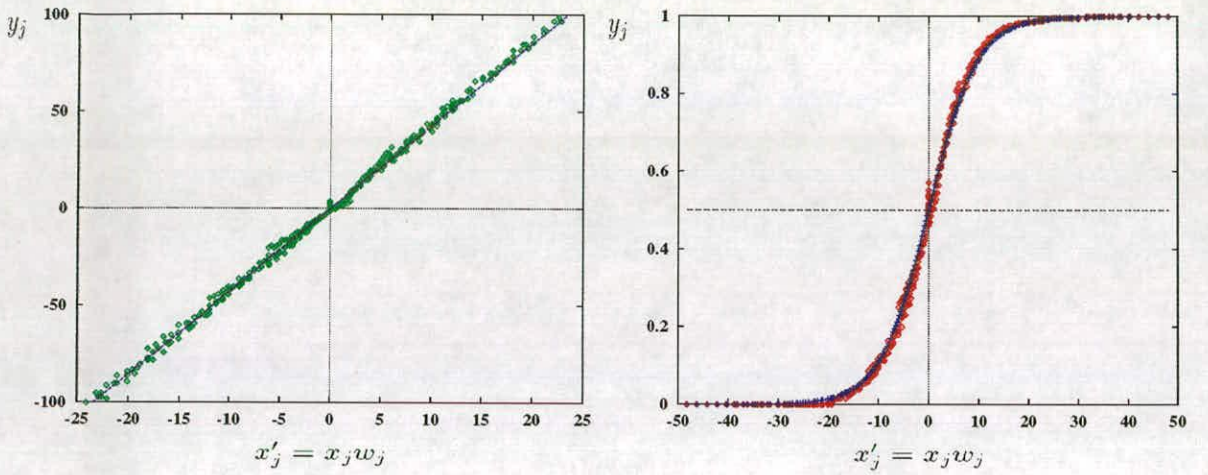
	PC	EPC	Conversion
inputs	$x_i \in [0, 1]$	$X_i \in [0, 0\text{xFF}]$	$X_i = (\text{int})(x_i * 0\text{xFF})$
outputs	$y_j \in (0, 1)$	$Y_j \in [0, 0\text{xFF}]$	$y_j = (Y_j + 1)/0\text{x}101$
weights	$w_{ij} \in$ float	$W_{ij} \in [0, 0\text{xFF}]$	$W_{ij} = (\text{int})(w_{ij} + 128)$

In order to train and simulate the neural network the PC needs an internal data representation. For this reason data is transformed into floating point numbers for use in the software. These transformations are shown in table 7–1 and allow the back-propagation learning algorithm and network simulation to operate on high precision floating point data. These data transformations follow on from those presented in table 6–1 which showed the transformation between

EPSILON II and the EPC. Note that for an ideal sigmoidal activation function the output (y_j) never reaches 1 or 0, rather it approaches these as network activity approaches plus or minus infinity. To approximate this behaviour the transformation for y_j is such that

$$y_{j_{max}} < 1 \text{ and } y_{j_{min}} > 0 \tag{7.1}$$

This is important for back-propagation learning as weight change is proportional to y'_j; if $y_j = 0$ or 1 $y'_j = 0$ which prevents any weight change.



a) Linear multiplier constant by least squares fit. **b) Comparison of sigmoid output for hardware and software.**

Figure 7–2: Synaptic Multiplier and Sigmoid Characteristic

To match the PC simulation of the EPSILON II chip, PC software characterises each EPC to determine a multiplier constant. Consider the transfer function of a column of synapses:

$$Y_j = f\left(\sum_i X_i W_{ij}\right) \tag{7.2}$$

This is modelled in software by:

$$y_j = f\left(\sum_i k x_i w_{ij}\right) \tag{7.3}$$

where k is a multiplicative constant. In actual fact this is only a first order approximation of the multiplier characteristic as the synaptic multipliers are not perfectly linear or matched. To calculate this factor $f(x)$ is set to a linear function and all weights and inputs are set to the same value, that is:

$$\begin{aligned} y_j &= kNx_j w_j \\ &= kNx'_j \end{aligned} \tag{7.4}$$

where N is number of neurons, and $x_i = x_j$, $w_{ij} = w_j \; \forall \; i$

By sweeping x_j and w_j through n different values (x'_j, y_j) pairs are obtained as plotted in figure 7–2–a. By using a linear least squares best fit[64] the "best" k for the neurons on a chip can be calculated:

$$k = \frac{1}{N} \frac{n \sum_j x'_j y_j - \sum_j y_j}{n \sum_j (x'_j)^2 - \sum_j x'_j} \tag{7.5}$$

The line corresponding to this calculated k is also plotted in figure 7–2–a.

Back-propagation training generally uses a sigmoidal transfer function of the form:

$$f(x) = \frac{1}{1 + e^{-\lambda x}} \tag{7.6}$$

where λ is the sigmoid gain. This function is implemented by loading appropriate values into the EPC ramp generator. Figure 7–2–b demonstrates the match achieved when the software model (solid line) is adjusted using the calculated multiplier constant.

7.3.3 Maximising Dynamic Range

Weights in hardware are restricted in the range and value they may take. For the EPC the dynamic range is limited by the physical [2.5,5V] range synapse circuitry allows while the precision is limited by the 8 bit resolution of refresh circuitry[1].

To maximise the dynamic range of the network implemented all available inputs and neurons should be utilised. For instance a single weight on the EPC can take 256 independent values, i.e. is of 8 bit precision. If there are *spare* inputs to a layer then they can be used to increase the available dynamic range. If two inputs are connected in parallel to two synapses then the dynamic range of that weight can be doubled, giving an effective weight of 512 independent values. If the number of inputs, n, to a layer is less than half the number of available inputs ($N = 32$), this is done for all inputs.

A bias unit is a network whose input remains at a fixed positive value. Any remaining inputs are designated bias units thus utilising the full available dynamic range of the hardware. The software controlling the EPC does this automatically given the network dimensions.

7.3.4 Accuracy Maximisation

In characterising the EPSILON II device in Chapter 5 some aspects of device accuracy were discussed, in particular the technique of autobiasing was presented to improve the zero offset of

[1]The ultimate limit to precision on EPSILON II is limited by capacitive decay and refresh rate coupled with noise level.

neurons. This technique improves the matching of the software model and the actual hardware; thus a weight set evolved on the software model is closer to the desired solution than would otherwise be the case without autobias. Autobias also alleviates an area of variability between chips; allowing a solution evolved for one chip to be close to the desired solution in another[44]. For these reasons autobias is performed in the experiments presented here to maximise accuracy and repeatability.

7.3.5 Summary

In this section:

- A first order software model of the hardware has been developed.
- A consistent set of data transformations was developed to allow communication and comparison between software and hardware.
- Matching of software model and hardware was demonstrated for linear and sigmoidal transfer functions.
- A technique for utilising the full available dynamic range of the EPSILON II chip was presented.
- The use of autobias was presented as a means to gain a more accurate matching between hardware and software solutions.

7.4 Back-propagation and Hardware Non-Idealities

The learning algorithm used in these experiments is the back-propagation or generalised delta rule algorithm[100,10]. Other training techniques such as weight perturbation[57], node perturbation[18] and stochastic error descent[5] have been developed more specifically for hardware use and have been shown to work well with chip-in-loop learning[18]. However all these methods require training times of at least an order of magnitude longer then back-propagation to produce only slightly better solutions[18]. In practice back-propagation is much faster for chip-in-loop training and produces comparable results. For these reasons it is used here.

Back-propagation seeks to minimise the network error by comparing actual outputs to given target values. The mean square error of a network is given by:

$$E = \frac{1}{2}\sum_j (y_j - t_j)^2 \tag{7.7}$$

where y_j is the j^{th} component of the network output and t_j similarly the target. y_j is a function of the inputs, x_i and the connection weights, w_{ij}. What back-propagation does is to attempt to decrease this error by adapting weights in proportion to the gradient of the error in weight space:

$$\Delta w_{ij} \propto -\frac{\partial E}{\partial w_{ij}} \tag{7.8}$$

$$= \eta x_i \delta_j \tag{7.9}$$

where η is the learning rate, a constant and δ_j is given by:

$$\delta_j = \begin{cases} (t_j - y_j) f'(\text{net}_j) & \text{if layer is an output layer.} \\ f'(\text{net}_j) \sum_k \delta_k w_{jk} & \text{if layer is a hidden layer.} \end{cases} \tag{7.10}$$

where δ_k and w_{jk} refer to the layer following the layer being calculated. For a derivation of these equations see [100].

7.4.1 Assumptions Concerning Hardware Non-Idealities

In this chapter back-propagation is applied to a hardware network. In doing this several assumptions and approximations are made. First of all note that equation 7.10 requires the values of the weights, w_{ij}. When weights are downloaded to hardware the exact value of the weight as represented in the hardware is not known, either through noise being present or component variation producing an offset or scaling in the weight. In this series of experiments it will be assumed that value of the weight is that stored in software.

Another area of uncertainty can be the neuron transfer function $f(x)$ and its derivative $f'(x)$. For the pulse-width modulating neuron of EPSILON II, it has been shown that the neuron function is well defined (Section 5.7.2). That is the non-linear function (such as a sigmoid) closely matches the desired function as it is determined by the reference ramp waveform rather than physical device characteristics.

Another assumption in the derivation of the back-propagation algorithm is that the error space is a continuous function. Of course when using hardware this is not the case as all values have a limited dynamic range and resolution as summarised in table 7–1. To limit the effects of quantization on the learning algorithm weights are manipulated in software at high resolution and only quantised as they are downloaded to the hardware.

7.4.2 Related Work

This problem of hardware uncertainty and limited precision has been studied for other analog hardware implementations [18,17,30,35,36,83,117] and by simulation for limited precision

digital networks [19,49,55,87]. Results from these studies have influenced the decisions made here. Frye *et al* [35,36], Hollis *et al* [52] and Holt *et al* [55] have all presented studies showing that successful back-propagation requires resolution in the range of 12–16 bits. However it has also been shown that with lower precision (6–8 bits) in the feed-forward component, coupled with high precision for the back-propagation of errors, effective training can be achieved [18, 36,44,55]. These results led to the chip-in-loop system used here of the EPC performing the forward pass at hardware resolution and the host performing error back-propagation at floating point precision. The studies above are further referenced when the results presented here relate to their results.

7.4.3 Summary

This section has introduced hardware non-idealities which may, or are known to, effect back-propagation training: limited dynamic range, mismatches, offsets, quantization and precision. The remainder of the chapter presents experimental work to investigate the ramifications of this on practical use of the EPC.

7.5 Character Recognition Experiments

This section presents a series of experiments carried out using the system explained above. Each experiment was designed to test various assumptions regarding the hardware and software emulation and determine the limits of the EPC. Three experiments are presented where the complexity of each subsequent one is increased by way of increasing the number of training patterns. The first experiment uses a simple problem to investigate the matching of software simulation and the neural hardware. The second experiment investigates the effects of limited dynamic range and sigmoid gain on the ability to train the network; while the third attempts to compare performance of the software and hardware networks after training.

7.5.1 Simple Character Recognition Problem

The problem used for this investigation is the artificial 1-of-N character recognition system shown in figure 7–3. Input data consists of a 5×5 pixel array on which characters are represented. These inputs are mapped to a hidden layer of a variable number of neurons. These then feed an output layer of one neuron for each of the N training patterns. This artificial problem offers several advantages in terms of experimenting with the EPC:

- Highly controlled problem.
- Easily generated training sets of varying size and complexity.

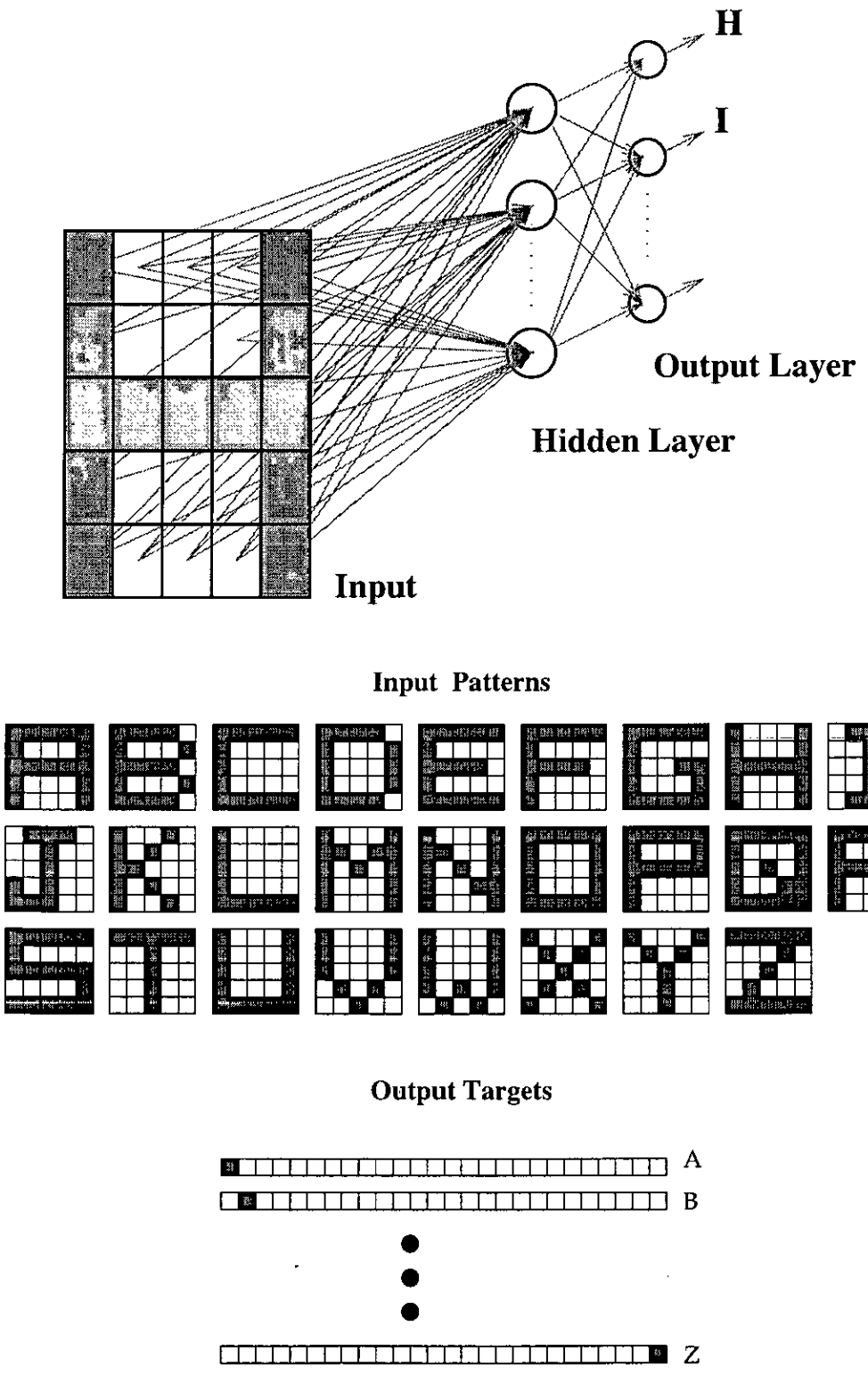


Figure 7–3: Simple Character Recognition Problem

- Problem represents a vehicle for investigating training hardware neural networks.
- Allows comparison of software (high resolution/accuracy) and hardware (low resolution/accuracy) neural networks.
- Allows exploration of the limits of the hardware using a problem of graded complexity.

7.5.2 Experiment One: Three Character Recognition

Table 7–2. Experimental Summary: 3 Character Recognition

Aim : Using a *minimal* problem look at the matching of the software model and hardware reality.

Network Structure : 25:14:3

Training : Error back-propagation training algorithm[10] on training set of 'H','J','K'.

Stopping criteria : MBE ≤ 0.05

Learning Parameters :
Learning rate $\eta = 50$
sigmoid gain $\lambda = 0.05054$

Results Summary : A comparison was made between the hardware and simulated networks for this simple problem.

- Both the ideal software model and the hardware evolved a solution in similar manner.
- Analog hardware non-idealities present translated to hardware training times that were longer than the ideal software network.
- When the hardware network was trained from the weights evolved in the software model, initial error was very high but further training quickly *trimmed* weights to a low error solution.

These points demonstrate that the software model of the hardware network is a reasonable first order approximation. It was demonstrated that weights evolved in software could be *trimmed* on the hardware to get a fast solution.

Initial experiments were performed on a training set of three input vectors, those representing 'H', 'J' and 'K'. These were chosen as input vectors *easy* to distinguish. The aim of these experiments was to investigate the software model of the network and compare results gained with runs performed on the hardware.

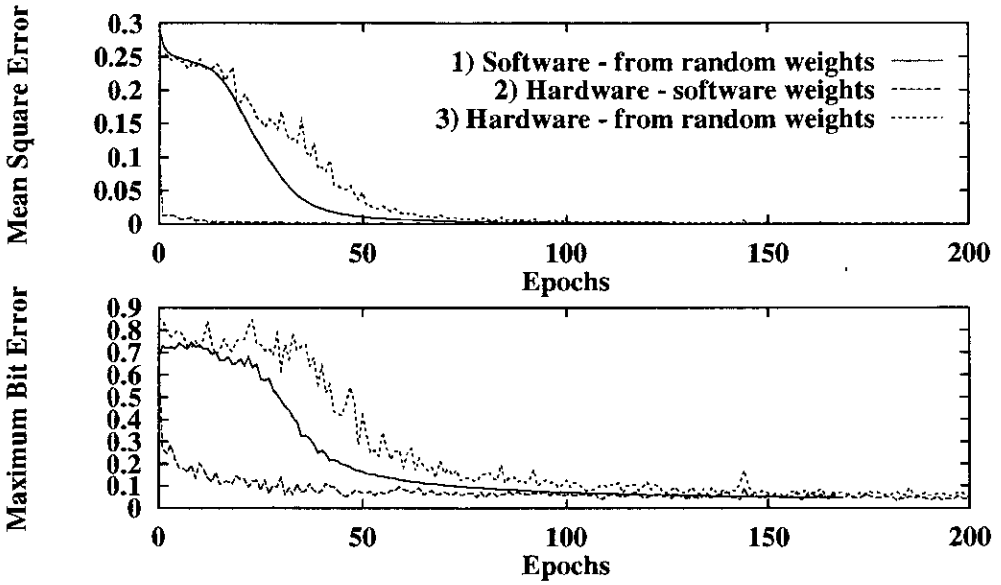


Figure 7–4: Results of Three Character Recognition Experiments

Results

Figure 7–4 shows the salient results from these experiments. The graphs show the evolution of the mean square error (MSE) and the maximum bit error[2] (MBE) of the network for each epoch or presentation of the training set. Results from three networks are shown in these graphs:

1. The benchmark for comparison is a software network in which all data is evaluated at 32 bit floating point precision. The error evolution of this network is shown in the green curve. The training proceeds smoothly to a low final error.

2. In the second network (blue curve) the weight set evolved by the software network is quantized and downloaded to the hardware then the network is trained. The fact that there is a low initial error which quickly trains to the target error demonstrates that the software model is a reasonable one. As expected non-idealities in the hardware introduce differences between the software and hardware networks as manifest in the initial error.

3. These differences can be seen in the third experiment where the weights are initialised to the same random values as the first software network. Here the forward pass is done on the hardware (red curve). The error characteristic follows the software closely at first but as differences due to hardware non-idealities accumulate, the characteristic departs from the idealised case.

[2]Maximum bit error is the maximum of the absolute error of any of the output neurons i.e. $\max_j (|y_j - t_j|)$

The effects of noise, offsets and gain variations are manifest in the fact that the characteristic is not smooth and monotonic as with the ideal case. These effects are most prevalent at high errors where the neurons are operating in the steep, high gain portion of their sigmoid characteristic, effectively amplifying the noise present. As the weights evolve, pushing outputs towards the flat, low gain, sections of the sigmoid these effects are less prevalent.

Performing multiple runs of the experiment confirms the general trend that the hardware network, due to the non-idealities present, takes substantially longer to train than the ideal software model.

Conclusions

From this series of experiments we can conclude that:

- The software model is a reasonable first order approximation of the hardware.
- Weights evolved in the software can be used as a starting point for quick hardware training (trimming).
- Training fully on hardware is possible and takes substantially longer than the ideal software case.

These results are comparable to those found on other hardware systems such as the optically controlled system of Frye *et al* [35] and the EPSILON system[44].

7.5.3 Experiment Two: Ten Character Recognition

The first series of experiments introduced three networks: a benchmark high resolution software network, a hardware network trained from software evolved weights and a hardware network evolved from random weights. All three converged to a similar final training error leading to the conclusion that the model and assumptions made were at least valid for simple problems.

In this series of experiments the complexity of the problem is increased by presenting a training set of the first ten letters of the alphabet. The problem is more difficult as not only are there more patterns to classify but certain characters (input vectors) in this set lie very close in input space. For example 'C' and 'G' differ by only two pixels and are thus difficult to distinguish. This serves to highlight the effects of limited dynamic range of weights as weights are forced to the extremes of their dynamic range. To model this a software network with similarly limited weights is introduced.

The first series of experiments concentrates on the effects of dynamic range limitations; to accomplish this a method of introducing variations in dynamic range is first presented.

Table 7–3. Experimental Summary: 10 Character Recognition

Aim : Using a more difficult problem, look at the effects of limited dynamic range on network performance.

Network Structure : 25:14:10

Training : Error back-propagation training algorithm[10] is used on the training set of the characters 'A'–'J'.

Stopping criteria : epoch = 7,000

Learning Parameters :
Learning rate η = 150,1000 & 2000
sigmoid gain λ = 0.05054 and 0.0382

Results Summary : A new software model developed to model the effects of limitations in precision and dynamic range in the weight set.
It was shown that the dynamic range of the weights directly effects the final error of the solution. By increasing sigmoid gain, dynamic range could be effectively increased resulting in a better solution, while with the low gain sigmoid error was large and variable.
It was also demonstrated that the stability of the hardware network was more sensitive to the learning rate parameter η than the software model. This was principally due to larger η values producing larger weights which were then clipped in hardware representation. The implications of this are that the minimum training time of the hardware is longer than that possible on software.

It was also found that the hardware network was more sensitive to the learning rate parameter η. This is the focus of the second set of results.

Weight saturation

In the previous experiment the weights of all the networks remained within the hardware's boundary of $w_{ij} \in [-128, 127]$. For the problem presented here, as several input vectors are separated by small distances in input space, larger weights evolve to separate these close patterns. In the case of the hardware this process is limited by the saturation of the weight as it reaches the physical limit of $T_{ij} \in [2.5, 5V]$ that is $w_{ij} \in [-128, 127]$. The precision of the weights on the EPC is 8 bits. To better investigate the effects of this a new software model is introduced with a likewise finite precision bounded weight set. This is done simply by hard-limiting or clipping the software weights to this range and quantizing the weights to 8 bit values for the forward pass.

Limited dynamic range

The physical characteristics of the hardware network imply a limit on the dynamic range of synapses and neurons. In an ideal network any single synapse may change a neuron's output from *off* to *on*. This is not the case with hardware as the effect of any single synapse is limited by the maximum weight value it can assume. If classification depends on a small change in only a few inputs, the dynamic range of these synapses may be insufficient to influence the result appropriately. There are ways by which this dynamic range can be altered however:

1. Alter the sigmoid gain λ (equation 7.6). If sigmoid gain is increased, a small change in network activity produces a larger swing in output. Effectively this means the neuron is more sensitive to the contribution of each individual synapse. This is equivalent to an increase in dynamic range. In practical terms, increasing λ compresses the reference ramp waveform. There are obviously limits to this in how accurately a compressed ramp waveform can be generated. Also a high gain sigmoid effectively amplifies any noise present in the network activation. Thus the trade-off here is increased dynamic range for a decrease in noise tolerance and a decrease in accuracy of ramp waveform and thus neuron transfer function.

2. While the physical value of weights in chip are limited, values of inputs are encoded in the time domain as pulse-widths. If the pulse-width of inputs is doubled, so is the effective dynamic range of synapses. Thus a trade-off of increased dynamic range for increased computation time can be made.

Note that in both these cases *precision* is unaffected; that is, there are still the same number of discrete values between maximum and minimum extremes. Frye *et al* looked at the effects of decreasing precision in [35] while keeping dynamic range constant. The effect of dynamic range in the analog network is analogous to the effect of dynamic range in fixed point digital implementations as studied by Hoehfeld *et al* [49] and Vincent *et al* [111]. In both these cases auto-scaling schemes are presented to increase dynamic range when necessary. In the experiments presented here dynamic range enhancement is performed manually by adjusting the sigmoid gain.

Results: Dynamic range limitation.

Figure 7–5 presents results for four network training runs which are:

1. An idealised floating point software simulation with $\eta = 150$[3].

[3]The η values here may seem high to the reader – this is because of the small values of the multiplier constant k and the sigmoid gain T, both usually unity in software simulations

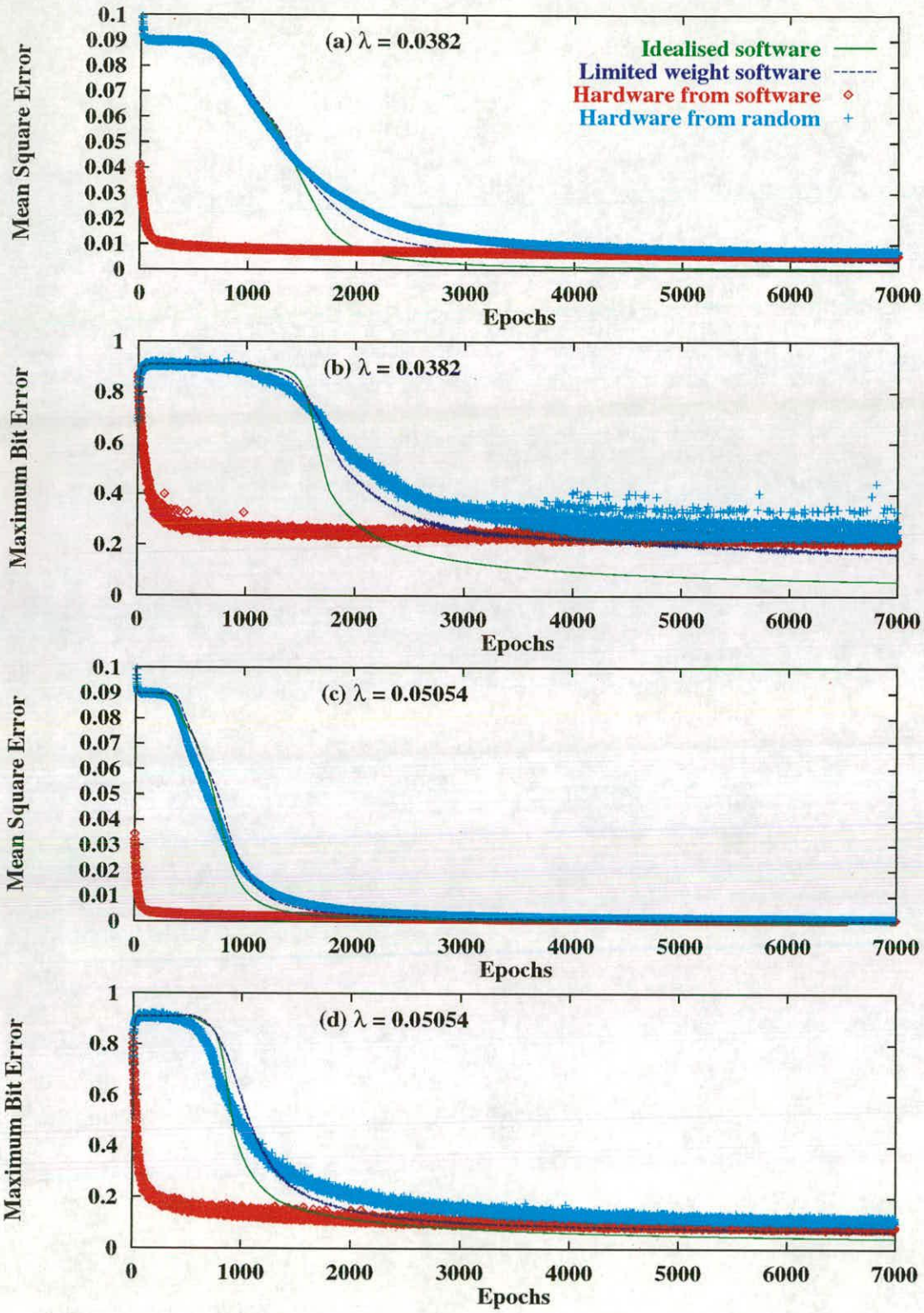


Figure 7–5: Results of Ten Character Recognition Experiments

2. A software simulation where the weights are hard-limited to the same range as the hardware, $\eta = 150$.

3. A hardware run continuing from the above evolved weights.

4. A hardware run from random weights.

The effects of dynamic range are investigated by repeating the experiments at sigmoid gains of $\lambda = 0.0382$[4] (figure 7–5 (a) and (b)) and $\lambda = 0.05054$ (figure 7–5 (c) and (d)).

Several key observations can be made concerning these results:

- The evolution of the software model and hardware networks is again similar and the network trimmed from the software model again finds a solution quickly. This indicates that the network models developed still form a good approximation of the hardware.

- For the low gain sigmoid ($\lambda = 0.0382$, figure 7–5 (a),(b)) weights become saturated before synaptic activity is sufficient to push neuron outputs to the relatively flat extremes of the sigmoid where the solution lies. This is indicated by the final, near flat, portion of the error graphs having relatively high maximum bit and mean square error.

- With neuron outputs in this high slope area of the sigmoid characteristic the effects of small weight changes are large, as is the effect of noise. In figure 7–5(b) several distinct levels of maximum bit error can be seen. This is due to limited dynamic range and quantization. The ideal solution lies outside the limits of the weights. The boundary around weight space forms several local minima which the network visits as the back-propagation algorithm tries to push the solution past the boundary of the limited weight set. This is what is seen in figure 7–5(b) as a small change in weights jumps the solution to different minima with different maximum bit error.

- When λ is increased (figure 7–5 (c),(d)) final mean square error is reduced significantly and maximum bit error reduces to under 0.2. Thus the network is much closer to an ideal solution implying that increasing the sigmoid gain has increased the dynamic range of the weights and promoted a better solution.

It can be seen that while the limited dynamic range and quantization of the hardware has a significant effect on the final solution of the network it is still able to reach a stable solution. Effects of limited dynamic range can be minimised by making the neurons more sensitive to small weight changes. This can be done by compressing the output ramp to the comparator neurons by way of the sigmoid gain, λ.

[4]These numbers are chosen due to hardware ranges - a λ of 0.0382 utilises the full voltage swing of the output neuron comparator, numbers greater than this reduce this voltage swing hence make the neuron more sensitive.

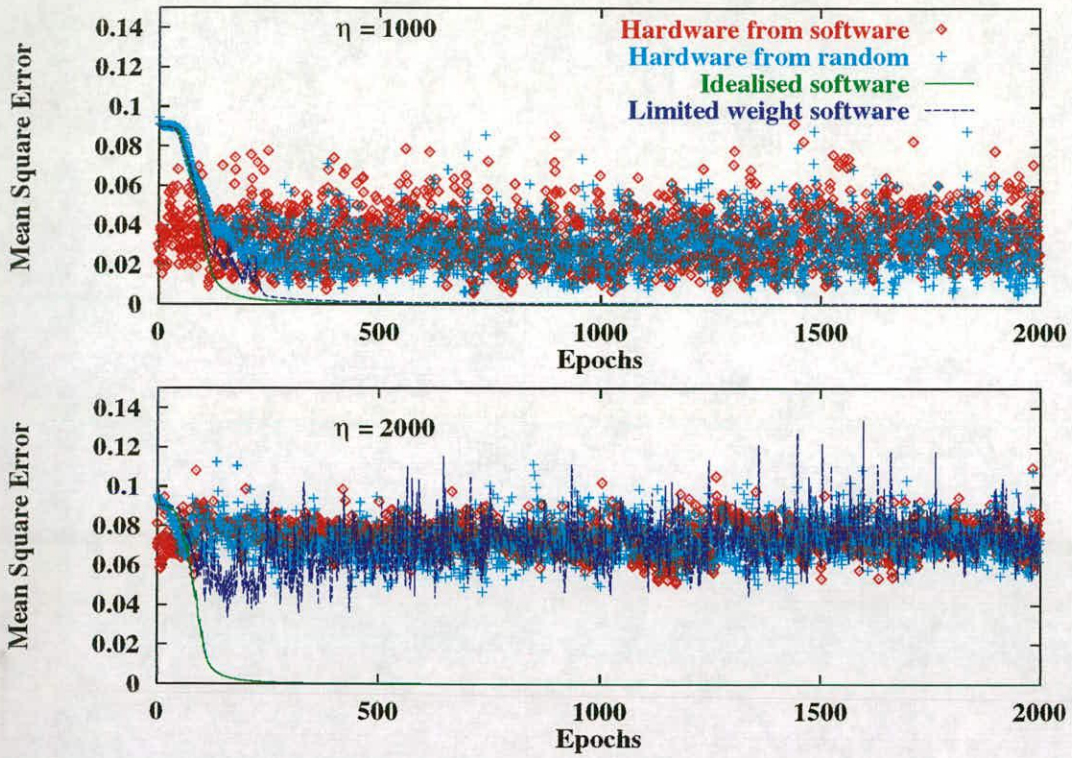


Figure 7–6: Results of η Variation Experiments

Effect of learning rate, η

The learning rate parameter η determines the magnitude of steps taken along the error gradient (equation 7.9). Hoehfeld and Fahlman [49] showed that with limited precision in the back-propagation path these steps could be quantised to zero and thus no learning occur. For the hardware here the back-propagation path is performed at high precision in software. If η is increased eventually the algorithm becomes unstable as the weight step jumps to a region with a radically different error gradient rather than taking small steps along the gradient to a minima.

For the hardware network, calculation of the error gradient is less accurate than the idealised and limited weight software model case. This is because the neuron outputs used in this calculation are quantised. Thus the error function, rather than being smooth and continuous, is also quantised. The effects of this are apparent in figure 7–6. Here the character recognition problem has been run with η set at 1,000 and 2,000. The key observations from these experiments are:

- For both $\eta = 1000$ and $\eta = 2000$, the idealised software network evolves a solution, thus η in itself is not too large.

- The hardware networks are unstable for both η values and do not evolve a solution.

- For $\eta = 1000$ the weight limited software model also evolves a solution while the hardware networks do not. This indicates that hardware non-idealities which are not modelled, such as neuron quantization and weight uncertainty, have a significant effect.

- For $\eta = 2000$ the weight limited software model is also unstable indicating that the hardware non-idealities modelled are responsible also for this instability.

A partial explanation was gained by examination of the weight sets. This revealed that larger η values led to larger values of individual weights. For $\eta = 2{,}000$, many weights are clipped and the training becomes unstable. At $\eta = 1{,}000$ only a moderate proportion of weights in the weight limited software model became clipped and a solution was evolved, however hardware failed to converge and became unstable. This tends to indicate that the effects not modelled in the software model, namely weight uncertainty and offset and quantization of inputs and outputs, are also limiting training performance. The effect of large η producing larger weights has been noted by Cairns in [18].

The ramifications of this are that the hardware neural networks are slower to train than high precision software networks where a larger η can get to a solution faster. Note that no experiments have been done here with other back-propagation derivatives designed for faster convergence, it would be interesting to see the effects of hardware non-idealities on these.

Summary

This section has introduces a new software model that includes the hardware non-idealities of a weight set limited in precision and dynamic range.

It was shown that the dynamic range of the weights directly effects the final error of the solution. By increasing sigmoid gain, dynamic range could be effectively increased resulting in a better solution, while with the low gain sigmoid error was large and variable.

It was also demonstrated that the bounded nature of the weights has an effect on the speed that the network can be trained in that the hardware becomes unstable with increasing η well before an idealised software network does.

All these results indicate that the hardware can produce adequate solutions but care must be taken in setting network parameters to achieve the best possible solution.

7.5.4 Experiment Three: Generalisation and Character Recognition

The final set of character recognition experiments takes the problem to its logical conclusion of the full twenty-six character set then compares the results of the experiments performed thus far. Following this the issue of generalisation ability is discussed.

Table 7–4. Experimental Summary: 26 Character Recognition and Response to Noise Generalisation

Aim : The experiment is performed with the 26 character alphabet and tested with noise corrupted inputs to compare the software and hardware networks

Network Structure : 25:14:26

Training : Error back-propagation training algorithm[10] on training set of 'A'–'Z'.

Stopping criteria : epoch = 10,000

Learning Parameters :
Learning rate $\eta = 50$
sigmoid gain $\lambda = 0.05054$

Results Summary : The full training set of 26 characters was trained on the network which was able to produce a solution, albeit with a higher error level. The four networks were then compared for generalisation ability and were found to be virtually equivalent.

Twenty-six character recognition

The results of training the character set 'A'–'Z' are shown in figure 7–7. The results are similar to those obtained in the ten character case except, as would be expected, the maximum bit error is greater still. A summary of average final error from the experiments presented so far is shown in figure 7–8.

It is possible to see some general trends in relation to back-propagation learning on hardware networks from these results:

- As the complexity of the problem increases so does the achievable minimum error. This is true for software as well, but with the further limitations placed on hardware the effects are manifest earlier.

- The dynamic range of the neurons is critical. This can be seen from the 10 character experiment where error was very large for the small dynamic range case.

- That the software model with limited weights exhibits an increase in error indicates the limited weight dynamic range is a significant factor in the performance of the hardware neural network.

- That the hardware error is greater than the limited software simulation indicates the other hardware factors, such as weight uncertainty, limited output resolution, offsets etc., also limit the performance of the hardware solution.

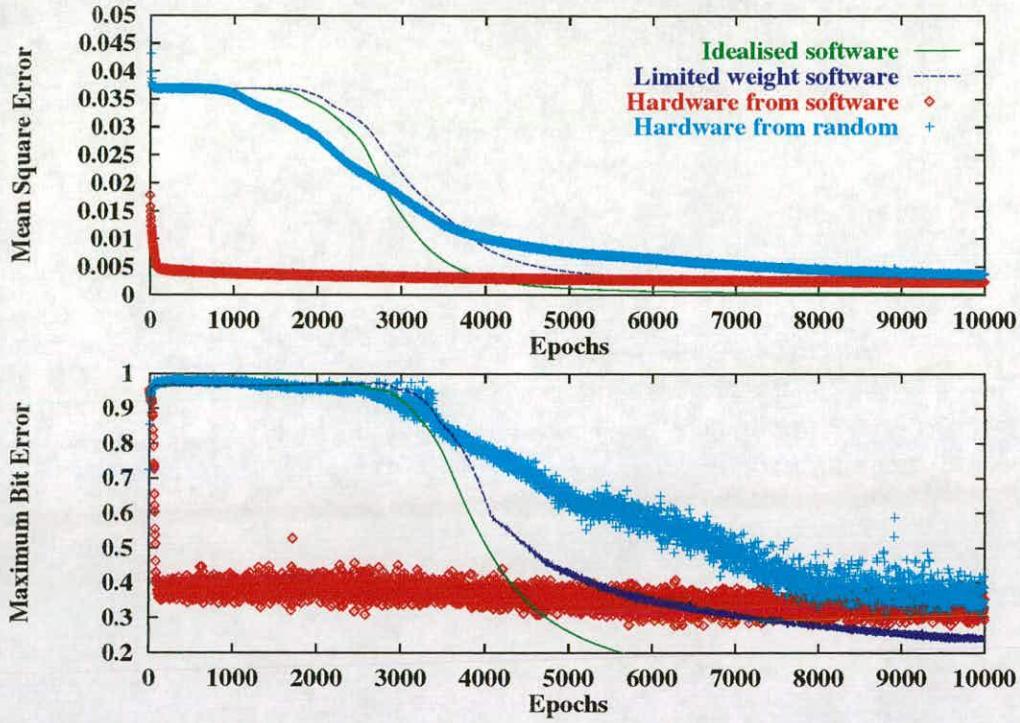


Figure 7–7: Results of Twenty Six Character Recognition Experiments

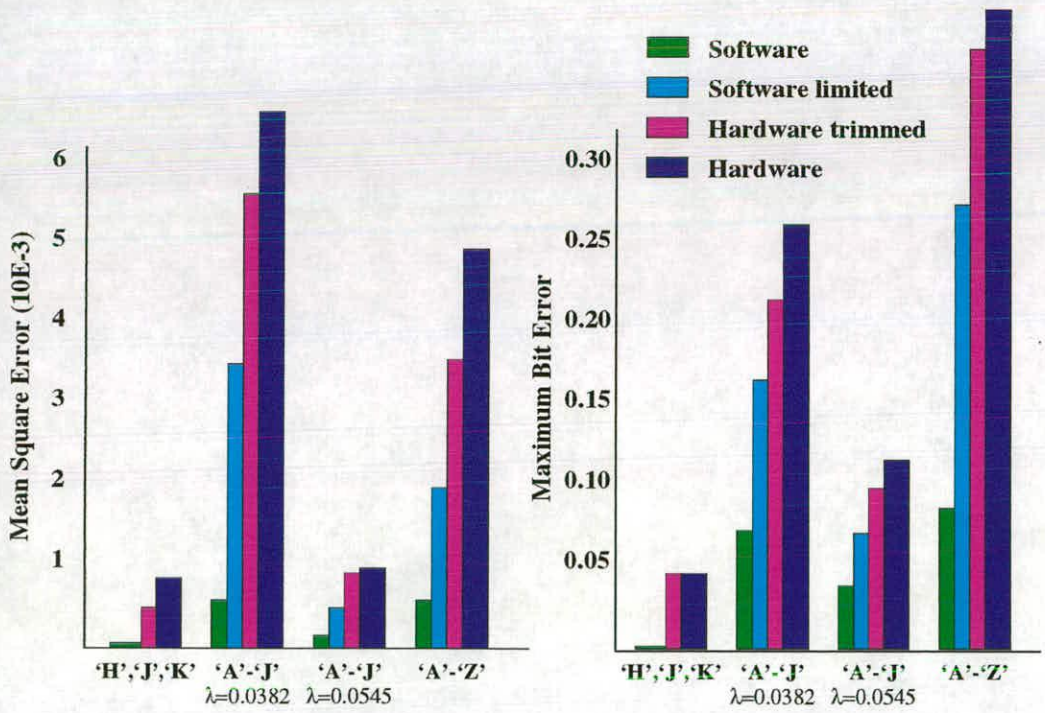


Figure 7–8: Summary of Errors from Character Recognition Experiments

Generalisation performance

For a *real-world* problem a training set is not as readily defined as in the artificial problem used here. In such a case the training set consists of only a sub-set of inputs that the network may encounter. The ability to classify inputs the network was not trained on is called *generalisation*. To gauge generalisation ability an examples database is subdivided into a training set and a generalisation set. The network is trained using the training set and then tested on the generalisation set. For the contrived problem presented here there is no generalisation set, instead the "generalisation ability" of the network may be gauged by testing the network on inputs corrupted by adding random noise. For the experiments presented here the peak value of this random noise is expressed as a percentage of full scale input. At each noise level 3000 trails are performed and the results are averaged over 10 training runs. Figure 7–9 shows the results of these experiments where the percentage of correct classifications is plotted against

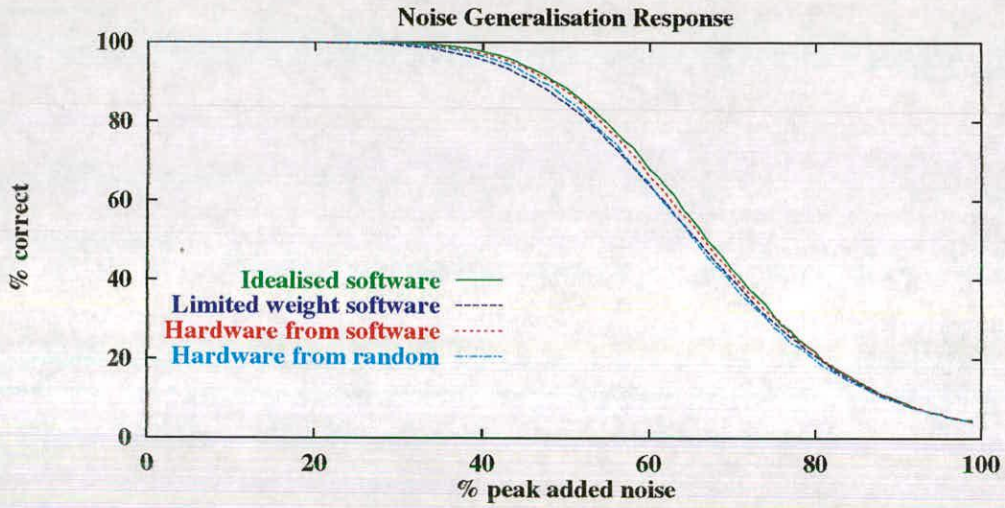


Figure 7–9: Noise Generalisation Experiment

the percentage of peak added noise. Surprisingly, considering the different final errors achieved by each network, the performance of each of the networks is almost identical. This tends to suggest that the increased error present in the hardware network has not significantly effected its generalisation ability. This will be further tested in the next section where a more complex problem is implemented on the network.

7.5.5 Summary

This section has used the artificial character recognition problem to study the effects of hardware non-idealities on performance and training of the EPC. It was found that learning rate must be kept small to form solutions with the limited hardware weight set and that dynamic range limitations had a strong effect on network performance in terms of final training error achievable.

For the character recognition problem, this larger training error did not adversely effect the generalisation ability.

The rest of this chapter continues by applying the lessons learnt thus far to more difficult, real-world problems. The next section examines the use of the EPC as a function approximator and classifier while Section 7.7 examines a range of 1-of-N classification problems.

7.6 Link Admission Control

This section tests the EPC on a more difficult *real-world* problem. This problem is that of *link admission control* in an ATM (Asynchronous Transfer Mode) communications network router, the neural solution of which has been developed by Nordström and Gällmo *et al* at Uppsala University, Sweden[37,38,91,92,93]. The problem is more demanding than the character recognition problem in that inputs and outputs are real valued rather than digital, coupled with a non-linear mapping. Here we investigate the performance of the hardware in providing a solution to this problem. The problem is first described, then a metric is introduced to evaluate network performance leading to the experimental results.

7.6.1 The Problem

In an ATM network the problem of link admission control is to determine if a new connection can be accepted on a link in the network. This is achieved by calculating an estimation of the *probability* of losing a data packet, P_{loss}, based on the current load present on the link along with traffic parameters that characterise the new connection. If this probability exceeds a certain set limit, $P_{loss} = 10^{-9}$, then the new connection is rejected, otherwise it is accepted. Hardware implementation would be an advantage in this case as each link must perform such a function and exact methods of calculation are too time consuming for real time operation[91]. For further background on this problem see [91,92].

The neural network solution to this problem involves estimating the probability of loss (P_{loss}) from six statistical measures of aggregate traffic on the link[93] along with three parameters defining the connection. These are processed by an MLP of 9:6:1 architecture. Hidden layers have a sigmoidal activation while the output neuron is linear as the network is performing a function approximation.

7.6.2 Network Performance

Though the decision to accept or reject a connection is a classification problem, the network performs as a function approximator to estimate P_{loss} as this estimate is useful elsewhere in

the ATM system. Performance of the network can be judged on a variety of measures; the most basic being whether the network classifies the input on the correct side of the decision boundary ($P_{loss} = 10^{-9}$). This is not necessarily a wholly effective measure of performance as wrong decisions near the boundary are not as detrimental to performance and ones further away. To take this into account a *width* statistic ω is defined by Gällmo in [38] that measures the mean square distance of a bad decision from the decision boundary:

$$\omega = \frac{\sum_{\forall\ errors} \left(\log_{10}(P_{loss}) - \log_{10}(10^{-9})\right)^2}{N_{errors}} = \frac{\sum_{\forall\ errors} (\log_{10}(P_{loss}) + 9)^2}{N_{errors}} \tag{7.11}$$

Where P_{loss} is the estimated probability of loss and N_{errors} is the total number of bad decisions.

These measures of performance will be used to compare the networks constructed here with results from Gällmo *et al* [38].

7.6.3 Training and Test Data

A database of traffic situations has been compiled by researchers at Uppsala University, Sweden[38,93]; the author is indebted to Jey Ngole and Olle Gällmo for arranging access to this data. In this data-set, a fluid flow model has been used to accurately calculate targets for a set of over 100,000 random traffic situations. A subset of 500 of these is used for training and a further 5000 for testing/generalisation. The input data for these sets is normalised on a [0,1] interval. Target values are normalised as shown in figure 7–10.

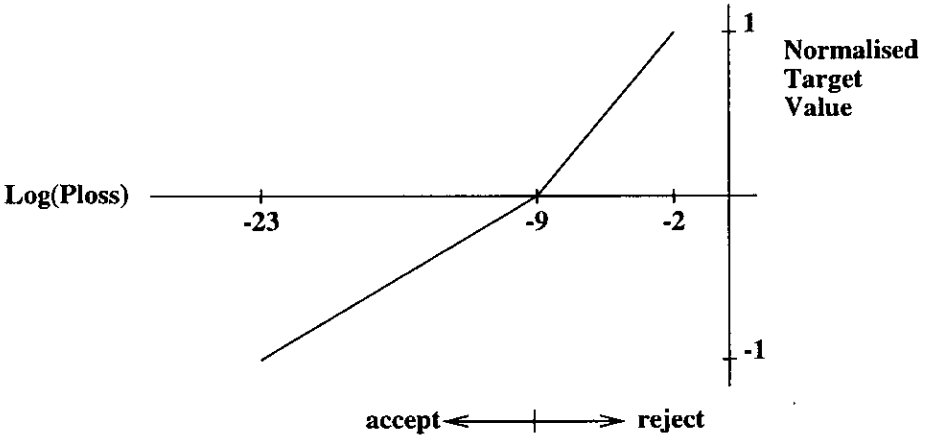


Figure 7–10: Normalisation of Target Values

Table 7–5. Experimental Summary: ATM Link Admission Control

Network	Width ω	Decisions Correct	Mean Sq. Error
Swedish results [38]	0.66	88.5 %	–
Idealised Software	0.36	89.0 %	0.0060
Weight Limited Software	0.48	88.4 %	0.0091
Hardware	1.06	85.6 %	0.0163

7.6.4 Results

The link admission control data was used to train three networks:

1. An idealised software network.
2. The weight limited software model.
3. The hardware neural network.

The results presented are the average performance on the generalisation test data of 10 training sessions per a network each trained for 5000 epochs[5]. Table 7–5 shows the results of these experiments while figure 7–11 shows the distribution of incorrect decisions.

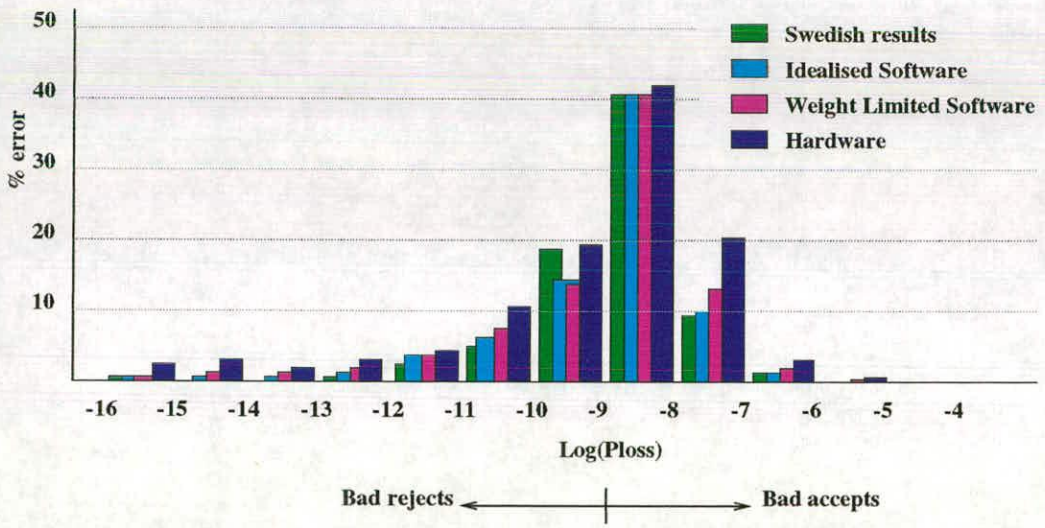


Figure 7–11: LAC Classification Experiment

[5]The Swedish results came from networks trained for 1000 epochs, however with the low η values used for hardware compatibility, the networks here were trained longer to converge to a low error.

The key observations from these experiments are:

- The results from Sweden and the idealised software model show similar performance. The larger width measurement, ω, stems principally from the poorer performance in the $P_{loss} = [10^{-10}, 10^{-9}]$ region.

- The weight limited software model shows a small degradation in performance and ω value.

- The hardware network shows a substantial degradation in performance both in the total percentage of correct decisions but more specifically in the spread (width measurement ω) of bad decisions.

The performance of the hardware is encouraging on this much more realistic and difficult problem. The decrease in performance in terms of total correct decisions is not very large (4.4% below the idealised software). The principle variation between hardware and software is in the width measurement ω. The ω measure for the hardware is more than double that of the software. Examining the histogram of figure 7–11 the reasons for this become clear:

- Firstly bad decisions at the extremes of the P_{loss} show a marked increase for hardware compared with software, this increases ω. This difference can be partly explained by the dynamic range limitations present in the hardware, confirmed by the increase in error for the limited weight simulation.

- A more significant difference in bad decisions is in the interval $[10^{-8}, 10^{-7}]$ close to the decision boundary on the bad accept side. Here hardware is significantly worse than any of the software simulations.

This second area of difference can be explained with reference to the normalisation of targets values as shown in figure 7–10. The steeper slope of the reject side (an incorrect reject is a bad accept) makes the network more sensitive to small variations of P_{loss} on the reject side. This is coupled with the fact that there are more training and generalisation examples on the reject side. For the software network this encourages better performance on the reject side, as can be seen by the low number of bad accepts. The hardware network follows this trend further from the decision boundary, but close to it shows much worse performance than the software. The effect manifest here is noise; that is any noise present close to the boundary can easily push the output past the decision boundary.

The effects of noise combined with linear ramps highlights a weakness of hardware when used as a function approximator. When used with a sigmoidal output noise effects are minimised as outputs are generally at the extremes of the sigmoid which has low gain and noise is suppressed. With the linear response, noise has the same effect everywhere and this leads to incorrect classification when it effects outputs close to the decision boundary.

Despite the degradation of performance, hardware results are encouraging and such a degradation may be offset by the benefits of real-time operation a hardware solution can provide. Having used the EPC to prototype a hardware solution, if a dedicated solution was considered feasible a highly integrated solution would be called for. As the network size for this problem is small support functions, such as the pulse conversion, could be integrated along with the EPSILON II neural circuits. It must be noted however that this problem, having purely digital I/O, is not in the primary applications area as outlined in Chapter 1. This implies that a dedicated digital solution would in all likelihood be more feasible. The problem though demonstrates a class of problem, function approximation, which occurs within the target area of analog/mixed signal problem – for example inferring a plant transfer function from sensor readings. This section has shown that the presence of noise in analog hardware significantly degrades the performance of such solutions. Edwards' [31] theoretical work on noise in MLPs also noted the detrimental effects of noise in networks with neurons operating in a linear fashion.

7.7 1–of–N Classification

The remainder of the problems implemented on the EPC are 1-of-N classification problems. Here the goal is to classify input space into one of N possible classes as was done in the earlier artificial character recognition problem. An example within the target applications area of the EPC would be sensor monitoring to classify operation of an engine as 'correct', 'uneconomical' or 'dangerous'. Three test problems are presented here in order of increasing difficulty[6]:

1. Speaker identification.

2. Medical data analysis.

3. Robot region classification.

These test problems are the same as used by Cairns [18] in his work on precision in analog MLPs. They were chosen as a good representation of 1-of-N classifiers and to allow a direct comparison to the work of Cairns. The data for the problems is in digital form, though it originally comes from analog sources followed by some pre-processing. A brief outline of each problem is given followed by the experimental results gained in implementing these on the EPC.

[6]This order is as presented by Cairns [18], the author considers the ATM problem to be of similar difficulty to the speaker identification problem. Problem difficulty is not easy to quantify as it is a combination of factors such as the number of inputs, resolution of data and *sharpness* of decision boundary.

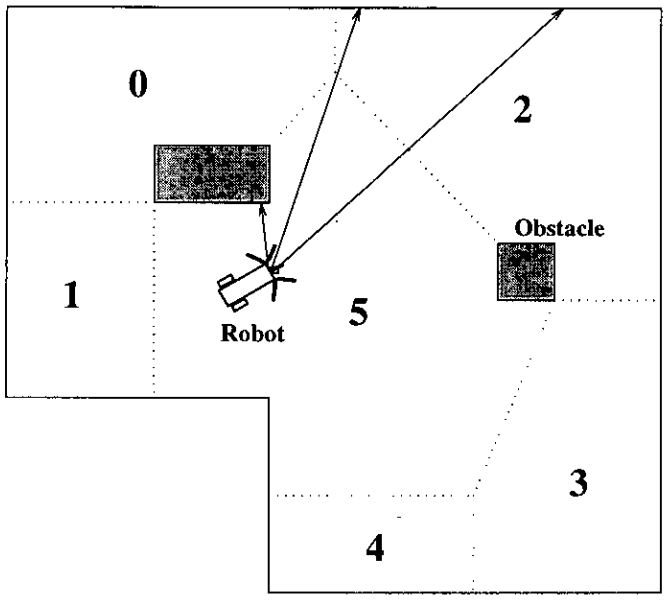


Figure 7–12: Robot Localisation Problem

7.7.1 Problem Outlines

Speaker identification

The objective of the speaker identification problem is to classify which one of three speakers was speaking given a short sample of speech input. The speech input is pre-processed to produce eight inputs and applied to an 8:8:3 MLP – more details on the problem and pre-processing can be found in Cairns [18].

Medical data analysis

The task of this problem is to classify the sleep state of patients from measurements derived from electroencephalogram (EEG) readings. A data-base of measurements classifying patients as 'awake', 'dreaming/light sleep' or in 'deep sleep' is used to train a 10:6:3 MLP. Five hundred random data points are used for training with a further 1,000 used for generalisation tests. Further details of this problem can be found in Tarassenko [104] and Cairns [18].

Region classification

The goal of this problem is to classify which one of six areas of a room a hypothetical robot is in. The floorplan of the room is shown in figure 7–12. The room contains two obstacles and the six regions are divided on the basis of the nearest visible corner. A set of eight features extracted from a 360° range scan are presented to a 8:25:6 MLP. The problem is extremely

difficult as at the boundaries of the regions, input data vectors are very similar. Further details of this problem can be found in Tarassenko [106] and Cairns [18].

7.7.2 Results

The experimental method used was similar to that of the ATM problem. Results are evaluated using the method used by Cairns [18] such that direct comparisons can be made. A summary of the experiments performed follows:

- Three networks were evaluated:
  1. An ideal floating point precision software network.
  2. A weight limited software network.
  3. The hardware network.
- Each experiment was repeated five times with different initial weights.
- Performance was evaluated on the correct classification of the generalisation set. This evaluation, or validation, was done every 200 epochs.

The results of the experiments are shown in table 7–6. For each problem and network, two metrics are presented[7]. These are the percentage error of classifications on the validation set (in **bold** type) and the mean square error on the validation set. These values are an average of the best validation results of each of the five runs. The key observations from these results are:

- A degradation of performance occurs as the problem complexity is increased. This degradation is present both in the limited weight software model and the hardware.
- For the speaker recognition and medical data problems, hardware performance is very close to the limited weight software simulation. This indicates that the principal factor in reduced performance here is the limited precision and dynamic range of the weight set.
- The very poor performance of the hardware on the region classification problem, much below that of the limited weight software model, demonstrates that for very difficult problems other hardware non-idealities have large effects and prevent a good solution.

[7]Note that it is not the absolute values of these metrics that is important but rather the degradation of performance between the different network types. The achievable error rate is problem dependent and does not necessarily reflect the difficulty of the problem – for instance the software network achieves an error rate of 6.5% on the difficult region classification problem but only 19.2% on the easier speaker identification problem.

Table 7–6. Experimental Summary: 1-of-N Classification Problems.

Problem	Software Results	Weight Limited Software Results	EPC Hardware results	Oxford Software Results	Oxford Software Model of Hardware	Oxford Hardware Results
Speaker Recognition	**19.2%** 0.114	**22.3%** 0.114	**22.9%** 0.151	**18.8%**	**21.8%**	**20.3%**
Medical Data Analysis	**17.5%** 0.087	**28.2%** 0.133	**29.2%** 0.146	**19.0%**	**22.6%**	—
Region Classification	**6.5%** 0.018	**20.5%** 0.055	**36.3%** 0.085	**6.8%**	**11.4%**	—

Bold font – % error on validation set
normal font – mean square error on validation set

- The speaker identification problem was the only problem small enough to be implemented on the hardware of Cairns. This hardware employed 12 bits of weight precision refreshing on-chip dynamic capacitors and utilised a pulse-width modulation scheme similar to the EPC. The results of the two hardware implementations are comparable showing a similar drop in performance over floating point networks.

7.7.3 Discussion

Two areas warrant further discussion here: The first is the poor performance on the region classification problem and the implications of this with respect to the types of problems suitable for hardware. The second area is, for the successfully solved problems, what are the the principal factors limiting the hardware performance.

Region Classification: High resolution problems.

The region classification problem was the toughest problem attempted on the EPC. Not only was performance of the hardware poor but also the weight limited software model performed poorly and did not provide a reasonable match to the hardware results as it did for the other problems.

Basically the problem is too difficult. Decision boundaries are straight hyper-lines through decision space so on either side of a classification region input vectors are virtually identical. This requires a very high resolution to distinguish; the poor solution provided by the weight limited model shows that the hardware can not provide this. In the hardware case, noise present

Table 7–7. Experimental Summary: Gain Variation Experiments for Medical Data Analysis Problem.

Network	gain = 0.1	gain = 0.2	gain = 0.4	gain = 0.8	gain = 1.6
Ideal software network	**17.3%** 0.088	**17.5%** 0.087	**17.1%** 0.087	**16.6%** 0.086	**17.0%** 0.088
Weight limited software network	**29.5%** 0.139	**28.2%** 0.133	**23.2%** 0.113	**21.2%** 0.105	**17.2%** 0.099
Hardware network	–	**29.2%** 0.146	**28.7%** 0.136	**26.6%** 0.133	–

Bold font – % error on validation set
normal font – mean square error on validation set

in the network will make close vectors indistinguishable – this explains the degradation in performance of the hardware compared to the weight limited model.

The data for this problem was generated artificially, allowing exact demarkation of regions. If the problem was implemented in the real-world, such sharp boundaries would be unrealistic as inevitable noise on sensor readings would *blur* boundaries. Still this problem does highlight a fundamental limitation of analog hardware: while lower resolution analog hardware can perform well for many problems, if input data is of high resolution and classification boundaries are arbitrary, low resolution solutions will be poor and noise will greatly affect the solution.

Limiting factor: Dynamic range or precision?

Turning to the problems where the hardware produced a good solution, the first point to note is that hardware and the weight limited model performed comparably. This indicates that effects such as noise, input/output quantization, offset and gain variations, have had minimal impact. This raises the question of which factor modelled in the weight limited software, dynamic range or precision, has the most significant effect on the solution. To investigate this some further simulations were carried out varying the dynamic range. The software models were kept the same and the multiplicative constant k was varied (see equation 7.3) to vary dynamic range of the weights. The medical data problem was chosen for these experiments as it was the most complex of the successfully solved problems. The results of these experiments are shown in table 7–7, the key observations are:

- Varying the gain has a minimal effect on the ideal software solution, both in terms of correct classification and mean square error.

- As gain is increased in the weight limited model both classification and mean square error improved. At a gain of 1.6 (8 times that of the hardware gain) classification performance was comparable to the ideal model.

These experiments demonstrate that dynamic range is the principal factor limiting performance in this problem and that low precision is acceptable.

To further test this the gain of the EPC was similarly increased by scaling the duration of the input pulses as outlined in Section 7.5.3. Inputs were scaled by a factor of two and four to produce gains of approximately 0.4 and 0.8 respectively. The bottom row of table 7–7 shows that an improvement in classification performance was achieved but not to the extent predicted by the weight limited simulation. The most likely reason for this is that in scaling the inputs the noise present is also increased along with the effects of offset and gain variations, this is due to the longer integration times. Despite this the general trend of increased weight dynamic range resulting in increased performance was shown. It is not envisaged that this increase would meet the software performance as effects of increased noise would soon outweigh the benefits of increased dynamic range. To further investigate dynamic range effects a mechanism whereby dynamic range can be increased without substantially increasing noise and inaccuracy needs to be devised.

7.8 Summary

This chapter began by investigating the practical aspects of back-propagation learning on the EPC by using an artificial character recognition problem. The graded complexity of this problem allowed easy investigation of:

- Software models to compare hardware results.
- Effects of dynamic range on training performance.
- Effects of the learning parameter η on network training.

The investigation showed that:

- With a high precision back-propagation path, back-propagation learning could be applied successfully to the non-ideal EPSILON II hardware.
- The effects of hardware non-idealities were manifest as slower training times and higher final training error.
- Limited dynamic range was highlighted as an issue in network performance and the need to vary this according to problem was demonstrated. The PWM scheme of EPSILON II offered a simple method of accomplishing this through variation of sigmoid gain.
- Generalisation ability did not exhibit as rapid a degradation as training error.

The lessons learnt from this study were next applied to a series of more difficult problems. Only in the most difficult of these problems did the hardware fail to provide an adequate solution. This was attributed to the hardware having insufficient resolution and the effects of noise and other analog non-idealities degrading the solution. This result would apply to other problems with high resolution input data and where classification regions have very close input vectors.

Hardware performance on the other problems compared well with the ideal software. The following observations account for differences in performance:

- Analog noise has a detrimental effect on linear output units as demonstrated in the ATM link admission control problem.

- Dynamic range of weights is a limiting factor in performance. It was shown that increasing dynamic range can improve the solution evolved.

- With sigmoid transfer functions, noise, input/output quantization and other analog non-idealities such as offset and gain variations had only minor detrimental influence on the problem solution.

Further investigation of the effects of weight dynamic range limitations would be a valuable contribution to the field. Due to time requirements, experiments such as investigations into the limits of the region classification problem and the effects of increased dynamic range on the solution to the speaker identification problem, were not carried out.

Chapter 8
Kryton: An Instinct-rule Robot

8.1 Introduction

This chapter presents the section of work aimed at utilising the EPC as a controller in an autonomous mobile robot named Kryton. The control methodology of Kryton is based on a control strategy and software driven exemplar (named *Alder*) developed by Nehmzow[88] from Edinburgh University's department of Artificial Intelligence.

The purpose of the work presented in this chapter is twofold: primarily it demonstrates the EPC operating in a real system with real-world analog inputs, thus achieving one of its design goals. The controller also utilises the mixed signal processing capabilities of the EPC. Secondly it offers a vehicle for exploring the instinct-rule control strategy. This is achieved by *enriching* the robot's sensory inputs using analog sensors, making changes to the control algorithm in response to this move to the analog domain and exploring other instinct-rules.

The first sections of this chapter introduce the instinct-rule controller architecture. Kryton is next specified and the differences between it and its predecessor, Alder, summarised. Next the extensions to the controller architecture that have arisen from this work are outlined before the experimental work is presented.

8.2 Background: Approaches to Robotic Control

Robotic control has been a principal testing ground for theories in Artificial Intelligence (AI); the field dedicated to producing intelligent behaviour in machines. Early, or *classic*, AI work in robotic control was characterised by a *vertical decomposition* of the control task in which sensor signals enter the controller and are processed in a pipelined manner by perceptual, modelling, planning and executional modules to produce action commands[73,89]. These controller techniques rely on an internal *world model* of the robots world and symbolic representation of intermediate steps. The effectiveness of the controller is inherently limited by the accuracy of this world model – problems arise when the robot encounters a situation not considered by designers of the world model. This type of conundrum led to the development of *behaviour based* robotic controllers which are characterised by the absence of internal world models and

symbolic representations, plus tight coupling between sensory input and effector output[88]. Brooks in [12] argues strongly against von Neumann based AI and states the case for behaviour based models. Exemplars of this type of work include the subsumption architecture of Brooks [11,13,73] and the instinct-rule controller of Nehmzow[88,89,90] which employs neural techniques. The principles on which this instinct-rule controller is based, and the robots built around, can be summarised as:

Experiment, rather than simulation. A computer simulation can only simulate what is known; by using *real* robots the need to model the environment is eliminated and the non-idealities of the real environment are present.

Minimise a priori knowledge. This is achieved by using a self-organising structure; in this case a neural network is trained with the aid of an instinct-rule teacher. The neural network forms a sensor-motor mapping with no predefinition. A priori knowledge is limited in a minimal way to the instinct-rules.

Rapid competence acquisition. Fundamental to the robots operation is the acquisition of basic competencies such as obstacle avoidance, contour following or direction biased motion. These competencies must be learned rapidly to effectively compensate for changes in environment. For a real robot these environmental changes can also include internal ones such as sensor failure or degradation.

These are the guiding principles behind all instinct-rule robots constructed to date[89]. The robot from which this work has evolved was the original robot *Alder*. The following section describes the controller architecture of Alder, then Kryton is introduced and the differences between Kryton and Alder summarised.

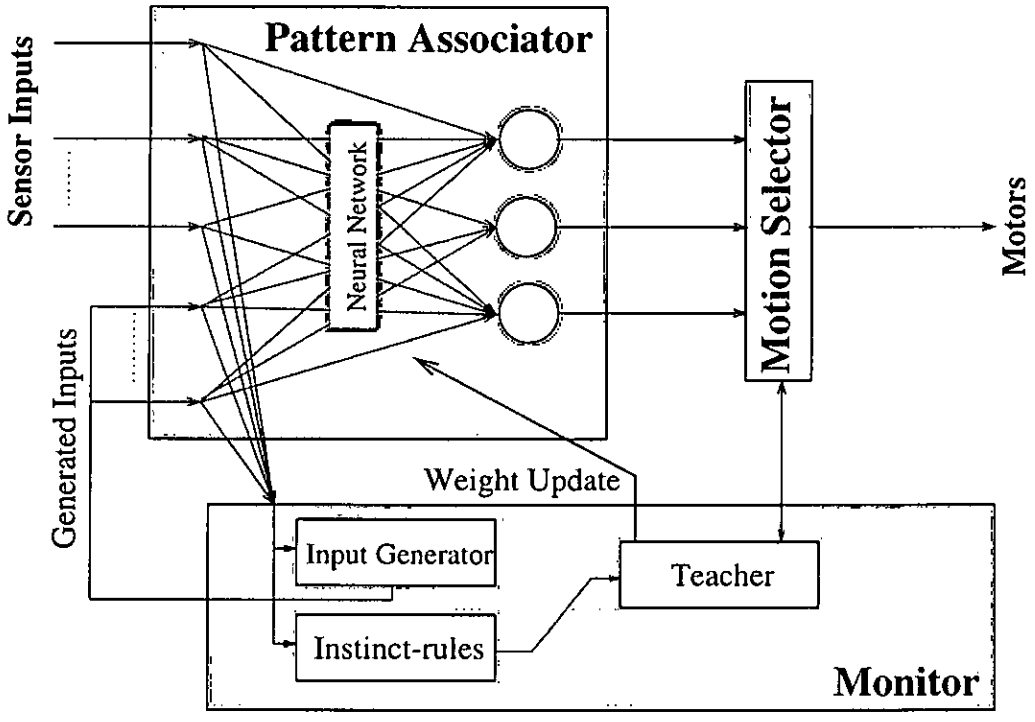


Figure 8–1: Architecture of Instinct-rule Robot Controller

8.3 Controller Architecture

The controller architecture used in Kryton was proposed and first implemented by Nehmzow in a robot called *Alder*[88,89,90]. The controller (shown in figure 8–1) consists of fixed and plastic elements. The fixed elements being the performance **monitor**, which contains the **instinct-rules** and the plastic, or adaptive, element the **pattern associator**; a single layer neural network. The adaptive behaviour of the neural network is under control of the **teacher** which responds to violations of the instinct-rules. The teacher acts in concert with the **motion selector** to train the neural network by trying alternative actions in an attempt to remove instinct-rule violations.

8.3.1 Pattern Associator

The pattern associator is the adaptive element which is trained to acquire the sensor–motor mapping that controls the robot. To satisfy the principle of rapid competence acquisition outlined in the previous section, adaption of the associator must be fast. For this reason a single layer perceptron (with linear output units) is used. This type of network was first used in the pioneering days of neural computation. Minsky and Papert in their seminal book

Perceptrons[80] proved that the fundamental limitation of such networks were that they could only solve linearly separable functions. Despite this, a single layer structure adapts more rapidly than its multilayer counterparts. To conform with the requirements of linear separability we must ensure that the sensory input linearly spans the input space. A demonstration of the linear separability of the basic problem can be found in [88].

8.3.2 Instinct-rules

To understand the function and basis of the instinct-rules, consider this definition:

> **Instinct** *n*:. . . complex and specific response on the part of an organism to environmental stimuli that is largely hereditary and unalterable though the pattern through which it is expressed may be modified by learning, that does not involve reason, and that has as its goal the removal of somatic tension or excitation.
>
> Webster's Third New International Dictionary 1981.

This, in essence, is what is required – a set of hardwired precepts which are used to judge performance of the *learned* associations between inputs (sensors) and output (motor actions). Consider the basic requirement of an autonomous mobile robot – obstacle avoidance; this can be encapsulated by the instinct-rule *"Keep crash sensors inactive"*. An urge to explore the environment can be instilled with the simple instinct *"Move forward"*. Basic competence acquisition was demonstrated on Alder using these instinct-rules.

8.3.3 Training Mechanism

The mechanism by which instinct-rule violations adapt the weights of the pattern associator is governed by the teacher and action selector (see figure 8–1). A flow chart of the operation of the teacher is shown in figure 8–2. When no rules are violated the action selector performs a winner-take-all function selecting the action of greatest activation of the pattern associator. If an instinct-rule is violated the teacher is activated and the winning action performed for a fixed period of time. If the violation is relieved in this time, the selected action is reinforced. If it is not, the teacher signals the motions selector to perform the action associated with the second strongest output of the network. This is performed for a slightly longer time than before to compensate for the action taken earlier. If this results in relief of the violated instinct-rule the network is trained to associate the initial sensor state to this output; if not, the next strongest action is selected and the process repeats.

This is the mechanism used in *Alder*. Later sections of this chapter will present certain enhancements to this initiated by the author.

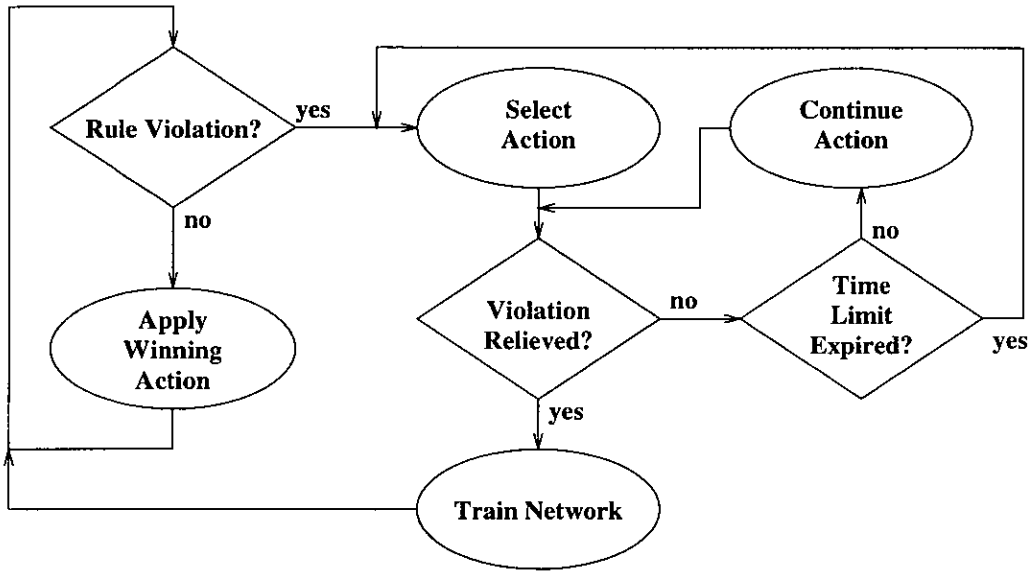


Figure 8–2: Flow Chart of Teacher Actions

8.4 Alder and Kryton

This hardware demonstrator has its basis, and is an extension of, Alder and Cairngorm, two robots developed by Ulrich Nehmzow[88]. These robots use the control architecture above and carried a variety of simple digital sensors. The simplest of these was a binary whisker which detected when a robot touched an object. Alder also used in some experiments an ultrasonic range-finder producing a ternary valued distance measurement.

When investigating possible demonstration applications the instinct-rule robot was selected for experimentation because:

- The single layer architecture offered an effective demonstration of the EPC hardware in action.
- Use of simple analog sensors could be used both to enrich the robots sensory environment as well as providing real-world analog input to verify this aspect of the EPC's operation.
- As a matter-of-course, this use of analog sensors raised issues of robotic control in relation to this control architecture. However it must be stressed that this work is primarily a demonstration of the hardware rather than an in depth study into robotics.

Table 8–1 presents a comparison between Alder and Kryton highlighting the differences between their sensory environment. The analog feelers of Kryton are constructed from a matched infra-red emitter and phototransistor. These are mounted at opposite ends of a flexible

	Alder	**Kryton**
Sensors	**digital:** 2 whiskers 1 forward motion **analog:** none.	**digital:** 2 crash sensors **analog:** 5 analog feelers 3 Light sensors
Neural Network	Software simulation	Hardware implementation
Computation	ARC52 micro-controller board with on-board BASIC interpreter.	EPC + 68020 board
Mechanics	Ficher Technic chassis Twin motor rear wheel drive Free rotating front castor	Technical Lego chassis Twin motor rear wheel drive Free rotating front castor

Table 8–1. Comparison of Alder and Kryton.

tube as shown in figure 8–3. As the tube is bent phototransistor current falls as less radiation reaches the phototransistor, whether directly or by internal reflection.

The response of one such feeler is shown in the graph of figure 8–3. These feelers can be seen mounted on the front of Kryton in the photograph of figure 8–4. The hardware platform for Kryton is an EPC under the control of a microprocessor board. The microprocessor board performs the functions of the monitor block and provides a communication interface to record the results of experiments. It is feasible that the monitors functions could be implemented within the FPGA of the EPC. This however was not attempted as the data logging functions of the microprocessor were central to the experiments.

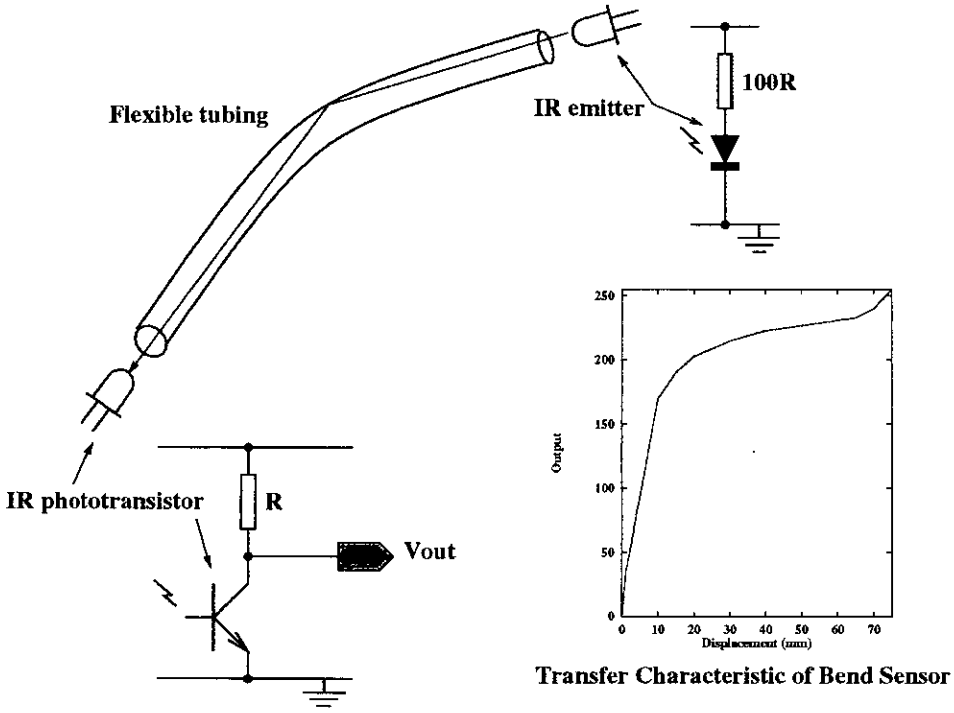


Figure 8–3: Analog Feelers used on Kryton

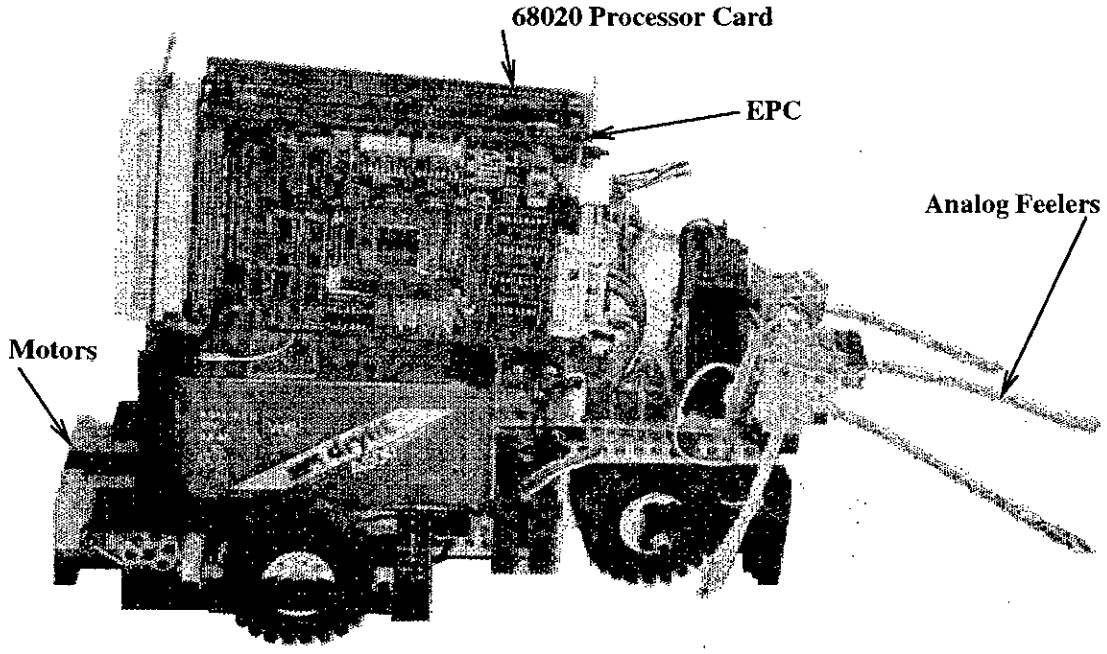


Figure 8–4: Photograph of Instinct-rule Robot

8.5 Controller Extensions

The primary motivation in using the EPC in the context of the instinct-rule controller was to interface directly to analog sensors. In doing so several issues were raised that resulted in extensions or improvements to the controller architecture. The following sections describe two such areas; the generation of additional input data to aid context detection and linear separability and the use of somatic tension as a measure of motor action performance.

8.5.1 Input Generation

The controllers task is to train the network to give a correct mapping between sensory inputs and motor actions. As mentioned before, the neural network architecture is a single layer one only capable of resolving inputs that are linearly separable. If inputs are not linearly separable the controller will *over-write* a previously learnt response to resolve the current situation. This has been noted in Alder where the robot was faced with a dead-end situation and learnt to turn always in one direction rather than the previous behaviour of turning away from the sensor that was excited[88,89]. Once out of the dead-end the robot had *forgotten* obstacle avoidance behaviour. This is not a particularly significant problem as the architecture adapts rapidly and the appropriate behaviour is re-learnt.

To minimise this over-writing of learnt responses there are some techniques that can be used. The obvious is to supply the network with inputs that can help to distinguish different situations or *contexts*. Nehmzow[88] demonstrated this by providing a *dead-end* signal manually as a network input. Alternatively, what is proposed here is that extra inputs are generated from the sensory input that may be useful in assisting the pattern associator to distinguish different situations. This practice is similar to the technique of providing a network with *hints*, defined by Abu-Mostafa[1] as *"auxiliary information about the target function that can be used to guide the learning process"*. It has been shown that networks can benefit from such hints, either as inputs or outputs, to speed or enhance training[1,38].

The first obvious *hints* are to give the robot some historical information. If the robot just hit a wall on one side for instance, it is likely to hit it on that side again soon. Thus historical data may provide an effective context hint. Other inputs are generated to assist in other behavioural goals, these are all derived from the input sensors and/or delays, and will be explained as they are used.

8.5.2 Somatic Tension

With Kryton's sensors being analog and continuous in nature, its sensory input is much enriched compared with the digital nature of previous robots such as Alder. With this transition into the

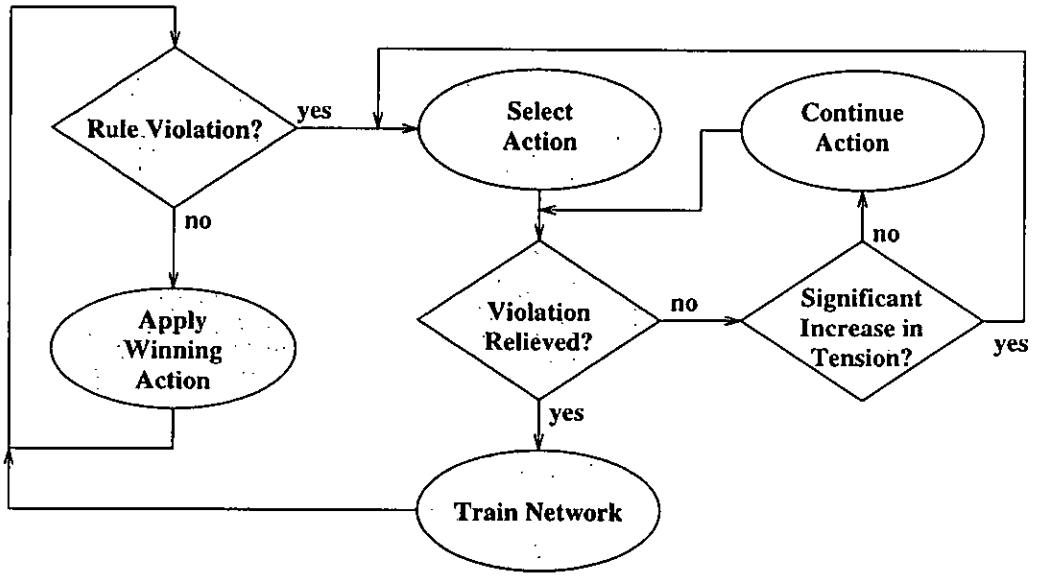


Figure 8–5: Flow Chart of Teacher Actions with Somatic Tension

analog domain other enhancements can also be made. The definition of instinct in section 8.3.2 leads to one such enhancement; paraphrased it says that the *goal of an instinct is the removal of somatic tension*. In Kryton's case a measure of somatic tension can be constructed as an aggregate of inhibitory sensors; that is the feelers. Such an aggregation or summation is natural to perform in the context of neural networks. It is achieved by using an extra linear neuron with fixed weights to provide a weighted summation of all feelers. This suggests a more effective way of gauging the correctness of alternative actions attempted by the teacher during instinct-rule violations: if somatic tension decreases it is likely that the action selected is a "good" one. If somatic tension increases it is unlikely to be "good". Utilising this concept, actions on Kryton are performed while monitoring somatic tension. The criteria (or threshold) under which an action is considered "bad" are relaxed as further actions are tried (or repeated). In this way the robot removes instinct-rule violations under minimal somatic.tension, that is it is less likely to damage itself (hit an obstacle) than with the previously used, purely time based, trial system. A revised flow-chart for the teacher functions incorporating the idea of somatic tension is shown in figure 8–5. The following experiments all use the concept of somatic tension to determine motor action performance.

8.6 Experiments in Competence Acquisition

The following sections describe the experimental work performed with Kryton. Three basic *behaviours* are investigated; obstacle avoidance, wall following and phototaxis. Obstacle avoidance is the most basic of competencies that enables the robot to stay operational by avoiding collisions. Wall following combined with obstacle avoidance gives the robot more complex objectives and would be useful in situations such as a domestic cleaning device. The experiments in phototaxis give the robot a navigational task, that is to move towards a light source.

8.6.1 Experiment: Obstacle Avoidance

The initial experiments implemented the most basic of competencies – obstacle avoidance. In these experiments Kryton used its feelers to learn mappings between its sensors and motor actions that avoided obstacles. At first this may appear a trivial task, but when performed under *real-world* conditions, or even the limited subset that is the laboratory, difficulties soon become apparent. Real sensors do not always perform as expected or designed. For instance the feelers used here may get stuck in gaps or bent into unforseen positions giving unexpected readings. One of the strengths of the instinct-rule controller is that complex behaviours are promoted by simple instinct-rules thus removing the need to pre-empt all possible contingencies.

Instinct-rules

The instinct-rules used in this experiment can be expressed as:

1. Keep feelers quiet.
2. Move forward.

The "keep feelers quiet" rule is triggered if any feeler exceeds a set threshold and promotes the obstacle avoidance behaviour. The "move forward" rule encourages the robot to explore its environment and is triggered if the robot has low sensory excitation (somatic tension) and is not moving forward.

Generated Inputs

It has been stated that to maximise the potential separability of the problem additional inputs are provided to the pattern associator. In these experiments the generated inputs consist of historical information and a low excitation bias input. The need for a bias input is obvious: if there is no sensory excitation all inputs to the network are zero and thus so are output states.

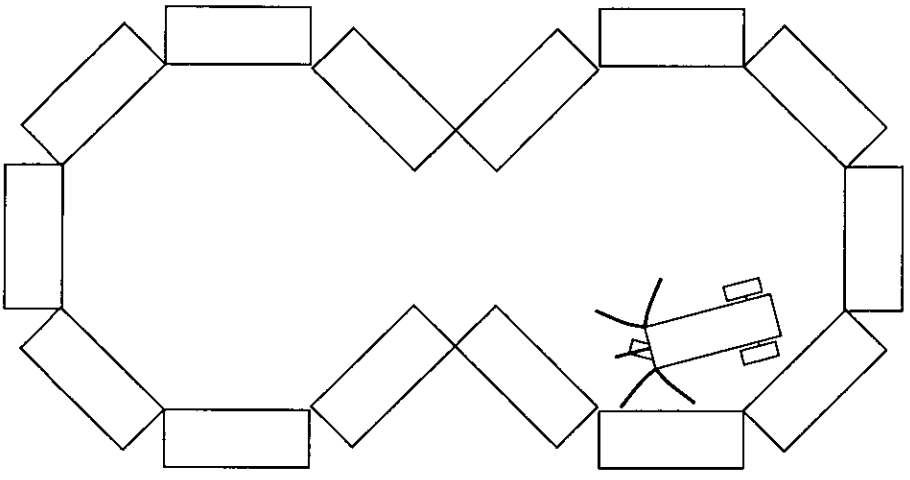


Figure 8–6: Obstacle Avoidance Experiment: Figure of 8

A constant high bias input is a possible solution to this. However to enrich the information content of this input, a low excitation bias is generated that is high if all feeler inputs are low (somatic tension low).

For each feeler, a record of activation is kept for twelve time-steps[1] the pattern associator is presented with three historical inputs for each feeler:

1. Previous time-step value $(t-1)$.
2. Maximum of time-steps $(t-2)$ to $(t-5)$.
3. Maximum of time-steps $(t-6)$ to $(t-12)$.

The effects of these generated inputs will be examined in the experiments below.

Experiment One: Figure of eight enclosure

The first results presented here were obtained by running Kryton in the enclosure depicted in figure 8–6. To gauge the performance and present results in a meaningful way the following statistics are uploaded from the robot every 100 time-steps:

1. Number of time-steps with rule violations per hundred samples.

[1]This is approximately 1 second in real time. The sampling rate of the controller is 12Hz. This is determined principally by the amount of data that is transmitted over the 9600bps serial link when logging the experiment. The rate is consistent with the physical time constants associated with Kryton' s motors. That is the time taken for a change in motor drive signal to manifest as change in motion.

2. Percentage of non-forward motor actions.

3. Number of new rule violations per hundred samples.

4. Number of new rule violations resulting in network training.

The nature of the system dictates that events happen in short bursts (such as coming into contact with an object) followed by longer periods of little activity (moving forward with no obstacles). To discover trends in activity the data was smoothed. Thus the data presented in the graphs in this chapter is the result of running window averages of fifteen data points. Figure 8–7 shows the trends in these statistics during a trial in the compound of figure 8–6. Kryton was

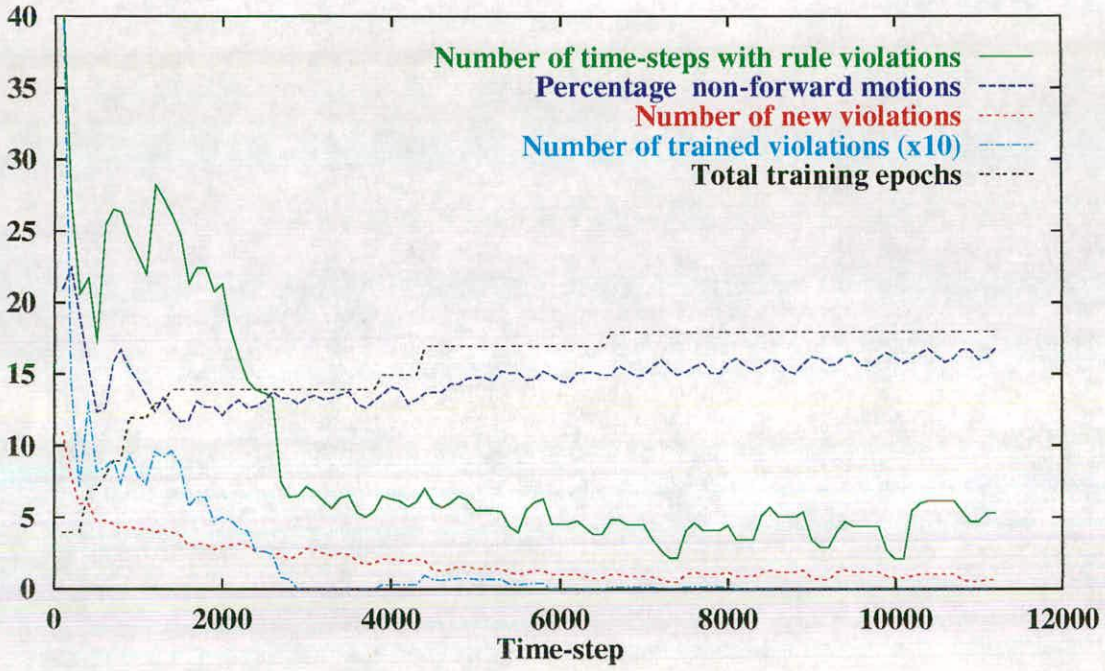


Figure 8–7: Obstacle Avoidance Experiment

initialised with random weights at time-step zero and to examine the training effectiveness of the controller, the monitor is disabled at time-step 7,000 such that the neural network directly controls the robot. The results plotted in the graph reveal the following about the experiment and controller:

- The number of time-steps with rule violations is initially high (green trace). This is as expected as there is only a random mapping between sensors and motor actions.

- The number of violations trained (cyan) starts high then rapidly drops away by time-step 3,000. This indicates that the network has been successfully trained to react to the situations the robot is encountering. Note that this takes only 15 or so training steps to achieve (black trace).

- The number of new violations (red) starts high and decreases to an essentially constant small number. As these violations are not resulting in training the network must already be performing the correct remedial action when the rule is violated.

- The similarity of the response before time-step 7,000 and after, together with the observed correct action, shows that the neural network has learnt the required mappings to successfully navigate this environment.

Summary

This experiment has shown:

- The EPC employed as a neural controller fulfilling the design goals of:
  - Direct interface to analog signals – sensors.
  - Direct interface to digital environment – the instinct-rule monitor and teacher.
- The instinct-rule controller was shown to promote simple obstacle avoidance behaviour with the modified somatic tension performance evaluation.

Experiment Two: Dead-end behaviour

The previous environment was relatively benign in that it had no tight corners or dead ends for the robot to negotiate. The neural network learnt a response that turned it away from any sensory excitation. If the robot approaches a dead-end or tight corner the physical positioning of the sensors means the response of some sensors must increase before the robot can turn out of the corner. Does the simple instinct rule of "keep feelers quiet" promote this behaviour? It was shown with Alder that it would; but once out of the dead-end had to *re-learn* the obstacle avoidance behaviour, indicating that the two situations, or contexts, were not linearly separable by the network.

This experiment confirms that the addition of the historical information can aid in providing contextual information. The robot was run in the environment pictured in figure 8–8. The statistics uploaded during one such experiment are depicted in figure 8–9. Like the previous experiment the robot was started in a random state and the monitor was disabled after time-step 10,000. The robot was trained to the correct behaviour where it would travel from end to end of the compound, avoiding the walls when moving between ends and turning out of the dead-end when encountering it. The graph of figure 8–9 shows the statistics of this experiment:

- The number of rule violations (green) and training epochs (cyan) is highest initially, dropping rapidly as basic competencies are acquired.

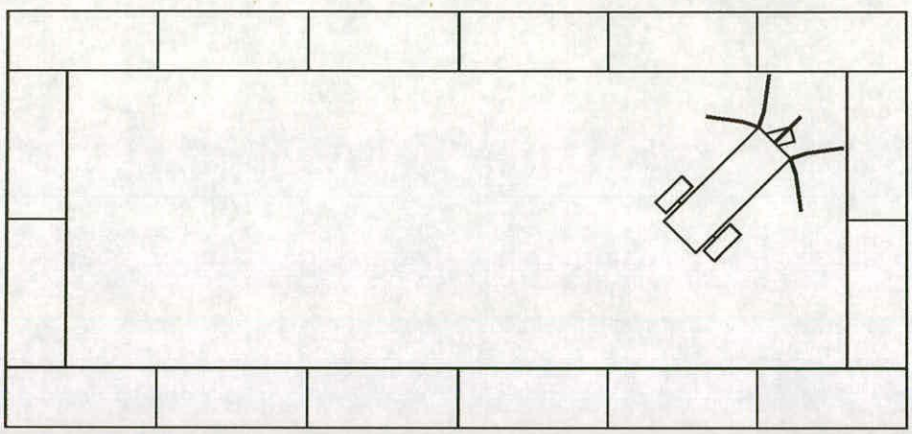


Figure 8–8: Obstacle Avoidance Experiment: Dead-end Enclosure

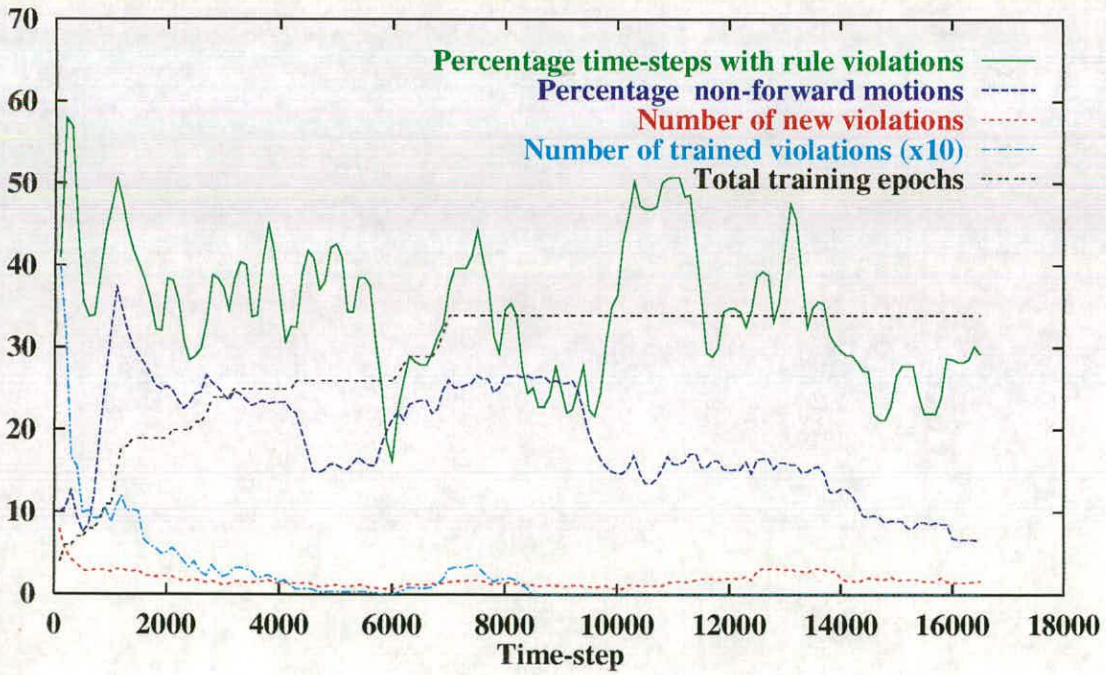


Figure 8–9: Dead End Experiment

- The majority of training is carried out in the first 3,000 time-steps in which approximately 30 training epochs take place (black trace).
- The number of violations (green) is not as close to constant as in the previous experiment. This is because the environment presents two distinct situations; the dead-end, where rules tend to be violated, and travel between ends where few rules are violated.
- The observed behaviour after the monitor is disabled proves that Kryton successfully learns both behaviours of obstacle avoidance and dead-end escape.

In this experiment Kryton evolves a mapping that recognises the *dead-end* context with the aid of history inputs. Figure 8–10 demonstrates the acquired behaviour – it shows the five general

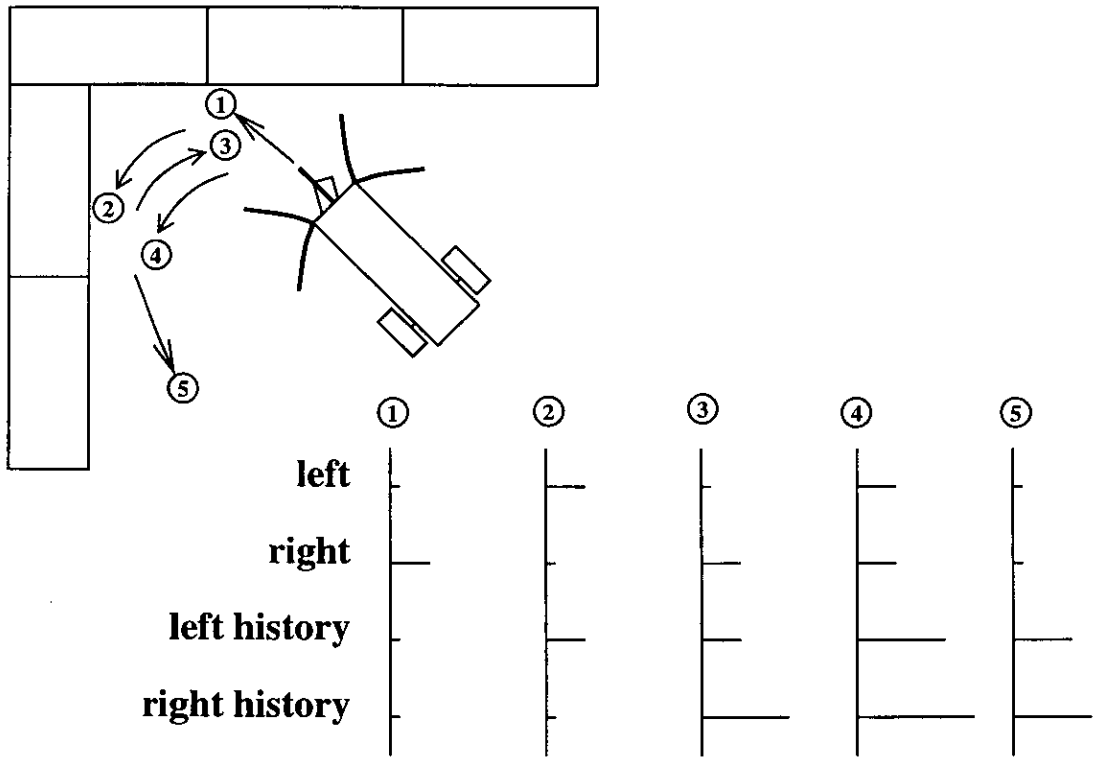


Figure 8–10: Behaviour in a Corner

movements that Kryton makes to detect and escape from a corner along with approximate sensor and history activations. When Kryton enters a corner ① it initially turns away (left ②) from the sensor activation. It turns until it encounters the wall on the left and turns right ③ to avoid this obstacle. By this stage a strong excitation has accumulated in the history information. This triggers action ④ which keeps the robot turning left until it gets low excitation ⑤.

Experiments conducted in other, more complex, compounds consisting of straight, convex and concave walls and corners produced the same ability to successfully avoid obstacles and get out of dead ends.

Summary

This experiment has shown historical data providing context information allowing Kryton to learn obstacle avoidance and dead-end escape behaviour from the simple "keep feelers quiet" instinct-rule. The mixed signal capabilities and the analog recovery feature of the EPC is used to mix analog sensor data and historical data in processing.

8.6.2 Experiment: Wall Following

This section investigates how an additional instinct-rule can be added to modify the behaviour of the robot. In this case an instinct-rule was added to promote a wall following behaviour.

Instinct-rules

The instinct-rules used in this experiment can be expressed as:

1. Keep feelers quiet.
2. Touch a wall.
3. Move forward.

The additional instinct-rule "touch a wall" is activated if in a certain period of time no sensor has become active. The violation is relieved when a sensor is activated or a time limit is reached where it is considered that the wall is lost.

Generated Inputs

In addition to the history inputs of the last section two additional types of generated input are used in this section:

1. A *find wall* input which increases every time-step that there is no sensor activation and is reset once a sensor is activated. The reasoning behind this is that the pattern associator will learn to associate this input with the "touch a wall" instinct.

2. Two long term memory inputs are generated to store on which side the robot last touched a wall. This is to allow Kryton to decide which direction to turn to touch a wall after history has decayed.

Results

Figure 8–11 shows an experimental run with Kryton in the same compound of figure 8–8. Kryton rapidly evolved a behaviour of turning back to the wall once all sensors left it and

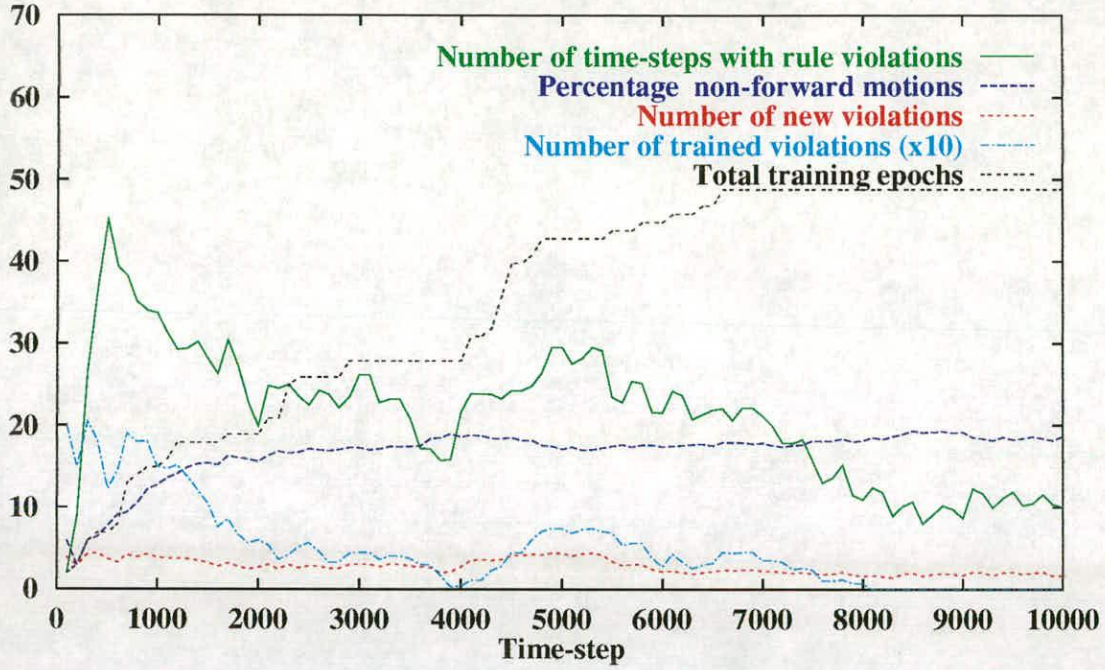


Figure 8–11: Wall Following Experiment

turning away from a wall when too close. This behaviour continued when the monitor is disabled after time-step 7,000. The analog nature of the sensors allows a *smooth* reaction to the problem – if sensors were digital as was the case in Alder, the robot in satisfying the "touch a wall" instinct-rule violates the "keep feelers quiet" rule, producing an oscillatory behaviour requiring constant intervention of the monitor block. With analog sensors this is not necessarily the case as the robot will move towards the wall before a "touch a wall" rule is violated and move away before a "keep feelers quiet" rule. – the pattern associator learns to pre-empt the instinct-rule violations. This can be seen in the fact that the number of time-steps with violations (green) of figure 8–11 is generally below 30. In the dead-end experiment it was rarely below 30. Also the percentage of non-forward motions (blue) is essentially constant as the robot smoothly follows the perimeter as against heading into corners in the dead-end case.

Summary

The EPC proved an effective processor for analog sensor inputs. Kryton was able to form a mapping between analog sensor, digitally generated data and delay signals that produced a wall following behaviour. The mapping was more effective than was the case in Alder as with analog sensors Kryton was able to pre-empt instinct-rules.

8.6.3 Experiment: Phototaxis

The basic behavioural tasks presented in the previous experiments had no navigational task associated with them. To be useful the robot must be able to carry out navigational tasks. In a subsumption architecture, navigational planning would be carried out be a higher level subsystem. The instinct-rule controller performs the basic low-level motor competency functions, yet must respond to directions from higher systems. This experiment demonstrates the ability of the instinct-rule controller to do just this by using an instinct-rule "maximise navigational signal". In this experiment rather than coming from a higher-level system, this navigational signal comes from a set of three light sensors to implement a light following behaviour – *phototaxis*. Figure 8–12 shows the arrangement and approximate response of the light sensors

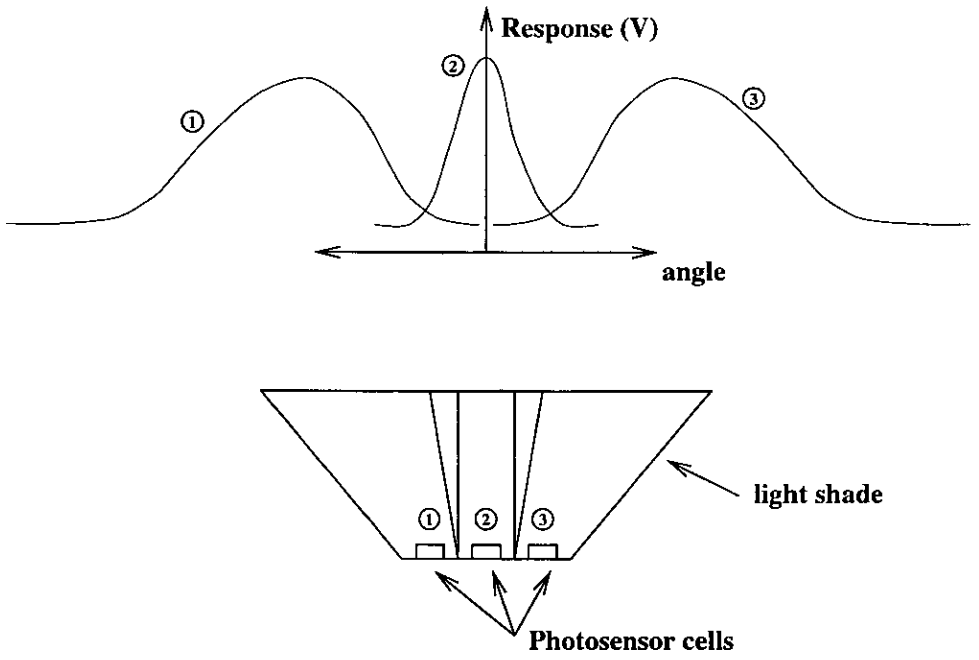


Figure 8–12: Photo-sensors for Phototaxis Experiments

which are mounted on top of Kryton. Outputs of these light sensors are fed as analog signals into the EPC.

Instinct-rules

1. Keep feelers quiet.
2. Maximise navigational signal.
3. Move forward.

In addition to the basic obstacle avoidance instinct-rules a new rule to maximise the navigational signal (photosensor ②) is added. The rule is violated when the output from photosensors ① or ③ is greater than photosensor ②.

Generated Inputs

Generated inputs are similar to the wall following ones:

1. A *find light* input which increases when there is no light sensor active.

2. Long term memory of last light sensor active.

Results

Figure 8–13 shows a typical experimental arrangement of obstacles and light source along with

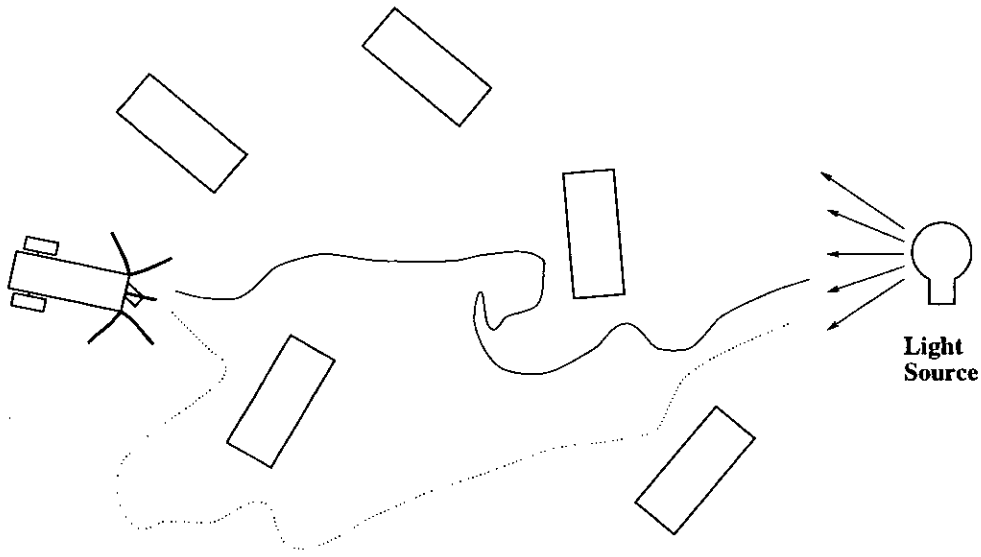


Figure 8–13: Example of Phototaxis Experiments

some observed paths of the robot. Experiments proved that the robot would lock and track the light source while avoiding obstacles. If the robot lost the light source in avoiding an obstacle it quickly learnt which way to turn to find the light again.

Summary

The important results from this experiment is that the instinct-rule controller will respond to directional signals while maintaining obstacle avoidance behaviour. This means that the EPC based controller can interact with higher navigational systems while autonomously handling the motor competency tasks.

8.7 Summary

The work presented in this chapter has placed the EPC into a real-world system. The EPC is performing in its primary mode of operation – as a processing interface between the analog domain and digital computing. It handles all analog input and presents this to the digital host. Along with this, neural processing is performed on the analog input along with inputs provided by the digital host. The results of this processing are made available as digital outputs. In the experiments presented here the EPC performed the vast majority of processing with the digital host idle or logging data over a serial line the majority of the time. This would allow the host to perform other tasks such as task planning and navigation, and it was shown that the instinct-rule controller could respond to such directional inputs.

The major difference between Kryton and its predecessor, Alder, was the direct interface to analog sensors. This prompted enhancements to the controller architecture such as the use of somatic tension to judge action performance. The mixed signal nature of the EPC allowed generation of additional inputs which enabled the robot to correctly differentiate situations such as obstacles and dead-ends. It also allowed finer control over competing tasks as in wall following where one instinct-rule promotes the robot to move towards a wall while another to move away.

Chapter 9
Discussion and Conclusions

9.1 Introduction

The objective of this thesis was to study the issues and practicalities of placing pulse stream neural hardware into applications. The issues raised by this work fall into three categories according to whether their effects are manifest at a VLSI, systems or applications level. This concluding chapter discusses the issues in these three categories and draws conclusions as to the success of the work and the future of hardware neural computation.

9.2 VLSI Issues

The investigation of the VLSI aspect commenced with a review of application based neural VLSI in Chapter 3 and showed no obvious leader in implementation methodology for neural VLSI. The benefits of pulse stream neural computation were particularly strong for the defined primary application area of analog/digital interface, due to its ability to provide mixed signal inputs and outputs suitable for direct digital processing.

The focus provided by the requirements of applications led to several VLSI design enhancements, principally mixed signal input structure and analog signal recovery. The performance of the EPSILON cells was also enhanced by layout improvements and refinements to the architecture. In particular the improvement in the distribution of synaptic power supply revealed operational characteristics of the synapse array that were masked in the original EPSILON chip.

The results gathered testing and characterising the EPSILON II device, together with the practical experience gained in putting the hardware to use, raised two major areas that concern future VLSI improvement:

1. Minimisation of neuron offsets through reduced amplifier input offset voltage. This in turn relates to the performance of the bias generation schemes.

2. The need for controlled variation of weight dynamic range.

9.2.1 Minimisation of Neuron Offset

In Chapter 5 variation of offsets between synaptic columns was quantified and simulations were used to illustrate the dominant role played by the distributed feedback amplifier's input offset in this variation. Here lies a practical consideration in using the distributed feedback synapse: it is very sensitive to supply and mid-point voltage variations. In fact these two factors are virtually equivalent as they are both manifest as unequal drain-source voltages across the transconductance pair. This leads to the practical consideration that both power supply variation and amplifier offset must be minimised for optimum operation. This problem also effects the bias generation scheme used for the distributed feedback synapse. If input offset is not minimised these references are unreliable. A scheme for minimising these offsets was presented in Chapter 5.

Despite the problems of neuron offsets and gain variations, chip-in-loop learning has been demonstrated in this work, and elsewhere, as being capable of compensating for analog inaccuracies. Though the presence of non-idealities prevented a software evolved weight set from providing a good solution in hardware, only relatively few training steps were required on such a weight set to *trim* it to a good solution. Thus the issue of minimising neuron offset is one of compatibility of weight sets between chips and software simulations to minimise training times, along with reliable generation of on-chip references.

9.2.2 Weight Dynamic Range

Chapter 7 highlighted the importance of restricted dynamic range in the weight set. It was found that while limitations in analog precision are acceptable in the feed-forward path, some control over the dynamic range is needed to allow solutions to a wide variety of problems. Two methods of dynamic range variation were presented for the pulse stream hardware:

1. Scaling of input pulse-widths to produce a dynamic range trade-off against speed and noise level.

2. Scaling of output ramp waveform to produce an accuracy/dynamic range trade-off.

For a practical learning system an automatic weight scaling and dynamic range adjustment scheme would be advisable. Such schemes have been proposed by Myers *et al*[87]. The above dynamic range enhancement methods were shown to have limitations or adverse effects in increasing noise and reducing network computation rate. To gain further control over dynamic range a third, circuit level, method could also be incorporated into future designs. This would involve adjusting the synaptic gain, which is set by the width-to-length ratios of the synapse

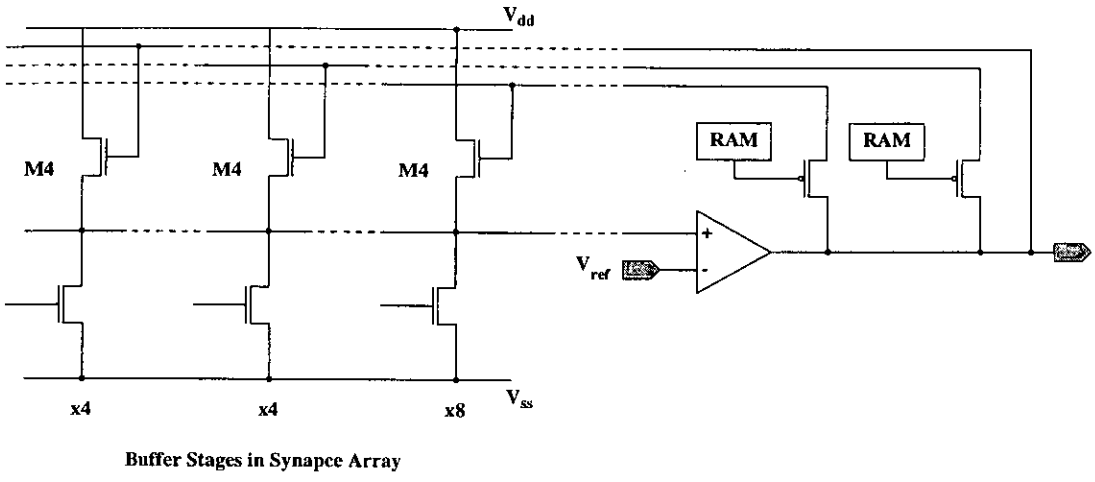


Figure 9–1: Gain Variation for EPSILON Distributed Feedback Synapse.

transconductance pair and the buffer transconductance pair:

$$\text{synaptic gain} = \frac{\beta_{\text{syn}}}{\beta_{\text{buf}}} = \frac{\left(\frac{W}{L}\right)_{\text{syn}}}{\left(\frac{W}{L}\right)_{\text{buf}}} \tag{9.1}$$

Thus reducing the width-to-length ratio of the buffer stage, equivalent to using less buffer stages, would increase the gain and thus dynamic range of the weights. Local digital storage in the neuron can control how many buffer stages are switched into the feedback path, the basic idea is shown in figure 9–1. The advantage of this arrangement is that it adds little complexity (thus area) to the synapse as no extra transistors are required, just extra routing. Apart from gaining extra control over dynamic range, this method does not entail significant negative trade-offs as the other methods presented in the thesis did.

9.2.3 VLSI Summary

Upon investigation of the applications area and system level needs, it was concluded that pulse stream methodology was the most suitable implementation strategy due to its:

- Mixed analog/digital domain input capabilities.
- Digital compatible outputs.
- Cascadable structure with digital communication.
- Simple, robust and compact synapse design.
- Existing technology available within the Edinburgh research group.

The chip presented, EPSILON II, represents an advance with regard to previous pulse stream neural chips in that it provides:

- Better performance achieved by layout improvements.
- Additional features for applications use such as:
  - Programmable input modes.
  - analog signal recovery mode.
  - simplified control strategy.

Amplifier offset was highlighted as a major source of non-ideality, and techniques to minimise this were presented for future designs. With limited weight dynamic range highlighted as a practical limitation on network capabilities, an electronic gain variation scheme was outlined.

9.3 System Level Issues

As part of the literature review, a variety of hardware neural systems were studied in order to illustrate system level issues. The most successful of these implementations showed a tight coupling between input structure and input domain. Examples of this were the direct analog sampling in the Kakadu system and the direct optical input of the Synaptics OCR. Similarly, matching between network outputs and output domain, which is invariably a digital system, was a factor in the most successful systems. Conversely, designs where large system overheads were needed showed the drawbacks of inappropriate use, one such example would be the ETANN chip in a digital environment requiring hundreds of D/A channels[61]. The same work also described problems in providing low noise analog busses to cascade ETANN chips.

The EPSILON/FENICS system, along with the ETANN systems discussed, demonstrate how data conversion can limit the data throughput of the system. In these cases conversion was slower than device computation. For EPSILON/FENICS, the host processor had to perform substantial processing to recover results and for ETANN, the large A/D overhead limited the computation rate. The following factors thus emerged as pointers for successful applications oriented neural hardware:

- Autonomy of operation from the host system is essential. For pulse stream systems this involves a system incorporating:
  - Chip support operations such as weight refresh and ramp generation.
  - Data conversion to and from host system data representation.
- Effective data interfaces to domains of operation, that is:

  - Analog real-world data implies the need for analog inputs.
  - Data input from a digital host requires digital inputs.
  - Chip operation is pulse stream so pulse stream communication is necessary to cascade chips.
  - The output domain is a digital host so digital outputs are required.

- Flexible structure to allow a variety of problems to be implemented.

The EPSILON Processor Card embodies these principles to provide a framework for prototyping applications. The strengths of the EPC are derived from the:

1. Parallel and autonomous nature of support functions allowing real-time operation.
2. Flexibility of its I/O structure.
3. Ability to customise internal digital processing.

Support functions and real-time operation

All chip support functions were performed at system level and parallel pulse conversion techniques were used to minimise data bottle-necks; the major source of performance degradation in previous systems. In an optimum configuration the EPC can operate at the top speed of the EPSILON II chip – a single layer computation cycle of $40\mu s$, or two layers in $60\mu s$. Some configurations of input and output (e.g. full digital input and output) require a serial pulse conversion approach which slows throughput somewhat.

Comparisons with software networks are dependent on the application and input structure. For instance a 66MHz processor could perform a two layer MLP computation at a similar rate of $65\mu s$[1]. Yet if inputs are analog then a significant A/D conversion overhead is required, both in hardware and computation time. Here lies the advantage of the hardware implementation: A/D conversion and neural processing are performed in the same step. This is the key factor promoting hardware use and development: if the input domain is the analog world there is significant advantage in using analog processing because the interface requirements are inherent in the hardware.

Input/output flexibility

The second key area in which the EPC improves on previous neural systems is in the flexibility of its I/O structure. The EPC can provide parallel channels for analog data directly to the chip.

[1] Assuming a multiply accumulate takes two clock cycles and a non-linear threshold four: total cycles $= 1024 \times 2 \times 2 + 32 \times 2 \times 3 \Rightarrow 65\mu s$ compute time. In reality program and memory access overheads would make this longer.

It can also configure inputs to process digital data from the standard bus to provide mixed signal processing. One extreme of this is the fully digital input configuration, though this is not seen as an efficient mode of operation when compared to a dedicated digital solution. The other extreme is fully analog input where the EPC performs as a processing A/D converter. In between, the mixed signal architecture allows a powerful tool to fuse analog and digital data. No other system examined has this flexibility of architecture, especially important is the ability to deal with mixed signal inputs.

The output of the EPC converts pulse stream outputs to digital values, these are presented to the host via a standard digital bus. It can also *recover* analog inputs and supply the host with digital conversions of these. Cascading chips or broadcasting inputs is achieved with a dedicated pulse stream bus which provides a relatively noise immune method of transferring neural states. Again no system examined presents digital output along with providing parallel, noise tolerant cascading ability.

Customisation

The use of FPGA technology for the digital support processing allows a great deal of flexibility and customisation for different applications. The digital support can be optimised to the highest degree of parallelism for the data and network structure of the applications. This is in contrast to providing support for all possible I/O variations which would be prohibitive with standard logic. Added to this is the advantage of being able to implement custom digital processing on the EPC itself; such as winner-take-all functions or delay loops for inputs. Spare FPGA pins can also be used for digital control purposes in the application, this was done for motor control in the instinct-rule robot demonstrator. This ability to match neural system to application has not been found in previous hardware neural systems.

Future system level work

Future advances in system level work is largely dependent on VLSI advances. The EPC architecture is ideal for applications such as remote sensor monitoring or control applications; for the EPC to be economic in such a role a near two chip solution is required. This implies substantial integration of analog functions onto the neural chip. Most of these are achievable, for instance:

- Ramp and weight refresh D/A converters.
- Bias and reference generation circuitry.
- Synaptic power supplies.

The largest system overhead is weight memory which implies that the ideal solution would involve on-chip non-volatile analog memory. Until technology provides such an analog solution external digital storage is required. For the EPC the problem of volatility of weights can be alleviated by storing the weights in the on-board EEPROM.

9.4 Applications Issues

Part III of the thesis investigated the practicalities of using the EPC. In Chapter 7 chip-in-loop learning with back-propagation was investigated and the limits of network ability studied. The study commenced using an artificial character recognition problem. The lessons learnt from this were then applied to a series of real-world problems. In Chapter 8 the EPC was placed in a real-world system, an autonomous mobile robot, to study its performance as a processing analog interface. The rest of this section discusses the issues raised by these studies.

9.4.1 Effects of Analog Non-Idealities on Back-Propagation Learning

The major areas to arise from the investigation of training the EPC chip-in-loop with back-propagation learning were:

1. Learning rate limitations of hardware.

2. The effect of limited dynamic range and precision on learning ability and performance of the hardware.

3. Pointers to the types of problems suitable for hardware neural solutions.

Learning rate

In the course of the investigation it was found that the hardware was inherently slower to train than an ideal software network because the learning rate parameter, η, required for stability was much lower than could be used in software. The principal reason for this was that a higher η promotes larger weights which become clipped in the bounded weight representation of the hardware, too high and the learning became unstable. Also the effects of analog non-idealities introduce inaccuracies in the error function that meant the training algorithm took longer to reach a low error solution. The best method to reach a good solution was found to be training a software model of the hardware which took into account the limited precision and dynamic range of the weight set then downloading this weight set to the hardware and *trimming* it by further training.

Table 9–1. Summary of Comparative Performance of EPC to Software Network

Problem	Difference in generalisation performance	Increase in mean square error
Character recognition	2.5	+797%
ATM link admission	3.4	+171%
Speaker identification	3.7	+32%
Medical data analysis	9.1	+9.7%
Region classification	29.8	+54%

Dynamic range and generalisation performance

Both finite precision and dynamic range of the hardware were found to affect network performance, though dynamic range was found to have greater practical ramifications. While precision in the weight set determined an absolute limit in network accuracy, variation of the dynamic range of the weights was shown to be essential in maximising the use of this precision. Experiments showed that the performance of low precision analog hardware will never match that of high precision software. For instance, the hardware non-idealities result in a significantly higher final training error for a problem. However, this was not found to translate directly as a severe degradation in generalisation ability. This is demonstrated in table 9–1 which summarises the results of the five problems presented in Chapter 7. The first column tabulates the difference between the percentage of correct classifications of the generalisation set for the floating point software network and the EPC hardware network. The second column shows the percentage increase in the mean square error of the hardware network compared to the software network[2]. As can be seen the hardware performed to a reasonable level when compared to the floating point software for all but the region classification problem.

The initial experiments for the medical data problem showed poorer performance which prompted an investigation into the effect of dynamic range on the solution to this problem. A series of software simulations demonstrated that increasing the dynamic range could produce a solution approaching that of the floating point network. This was attempted on hardware by scaling the inputs to increase dynamic range. This produced an improvement in performance but the corresponding increase in noise and the effects of gain and offset variation limited the effectiveness of this method.

These experiments again highlighted the importance of limited dynamic range on network performance and this is perhaps the major result of this thesis as weight dynamic range is a limited hardware resource dependent on the physical characteristics of the design. Careful

[2]The figure for the character recognition problem is from the training set while the rest are calculated from the generalisation test sets.

design is needed to maximise weight dynamic range without adversely affecting noise tolerance, speed and power consumption. Note that this is a practical issue relating to hardware only – such scalings of dynamic range can be easily achieved in software networks without introducing errors as long as sufficient *headroom* is available in the calculations. The role of dynamic range in hardware performance as demonstrated by the experimental work of this study has not been seen reported in the literature.

The next section discusses the issues of problem suitability that arose from the experiments of Chapter 7.

Suitability of problems

The experiments in Chapter 7 demonstrated the type of problems most suited to hardware implementation. The 1-of-N classification problems, bar the region classification problem, performed well as the output units, being sigmoidal, suppressed the effects of noise. The effects of noise were much more prevalent where the network is in the role of a function approximator with linear outputs such as the ATM problem. Though the hardware performed satisfactorily in classification, the larger error may mean its value as a function approximator is limited.

The region classification problem demonstrated that there is a limit to the difficulty of problems that can be solved with the hardware. This problem required higher resolution than was available to differentiate very close input vectors. This highlights that problem selection, or equally input data selection, must be examined to determine suitability of the problem for hardware implementation. The simple software models of the hardware presented in the thesis offer a convenient first step in determining problem suitability.

9.4.2 Kryton – The EPC Processing Analog, Real-World Data

Chapter 8 presented the instinct-rule robot, Kryton. The aim of this investigation was to place the EPC in a real system and utilise its ability to interface directly to analog data. The problem represents a class of potential applications where analog sensor data must be interfaced and processed to perform a control task. The instinct-rule application demonstrated the ease of use of the EPC; it was possible to wire Kryton's *analog feelers* directly to the EPC analog input bus and three unused digital outputs were used as control signals to motor actuators.

While it is difficult to make a direct comparison between Kryton and the software controlled exemplar Alder from which it was modelled, as Alder used digital or ternary sensors and a software neural network. However the use of real-valued sensor readings did lead to refinements in the control architecture which were shown to have such effects as fewer actual contacts with obstacles and a *smooth* action in wall following.

The mixed signal architecture of the EPC was also utilised in interaction with the instinct-rule controller. Here additional inputs conveying time dependent or historical information were presented to the network which allowed the robot to make context dependent decisions.

As a demonstrator Kryton performed well and was the only example found in the literature of dedicated neural hardware controlling an autonomous vehicle.

9.4.3 Future Work – Applications

An obvious area for future work in the applications field is to use the EPC to prototype further real-world problems such as sensor monitoring tasks, local control problems and intelligent A/D. The demonstration applications presented in this thesis show that the EPC is suitable for prototyping such problems. The analog recovery ability enables sampling of analog inputs of an application to build a data–set for training. The software models developed in Chapter 7 allow the developer to predict likely performance of the EPC and gauge problem suitability. These software models also allow the off-line development of weight sets that may be quickly trimmed on the hardware.

Chapter 7 raised issues concerning the ability to train hardware neural networks. Further work here would be to examine the performance of other MLP learning algorithms.

The issue of weight dynamic range could be further investigated by studying its effects on other problems and determining the limits to which increasing dynamic range is effective.

9.5 Overall Conclusions

To make the transition from laboratory research to real-world use and acceptance, hardware neural networks must fill an applications niche where they outperform existing methods or for which no present method is available. In evaluating potential applications areas, points to consider are:

- Most neural applications will be served optimally by fast, generic digital computers. This is because data is in digital form and conventional computing continues to improve in performance which quickly outpaces the performance of a dedicated hardware solution.

- Analog neural VLSI is applied optimally at the interface between the real world and higher-level digital processing. It is here that the ability to deal efficiently with analog inputs can give neural technology an edge.

To operate in this area it is essential that neural hardware:

- Interface directly to analog inputs.

- Deal equally well with digital inputs.
- Provide output in a host readable form.
- Operate without consuming host resources

The EPC proved an effective solution to most of the problems studied in the thesis and demonstrated a high degree of flexibility and ease of use:

- The EPC was demonstrated solving 1–of–N classification problems such as the speaker identification and medical data problem. These problems represent a class in the applications area where real-world analog signals must be classified and outputs made available to a digital host.
- The limits of the EPC's performance was probed with the region classification problem where it failed to provide a good solution. This demonstrated a class of problem unsuited to analog hardware implementation: those with high resolution input data and sharp decision boundaries requiring high resolution weights.
- The success of Kryton demonstrated the advantages of the EPC in interfacing to analog signals. Kryton represents a broad class of problem where analog and digital signals are fused in neural processing.
- The combination of neural VLSI combined with the digital support provided by an FPGA allowed the EPC to interface easily with digital systems. This gives the EPC great flexibility for prototyping applications as custom digital control can be implemented on the board.

The above points demonstrate that analog pulse stream neural processing, embedded in a support system can provide an effective solution to real-world problems.

Appendix A
EPSILON II Chip Details

A.1 Layout Plot Legend

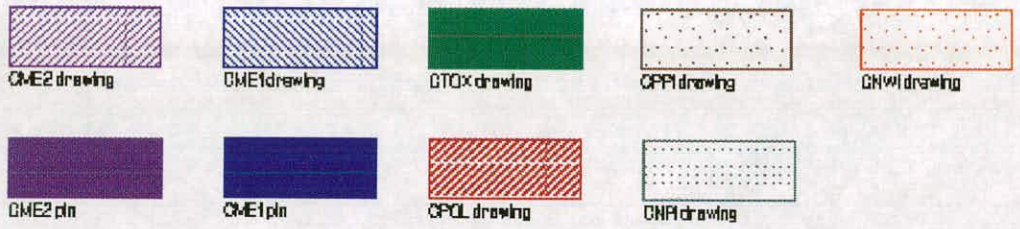


Figure A–1: Key to Geometrical Layers in Layout Plots

A.2 EPSILON II Input Neuron SRAM Cell

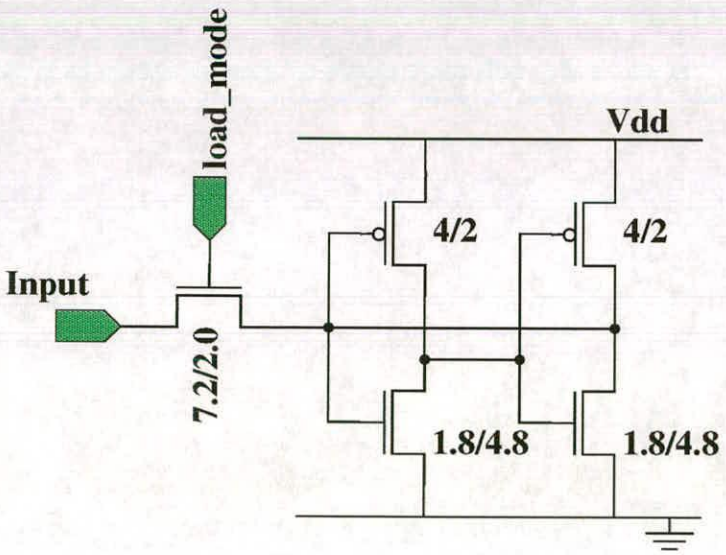


Figure A–2: EPSILON II Input Neuron SRAM Circuit with Transistor W/L Ratios

A.3 EPSILON II Input Neuron Cell

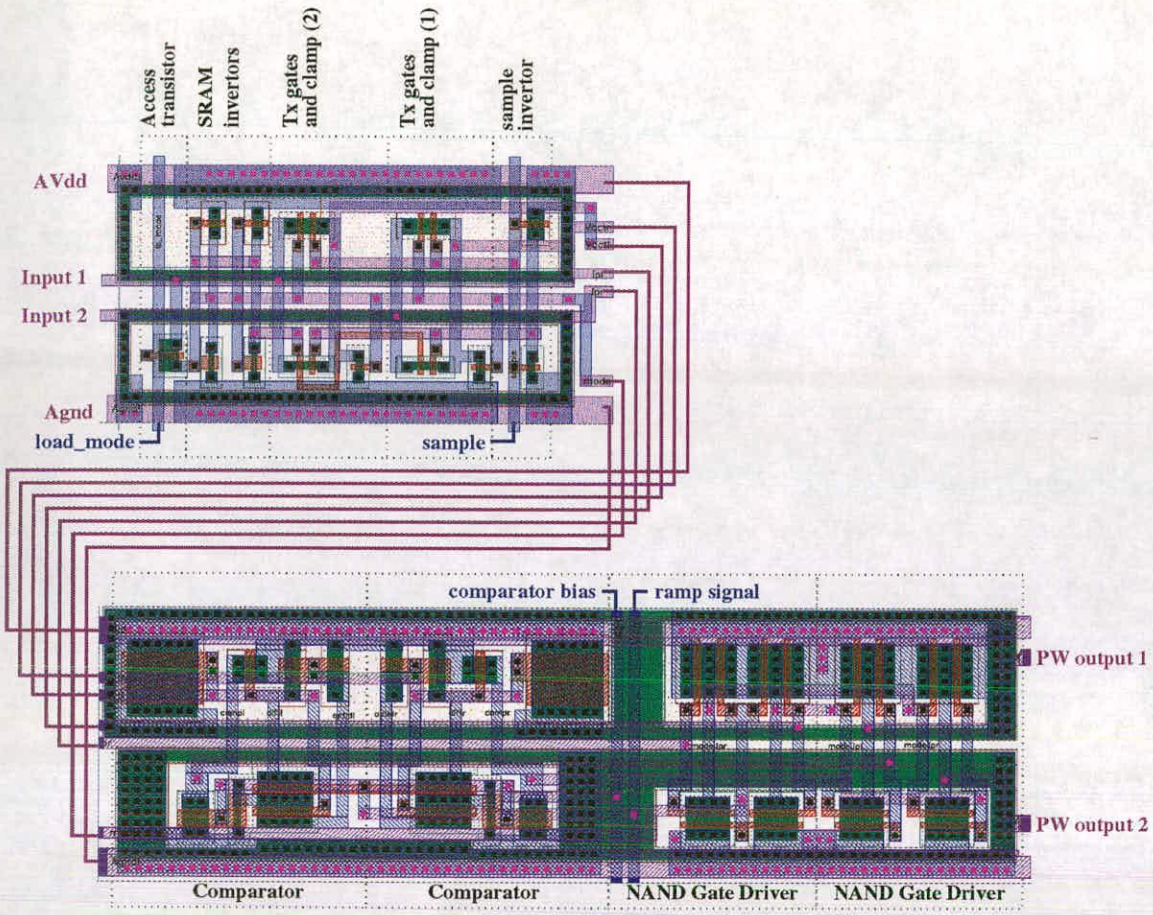


Figure A–3: EPSILON II Input Neuron

A.4 EPSILON II Synapse Cell

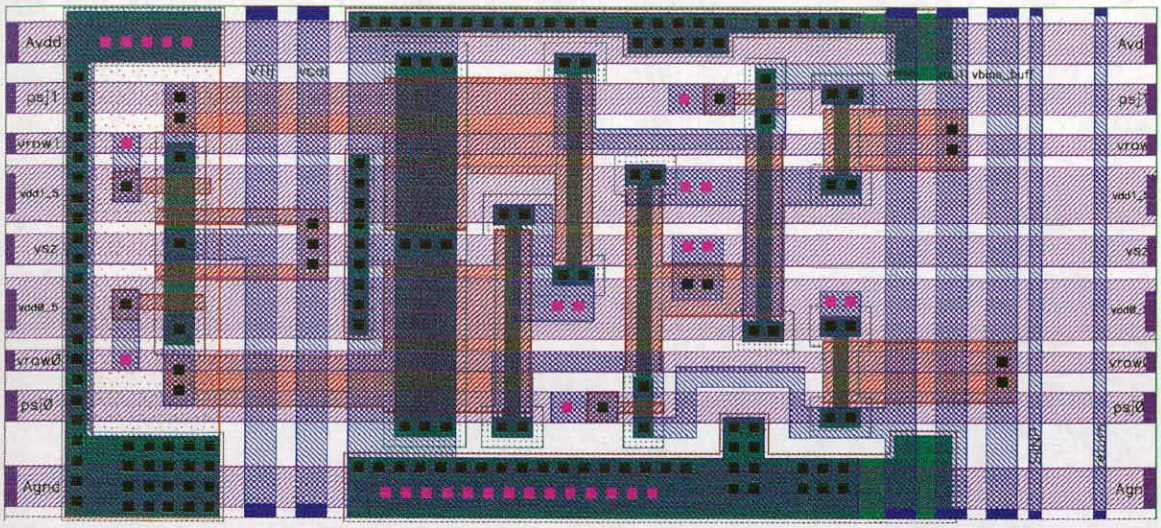


Figure A–4: EPSILON II Synapse Cell

A.5 EPSILON II Output Neuron

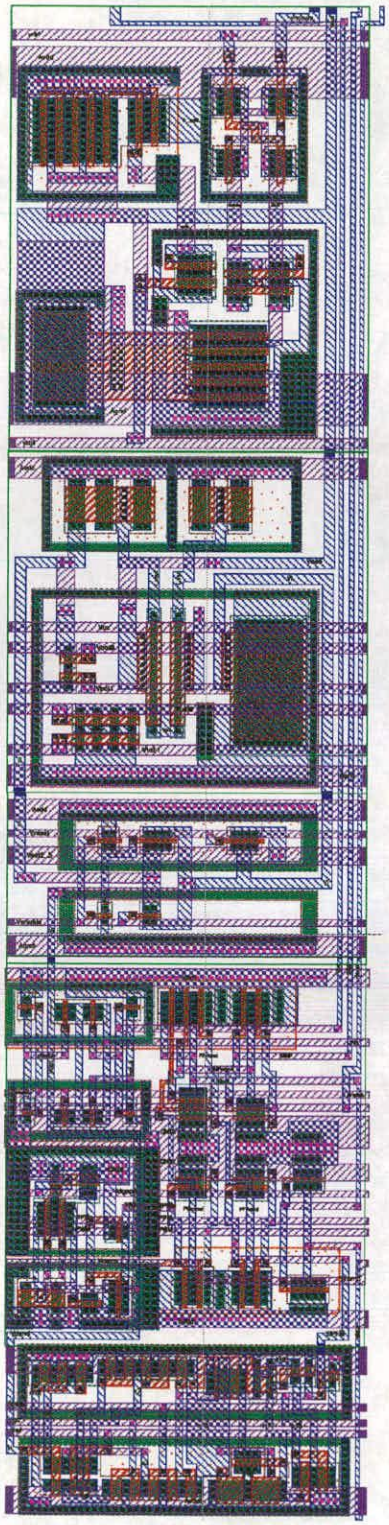


Figure A–5: EPSILON II Output Neuron Cell

A.6 EPSILON II Shift Registers

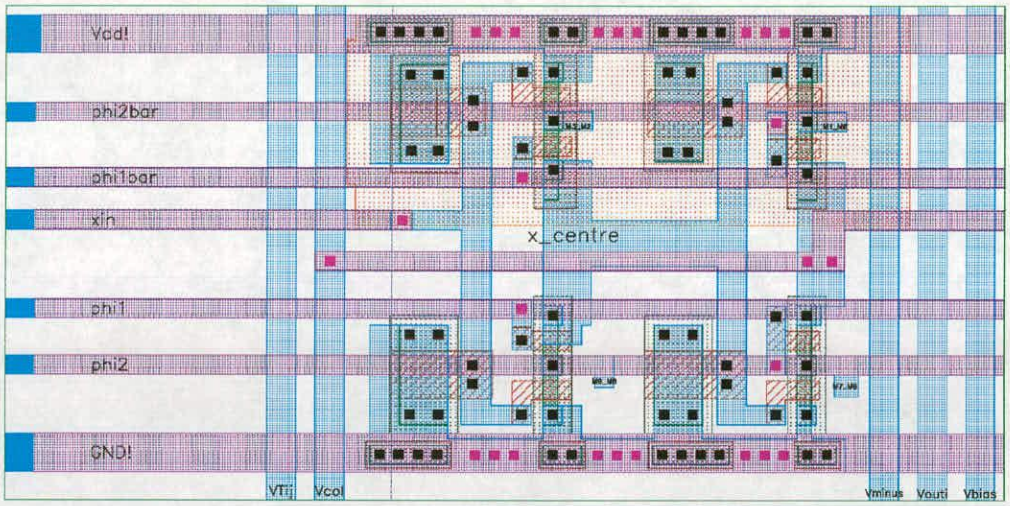


Figure A–6: EPSILON II X-Shift Register

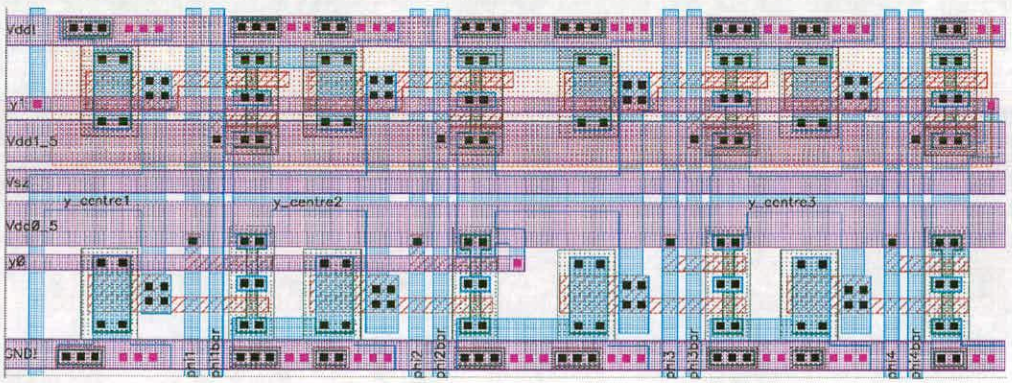


Figure A–7: EPSILON II Y-Shift Register

A.7 EPSILON II Pin Out

Description	Name	Pad #	PGA	SCM
Network Input	IN28	1	C3	29
Network Input	IN27	2	B2	15
Network Input	IN26	3	B1	14
Network Input	IN25	4	D3	42
Network Input	IN24	5	C2	28
Network Input	IN23	6	C1	27
Network Input	IN22	7	D2	41
Network Input	IN21	8	E3	48
Network Input	IN20	9	D1	40
Digital Ground	GND	10	E2	47
Digital Supply	Vdd	11	E1	46
Network Input	IN19	12	F3	54
Network Input	IN18	13	F2	53
Network Input	IN17	14	F1	52
Network Input	IN16	15	G2	59
Network Input	IN15	16	G3	60
Network Input	IN14	17	G1	58
Network Input	IN13	18	H1	64
Network Input	IN12	19	H2	65
Network Input	IN11	20	H3	66
Network Input	IN10	21	J1	70
Network Input	IN9	22	J2	71
Network Input	IN8	23	K1	76
Network Input	IN7	24	J3	72
Network Input	IN6	25	K2	77
Network Input	IN5	26	L1	82
Network Input	IN4	27	M1	95
Network Input	IN3	28	K3	78
Network Input	IN2	29	L2	83
Network Input	IN1	30	N1	108

Table A–1. EPSILON II Pin out part I

Description	Name	Pad #	PGA	SCM
Network Input	IN0	31	L3	84
Autobias Input	In_bias	32	M2	96
Input Ramp	Vramp_ip	33	N2	109
Digital Ground	GND	34	L4	85
Analog Ground	AGND	35	M3	97
Digital Supply	Vdd Pry	36	N3	110
Mode select	ld_mode	37	M4	98
Sample input	Vsample	38	L5	86
Neuron Output	OUT0	39	N4	111
Neuron Output	OUT1	40	M5	99
Neuron Output	OUT2	41	N5	112
Neuron Output	OUT3	42	L6	87
Neuron Output	OUT4	43	M6	100
Neuron Output	OUT5	44	N6	113
Neuron Output	OUT6	45	M7	101
Neuron Output	OUT7	46	L7	88
Neuron Output	OUT8	47	N7	114
Neuron Output	OUT9	48	N8	115
Neuron Output	OUT10	49	M8	102
Neuron Output	OUT11	50	L8	89
Neuron Output	OUT12	51	N9	116
Neuron Output	OUT13	52	M9	103
Neuron Output	OUT14	53	N10	117
Neuron Output	OUT15	54	L9	90
Enable integrator	Venable	55	M10	104
Reset integrator	Vreset	56	N11	118
PF phase filter	rfp	57	N12	119
PF gain filter	rfg	58	L10	91
PF gain VCO O/P	vcog	59	M11	105
PF phase VCO O/P	vcopw	60	N13	120

Table A–2. EPSILON II Pin out part II

Description	Name	Pad #	PGA	SCM
Digital Ground	GND Pry	61	L11	92
Analog Supply	AVdd	62	M12	106
N/C	N/C	63	M13	107
Digital Ground	GND Pry	64	K11	79
1v0 synapse reference	1v0ref	65	L12	93
Output ramp	Vramp_op	66	L13	94
Phase PLL in	rip	67	K12	80
Phase PLL out	rop	68	J11	73
Gain PLL in	rig	69	K13	81
PLL reference current	ipg	70	J12	74
Gain PLL out	rog	71	J13	75
PLL reference voltage	vig	72	H11	67
Integrator O/P zero	Voz_int	73	H12	68
Weight load	Vwt	74	H13	69
0.5V Supply	0v5	75	G12	62
1.5V Supply	1v5	76	G11	61
Vsz generator	VszOUT	77	G13	63
Voz generator	Voz_out	78	F13	57
syn buffer bias	vbias	79	F12	56
2.5V reference	Vset2v5	80	F11	55
Zero weight reference	Vtijz	81	E13	51
PWM Comparator reference	Iref_PW	82	E12	50
Int Balance I	Ibal_int	83	D13	45
Int tail I	Itail_int	84	E11	49
Synapse zero ref	Vsz	85	D12	44
Opamp tail current	Itail_op	86	C13	39
Opamp tail voltage	Vtail_op	87	B13	26
Digital Supply	Vdd Pry	88	D11	43
N/C	N/C	89	C12	38
Digital Ground	GND Pry	90	A13	13

Table A–3. EPSILON II Pin out part III

Description	Name	Pad #	PGA	SCM
Analog Supply	AVdd	91	C11	37
Preset control signal	Preset	92	B12	25
Neuron Output	OUT31	93	A12	12
Neuron Output	OUT30	94	C10	36
Neuron Output	OUT29	95	B11	24
Neuron Output	OUT28	96	A11	11
Neuron Output	OUT27	97	B10	23
Neuron Output	OUT26	98	C9	35
Neuron Output	OUT25	99	A10	10
Neuron Output	OUT24	100	B9	22
Neuron Output	OUT23	101	A9	9
Neuron Output	OUT22	102	C8	34
Neuron Output	OUT21	103	B8	21
Neuron Output	OUT20	104	A8	8
Neuron Output	OUT19	105	B7	20
Neuron Output	OUT18	106	C7	33
Neuron Output	OUT17	107	A7	7
Neuron Output	OUT16	108	A6	6
Output mode control	PWM_select	109	B6	19
Output mode control	PFM_select	110	C6	32
Control test output	SR_check	111	A5	5
Refresh control signal	refresh	112	B5	18
Refresh clock 1	xphi2	113	A4	4
Refresh clock 2	xphi1	114	C5	31
Digital Supply	Vdd Pry	115	B4	17
Analog Ground	AGND	116	A3	3
Digital Supply	Vdd Pry	117	A2	2
Network Input	IN31	118	C4	30
Network Input	IN30	119	B3	16
Network Input	IN29	120	A1	1

Table A–4. EPSILON II Pin out part IV

A.8 Summary of Chip Functionality

Chip Number	Number of out-of-range neurons	Comments
0	1	
1	3	
2	0	
3	-	Originally working but damaged during testing
4	4	
5	4	
6	-	Not operational
7	1	
8	3	
9	2	
10	3	
11	1	
12	4	
13	1	
14	4	
15	-	Not operational
16	3	
17	-	Not operational
18	4	
19	-	Not operational

Table A–5. Summary of Chip Functionality

Appendix B

Xilinx Chip Design

This appendix gives the schematic designs of Xilinx functional blocks. Additional information on Xilinx building blocks and chip configuration can be found in the libraries guide [118] and data book [117].

B.1 Top Level Design

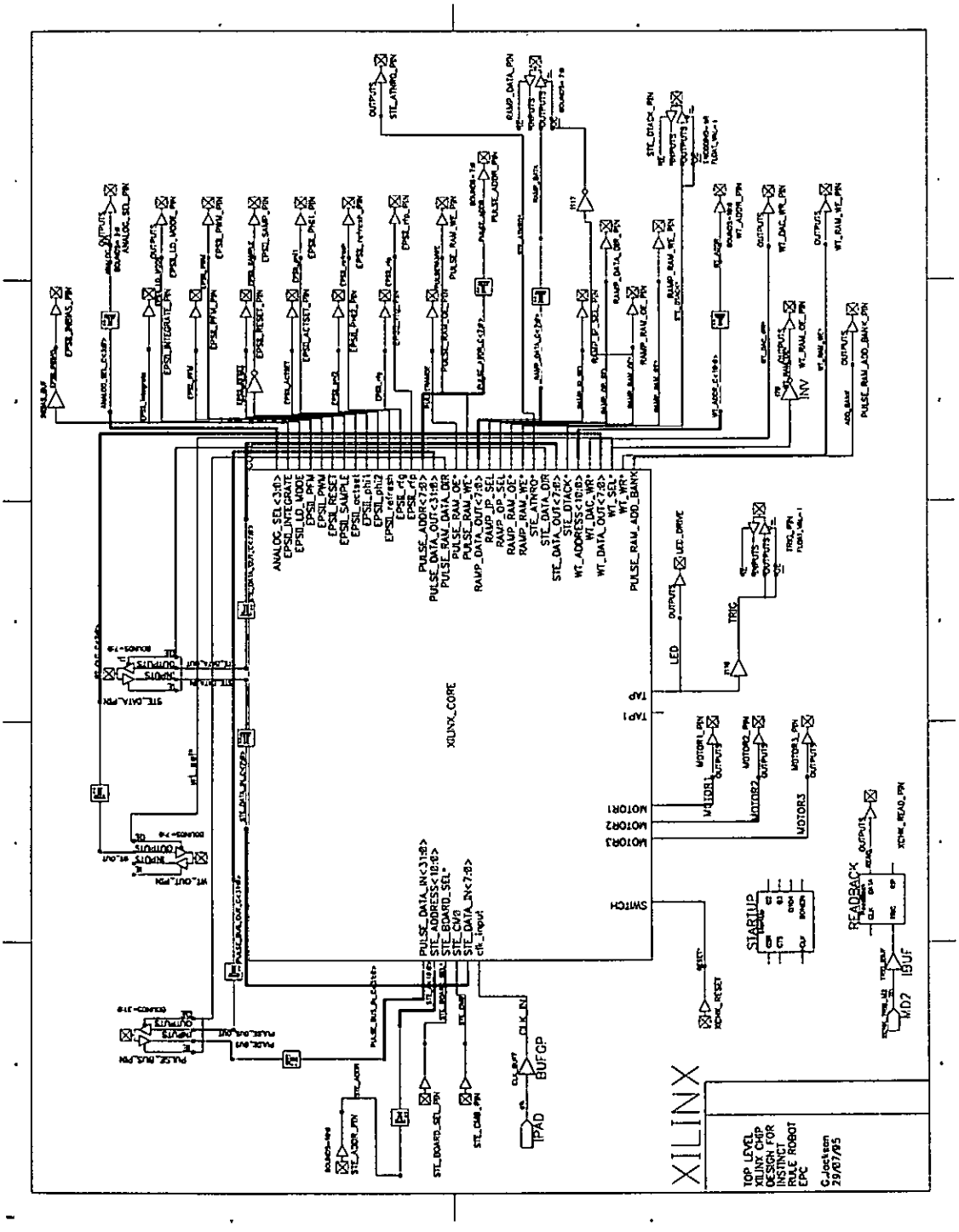


Figure B–1: Top Level Design

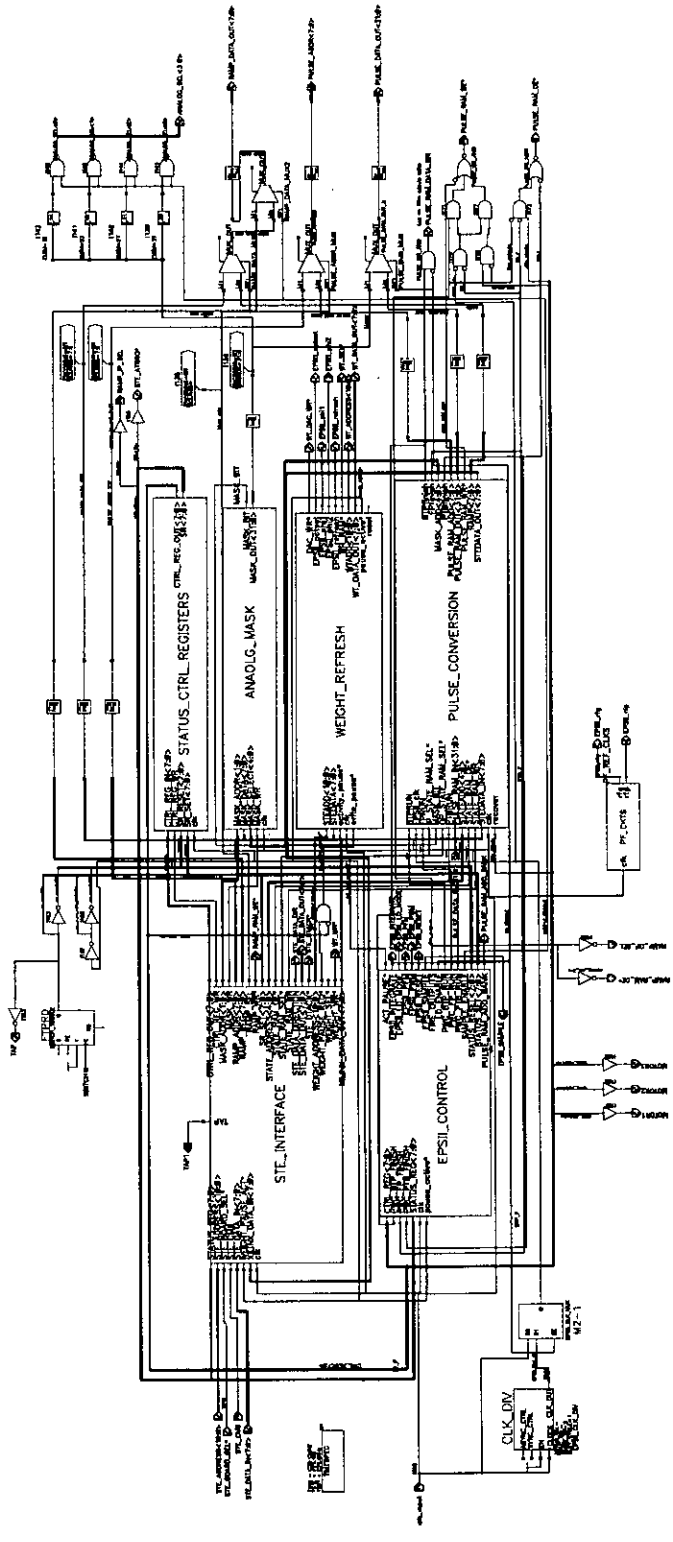


Figure B–2: Top Level Core Design

B.2 Pulse Conversion

Consists of five blocks:

1. BIN_TO_PW for binary-to-pulse-width conversion.
2. FIRE_PULSES to fire input and output ramps and drive pulse RAM.
3. UP_DOWN_COUNTER to generate linear input ramp.
4. EPSII_ADDRESS_COUNTER which clocks addresses for ramps and pulse RAM.
5. PW_TO_BIN for pulse-width-to-binary conversion.

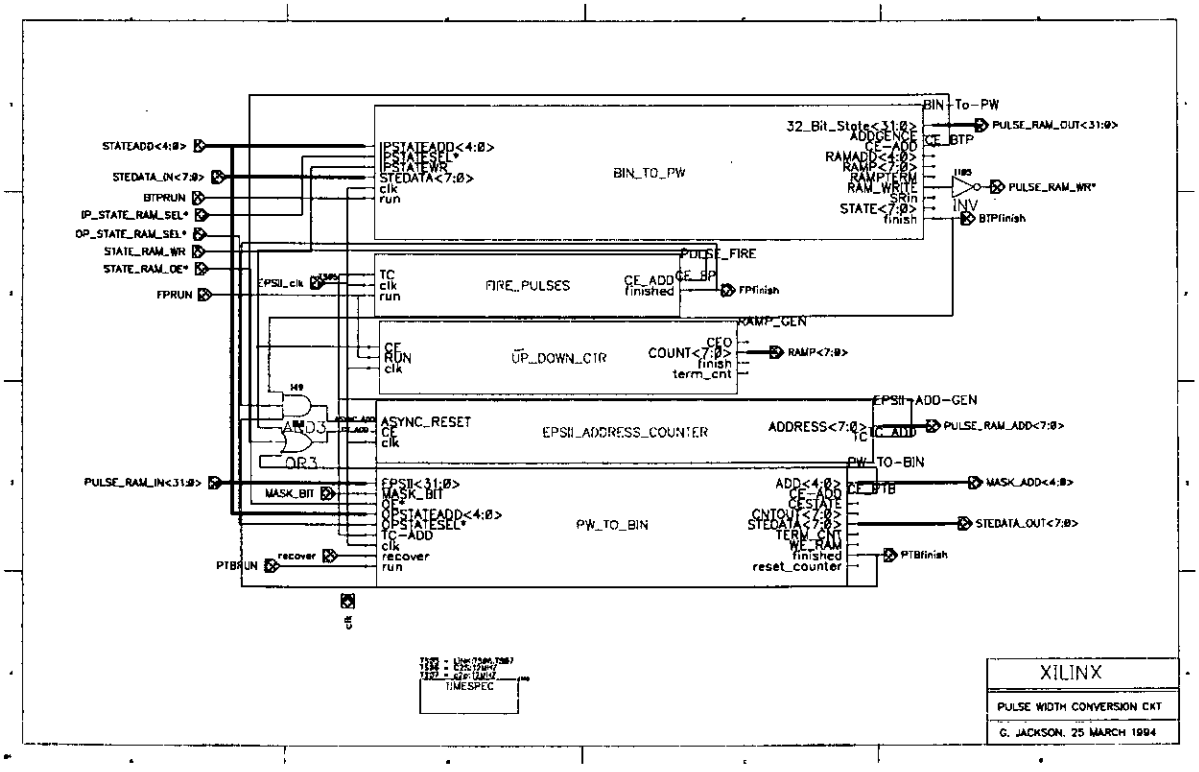


Figure B–3: Pulse Conversion Design

B.2.1 Binary to Pulse Stream Conversion

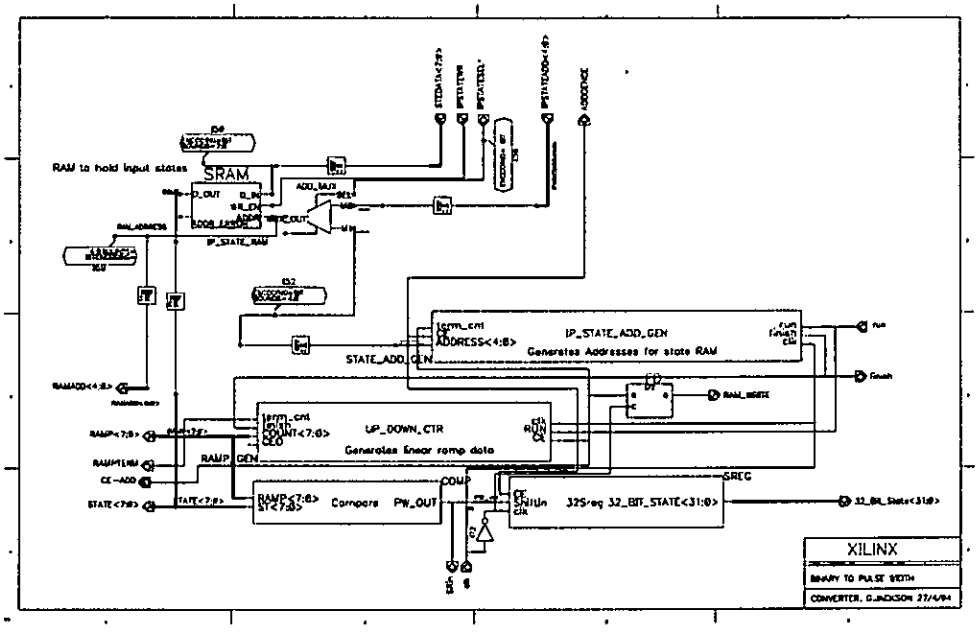


Figure B–4: Binary-to-pulse stream Design

The UP_DOWN_CTR begins counting on a **run** signal down from 0x0FF to 0x00 in steps of two, it stays at 0x00 for one clock cycle then counts up from 0x01 to 0xFF then stops.

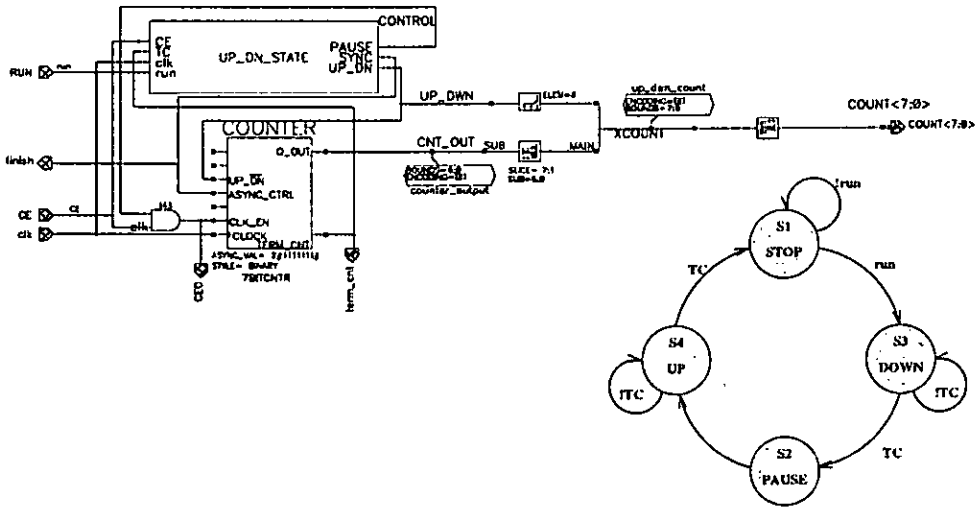


Figure B–5: Up-Down Counter Design

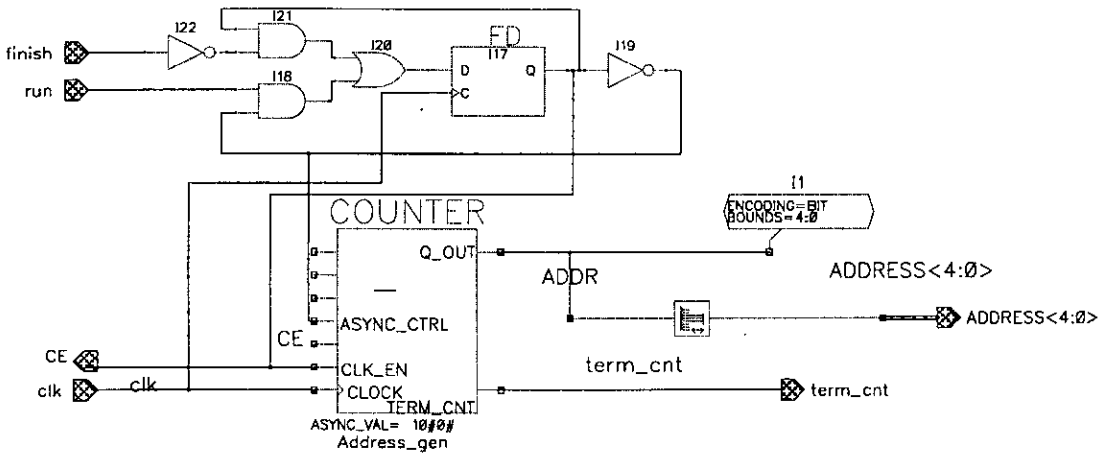


Figure B–6: Address Generation and Control Design

B.2.2 Pulse Stream to Binary Conversion

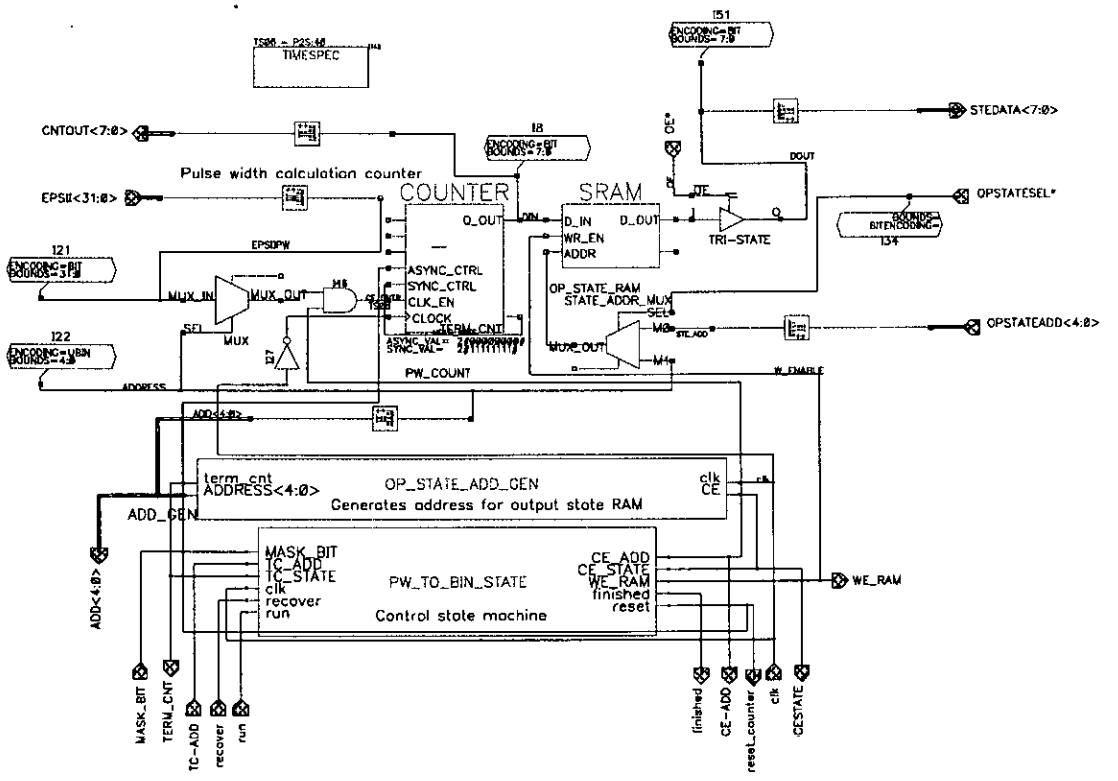


Figure B–7: Pulse-width-to-Binary Conversion Design

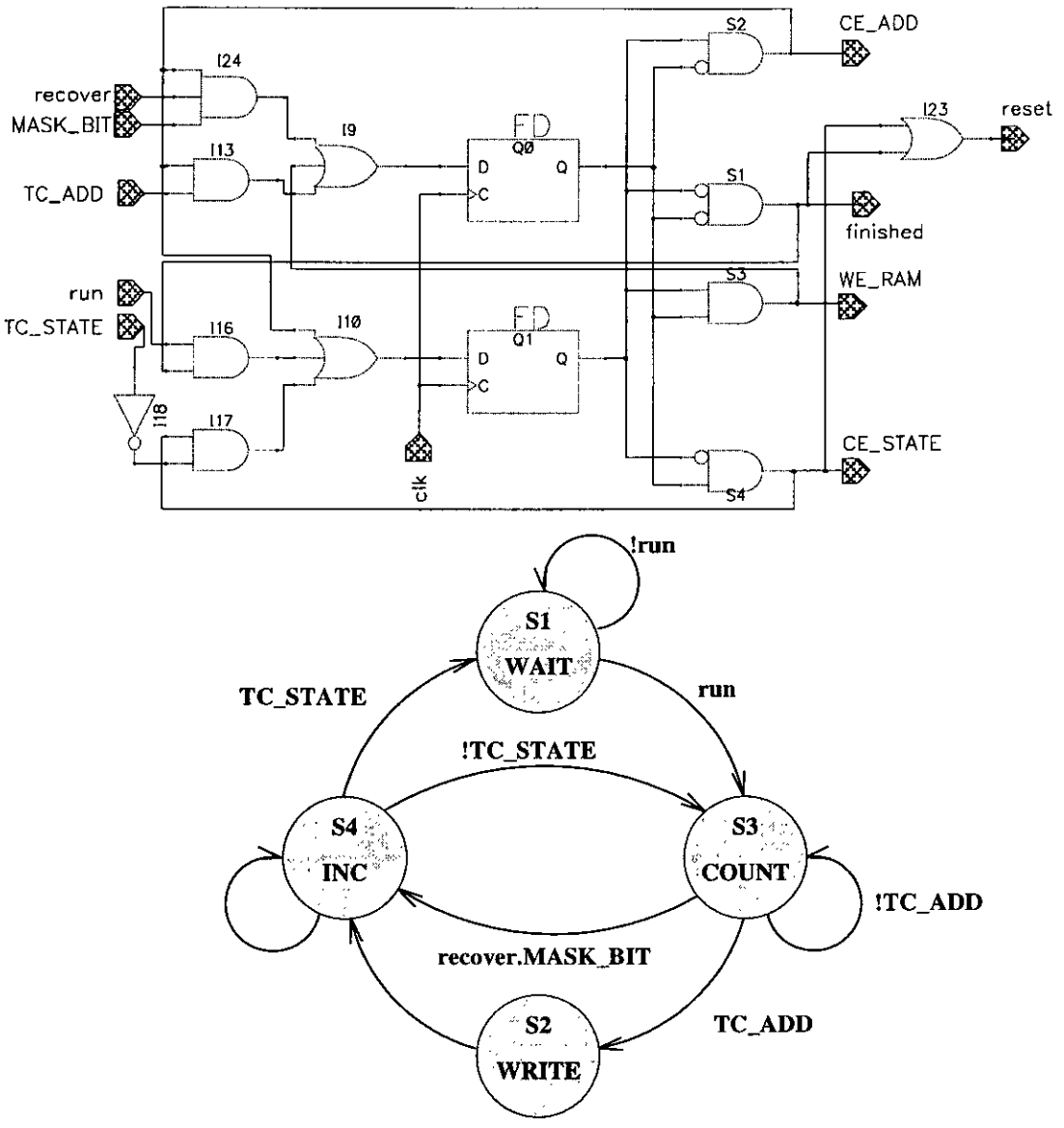


Figure B–8: Control State Machine Design

This state machine controls pulse-width-to-binary conversion. Pulse-widths are processed sequentially by scanning pulse RAM (CE_ADD enables address generation in COUNT state) then writing the result into state ram (WRITE state). State RAM address counter is then incremented and (INC) process repeated until all states done (as defined by TC_STATE signal). For a recovery of analog inputs (recover) only analog inputs are processed as defined by mask register.

B.3 Weight Refresh

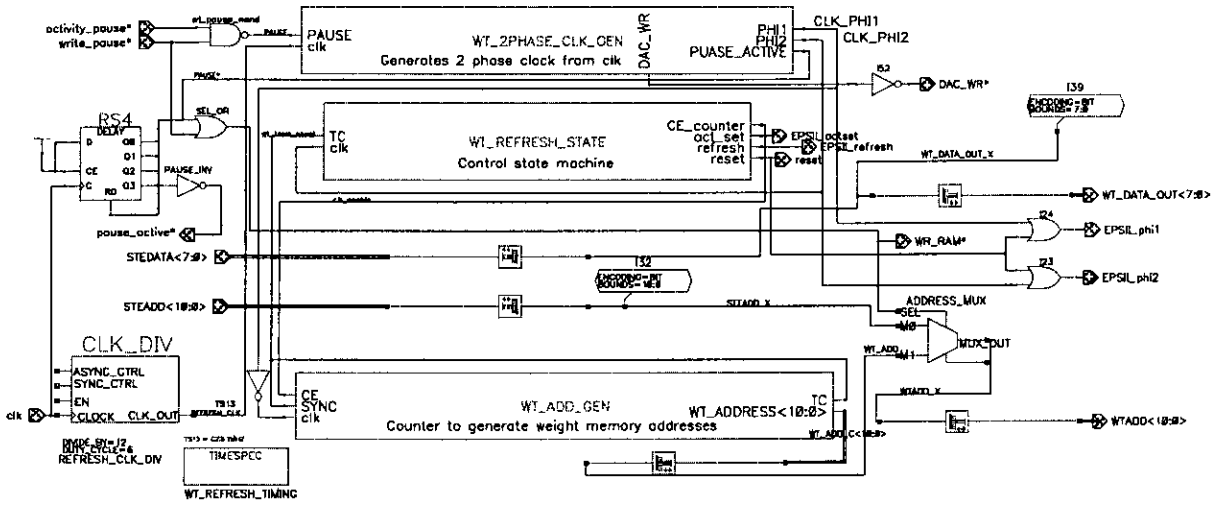


Figure B–9: Weight Refresh Design

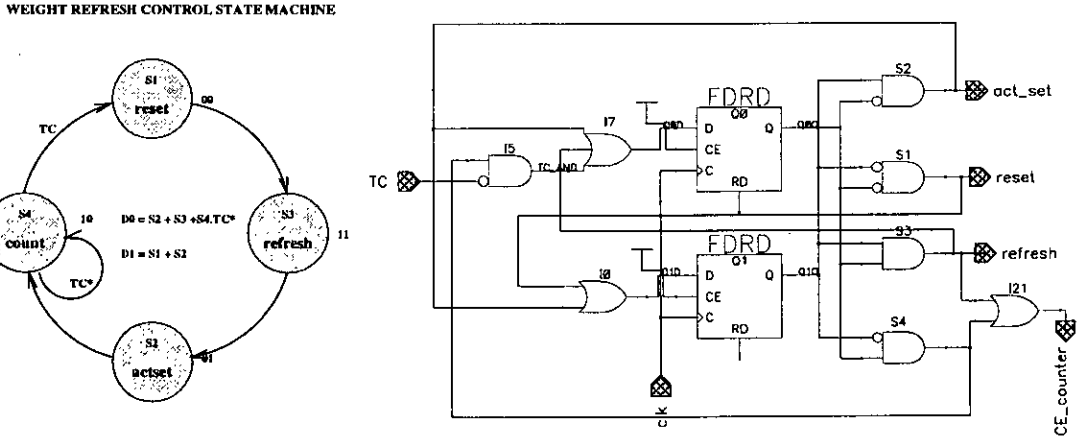


Figure B–10: Weight State Machine Design

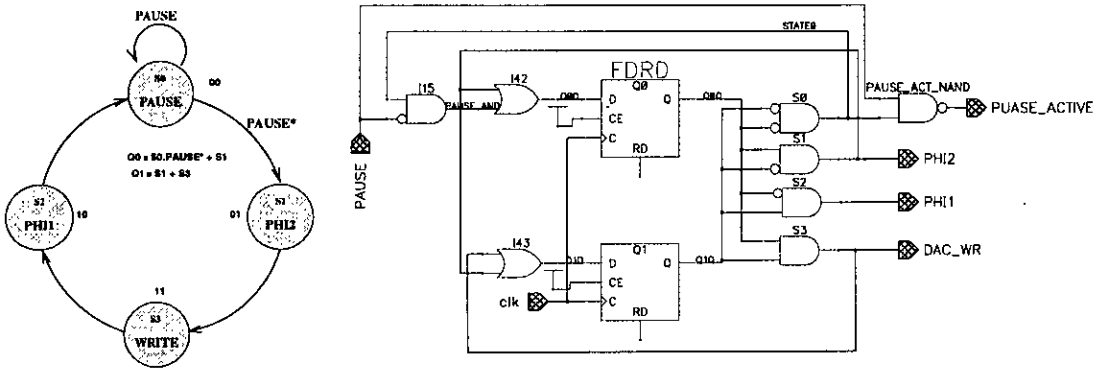


Figure B–11: Two Phase Clock Generator Design

B.4 STE Interface

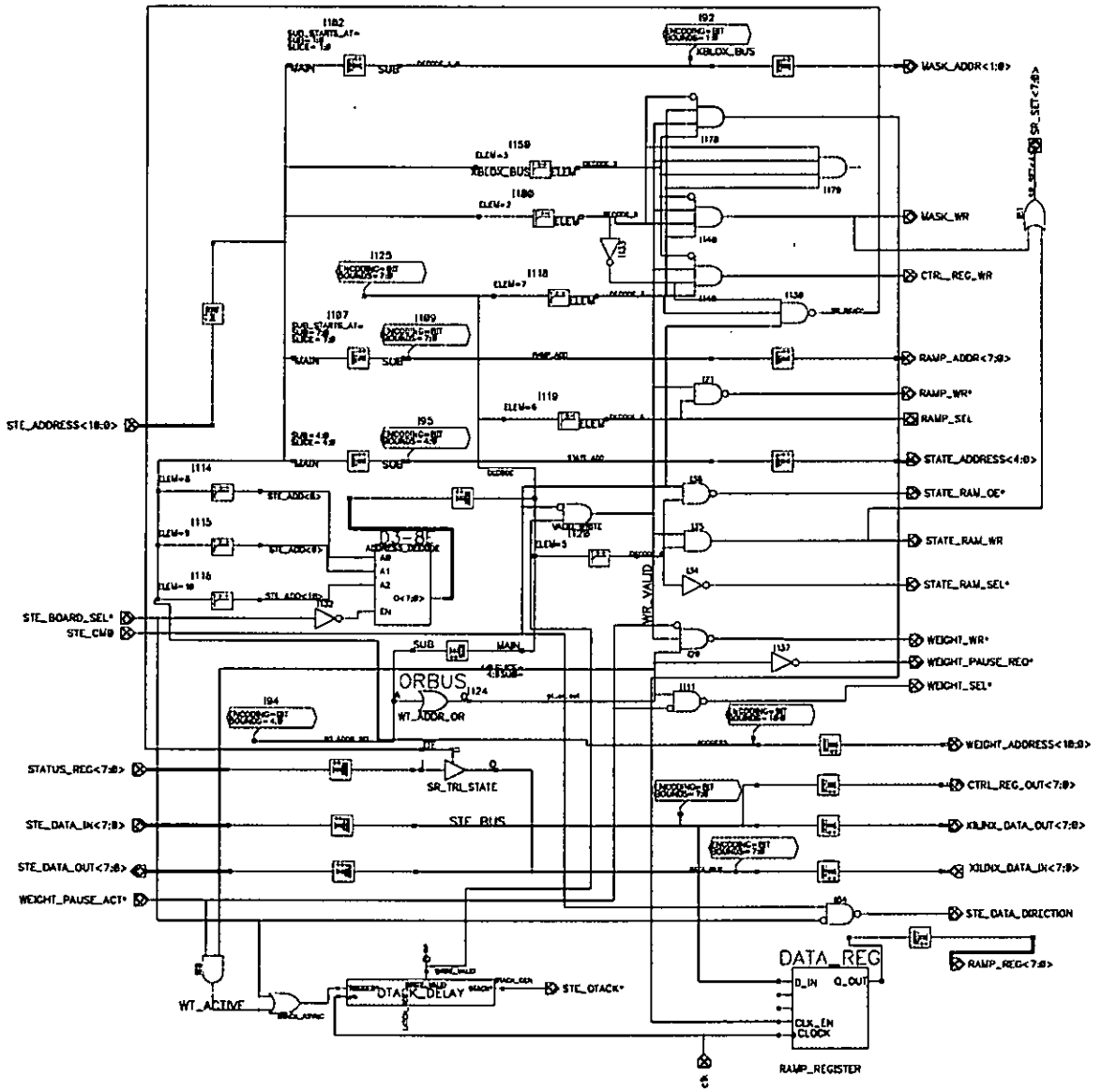


Figure B–12: STE Interface Design

This block decodes addresses and produces select and write signals for various registers and RAM blocks memory mapped in the EPC. It generates the STE data acknowledge (DTACK) signal to indicate the completion of a bus cycle.

B.5 Control State Machine

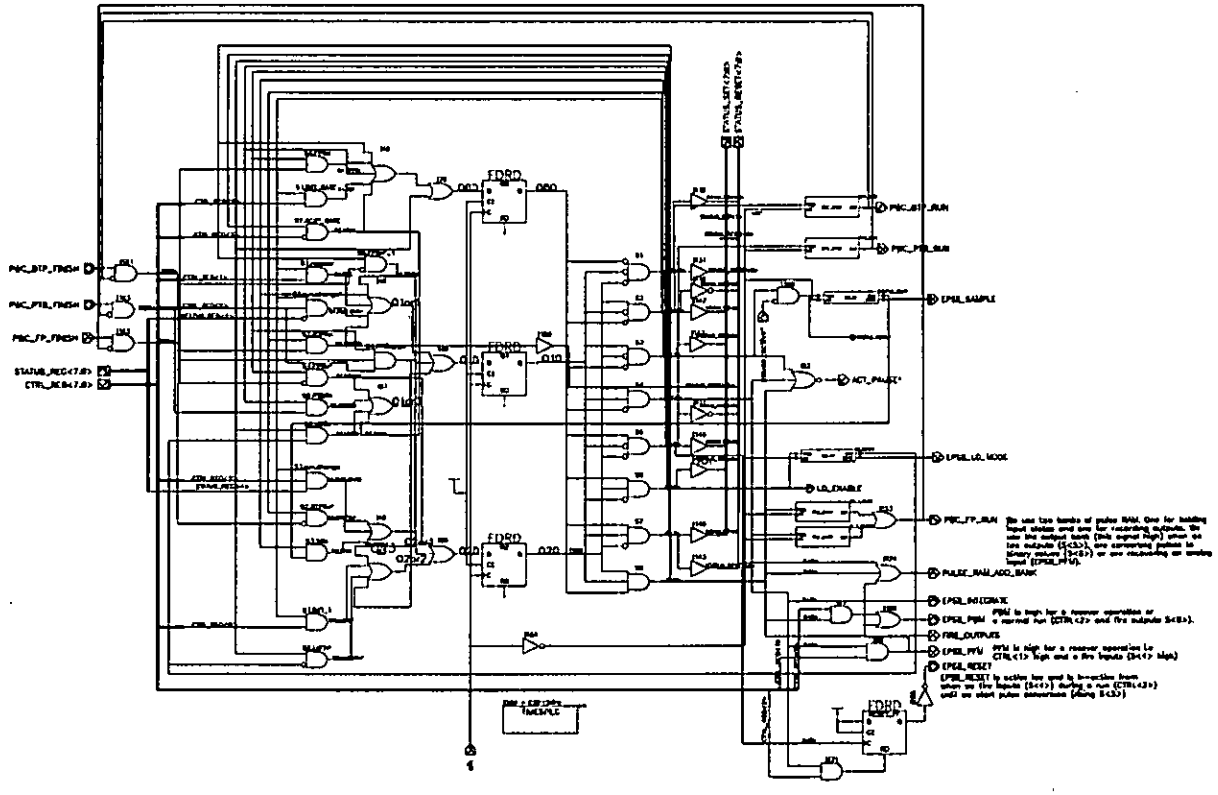


Figure B–13: Control State Machine Design

This state machine sequences EPC operations. It is triggered by bits in the **control** and **status** registers along with finish signals from the EPC functional blocks. It provides **run** signals for functional blocks and the main control signals for the EPSILON II chip.

Appendix C

EPC Documentation

C.1 Parts List

PART NO.	DESCRIPTION	QTY
STECON	STE BUS CONNECTOR	1
M-TERM	MOTHER BOARD SOCKET	2
IDC40	40 WAY IDC CONNECTOR	1
XCHECK1	XCHECKER CONNECTOR 1	1
XCHECK2	XCHECKER CONNECTOR 2	1
0.1μF	0.1μF CAP	11
XC4006	XILINX XC4006PG156 FPGA	1
74HC245	OCTAL BUS TRANSCEIVERS	4
M5M5178-25	8192 X 8-BIT HIGH SPEED STATIC	6
74HC688	8 BIT MAGNITUDE COMPARATOR	2
27C256	32K X 8 UV EPROM CMOS	1
JUMP4	4 WAY JUMPER	1
JUMP	8 WAY GND/VCC JUMPER	1
SM-LED	SMALL LED, INT RES	2
OSC24	DIL OSCILLATOR 24.0'MHz	1
ME8-47K	8 RESISTORS,9 PIN SIL 47K	1
10K	10K 1/4W RESISTOR	1
DUALSW	DUAL DPDT SWITCH	1
PB-SW	PUSH BUTTON SWITCH N/O	2

Table C–1. EPC Mother Board Parts List

PART NO.	DESCRIPTION	QTY
TAP	TAP POINT TERMINAL	7
D-TERM	DAUGHTER BOARD HEADER	2
IDC20	20 WAY IDC CONNECTOR	1
PWRCON	POWER CONNECTOR	1
33pF	33pF CERAMIC CAP	5
0.1μF	0.1μF CAP	19
1μF-TANT	1μF TANTALUM CAP	9
C	CAPACITOR	2
1PF5	1.5pF CERAMIC CAP	1
82PF	82pF CERAMIC CAP	2
10PF	10pF CERAMIC CAP	2
10μF-TANT	10μF TANTALUM CAP	4
EPSII	PULSE STREAM NN	1
LF347	QUAD FET INPUT OPAMP	2
DG211	DG211CJ QUAD SPST ANALOGUE SWI	4
AD7524	AD7524 8 BIT MULT DAC	1
EL244	QUAD VIDEO OPAMP 350V/μS SLEW	1
DAC08	8 BIT HI SPEED DAC	2
DG303	DG303ACJ DUAL SPDT ANALOGUE SW	1
AD828	DUAL HIGH SPEED OPAMP	2
JUMP2	2 WAY JUMPER	1
LINK8	8 WAY LINKS	1
10KPOT	10K POT	12
100RPOT	100R POT	1
1KPOT	1K POT	1
5KPOT	5K POT	2
LM337LZ	Negative Voltage Regulator	1
LM317LZ	Positive Voltage Regulator	1
MD4-2K2	4 2K2 RESISTORS,8 PIN SIL	2
R	1/4W RESISTOR	9
220R	220R 1/4W RESISTOR	2

Table C–2. EPC Daughter Board Parts List

C.2 PCB Layout

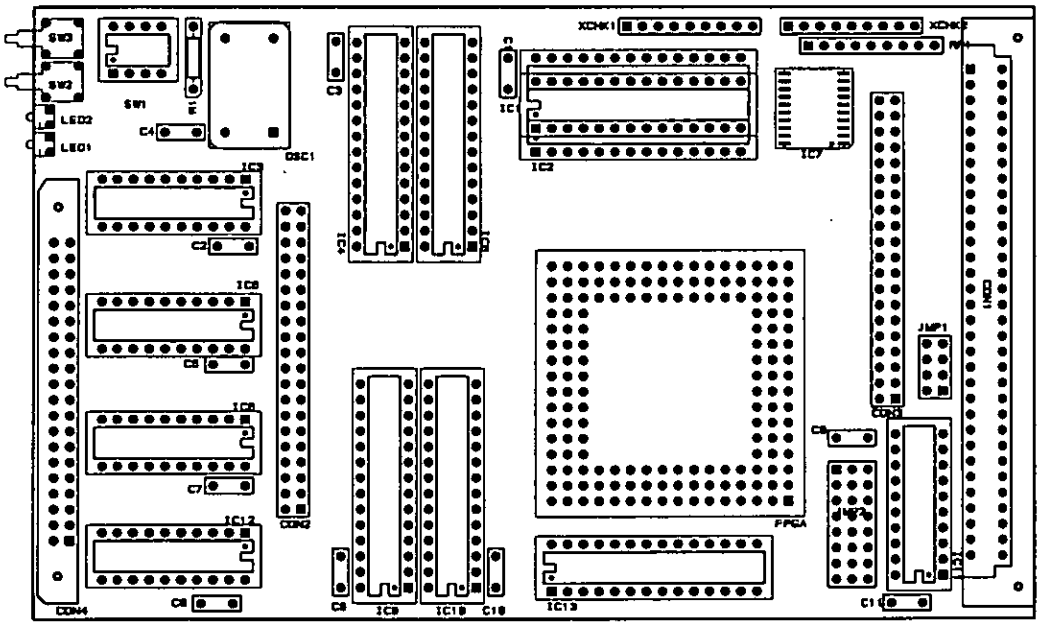


Figure C–1: Mother Board PCB Layout

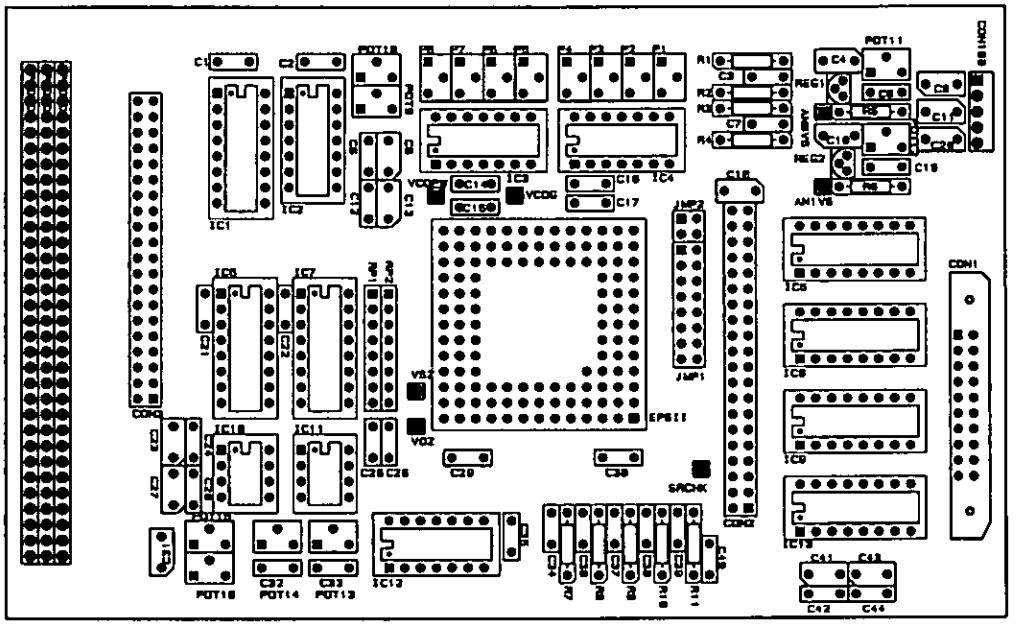


Figure C–2: Daughter Board PCB Layout

C.3 EPC Schematics

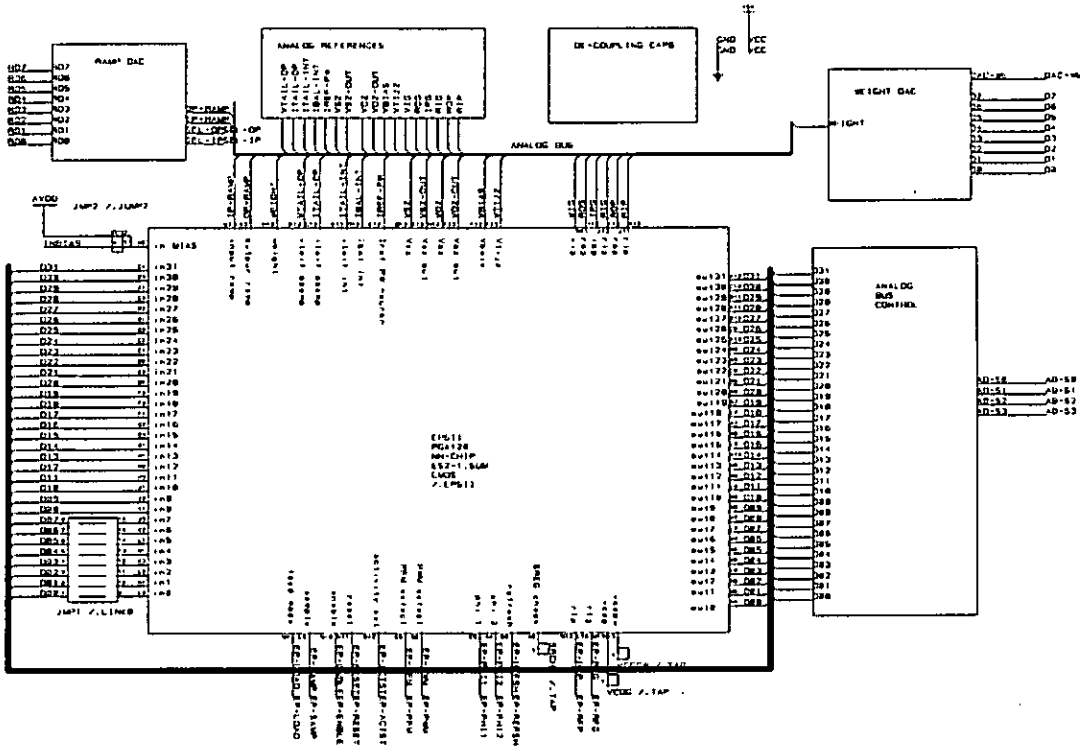


Figure C–3: Daughter Board Schematic

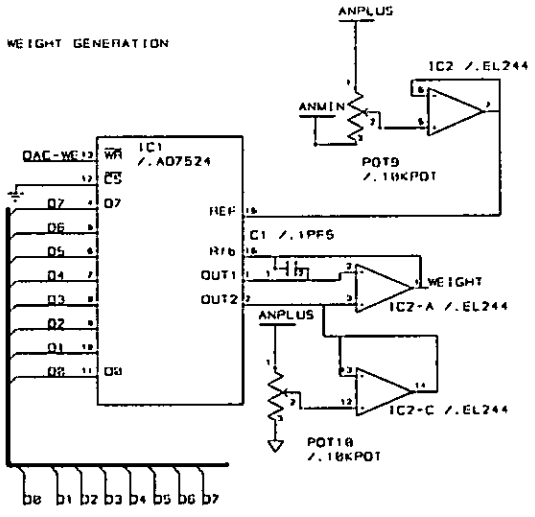


Figure C–4: Weight Generation Schematic

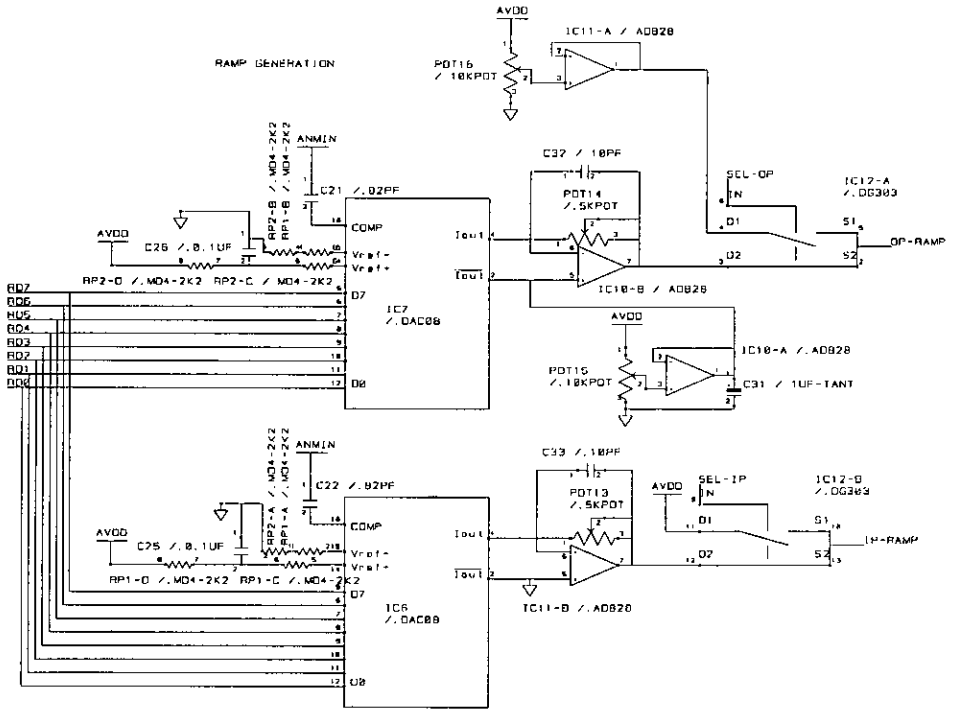


Figure C–5: Ramp Generation Schematic

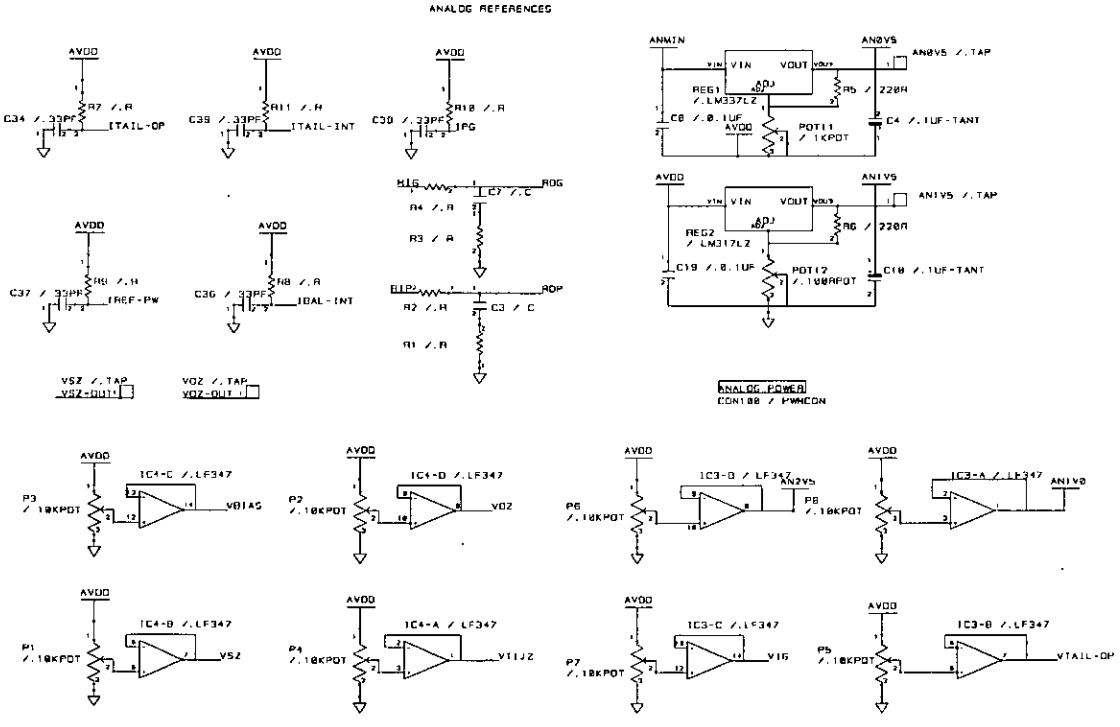


Figure C–6: Reference and Supply Generation Schematic

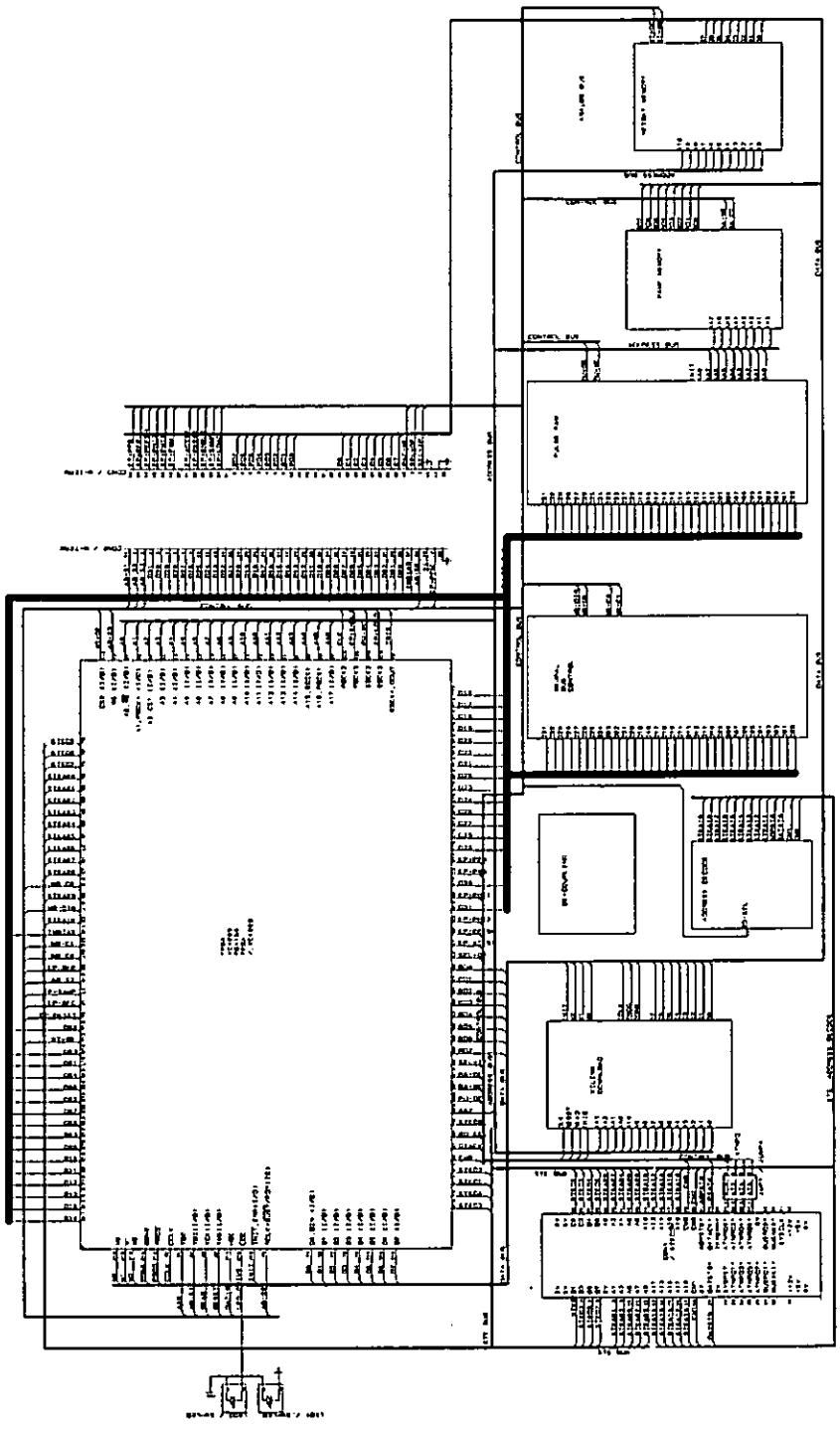


Figure C–7: Mother Board Schematic

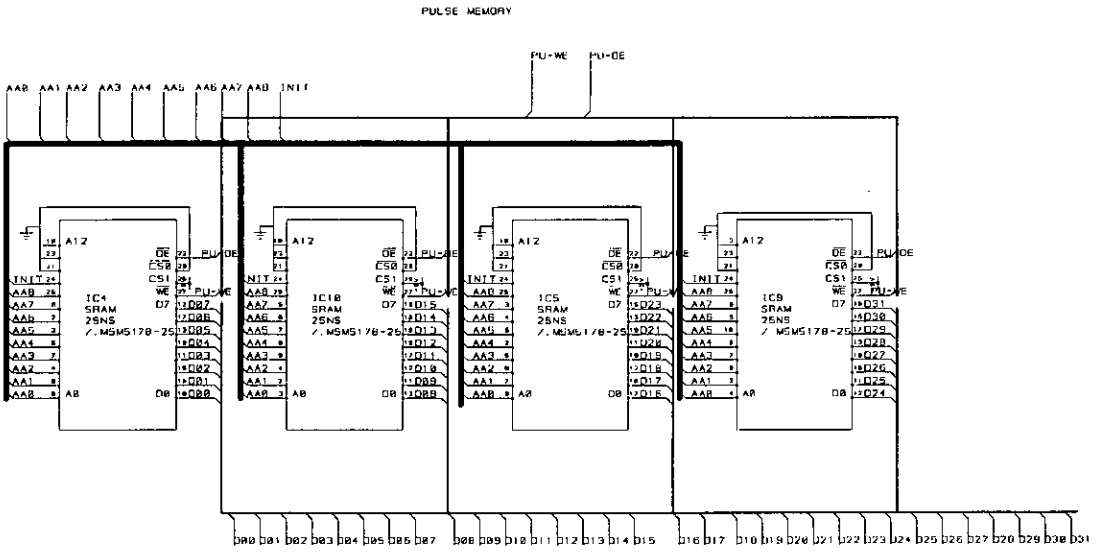


Figure C–8: Pulse RAM Schematic

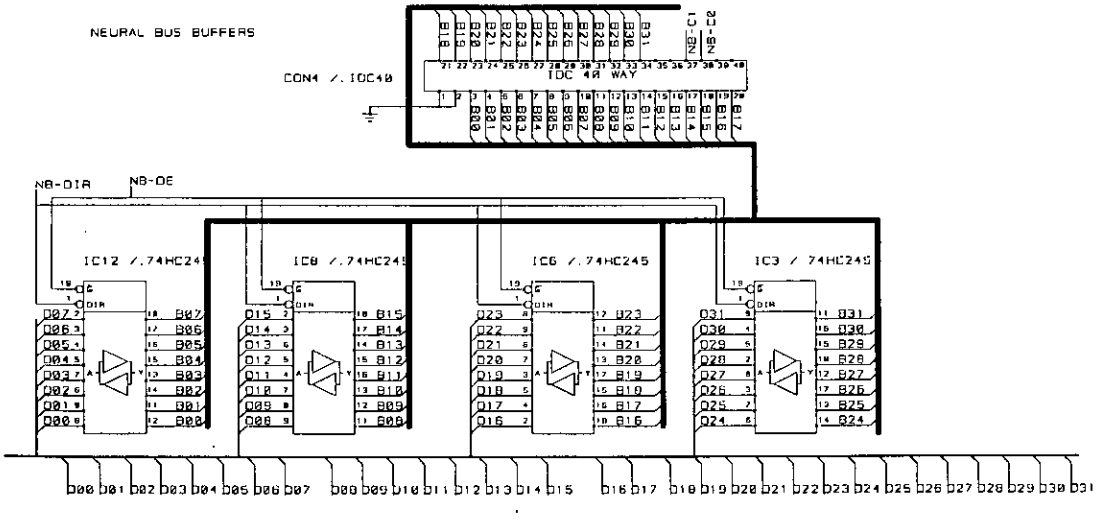


Figure C–9: Neural Bus Control Schematic

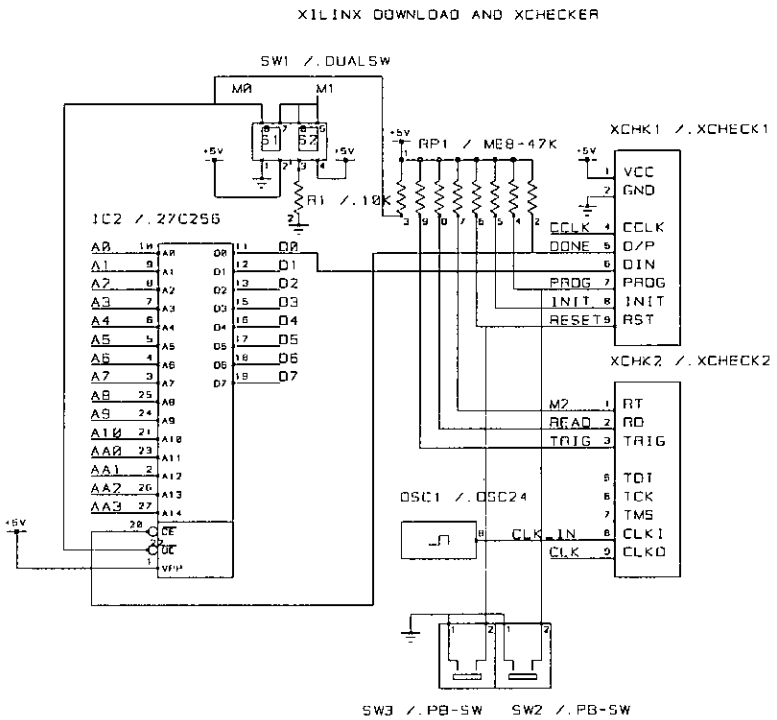


Figure C–10: Xilinx Download Schematic

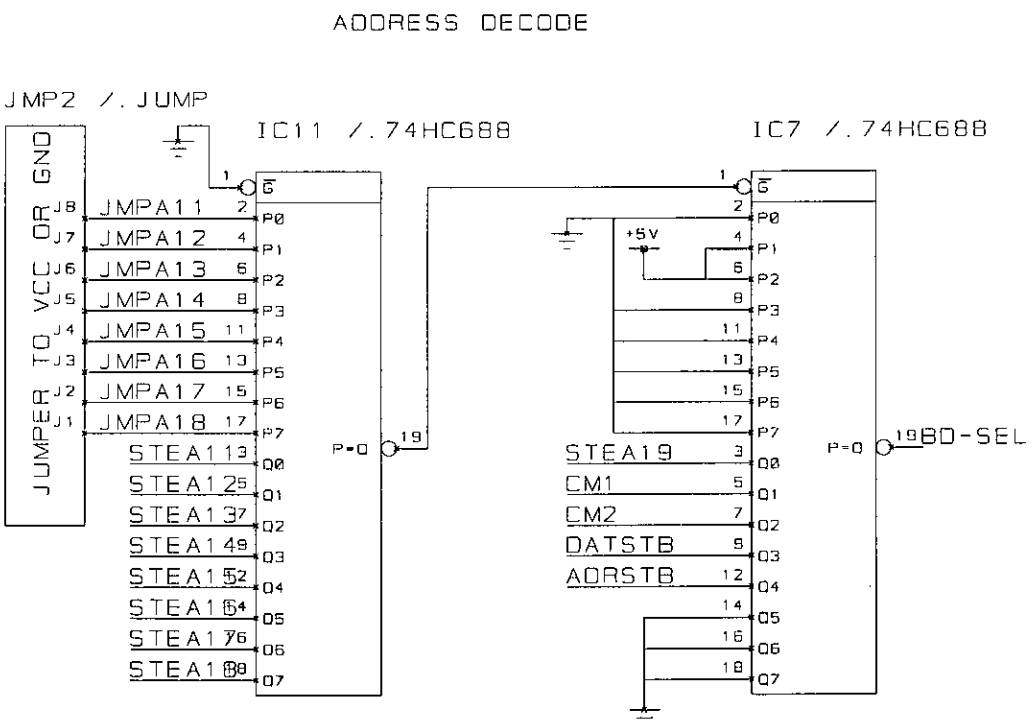


Figure C–11: Address Decoding Schematic

C.4 Board Set-up Procedure

The following procedures describe how to set up the EPC board for operation.

C.4.1 Xilinx Microcode Selection.

The microcode that determines the functionality of the Xilinx chip can be loaded into the chip from two sources:

1. Xilinx Xchecker serial cable: This allows downloading and de-bugging from a SUN workstation via the Xchecker serial cable. To select this mode the mode switches are in the **XCHK** and **DWN** positions. The Xchecker cable is plugged onto connectors **XCHK1** and **XCHK2**.

2. On-board EEPROM: To load the Xilinx device from the on-board EEPROM the first switch is placed into the **EEPROM** position. A single EEPROM can store two possible configurations. Select switch to **UP** to load configuration from EEPROM address **0x0000** select **DWN** to load configuration from EEPROM address **0x7FFF**

C.4.2 EPC Base Address Selection

The base address is set using jumpers marked **A11–A18** these are set to 0 or 1 depending on position of jumper.

C.4.3 Analog Supplies

The two analog synaptic supplies as set using **POT11** (0.5V supply) and **POT12** (1.5V supply) these can be monitored at tap points marked **AN0V5** and **AN1V5** and should be set to $1mV$ accuracy.

C.4.4 DAC Setup

Weight Refresh DAC

Apply a weight set consisting of weights of T_{ij}=0x00,0x80 & 0xFF. Adjust POT10 to give minimum weight voltage of 2.5V then adjust POT9 which controls gain to set T_{ij}=0x80 to be 3.75V. T_{ij}=0xFF should be close to 5V.

Ramp DACs

Input ramp is adjusted by POT13 to give 0–5V ramp. Output ramp is offset by POT15 to start at 1.0V and end at 4.0V.

C.4.5 Analog References

Potentiometers **P1** to **P8** set the analog references, theses are monitored from the appropriate op-amp pins (see figure C–6) and set to:

P1 VSZ set to achieve a $10\mu s$ output pulse-width with zero weight and $10\mu s$ input pulse.

P2 VOZ set to achieve a $10\mu s$ output pulse-width with zero weight and $0\mu s$ input pulse.

P3 VTijz set to 3.75V

P4 Vbias set to 3.10V

P5 Vt-op set to 1.60V

P6 2V5 set to 2.500V

P7 1V0 set to 1.000V

P8 Vig set to 0V, not used for pulse-width circuits.

Appendix D
Publications

G Jackson and A F Murray. "Competence acquisition in an autonomous mobile robot using hardware neural techniques", In *Advances in Neural Information Processing Systems 8*, In print, 1995.

G Jackson, A F Murray, and A Hamilton. "Pulse stream neural networks for applications", In *Australian Conference on Neural Networks*, 1995.

G Jackson, A Hamilton, and A F Murray. "The EPSILON processor card: A framework for analog neural computation", In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems.*, pages 280–286. IEEE Computer Society Press, 1994.

A F Murray, S Churcher, A Hamilton, A J Holmes, G B Jackson, and R J Woodburn. "Applications of pulsed neural VLSI", *IEEE MICRO*, 14(3):29–39, June 1994.

G Jackson, A Hamilton, and A F Murray. "Pulse stream VLSI neural systems: into robotics", In *Proceedings ISCAS'94*, volume 6, pages 375–378. IEEE Press, May 1994.

A Hamilton, S Churcher, P J Edwards, G B Jackson, Alan F. Murray, and H. Martin Reekie. "Pulse stream VLSI circuits and systems: The EPSILON neural network chipset", *International Journal of Neural Systems*, 4(4):395–405, Dec 1993.

G Jackson and A F Murray. "Competence acquisition in an autonomous mobile robot using hardware neural techniques", In *Advances in Neural Information Processing Systems 8*, In print, 1995.

Competence Acquisition in an Autonomous Mobile Robot using Hardware Neural Techniques.

Geoff Jackson and Alan F. Murray
Department of Electrical Engineering
Edinburgh University
Edinburgh, EH9 3JL
Scotland, UK
gbj@ee.ed.ac.uk,afm@ee.ed.ac.uk

Abstract

In this paper we examine the practical use of hardware neural networks in an autonomous mobile robot. We have developed a hardware neural system based around a custom VLSI chip, EPSILON II[1], designed specifically for embedded hardware neural applications. We present here a demonstration application of an autonomous mobile robot that highlights the flexibility of this system. This robot gains basic mobility competence in very few training epochs using an "instinct-rule" training methodology.

1 INTRODUCTION

Though neural networks have been shown as an effective solution for a diverse range of real-world problems, applications and especially hardware implementations have been few and slow to emerge. For example in the DARPA neural networks study of 1988; of the 77 neural network applications investigated only 4 had resulted in field tested systems [Widrow, 1988]. Furthermore, none of these used dedicated neural network hardware. It is our view that this lack of tangible successes can be summarised by the following points:

- Most neural applications will be served optimally by fast, generic digital computers.
- Dedicated digital neural accelerators have a limited lifetime as "the fastest", as standard computers develop so rapidly.

[1] Edinburgh Pulse Stream Implemenation of a Learning Oriented Network.

- Analog neural VLSI is a niche technology, optimally applied at the interface between the real world and higher-level digital processing.

This attitude has some profound implications with respect to the size, nature and constraints we place on new hardware neural designs. After several years of research into hardware neural network implementation, we have now concentrated on the areas in which analog neural network technology has an "edge" over well established digital technology.

Within the pulse stream neural network research at the University of Edinburgh, the EPSILON chip's areas of strength can be summarised as:

- Analog or digital inputs, digital outputs.
- Modest size.
- Scaleable and cascadeable design.
- Compact, low power.

This list points naturally and strongly to problems on the boundary of the real, analog world and digital processing, such as pre-processing/interpretation of analog sensor data. Here a modest neural network can act as an *intelligent analog-to-digital converter* presenting preprocessed information to its host. We are now engaged in a two pronged approach, whereby development of technology to improve the performance of pulse stream neural network chips is occurring concurrently with a search and development of applications to which this technology can be applied. The key requirements of this technological development are that devices must:

- Work directly with analog signals.
- Provide a moderate size network.
- Have the potential for a fully integrated solution.

In working with the above constraints and goals we have developed a new chip, EPSILON II, and a bus based processor card incorporating it. It is our aim to use this system to develop applications. As our first demonstration the EPSILON processor card has been mounted on an autonomous mobile robot. In this case the network utilises a mixture of analog and digital sensor information and performs a mapping between input/sensor space, a mixture of analog and digital signals, and output motor control.

2 THE EPSILON II CHIP

The EPSILON II chip has been designed around the requirements of an application based system. It follows on from an earlier generation of pulse stream neural network chip, the EPSILON chip [Murray, 1992].

The EPSILON II chip represents neural states as a pulse encoded signal. These pulse encoded signals have digital signal levels which make them highly immune to noise and ideal for inter and intra-chip communication, facilitating efficient cascading of chips to form larger systems. The EPSILON II chip can take as inputs either pulse encoded signals or analog voltage levels, thus facilitating the fusing of analog and digital data in one system. Internally the chip is analog in nature allowing the synaptic multiplication function to be carried out in compact and efficient analog cells [Jackson, 1994].

Table 1 shows the principal specifications of the EPSILON II chip. The EPSILON II chip is based around a 32x32 synaptic matrix allowing efficient interfacing to digital systems. Several features of the device have been developed specifically for applications based usage. The first of these is a programmable input mode. This

Table 1: EPSILON II Specifications

EPSILON II Chip Specifications	
No. of state input pins	32
Input modes	Analog, PW or PF
Input mode programmability	Bit programmable
No. of state outputs	32 pinned out
Output modes	PW or PF
Digital recovery of analog I/P	Yes - PW encoded
No. of Synapses	1024
Additional *autobias* synapses	4 per output neuron
Weight storage	Dynamic
Programmable activity voltage	Yes
Die size	$6.9mm \times 7mm$

allows each of the network inputs to be programmed as either a direct analog input or a digital pulse encoded input. We believe that this is vital for application based usage where it is often necessary to *fuse* real–world analog data with historical or control data generated digitally. The second major feature is a pulse recovery mode. This allows conversion of any analog input into a digital value for direct use by the host system. Both these features are utilised in the robotics application described in section 4 of this paper.

3 EPSILON PROCESSOR CARD

The need to embed the EPSILON chip in a processor card is driven by several considerations. Firstly, working with pulse encoded signals requires substantial processing to interface directly to digital systems. If the neural processor is to be transparent to the host system and is not to become a substantial processing overhead, then all pulse support operations must be carried out independently of the host system. Secondly, to respond to further chip level advances and allow rapid prototyping of new applications as they emerge, a certain amount of flexibility is needed in the system. It is with these points in mind that the design of the flexible EPSILON Processor Card (EPC) was undertaken.

3.1 DESIGN SPECIFICATION

The EPC has been designed to meet the following specifications. The card must:

- Operate on a conventional digital bus system.
- Be transparent to the host processor, that is carry out all the necessary pulse encoding and decoding.
- Carry out the refresh operations of the dynamic weights stored on the EPSILON chip.
- Generate the ramp waveforms necessary for pulse width coding.
- Support the operation of multiple EPC's.
- Allow direct input of analog signals.

As all data used and generated by the chip is effectively of 8-bit resolution, the STE bus, an industry standard 8-bit bus, was chosen for the bus system. This is also cost

effective and allows the use of readily available support cards such as processors, DSP cards and analog and digital signal conditioning cards.

To allow the transparency of operation the card must perform a variety of functions. A block diagram indicating these functions is shown in figure 1.

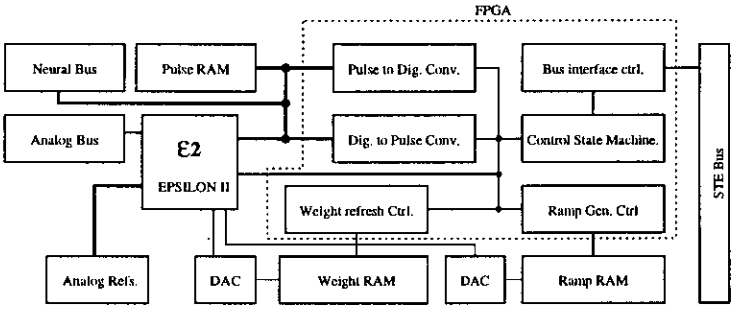


Figure 1: EPSILON Processor Card

A substantial amount of digital processing is required by the card, especially in the pulse conversion circuitry. To conform to the *Eurocard* standard size of the STE specification an FPGA device is used to "absorb" most of the digital logic. A twin mother/daughter board design is also used to isolate sensitive analog circuitry from the digital logic. The use of the FPGA makes the card extremely versatile as it is now easily reconfigurable to adapt to specialist application. The dotted box of figure 1 shows functions implemented by the FPGA device. An on board EPROM can hold multiple FPGA configurations such that the board can be reconfigured "on the fly". All EPSILON support functions, such as ramp generation, weight refresh, pulse conversion and interface control are carried out on the card. Also the use of the FPGA means that new ideas are easily tested as all digital signal paths go via this device. Thus a card of new functionality can be designed without the need to design a new PCB.

3.2 SPECIALIST BUSES

The digital pulse bus is buffered out under control of the FPGA to the neural bus along with two control signals. Handshaking between EPC's is done over these lines to allow the transfer of pulse stream data between processors. This implies that larger networks can be implemented with little or no increase in computation time or overhead. A separate analog bus is included to bring analog inputs directly onto the chip.

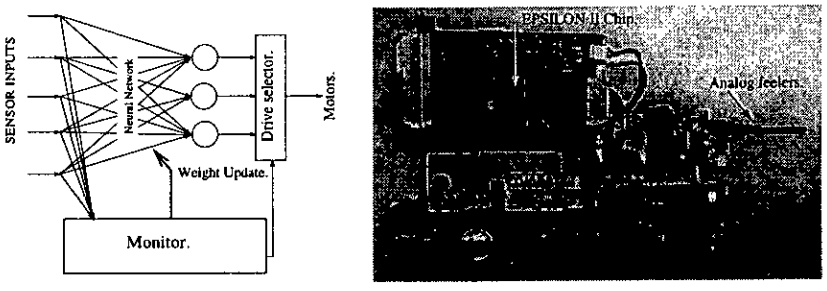
4 APPLICATIONS DEVELOPMENT

The over-riding reason for the development of the EPC is to allow the easy development of hardware neural network applications. We have already indicated that we believe that this form of neural technology will find its niche where its advantages of direct sensor interface, compactness and cost-effectiveness are of prime importance. As a good and intrinsically interesting example of this genre of applications, we have chosen autonomous mobile robotic control as a first test for EPSILON II. The object of this demonstrator is not to advance the state-of-the-art in robotics.

Rather it is to demonstrate analog neural VLSI in an appropriate and stimulating context.

4.1 "INSTINCT-RULE" ROBOT

The "instinct-rule" robotic control philosophy is based on a software-controlled exemplar from the University's Department of Artificial Intelligence [Nehmzow, 1992]. The robot incorporates an EPC which interfaces all the analog sensor signals and provides the programmable neural link between sensor/input space and the motor drive actuators.



a) Controller Architecture. b) Instinct rule robot.

Figure 2: "Instinct Rule" Robot

The controller architecture is shown in figure 2. The neural network implemented on the EPC is the *plastic* element that determines the mapping between sensory data and motor actions. The majority of the monitor section is currently implemented on a host processor and monitors the performance of the neural network. It does this by regularly evaluating a set of *instinct rules*. These rules are simple behaviour based axioms. For example, we use two rules to promote simple obstacle avoidance competence in the robot, as listed in column one of table 2

Table 2: Instinct Rules

Simple obstacle avoidance.	Wall following
1. **Keep crash sensors inactive.**	1. **Keep crash sensors inactive.**
2. **Move forward.**	2. **Keep side sensors active.**
	3. **Move forward.**

If an instinct rule is violated the drive selector then chooses the next strongest output (motor action) from the neural network. This action is then performed to see if it relieves the violation. If it does, it is used as targets to train the neural network. If it does not, the next strongest action is tried. The mechanism to accomplish this will be described in more detail in section 4.2.

Using this scheme the robot can be initialised with random weights (i.e. no mapping between sensors and motor control) and within a few epochs obtains basic obstacle avoidance competence.

It is a relatively easy matter to promote more complex behaviour with the addition of other rules. For example to achieve a wall following behaviour a third

rule is introduced as shown in column two of table 2. Navigational tasks can be accomplished with the addition of a **"maximise navigational signal"** rule. An example of this is a light sensor mounted on the robot producing a behaviour to move towards a light source. Equally, a signal from a more complex, higher level, navigational system could be used. Thus the instinct rule controller handles basic obstacle avoidance competence and motor/sensory interface tasks leaving other resources free for intensive navigational tasks.

4.2 INSTINCT RULE EVALUATION USING SOMATIC TENSION

The original instinct rule robot used binary sensor signals and evaluated performance of alternative actions for fixed, and progressively longer, periods of time [Nehmzow, 1992]. With the EPC interfacing directly to analog sensors an improved scheme has been developed. If we sum all sensors onto a neuron with fixed and equal weights we gain a measure of total sensory activity. Let us call this *somatic tension* as an analogy to biological signal aggregation on the soma. If we have an instinct violation and an alternative action is performed we can monitor this somatic tension to gauge the performance of this action. If tension decreases significantly we continue the action. If it increases significantly we choose an alternative action. If tension remains high and roughly the same, we are in a *tight* situation, for example say a corner. In this case we perform actions for progressively longer periods continuing to monitor somatic tension for a drop.

4.3 RESULTS AND DISCUSSION

The instinct rule robot has been constructed and its performance is comparable with software-controlled predecessors. Unfortunately direct comparisons are not possible due to unavailability of the original exemplars and differing physical characteristics of the robots themselves. In developing the application several observations were made concerning the behaviour of the system that would not have come to light in a simulated environment.

In any system including real mechanics and real analog signals, imperfections and noise are present. For example, in a real robot we cannot guarantee that a forward motion directive will result in perfect forward motion due to inherent asymmetries in the system. The instinct rule architecture does not assume a-priori knowledge such as this so behaviour is not affected adversely. This was tested by retarding one drive motor of the robot to give it a bias to one side.

In early development, as the monitor was being *tuned*, the robot showed a tendency to oscillatory motion, thus exhibiting undesirable behaviour that satisfies its instincts. It could, for example, oscillate back and forth at a corner. In a simulated environment this continues indefinitely. However, with real mechanics and noisy analog sensors the robot breaks out of this undesirable behaviour.

These observations strengthen the arguments for hardware development aimed at embedded systems. The robot application is but an example of the different, and often surprising conditions that pertain in a "real" system. If neural networks are to find applications in real-world, low-cost and analog-interface applications, these are the conditions we must deal with, and appropriate, analog hardware is the optimal medium for a solution.

5 CONCLUSIONS

This paper has described pulse stream neural networks that have been developed to a system level to aid development of applications. We have therefore defined areas of strengths of this technology along with suggestions of where this is best applied. The strengths of this system include:

1. Direct interfacing to analog signals.
2. The ability to fuse direct analog sensor data with digital sensor data processed elsewhere in the system.
3. Distributed processing. Several EPC's may be embedded in a system to allow multiple networks and/or multi layer networks.
4. The EPC represents a flexible system level development environment. It is easily reconfigured for new applications or improved chip technology.
5. The EPC requires very little computational overhead from the host system and can operate independently if needed.

A demonstration application of an instinct rule robot has been presented highlighting the use of neural networks as an interface between real-world analog signals and digital control.

In conclusion we believe that the immediate future of neural analog VLSI is in small applications based systems that interface directly to the real-world. We see this as the primary niche area where analog VLSI neural networks will replace conventional digital systems.

Acknowledgements

Thanks are due to Ulrich Nehmzow, University of Manchester, for discussions and information on the instinct-rule controller and the loan of his original robot – Alder.

References

[Caudell, 1990] Caudell, M. and Butler, C. (1990). *Naturally Intelligent Systems.* MIT Press, Cambridge, Ma.

[Jackson, 1994] Jackson, G., Hamilton, A., and Murray, A. F. (1994). Pulse stream VLSI neural systems: into robotics. In *Proceedings ISCAS'94*, volume 6, pages 375–378. IEEE Press.

[Maren, 1990] Maren, A., Harston, C., and Pap, R. (1990). *Handbook of Neural Computing Applications.* Academic Press, San Diego, Ca.

[Murray, 1992] Murray, A. F., Baxter, D. J., Churcher, S., Hamilton, A., Reekie, H. M., and Tarassenko, L. (1992). The Edinburgh pulse stream implementation of a learning-oriented network (EPSILON) chip. In *Neural Information Processing Systems (NIPS) Conference.*

[Nehmzow, 1992] Nehmzow, U. (1992). *Experiments in Competence Acquisition for Autonomous Mobile Robots.* PhD thesis, University of Edinburgh.

[Widrow, 1988] Widrow, B. (1988). *DARPA Neural Network Study.* AFCEA International Press.

G Jackson, A F Murray, and A Hamilton. "Pulse stream neural networks for applications", In *Australian Conference on Neural Networks*, 1995.

Pulse Stream Neural Networks for Applications

Geoff Jackson
Department of Electrical Engineering
University of Edinburgh
Mayfield Road, EH9 3JL
United Kingdom

Alister Hamilton and Alan F. Murray
Department of Electrical Engineering
University of Edinburgh
Mayfield Road, EH9 3JL
United Kingdom

***Abstract*— Current research at the University of Edinburgh is focused on developing pulse stream neural systems to operate on the boundary between the analog sensory environment and that of conventional digital processors. The issues of where, how, and why pulse stream neural hardware should be applied are examined in this paper. We present here a chip, EPSILON II[1] and a processor card incorporating it, that has been designed to bring pulse stream neural hardware to bear on real applications. As an example, an autonomous mobile robot is described.**

I. Introduction

NEURAL networks have been heralded as an effective solution for a diverse range of real-world problems[1], [2]. Yet despite this, applications and especially hardware implementations have been few and slow to emerge. For instance in the DARPA[2] neural networks study of 1988; of the 77 neural network applications investigated only 4 had resulted in field tested systems[3]. Furthermore, none of these used dedicated neural network hardware. It is our view that this lack of tangible successes can be summarised by the following points:

- Most neural applications will be served optimally by fast, generic digital computers.
- Dedicated digital neural accelerators have a limited lifetime as "the fastest", as standard computers develop so rapidly.
- Analog neural VLSI is a niche technology, optimally applied at the interface between the real world and higher-level digital processing.

This attitude has some profound implications with respect to the size, nature and constraints we place on new hardware neural designs. After several years of research into hardware neural network implementation, have now concentrated on the areas in which analog neural network technology has an "edge" over well established digital technology.

Clearly, neural network technology must compete with more conventional digital techniques in solving real-world problems, and neural networks must concentrate on areas where their advantages are most prominent, and their disadvantages (the inability to interrogate a solution fully, for example) are least bothersome.

Within the pulse stream neural network research at the University of Edinburgh, the EPSILON chip's areas of strength can be summarised as:

- Analog or digital inputs, digital outputs.
- Compact, low power.
- Modest size.
- Scaleable and cascadeable design.

This list points naturally and strongly to problems on the boundary of the real, analog world and digital processing, such as pre-processing/interpretation of analog sensor data. Here a modest neural network can act as an *intelligent analog-to-digital converter* presenting preprocessed information to its host. It is our conclusion that this is an area where analog neural networks will make the most significant impact. We are now engaged in a two pronged approach, whereby development of technology to improve the performance of pulse stream neural network chips is occurring concurrently with a search and development of applications to which this technology can be applied.

The key requirements of this technological development are that devices must:

- Work directly with analog signals.
- Provide a moderate size network to process data for further digital processing.
- Have the potential for a fully integrated solution.

The next section describes the EPSILON II chip, or more specifically, the features of the chip that have been developed to make the hardware more amenable to use in real applications. The following section examines the system level considerations and the specifics of the EPSILON processor card (EPC), a flexible environment for applications and chip level development. Finally the nature of appropriate applications is discussed and a demonstration application of an autonomous mobile robot is presented.

II. The EPSILON II Chip

The EPSILON II chip has been designed around the requirements of an application based system. It follows on from an earlier generation of pulse stream neural network chip, the EPSILON chip[4].

The EPSILON II chip represents neural states as a pulse encoded signal. These pulse encoded signals have digital signal levels which make them highly noise immune and ideal for inter and intra-chip communication, facilitating efficient cascading of chips to form larger systems. The EPSILON II chip can take as inputs either pulse encoded signals or analog voltage levels, thus facilitating the fusing of analog and digital data in one system. Internally the

email: gbj@ee.ed.ac.uk

[1] Edinburgh Pulse Stream Implemenation of a Learning Oriented Network.

[2] Defence Advanced Research Projects Agency

chip is analog in nature allowing the synaptic multiplication function to be carried out in compact and efficient analog cells[5].

EPSILON II Chip Specifications	
No. of state input pins	32
Input modes	Analog, PW or PF
Input mode programmability	Bit programmable
No. of state outputs	32 pinned out
Output modes	PW or PF
Digital recovery of analog I/P	Yes - PW encoded
No. of Synapses	1024
Additional *autobias* synapses	4 per output neuron
No. of weight load channels	1
Weight load time	$2.3ms$
Weight storage	Dynamic
Programmable activity voltage	Yes
Maximum speed (cps)	$102.4Mcps$
Technology	ES2 $1.5\mu m$ CMOS
Die size	$6.9mm \times 7mm$
Packaging	120 pin PGA
Maximum power dissipation	$320mW$

TABLE I
EPSILON II SPECIFICATIONS

Table I shows the principal specifications of the EPSILON II chip. The EPSILON II chip is based around a 32x32 synaptic matrix allowing efficient interfacing to digital systems. A plot of the layout of the chip (figure1) shows the structure of, and the signal flow within the chip. Several features of the device have been developed specifically for applications based usage. The first of these is a programmable input mode. This allows each of the network inputs to be programmed as either a direct analog input or a digital pulse encoded input. We believe that this is vital for application based usage where it is often necessary to *fuse* real-world analog data with historical or control data generated digitally. The second major feature is a pulse recovery mode. This allows conversion of any analog input into a digital value for direct use by the host system. Such a facility is necessary if learning is to be done with the system in operation using say the back propagation algorithm as input state values are needed for learning.

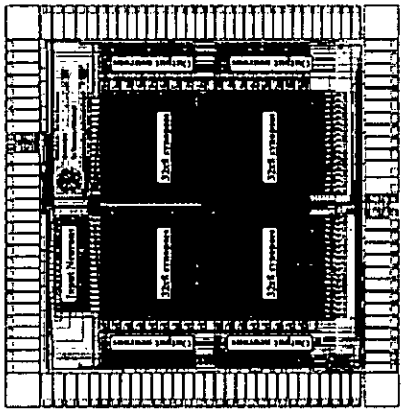


Fig. 1. EPSILON II Chip Layout.

Other concurrent work in the neural group in Edinburgh seeks to make future chips more "application friendly", by using amorphous silicon for non-volatile weight storage [6] and developing on-chip learning circuits to render chips more autonomous[7].

An example of the characteristics of the EPSILON II device is shown in figure 2. This plot shows the characteristics of an individual synapse/neuron on the chip, as a plot of output pulse width against the input range for various weight values. This characteristic represents a significant improvement over the earlier EPSILON pulse stream neural network chip[4]. This improvement arises from careful layout and architecture changes while still using the same basic circuits.

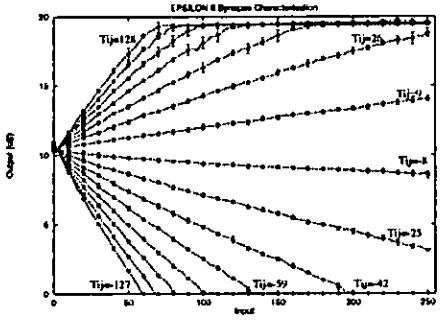


Fig. 2. EPSILON II Synapse Characteristics.

III. EPSILON PROCESSOR CARD

The need to embed the EPSILON chip in a processor card is driven by several considerations. Firstly, working with pulse encoded signals requires substantial processing to interface directly to digital systems. If the neural processor is to be transparent to the host system and is not to become a substantial processing overhead, then all pulse support operations must be carried out independently of the host system. Secondly, to respond to further chip level advances and allow rapid prototyping of new applications as they emerge, a certain amount of flexibility is needed in the system. It is with these points in mind that the design of the flexible EPSILON Processor Card (EPC) was

undertaken.

A. Design Specification

The EPC has been designed to meet the following specifications. The card must:

- Operate on a conventional digital bus system.
- Be transparent to the host processor, that is carry out all the necessary pulse encoding and decoding.
- Carry out the refresh operations of the dynamic weights stored on the EPSILON chip.
- Generate the ramp waveforms necessary for pulse width coding.
- Support the operation of multiple EPC's.
- Allow direct input of analog signals.

As all data used and generated by the chip is effectively of 8-bit resolution, the STE bus, an industry standard 8-bit bus, was chosen for the bus system. This is also cost effective and allows the use of readily available support cards such as processors, DSP cards and analog and digital signal conditioning cards.

To allow the transparency of operation the card must perform a variety of functions. A block diagram indicating these functions is shown in figure 3.

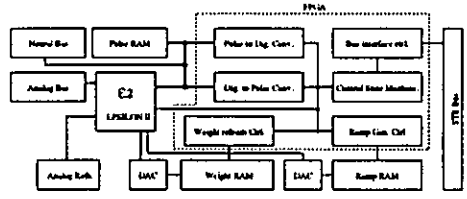


Fig. 3. EPSILON Processor Card

A substantial amount of digital processing is required by the card, especially in the pulse conversion circuitry. To conform to the *Eurocard* standard size of the STE specification an FPGA device is used to "absorb" most of the digital logic. A twin mother/daughter board design is also used to isolate sensitive analog circuitry from the digital logic. The use of the FPGA makes the card extremely versatile as it is now easily reconfigurable to adapt to specialist application. The dotted box of figure 3 shows functions implemented by the FPGA device. An on board EPROM can hold multiple FPGA configurations such that the board can be reconfigured "on the fly". All EPSILON support functions, such as ramp generation, weight refresh, pulse conversion and interface control are carried out on the card. Also the use of the FPGA means that new ideas are easily tested as all digital signal paths go via this device. Thus a card of new functionality can be designed without the need to design a new PCB.

B. Specialist Buses

The digital pulse bus is buffered out under control of the FPGA to the neural bus along with two control signals. Handshaking between EPC's is done over these lines to allow the transfer of pulse stream data between processors. This implies that larger networks can be implemented with little or no increase in computation time or overhead.

A separate analog bus is included to bring analog inputs directly onto the chip.

C. Future Extensions.

As all control and pulse stream signals are generated by the FPGA the EPC stands ready to accept the next generation in the EPSILON chipset. By judicious chip design, chips incorporating on-chip learning or non-volatile analog storage currently being developed at Edinburgh (see [8]) will readily plug into the EPC for evaluation in a stable environment.

IV. Applications

The over-riding reason for the development of the EPC is to allow the easy development of hardware neural network applications. We have already indicated that we believe that this form of neural technology will find its niche where its advantages of direct sensor interface, compactness and cost-effectiveness are of prime importance. As a good and intrinsically interesting example of this genre of applications, we have chosen autonomous mobile robotic control as a first test for EPSILON II. The object of this demonstrator is not to advance the state-of-the-art in robotics. Rather it is to demonstrate analog neural VLSI in an appropriate and stimulating context.

The robot itself is of a form that could perform simple tasks (pipe-following, for example) in an unknown environment, or have the ability to "get out of trouble" when a higher-level camera-based control system fails. The ultimate reason for the development of the EPC is to allow the easy development of hardware neural network applications.

A. "Instinct" Rule Robot

The "instinct-rule" robotic control philosophy is based on a proven software-controlled exemplar in the University's Department of Artificial Intelligence [9] (see Figure 4). The robot will incorporate an EPC to implement the essential programmable neural link between the analog sensors and the drive actuators that underpins the robot's adaptive behaviour.

The original instinct-rule robot used two *feelers* mounted at the front of the robot and a simple detector on the front free-rotating castor to register forward motion. The *feelers* are implemented as simple binary switches giving the robot an indication of obstacles in its path. A hard-wired network determines any instinct rule violations, this then supplies a training signal to the neural network linking sensor data to drive motors to train the network to avoid these rule violations. Instinct-rules such as "keep crash sensors inactive", "get bored – change direction" allow the robot to learn corridor following. The additional use of historical information allows maze following tasks to be accomplished.

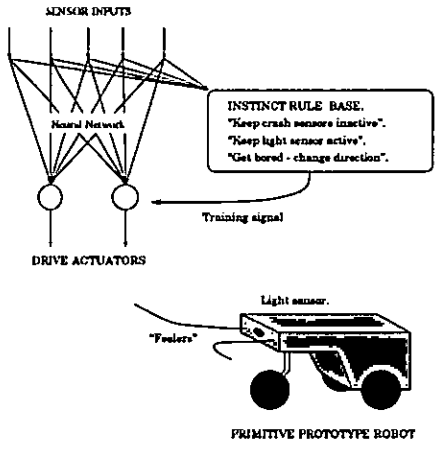


Fig. 4. "Instinct Rule" Robot

A directional instinct allows the robot to carry out navigational tasks. This could be in the form of following a light source. A photo-sensor mounted on the robot together with the instinct rule "keep light sensor active" can be used to achieve this. Alternatively the directional information may come from some higher level navigational controller. In this way the the instinct rule controller handles all *low level* behaviour such as avoiding obstacle or dangerous situations while a higher level controller determines navigational tasks.

Our intention is to extend the sensitivity and range of sensors interfaced to the neural network and increase the scope of the instinct-rules. Using analog sensor data directly means the use of more complex and numerous sensors can be easily achieved.

V. Discussion

This paper has discussed the use of pulse stream neural networks in practical applications. The paper has two main aims:

- To present new results from a novel analog neural chip.
- To offer reasoned opinions regarding the optimal use of neural analog VLSI.

We have therefore defined areas of strengths of this technology along with suggestions of where this is best applied.

To aid the development of practical applications the EPSILON II chip and the EPSILON Processor Card have been designed. These resources have been designed to process data on the boundary between the analog real-world and the digital world of conventional computing. The analog VLSI nature of the neural hardware make it extremely versatile for this type of purpose. Reasons for this include:

1. Direct interfacing to analog signals.
2. The ability to fuse direct analog sensor data with digital sensor data processed elsewhere in the system.
3. Distributed processing. Several EPC's may be embedded in a system to allow multiple networks and/or multi layer networks.
4. Speed. Guaranteed calculation times (as per Table I). The speed of software solutions is not so readily defined or achievable in a compact unit. This has implications for real-time applications.
5. The EPC represents a flexible system level development environment.
6. The EPC requires very little computational overhead from the host system and can operate independently if needed.
7. The flexibility of the EPC with major digital functions carried out in programmable logic means that it is easily reconfigured for new applications or improved chip technology.

It is envisaged that the robot control application of the EPC will be the first amongst many. Further advances in non-volatile analog memory technology and on-chip learning currently being investigated at Edinburgh University will further enhance the capabilities of our neural network VLSI.

In conclusion we believe that the immediate future of neural analog VLSI is in small applications based systems that interface directly to the real-world. We see this as the niche area where the VLSI neural networks can compete most effectively with conventional digital systems.

Acknowledgements

Geoff Jackson would like to thank the Commonwealth Scholarship Commission and the British Council for financial support. The early stages of this work has also enjoyed support from the Engineering and Physical Sciences Research Council (EPSRC).

References

[1] A Maren, C Harston, and R Pap, *Handbook of Neural Computing Applications*, Academic Press, San Diego, Ca, 1990.

[2] M Caudell and C Butler, *Naturally Intelligent Systems*, MIT Press, Cambridge, Ma, 1990.

[3] B Widrow, *DARPA Neural Network Study*, AFCEA International Press, Nov. 1988.

[4] Alan F. Murray, Donald J. Baxter, S. Churcher, A. Hamilton, H. Martin Reekie, and L. Tarassenko, "The Edinburgh pulse stream implementation of a learning-oriented network (EPSILON) chip", in *Neural Information Processing Systems (NIPS) Conference*, 1992.

[5] Geoff Jackson, Alister Hamilton, and Alan F Murray, "Pulse stream VLSI neural systems: into robotics", in *Proceedings ISCAS'94*, May 1994, vol. 6, pp. 375-378, IEEE Press.

[6] A.J. Holmes, R.A.G. Gibson, et al., "Use of a-Si:H memory devices for non-volatile weight storage in artificial neural networks", in *15th International Conference on Amorphous Semiconductors*, 1993.

[7] R Woodburn, H. Martin Reekie, and Alan F. Murray, "Pulse stream circuits for on-chip learning in analogue VLSI neural networks", in *Proceedings ISCAS'94*, May 1994, vol. 4, pp. 103-106.

[8] Alan F. Murray, S. Churcher, A. Hamilton, A.J. Holmes, G.B. Jackson, and R.J. Woodburn, "Applications of pulsed neural VLSI", *IEEE MICRO*, 1994.

[9] U. Nehmzow, *Experiments in Competence Acquisition for Autonomous Mobile Robots*, PhD thesis, University of Edinburgh, 1992.

G Jackson, A Hamilton, and A F Murray. "The EPSILON processor card: A framework for analog neural computation", In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems.*, pages 280–286. 1994.

The EPSILON Processor Card: A Framework for Analog Neural Computation

Geoff Jackson*

Department of Electrical Engineering
University of Edinburgh
Mayfield Road, EH9 3JL
United Kingdom

Alister Hamilton and Alan F. Murray

Department of Electrical Engineering
University of Edinburgh
Mayfield Road, EH9 3JL
United Kingdom

Abstract

This paper presents new advances in pulse stream neural processing. These advances are made possible by a new neural chip EPSILON II[1]. EPSILON II *is made using a standard digital ES2 process and though essentially analog in nature exhibits process invariance. The new chip has been designed specifically with a system level view in mind. The* EPSILON Processor Card (EPC) *a powerful neural processing tool with the* EPSILON II *chip at its heart is presented for the first time. Multiple EPC's can be used in parallel to implement large networks and can take inputs directly from the analog world.*

To demonstrate the flexibility of the EPC an implementation of an autonomous mobile robot is under construction using the card.

1 Introduction

Like their biological counterparts, pulse stream networks use pulse code modulation to represent the neural states of the network. Over the years much research has been undertaken in this field, both at Edinburgh [1, 2] and elsewhere, for example[3].

Pulse stream networks offer many advantages when applied to VLSI hardware implementations. The essentially digital levels of pulse stream signals render them largely immune to noise. This, together with simple buffering, make them ideal for inter- and intra-chip communications.

Of course no advantage in engineering is without its cost. Pulse stream systems, as with any other modulation scheme, requires signals to be encoded and decoded to present and recover information to and from the network. The new chip, EPSILON II, and the EPSILON Processor Card (EPC) presented here have been designed to make this process as transparent as possible to the host system.

It is envisaged that analog VLSI neural networks will perform their most useful tasks on the boundary between the analog and digital world. For this reason EPSILON and the EPC have the ability to fuse both digital and *real world* analog data together to work on this boundary. To demonstrate this approach the EPC is being used in a robotic control application fusing digital and real-world analog sensor data to carry out the task.

2 The EPSILON II Chip

The EPSILON II chip is a descendent of the EPSILON chip developed at the University of Edinburgh. Internal circuitry of the chip operates on the same principles of this earlier chip and has been reported elsewhere[4] and only a brief summary will be given here. The new features of the EPSILON II chip are presented along with the characterisation results gained from the chip. A comparison with EPSILON is also given.

2.1 The EPSILON Synapse

The synapse circuit incorporates a standard transconductance multiplier previously used in analog signal processing applications[5]. Figure 1 shows the synapse circuit. Here the transistors M1 and M2 are operated in their linear region to produce an output current proportional to the weight voltage T_{ij}.

*email: gbj@ee.ed.ac.uk

[1] Edinburgh Pulse Stream Implementation of a Learning Orientated Network

This current is gated through M3 under control of the neural state, V_j, to dump charge packets on a summation line common to all synapses of a neuron.

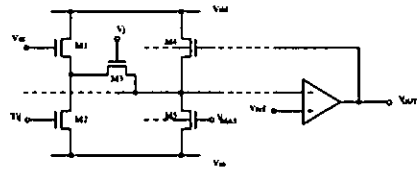


Figure 1: The EPSILON Distributed Feedback Synapse.

The transistors M4 & M5 along with the common op-amp form a distributed feedback system which ensures that drain-source voltages of M1 & M2 remain constant and equal. This has the effect of cancelling non-linearities in the transistors responses making I_{out} linearly dependent on $V_{g,1}$ and $V_{g,2}$[6]. V_{zt} is a zero weight reference derived from automatic bias circuitry as described in [7]. V_{out} represents the combined activity of the synapses at a particular moment in time. This voltage is integrated by a voltage integrator to produce the total synaptic activity.

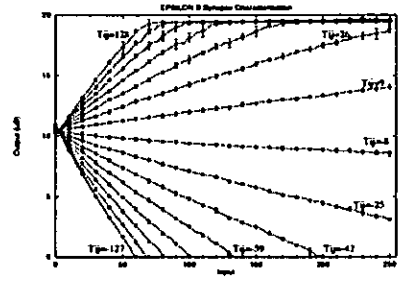


Figure 2: Synapse Multiplier Characteristics of EPSILON II.

Figure 2 show characterisation results of the EPSILON II synapse. The figure shows the output pulse width over the input state range for different weight voltages. The figure demonstrates the excellent linearity with respect to input state gained from the distributed feedback synapse. The plot represents an average of 50 excitations and the associated standard deviations are essentially constant and of the order of 120ns. This characteristic represents a significant improvement over the original EPSILON design (see [2]) due to judicious architectural and analog performance improvements incorporated into EPSILON II.

2.2 EPSILON Neurons

When considering pulse stream systems, two modulation schemes spring naturally to mind, pulse width modulation (PWM) and pulse frequency modulation (PFM).

PWM is essentially *synchronous* in nature as it entails a fixed period. PFM on the other hand is *asynchronous*. Neurons performing both these modulation schemes are incorporated in EPSILON II. The choice of which modulation scheme is used depends greatly on the problem at hand. The asynchronous, temporal characteristics of PFM make it the scheme of choice for feedback or recurrent networks. The synchronous characteristics of PWM guarantee a maximum computation time thus making it ideal for high speed, repetitious applications such as vision processing and control.

2.3 Pulse Width Neurons

The result of the multiplication of neural state by synaptic weight is a voltage stored on a capacitor. This voltage may be converted into a pulse width modulated signal by the use of a global ramp signal and comparator as shown in Figure 3. The use of a dual sloped ramp prevents a large number of synchronised switching transients, reducing power supply spikes and therefore noise within the device. While this approach is simple and flexible – any neuron transfer characteristic may be employed – it currently requires off-chip RAM and DAC circuitry to generate the ramp signals.

2.4 Pulse Frequency Neurons

The pulse frequency neuron circuit (Figure 4) uses a non-symmetrical differential input stage to perform a linear voltage to "sigmoid" current conversion. This conversion results in a "sigmoid" activity voltage to duty cycle neuron transfer characteristic.

Phase lock loop techniques have been employed [7, 2] to fix the capacitor charge current, I_H, and therefore the output pulse width of the neuron to a constant 1 microsecond.

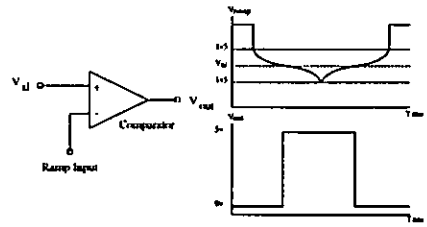


Figure 3: Pulse-Width Modulation Neuron

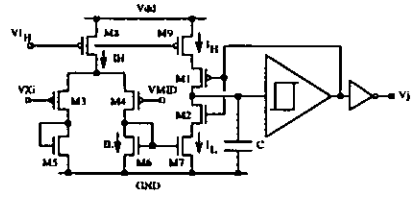


Figure 4: Pulse-Frequency Modulation Neuron

The current, I_L, determined by the differential input voltages is used to discharge the capacitor and sets the output pulse spacing.

A further extension of this circuit (not shown in Figure 4 for clarity, but implemented on EPSILON[7]) allows limited electronic gain control of the sigmoid characteristic using phase lock loop techniques.

2.5 EPSILON II Input/Output Modes and Pulse Recovery

We have already discussed EPSILON II's various output modes, that is either pulse width or pulse frequency. The versatility of EPSILON II is further enhanced by offering a choice of input modes. The two choices offered are either a digital pulse stream input or an analog voltage. Each input is individually programmable to either of these modes to allow both analog and digital data to be *fused* within the network.

The pulse stream data can be either PW or PF modulated. Its source can be pulse streams generated by other circuitry, other EPSILON II chips or feedback connections from the current output states to allow recurrent configurations. The analog input range is a voltage between 0-5V. Internally within the chip these values are converted to pulse widths via a global ramp and comparator scheme similar to that used in the pulse width neuron (see section 2.3).

In addition, the chip incorporates a "pulse recovery" mode. This allows pulse widths from analog conversions to be fed off chip via the neuron outputs to recover the analog values. Thus multiple analog values can be converted to digital form in parallel in a very efficient manor, a highly desirable feature when many learning algorithms require knowledge of input states to operate.

2.6 EPSILON II SPECIFICATIONS

The EPSILON II chip was fabricated using the European Silicon Structures (ES2) ECPD15 ($1.5\mu m$ double metal n-well) CMOS process. The dimensions of the EPSILON II chip were chosen to be 32 inputs by 32 outputs giving a synaptic array of 1024 synaptic connections. This configuration was chosen to enhance the ease with which EPSILON II may be interfaced to external hardware while maintaining a "useful" size. A table showing the salient features of the EPSILON II device and comparing them with the original EPSILON chip is given in table 1

3 Neural Processing at System Level

Let us look at the relationship between analog VLSI neural networks in general, EPSILON in particular, and conventional computing:

The EPSILON chip is a neural array. It contains (at present) no in-built learning mechanism or algorithm. At system level this overall control on how the network evolves or learns is the responsibility of some other conventional processing device.

EPSILON can take as input direct analog signals. Fed through a trained network, EPSILON can perform some type of classification, recognition or preprocessing task and present its results in digital form. This data can be acted on by conventional digital processors or used as a starting point for further processing. Thus at system level, the neural processor is only one part of the whole. An example of this system level framework wherein the neural network resides is shown in figure 5 Choosing a standard bus system allows the use of cheap, powerful and readily available digital systems to be flexibly integrated with the neural hardware. To maximise the efficiency of such a system the operation of the neural processor

EPSILON Chip Specifications		
	EPSILON I	EPSILON II
No. of State Input Pins	30	32
No. of Actual State Inputs	120, MUX'd in banks of 30	32
Input Modes	analogue, PW or PF	analogue, PW or PF
Input Mode Programmability	All analogue/All digital	Bit programmable
No. of State Outputs	30 Directly Pinned Out	32 Directly Pinned Out
Output Modes	PW or PF	PW or PF
Digital Recovery of Analogue Inputs	No	Yes - PW Modulated
No. of Synapses	3600	1024
Additional *Autobias* Synapses	None	4 per output neuron
No. of Weight Load Channels	2	1
Weight Load Time	$3.6ms$	$2.3ms$
Weight Storage	Dynamic	Dynamic
Programmable Activity Voltage	No	Yes
Maximum Speed (cps)	$360Mcps$	$102.4Mcps$
Technology	$1.5\mu m$ CMOS	$1.5\mu m$ CMOS
Die Size	$9.5mm \times 10.1mm$	$6.9mm \times 7mm$
Packaging	144 pin PGA	120 pin PGA
Maximum Power Dissipation	$350mW$	$320mW$

Table 1: Comparison of EPSILON I and EPSILON II Specifications

must be transparent to the rest of the system, to this end the EPC has been designed.

4 The EPSILON Processor Card

The principle overhead of a pulse stream system arises from the large amount of data communication and data conversion that this modulation entails. To prevent this becoming the principle processing bottleneck, hardware must be dedicated to supporting this pulse code modulation and control such that the neural computation is transparent to the host processor. Other support functions necessary for operation, such as weight refresh and ramp generation, must also be handled locally. To this end the EPSILON Processor Card (EPC) has been designed.

4.1 Design Specification

The EPC has been designed to meet the following specifications:

- Operate on a conventional digital bus system.
- Be transparent to the host processor, that is carry out all the necessary pulse encoding and decoding.
- Carry out the refresh operations of the dynamic weights stored on the EPSILON chip.
- Generate the ramp waveforms necessary for pulse width coding.
- Support the operation of multiple EPC's such that larger networks or multilayer networks can be synthesised using pulse code communication.
- Allow direct input of analog signals as an interface to the *real world*.

As all data used and generated by the chip is effectively of 8-bit resolution, the STE bus, an industry standard 8-bit bus, was chosen for the bus system. This is also cost effective and allows the use of readily available support cards such as processors, DSP cards and analog and digital signal conditioning cards.

To allow the transparency of operation the card must perform a variety of functions. A block diagram indicating these functions is shown in figure 6.

A substantial amount of digital processing is required by the card, especially in the pulse conversion circuitry. To conform to the *Eurocard* standard size of the STE specification an FPGA device is used to "soak" up most of the digital logic. A twin mother/daughter board design is also used to isolate sensitive analog circuitry from the digital logic. The

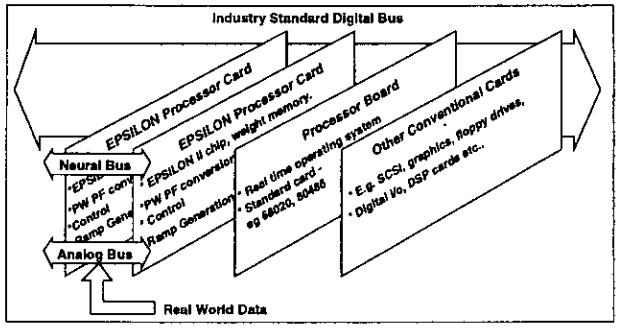


Figure 5: A system level framework for neural processing.

use of the FPGA makes the card extremely versatile as it is now easily reconfigurable to adapt to specialist application. The dotted box of figure 6 shows functions implemented by the FPGA device. An on board EPROM can hold multiple FPGA configurations such that the board can be reconfigured "on the fly". All EPSILON support functions, such as ramp generation, weight refresh, pulse conversion and interface control are carried out on the card. Also the use of the FPGA means that new ideas are easily tested as all digital signal paths go via this device. Thus a card of new functionality can be designed without the need to design a new PCB.

4.2 Pulse Conversion Techniques

The basic logic needed for the pulse width conversion hardware is shown in figure 7. For 32 channels of information as on the EPSILON II chip a parallel implementation of this would require 32x8-bit counters, 32x8-bit comparators plus the ramp generator. Despite the fact that the Xilinx FPGA used has the equivalent of approximately 5000 gates and 768 flip-flops, a fully parallel implementation is not practical. Thus a multiplexing scheme is used to carry out blocks of conversions. The FPGA can be reconfigured to offer the greatest degree of parallelism consistent with the control requirements of any particular application. For the worst case, that is a totally serial approach, a conversion takes 32x256 clock cycles or $341\mu s$ at $24MHz$. Done in parallel blocks of 8 channels the conversion takes $85\mu s$.

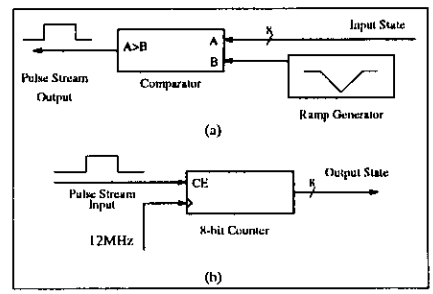


Figure 7: (a) Binary to pulse width conversion. (b) Pulse width to binary conversion.

4.3 Specialist Buses

The digital pulse bus is buffered out under control of the FPGA to the neural bus along with two control signals. Handshaking between EPC's is done over these lines to allow the transfer of pulse stream data between processors. This implies that larger networks can be implemented with little or no increase in computation time or overhead.

A separate analog bus is included to bring analog inputs directly onto the chip.

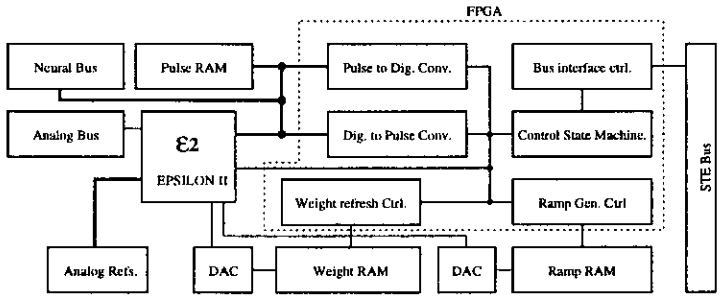


Figure 6: EPSILON Processor Card

4.4 Future Extensions.

As all control and pulse stream signals are generated by the FPGA the EPC stands ready to accept the next generation in the EPSILON chipset. By judicious chip design, chips incorporating on-chip learning or non-volatile analog storage currently being developed at Edinburgh (see [8]) will readily plug into the EPC for evaluation in a stable environment.

5 Applications

The ultimate reason for the development of the EPC is to allow the easy development of hardware neural network applications. The first test of this philosophy is in autonomous mobile robotic control.

5.1 "Instinct" Rule Robot

The "instinct-rule" robotic control philosophy is based on a software-controlled exemplar in the University's Department of Artificial Intelligence [9] (see Figure 8). The robot will incorporate an EPC to implement the essential programmable neural link between the analogue sensors and the instinct rule base that underpins the robot's adaptive behaviour.

The original instinct-rule robot used two *feelers* mounted at the front of the robot and a simple detector on the front free-rotating castor to register forward motion. The *feelers* are implemented as simple binary switches giving the robot an indication of

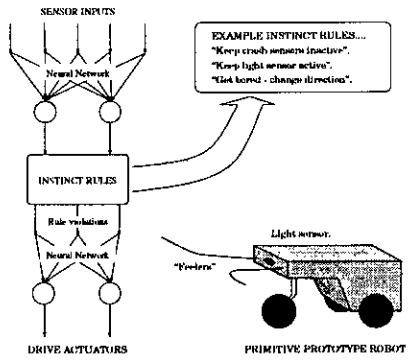


Figure 8: "Instinct Rule" Robot

obstacles in its path. A pattern associator neural network links the sensor data to the instinct-rule controller. Instinct-rules such as "keep crash sensors inactive", "get bored – change direction" allow the robot to learn corridor following. The additional use of historical information allows maze following tasks to be accomplished.

Our intention is to extend the sensitivity and range of sensors interfaced to the neural network and increase the scope of the instinct-rules. For example, the use of force sensitive resistors as bend sensors will allow the implementation of an analogue feeler. The

use of a small camera or an array of photo-sensors and appropriate instinct-rules such as "keep light sensor active" will allow the robot to learn tasks such as following a light source.

6 Discussion

The EPSILON II chip and the EPSILON Processor Card have been designed to process data on the boundary between the analog real-world and the digital world of conventional computing. The analog VLSI nature of the neural hardware make it extremely versatile for this type of purpose. Reasons for this include:

1. Direct interfacing to analogue signals without the requirement for analogue-to-digital converters and analogue signal multiplexing. This has implications for the size, speed and power consumption of the system.

2. The ability to fuse direct analogue sensor data with digital sensor data processed elsewhere in the system. In the case of the robot application, this may be historical sensor data or data conventionally processed from a camera.

3. Hardwired neural algorithm. There is no requirement to *program* the neural algorithm in software as it is hardwired in VLSI. Learning is currently performed off-chip by a host processor.

4. Distributed processing. Several EPC's may be embedded in a system to allow multiple networks and/or multi layer networks. The real-time applications environment described makes this an attractive possibility.

5. Speed. Guaranteed calculation times defined in Table 1 in *connections per second*. The speed of software solutions is not so readily defined or achievable in a compact unit. This has implications for real-time applications.

6. A flexible system level development environment has evolved where additional EPC's can be easily incorporated to increase neural processing power.

7. The EPC requires very little computational overhead from the host system allowing efficient solutions to problems where a neural processor can perform significant portions of the computational processing.

It is envisaged that the robot control application of the EPC will be the first amongst many. The flexible real-time environment has been designed with this consideration. Further advances in non-volatile analogue memory technology and on-chip learning currently being investigated at Edinburgh University will further enhance the capabilities of our neural network VLSI.

References

[1] A. F. Murray, "Pulse arithmetic in VLSI neural networks," *IEEE MICRO*, vol. 9, no. 6, pp. 64–74, 1989.

[2] A. Hamilton, S. Churcher, P. J. Edwards, G. B. Jackson, A. F. Murray, and H. M. Reekie, "Pulse stream VLSI circuits and systems: The EPSILON neural network chipset," *International Journal of Neural Systems*, vol. 4, pp. 395–405, Dec 1993.

[3] 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.

[4] S. Churcher, D. J. Baxter, A. Hamilton, A. F. Murray, and H. M. Reekie, "Generic analog neural computation – the EPSILON chip," *Neural Information Processing Systems (NIPS) Conference*, pp. 773–780, 1993.

[5] P. Denyer and J. Mavor, "MOST transconductance multipliers for array applications," *IEE Proc. Pt. 1*, vol. 128, pp. 81–86, June 1981.

[6] 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 Transactions on Neural Networks*, pp. 385–393, 1992.

[7] A. Hamilton, *Pulse Stream Circuits and Techniques for the Implementation of Neural Network*. PhD thesis, University of Edinburgh, 1993.

[8] A. F. Murray, S. Churcher, A. Hamilton, A. Holmes, G. Jackson, and R. Woodburn, "Applications of pulsed neural VLSI," *IEEE MICRO*, 1994.

[9] U. Nehmzow, *Experiments in Competence Acquisition for Autonomous Mobile Robots*. PhD thesis, University of Edinburgh, 1992.

References

[1] Said Abu-Mostafa. Hints. *Neural Computation*, 7:639–671, 1995.

[2] Adaptive Solutions Inc., 1400 N.W. Compton Drive, Beverton, OR 97006. *Getting Acquainted with CNAPS*, December 1994. PN 801-30008-04. Available at http://www.asi.com.

[3] Adaptive Solutions Inc., 1400 N.W. Compton Drive, Beverton, OR. *CNAPS Data Book*, March 1995. PN: 801-20063-03. Available http://www.asi.com.

[4] Phillip E Allen and Douglas R Holberg. *CMOS Analog Circuit Design*, chapter 6, pages 273–287. Holt, Rinehart and Winston, INC., 1987.

[5] J Alspector, R Meir, B Yuhas, A Jayakumar, and D Lippe. A parallel gradient descent method for learning in analog VLSI neural networks. In *Advances in Neural Information Processing Systems*, volume 5, pages 836–844. Morgan Kaufmann, 1993.

[6] Charles W Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, pages 31–37, April 1989.

[7] Andreas G Andreou, Kwabena A Boahen, Philippe O Pouliquen, Aleksandra Pavasović, Robert E Jenkins, and Kim Stronbehn. Current-mode subthreshold MOS circuits for analog VLSI neural systems. *IEEE Transactions on Neural Networks*, 2(2):205–13, March 1991.

[8] B Angeniol, G De La Criox Vaubios, and J Le Texier. Self-organising feature maps and the travelling salesman problem. *Neural Networks*, 1:289–293, 1988.

[9] Donald J Baxter. *Process-Tolerant VLSI Neural Networks for Applications in Optimisation*. PhD thesis, The University of Edinburgh, 1993.

[10] R Beale and T Jackson. *Neural Computing: An Introduction*. Adam Hilger, 1991.

[11] Rodney A Brooks. A robot that walks; emergent behaviours from a carefully evolved network. *IEEE Journal of Robotics and Automation*, pages 692–695, 1989.

[12] Rodney A Brooks. Intelligence without reason. In *Twelfth International Joint Conference on Artificial Intelligence*, pages 569–595, San Mateo, Ca, August 1991. Morgan Kaufmann.

[13] Rodney A Brooks, Jonathan H Connell, and Peter Ning. Herbet: A second generation mobile robot. Technical report, MIT AI Memo 1016, January 1988.

[14] M Brownlow, L Tarassenko, and A F Murray. Analogue computation using VLSI neural network devices. *Electronic Letters*, 26(16):1297–1299, August 1990.

[15] M Brownlow, L Tarassenko, and A F Murray. Results from pulse stream neural network devices. In *Proceedings International Workshop on VLSI for Artificial Intelligence and Neural Networks*, pages A2/1–A2/11, September 1990.

[16] HH Busta, OK Ersoy, JE Pogemiller, KD Mackenzie, and RW Standley. Hardware implementation of a wired-once neural net in thin-film technology on a glass substrate. *IEEE Transactions on Electron Devices*, 37(4):1039–1045, 1990.

[17] Graham Cairns and Lionel Tarassenko. Precision issues for learning with analog VLSI multilayer perceptrons. *IEEE Micro*, 15(3):54–56, June 1995.

[18] Graham A Cairns. *Learning with Analogue VLSI Multi-Layer Perceptrons*. PhD thesis, University of Oxford, Department of Engineering Science, 1995.

[19] Jakob Carlström. Minimisation of quantization errors in digital implementations of multi layer perceptrons. In L F Niklasson and M B Bodèn, editors, *Current Trends in Connectionism – Proceedings of the 1995 Swedish Conference on Connectionism*, pages 191–202. Lawrence Erlbraum, 1995.

[20] Gail A Carpenter. Neural network models for pattern recognition and associative memory. *Neural Networks*, 2:243–257, 1989.

[21] S Chakrabarti, N Bindal, and K Theagharajan. Robust radar target classifier using artificial neural networks. *IEEE Transactions on Neural Networks*, 6(3):760–766, May 1995.

[22] Alice M Chiang, Michael L Chuang, and Jeffrey R LaFranchise. CCD neural network processors for pattern recognition. In *Advances in Neural Information Processing Systems*, volume 4, 1992.

[23] S Churcher, D J Baxter, A Hamilton, A F Murray, and H M Reekie. Generic analog neural computation – the EPSILON chip. In *Advances in Neural Information Processing Systems*, volume 5, pages 773–780, 1993.

[24] Stephen Churcher. *VLSI Neural Networks for Computer Vision*. PhD thesis, The University of Edinburgh, 1993.

[25] U Cilingiroglu. A purely capacitive synaptic matrix for fixed-weight neural networks. *IEEE Transactions on Circuits and Systems*, 38(2):210–217, 1991.

[26] R J Coggins, M A Jabri, and S J Pickard. A comparison of three on chip neuron designs for a low power VLSI MLP. In *Proceedings Microneuro'93*, pages 97–103, 1993.

[27] Richard Coggins, Marwan Jabri, Barry Flower, and Stephen Pickard. A hybrid analog and digital VLSI neural network for intracardiac morphology classification. *IEEE Journal of Solid State Circuits*, 30(5):542–550, May 1995.

[28] R H Crites and A G Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 8, 1995.

[29] P B Denyer and J Mavor. MOST transconductance multipliers for array applications. *IEE Proceedings Part I*, 128(3):81–86, June 1981.

[30] Brion K Dolenko and Howard C Card. Tolerance to analog hardware of on-chip learning in backpropagation networks. *IEEE Transactions on Neural Networks*, 6(5):1045–52, September 1995.

[31] Peter J Edwards. *Analogue Imprecision in MLPs – Implications and Learning Improvements*. PhD thesis, Department of Electrical Engineering, University of Edinburgh, 1994.

[32] H Eguchi, T Furuta, S Horiguchi, S Oteki, and T Kitaguchi. Neural network LSI with on-chip learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 453–456, Seattle, 1991.

[33] F Faggin. Commercialisation of neural networks. In *Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*. IEEE Computer Society Press, 1994.

[34] Federico Faggin. VLSI implementation of neural networks. In *Tutorial texts, IJCNN'93*, pages 260–281. IEEE catalog number 93CH3353-0, October 1993.

[35] R C Frye, E A Rietman, C C Wong, and B L Chin. An investigation of adaptive learning implemented in an optically controlled neural network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 457–63, 1992.

[36] Robert C Frye, Edward A Rietman, and Chee C Wong. Back-propagation learning and nonidealities in analog neural network hardware. *IEEE Transactions on Neural Networks*, 2(1):110–117, January 1991.

[37] O Gällmo, E Nordström, M Gustafsson, and L Asplund. Neural networks for preventive traffic control in broadband ATM networks. In *International Workshop on Mechatronical Computer Systems for Perception and Action (MCPA-93)*, pages 139–145, Halmstad, Sweden, June 1993.

[38] Olle Gällmo and Jakob Carlström. Some experiments in using extra output learning to hint multi layer perceptrons. In L F Niklasson and M B Bodén, editors, *Current Trends in Connectionism – Proceedings of the 1995 Swedish Conference on Connectionism*, pages 179–190. Lawrence Erlbaum, 1995.

[39] Randall L Geiger, Phillip E Allen, and Noel R Strader. *VLSI Design Techniques for Analog and Digital Circuits*, chapter 6, pages 514–518. McGraw-Hill, 1990.

[40] H P Graf, L D Jackel, R E Howard, B Straughn, J S Denker, W Hubbard, D M Tennant, and D Schwartz. VLSI implementation of a neural network memory with several hundreds of neurons. In J S Denker, editor, *1986 Conference on Neural Networks for Computing Snowbird, UT*, volume 151, pages 182–187, 1986.

[41] Hans Peter Graf, E Sackinger, and L D Jackel. Recent developments of electronic neural nets in north america. *Journal of VLSI Signal Processing*, 5(19-31):19–31, 1993.

[42] A Gruber, J Fent, W Fröchtenicht, C Kiesling, J Möck, P Ribarics, D Goldner, H Kolanoski, and T Krämerkämper. A neural network architecture for the second level trigger in the H1-Experiment at the electron proton collider HERA. In *Proceedings 6th International Conference on Tools with Artificial Intelligence*, pages 325–332, Los Alamitos CA, Nov 1994. IEEE Computer Society Press.

[43] 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 Transactions on Neural Networks*, 3(3):385–393, May 1992.

[44] Alister Hamilton. *Pulse Stream Circuits and Techniques for the Implementation of Neural Network.* PhD thesis, University of Edinburgh, 1993.

[45] Alister Hamilton, Stephen Churcher, Peter J. Edwards, Geoffrey B. Jackson, Alan F. Murray, and H. Martin Reekie. Pulse stream VLSI circuits and systems: The EPSILON neural network chipset. *International Journal of Neural Systems*, 4(4):395–405, Dec 1993.

[46] Dan Hammerstrom. A VLSI architecture for high-performance, low-cost, on-chip learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 537–44, 1990.

[47] Dan Hammerstrom and Steve Rehfuss. Neurocomputing hardware: present and future. *Artificial Intelligence Review*, 7:285–300, 1993.

[48] Donald Hebb. *The Organisation of Behaviour.* Wiley, 1949.

[49] Markus Hoehfeld and Scott E Fahlman. Learning with limited numerical precision using the cascode-correlation algorithm. *IEEE Transactions on Neural Networks*, 3(4):602–611, July 1992.

[50] M Holler, A Shmurun, and J Brauch. Neural network recognition of objects based on impact dynamics. In *Conference record, 1992 IEEE Nuclear Science Symposium and Medical Imaging Conference*, volume 2, NY, 1992. IEEE.

[51] Mark Holler, Simon Tam, Hernan Castro, and Ronald Benson. An electrically trainable artificial neural network ETANN with 10240 "floating gate" synapses. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages II–191–6, NY, USA, 1989. IEEE Neural Network Committee.

[52] P W Hollis, J S Harper, and J J Paulos. The effects of precision constraints in a backpropagation learning network. *Neural Computation*, 2:363–73, 1992.

[53] Andrew J Holmes. *The Use of Non-Volatile a-Si:H Memory Devices for Synaptic Weight Storage in Artificial Neural Networks.* PhD thesis, Department of Electrical Engineering, University of Edinburgh, January 1995.

[54] Jordan Holt, Toby Skinner, and Nguyen Nguyen. Automating operator services using automatic speech recognition. In *Conference records of the 26th Asilomar conference on signals, systems and computers*, volume 2, pages 1096–99. IEEE Computer Society Press, 1992.

[55] Jordan L Holt and Jenq-Neng Hwang. Finite precision error analysis of neural network hardware. Technical Report FT-10, Department of Electrical Engineering, University of Washington, Seattle, WA 98195, 1991.

[56] W Hubbard, D Schwartz, J Denker, HP Graf, R Howard, L Jackel, B Straughn, and D Tennant. Electronic neural networks. *Neural Networks for Computing*, 151:227–234, 1986.

[57] M Jabri and B Flower. Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks. *Neural Computation*, 3(4):546–565, 1991.

[58] M Jabri, S Pickard, Z Leong, P ans Chi, B Flower, and Y Xie. ANN based classification for heart defibrillators. In *Advances in Neural Information Processing Systems*, volume 4, 1992.

[59] Marwan A Jabri and R J Wang. A novel channel selection system in cochlear implants using artificial neural networks. In *Advances in Neural Information Processing Systems*, volume 8, 1995.

[60] Geoffrey B Jackson. VLSI implementation of neural networks: A switched capacitor approach. Masters thesis, Department of Electrical Engineering, University of New South Wales, Sydney, Australia, 1992.

[61] Lynn R Kern. Design and development of a real-time neural processor using the Intel 80170NX ETANN. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 684–9, NY, 1992.

[62] T Kohonen. An introduction to neural computing. *Neural Networks*, 1:3–16, 1988.

[63] A Kramer, C K Sin, R Chu, and P K Ko. Compact EEPROM based weight functions. In *Advances in Neural Information Processing Systems 3*, pages 1001–7, 1990.

[64] Erwin Kreeyszig. *Advanced Engineering Mathematics*, chapter 19, pages 1001–61. John Wiley & Sons, 6th edition, 1988.

[65] J Lazzaro, S Ryckebusch, M A Mahowald, and C A Mead. Winner-take-all networks of $O(N)$ complexity. In David S Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 703–11. Morgan Kaufmann Publishers, 2929 Campus Drive, San Mateo CA, 1989.

[66] Y Lecun, L D Jackel, B Boser, J S Denker, H P Graf, I Guyon, D Henderson, R E Howard, and W Hubbard. Handwritten digit recognition – applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

[67] Philip H W Leong and Marwan A Jabri. Kakadu – a low power analogue neural network classifier. *International Journal of Neural Systems*, 4(4):381–394, Dec 1993.

[68] Phillip H W Leong and Marwan A Jabri. A VLSI neural network for morphology classification. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 678–683, 1992.

[69] Clark S Lindsey, Bruse Denby, and Herman Haggerty. Drift chamber tracking with neural networks. In *Conference Record, 1992 IEEE Nuclear Science Symposium and Medical Imaging Conference*, volume 2, pages 835–7, NY, 1992. IEEE.

[70] R P Lippmann. An introduction to computing with neural networks. *IEEE ASSP magazine*, pages 4–22, apr 1987.

[71] S Liu and K Boahen. Adaptive retina with centre-surround receptive field. In *Advances in Neural Information Processing Systems*, volume 8, 1995.

[72] Weimin Liu, Andreas G Andreou, and H Goldstein Moise. Analog cochlear model for multistream speech analysis. In *Advances in Neural Information Processing Systems*, volume 5, 1993.

[73] Pattie Maes and Rodney A Brooks. Learning to coordinate behaviours. In *AAAI*, pages 797–802, 1990.

[74] Armando Manduca, Paul Christy, and Richard Ehman. Neural network diagnosis of avascular necrosis form magnetic resonance images. In *Advances in Neural Information Processing Systems*, volume 4, 1992.

[75] Yoshihiro Matsuura, Hideki Miyazawa, and Toby E Skinner. Word recognition using a neural network and a phonetically based DTW. In *Neural Networks for Signal Processing IV, Proceedings of 1994 IEEE Workshop*, pages 329–34, 1994.

[76] Hal McCartor. Back propagation implementation on the Adaptive Solutions neurocomputer chip. In Richard P Lippmann, John E Moody, and Davis S Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 1028–31. Morgan Kaufmann Publishers, 1991.

[77] W S McCulloch and W A Pitts. A logical calculus if ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[78] Carver Mead. *Analog VLSI and neural systems*. Addison-Wesley, 1989.

[79] Eric Means and Dan Hammerstrom. Piriform model execution on a neurocomputer. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 575–80, 1991.

[80] M Minsky and S Papert. *Perceptrons*. MIT Press, 1969.

[81] R J Mitchell. *Microcomputer Systems Using the STE Bus*. Macmillan, 1989.

[82] Dean Mueller and Dan Hammerstrom. A neural network systems component. In *Proceedings IEEE International Conference on Neural Networks*, volume 3, pages 1258–64, NY, USA, 1993. IEEE.

[83] D B Mundie and L W Massengil. Threshold non-linearity effects on weight decay tolerance in analog neural networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 583–7, 1992.

[84] Alan F. Murray, S. Churcher, A. Hamilton, A.J. Holmes, G.B. Jackson, and R.J. Woodburn. Applications of pulsed neural VLSI. *IEEE MICRO*, 14(3):29–39, June 1994.

[85] Alan F Murray and A V W Smith. Asynchronous arithmetic for VLSI neural systems. *Electronic Letters*, 23(12):642–43, June 1987.

[86] Alan F Murray and Anthony V W Smith. Asynchronous VLSI neural networks using pulse stream arithmetic. *IEEE Journal of Solid State Circuits and Systems*, 23(3):688–697, 1988.

[87] D J Myers, J M Vincent, and D A Orrey. A VLSI architecture for implementing neural networks with on-chip backpropagation learning. In D J Myers, C Nightingale, and R Linggard, editors, *Neural Networks for Vision, Speech and Natural Language*, 1, chapter 15, pages 312–329. Chapman and Hall, 1992.

[88] Ulrich Nehmzow. *Experiments in Competence Acquisition for Autonomous Mobile Robots*. PhD thesis, University of Edinburgh, 1992.

[89] Ulrich Nehmzow. Flexible control of mobile robots through autonomous competence acquisition. *Measurement and Control*, 28:48–54, 1995.

[90] Ulrich Nehmzow, John Hallam, and Tim Smithers. Really useful robots. In T Kanade, F C A Groen, and L O Hertzberger, editors, *Intelligent Autonomous Systems*, Amsterdam, 1989. ISBN 90-8000410-1-7.

[91] E Nordström. A hybrid admission control scheme for broadband ATM traffic. In J Alspector, R Goodman, and T X Brown, editors, *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications*, pages 77–84. Lawrence Erlbaum, 1993.

[92] E Nordström, O Gällmo, L Asplund, M Gustafsson, and B Eriksson. Neural networks for admission control in an ATM network. In L F Niklasson and M B Bodén, editors, *Connectionism in a Broad Perspective: Selected papers from the Swedish Conference on Connectionism - 1992*, pages 239–250. Ellis Horwood, 1994.

[93] E Nordström, O Gällmo, M Gustafsson, and L Asplund. Statistical preprocessing for service quality estimation in a broadband network. In *World Congress on Neural Networks (WCNN-93)*, volume I, pages 295–299, Portland, Oregon, USA, 1993.

[94] John C Platt and TImothy P Allen. A neural network classifier for the I1000 chip. In *Advances in Neural Information Processing Systems*, volume 8, 1995.

[95] Dean A Pomerleau. Rapidly adapting artificial neural networks for autonomous navigation. In *Advances in Neural Information Processing Systems*, volume 3, 1991.

[96] L Reyneri, H C A M Withagen, J A Hegt, and M Chiaberge. A comparison between analog and pulse stream VLSI hardware for neural network s and fuzzy systems. In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 77–86. IEEE Computer Society Press, 1994.

[97] L M Reyneri and M Sartori. A neural vector matrix multiplier using pulse width modulation techniques. In *Proc. of the 2nd International Conference on Microelectronics for Neural Networks*, pages 269–272, Munich, October 1991.

[98] A Rodríguez-Vázquez, A Reuda, J L Huertas, and R Domínguez-Castro. Switched-capacitor neural networks for linear programming. *Electronic Letters*, 24(8), April 1988.

[99] F Rosenblatt. *Principles of Neurodynamics*. Spartan Books, New York, 1962.

[100] D E Rumelhart, G E Hinton, and R J Williams. Learning internal representations by error propagation. In D E Rumelhart and J L McLelland, editors, *Parallel Distributed Processing*, volume 1. M.I.T. Press, Cambridge, MA, 1986.

[101] M R Rutenberg. PAPNET: A neural net based cytological screening system. In *Neural Information Processing Systems*, volume 3, 1991.

[102] S Tam, M Holler, J Brauch, A Pine, A Peterson, S Anderson, and S Deiss. A reconfigurable multi-chip analog neural network; recognition and back-propagation training.

In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 625–30, 1992.

[103] Simon M Tam, Bhusan Gupta, Herman A Castro, and Mark Holler. Learning on an analog VLSI neural network chip. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 701–3, NY, 1990. IEEE.

[104] L Tarassenko and S Roberts. Supervised and unsupervised learning in radial basis function classifiers. *IEE Proc. on Vision, Image and Signal Processing*, 141(4):210–216, 1994.

[105] Lionel Tarassenko, Michael Bronlow, Gillian Marshall, Jon Tombs, and Alan F Murray. Real-time autonomous robot navigation using VLSI neural networks. In *Advances in Neural Information Processing Systems*, volume 3, 1991.

[106] Lionel Tarassenko, Jon Tombs, and Graham Cairns. On-chip learning with analogue VLSI neural networks. *International Journal of Neural Systems*, 4(4):419–426, December 1993.

[107] Viral V Tolat and Bernard Widrow. An adaptive "broom balancer" with visual inputs. In *IEEE International Conference on Neural Networks*, volume 2, pages 641–647, July 1988.

[108] J Tombs and L Tarassenko. A fast, novel, cascadable design for multi-layer networks. In *IEE Second International Conference on Artificial Neural Networks*, pages 64–68, November 1991.

[109] M S Tomlinson, D J Walker, and M A Sivilotti. A digital neural network architecture for VLSI. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 545–550, San Diego, 1990.

[110] Y P Tsividis and D Anastassiou. Switched-capacitor neural networks. *Electronic Letters*, 23(18), 1987.

[111] J M Vincent and D J Myers. Weight dithering and wordlength selection for digital backpropagation networks. *BT Technology Journal*, 10(3):124–133, 1992.

[112] Eric Vittoz. Analog VLSI implementation of neural networks. In *Proceedings Journées D'Électronique 1989 Artificial Neural Networks*, pages 224–50. Presses Polytechniques Romandes, CH–1015 Lausanne, Suisse., October 1989.

[113] Eric A Vittoz. Micropower analog design: Limits and basic techniques. In *Proceedings ISCAS'94*, volume 4, pages 3–4. IEEE Press, May 1994.

[114] M H White and C Y Chen. Electrically modifiable nonvolatile synapses for neural networks. In *IEEE International Symposium on Circuits and Systems*, volume 3,, pages 1213–16, 1989.

[115] B Widrow. *DARPA Neural Network Study*. AFCEA International Press, November 1988.

[116] B Widrow and R Winter. Neural nets for adaptive filtering and adaptive pattern recognition. *Computer*, 21:25–39, 1988.

[117] Yun Xie and Marwan A Jabri. Training algorithms for limited precision feedforward neural networks. SEDAL Technical Report 1991-8-3, Department of Electrical Engineering, The University of Sydney, 1991.

[118] Xilinx Inc., 2100 Logic Drive, San Jose, CA 95124. *The Programmable Logic Data Book*, 1993.

[119] Xilinx Inc. *XACT Libraries Guide*, January 1993.