Implementing Radial Basis Function Neural Networks in Pulsed Analogue VLSI

David J. Mayes

Thesis submitted for the degree of
Doctor of Philosophy
The University of Edinburgh
January 1997

Abstract

The Radial Basis Function (RBF) neural network architecture is a powerful computing paradigm that can solve complex classification, recognition and prediction problems. Although the RBF is similar in structure to the ubiquitous Multilayer Perceptron (MLP) neural architecture, it operates in a different way.

This thesis discusses the issues addressed, and the findings from, a project that involved implementing a Radial Basis Function neural network in analogue CMOS VLSI. The developed hardware exploits the pulse width modulation (PWM) neural method, which allows compact, low power hardware to be realised through a combination of analogue and digital VLSI techniques.

Novel pulsed circuits were designed and developed, fabricated and tested in pursuit of a fully functioning RBF demonstrator chip. The theory underpinning the designs is discussed and measured hardware results from two test chips are presented along with an assessment of circuit performance. Although the circuits generally functioned as required, discrepancies between the actual and theoretical operation were observed. Thus suggested improvements to the original designs are made and the circuit and system level considerations for the final demonstrator chip are discussed.

Measured results are presented from the final demonstrator chip, confirming the correct operation of its constituent circuits, along with results from experiments showing that, when modelled in software, the developed circuitry is capable of performing as well as an identically trained RBF with Gaussian non-linearities. However, further results indicated that the expected network performance would degrade when the neural parameters are quantised.

Hardware experiments with the demonstrator chip indicated that it could be used as an RBF classifier, but its performance degraded for more complex problems. A summary of the probable reasons for the performance degradation is provided.

The final conclusion reached as a result of this work is that, provided care is taken when designing and laying out the circuits, it is possible to produce pulsed hardware capable of reproducing the required RBF operations. However, successfully applying the hardware to *real* problems is not trivial, as indicated by a discussion of the pertinent issues.

Declaration

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

David J. Mayes

Acknowledgements

I would like to acknowledge the help and support I received from the following people during the course of my PhD project.

- My academic supervisors, Alan Murray and Martin Reekie, for their advice, patience and encouragement.
- Alister Hamilton, for providing me with an initial route into analogue VLSI design and the subsequent discussions on the DYMPLE synapse and hardware testing.
- Jean E. Louvet for laying out the original DYMPLES multiplier cell.
- Bill Seddon and Lisa Wong at Rutherford Research Labs for ensuring the final submission of my completed chip designs went smoothly.
- My fellow researchers with whom I had many fruitful discussions at the conferences I attended during the course of my work.
- Finally, I would like to thank my fellow students and friends, both inside and outside of Electrical Engineering, for making the last three years so enjoyable and rewarding, including the CHV kids from Niddrie, Pilton and Muirhouse for reminding me there are more important things in life than academic exercises.

Table of Contents

Chapter 1

Introduction

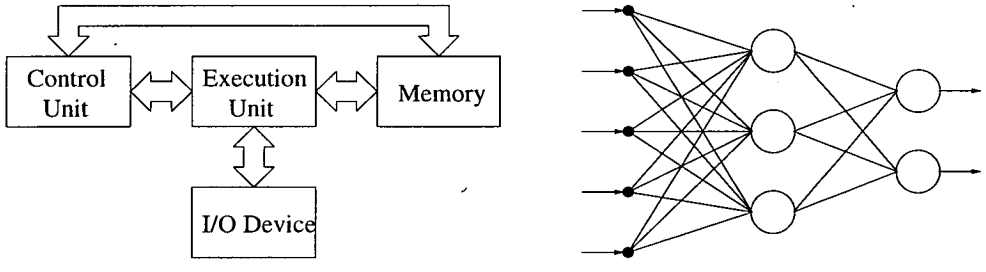
1.1. Background

Research into Artificial Neural Networks (ANNs) is driven by two main aims:

- the desire to model the operation of the brain in order to derive a better understanding of cognitive function
- the desire to automate the processes of recognition, classification and time-series analysis for solving increasingly complicated real world problems.

While biologists, psychologists, neurologists and cognitive scientists investigate the finer nuances of thought processes, it is the application of such biologically-inspired computation that concerns the electronics engineer and which is explored here.

Although recent advances in neural network research have only been achievable because of the rapid developments witnessed in electronic integration and computer architectures over the last quarter of a century, the structure and operation of ANNs is in marked contrast to that of the traditional, ubiquitous, von Neumann computer architecture, Figure 1.1.



The von Neumann Architecture. An ANN Architecture.

Figure 1.1 - The von Neumann Computer Architecture and an ANN Architecture.

Whereas a von Neumann computer consists of a single, complex, multi-functional processor, ANNs are densely interconnected topologies of simple computational units called *neurons*. Whilst von Neumann computers operate serially, processing a list of instructions in sequential order, ANNs operate in parallel, processing all the inputs at the same time and distributing the results along their interconnections.

Furthermore, artificial neural networks have the ability to learn - they can be trained, via learning algorithms, to solve complex non-linear input to output mappings through the evolution of a number of adaptable parameters, or *weights*, associated with each computational unit. In contrast, although the von Neumann computer has found widespread use due to its ability to process numbers quickly and to arbitrary accuracy, even simple classification tasks have proven to be troublesome.

So herein lies the aspiration for ANNs: if computing engines can be developed which draw inspiration from the understood and observed operation of the brain, will similar highly developed, "human-like", performance be achieved ?

1.2. Historical Summary

Although often considered as a new, "leading-edge" technology, ANNs have had a long and chequered history. The first model of a biological neuron was conceived in the 1940's by McCulloch and Pitts [1], and was developed from their research into the operation of the brain. Rosenblatt later termed these computational units *perceptrons* [2]. Interest in the neural network field continued from the 1940s until the late 1960s when a publication by Minsky and Papert [3] examined single layers of perceptrons in detail and highlighted that they could only solve linearly separable problems. This was seen as a fundamental limitation and research all but ceased for fifteen years.

However, work by Werbos [4], Parker [5] and Rumelhart *et al* [6] demonstrated that the crippling limitation of the single layer perceptron could be circumvented and the latter group demonstrated that trained ANNs could produce superior classification performance in high-level tasks compared to traditional computers. There then followed an explosion of research interest in the area of adaptive, parallel, interconnected structures and many "neural" paradigms, with associated learning schemes, have since been developed.

Although numerous architectures have been developed since 1986, one has dominated the field: the multilayer perceptron or MLP. This was the architecture originally proposed by Rumelhart *et al* and, along with its *back-propagation* learning scheme, was the most utilised architecture during the 1980's.

Research by Broomhead and Lowe [7] and Moody and Darken [8] in the late 1980s produced a new neural architecture based on *radial basis functions.* The Radial Basis Function (RBF) neural architecture has a similar structure to the MLP, and

possesses the same ability to universally approximate any function given enough computational units. However, it can be trained faster than the MLP and its training mechanisms do not suffer from the same pathologies as back-propagation. Due to these advantages, the RBF has reduced the MLP's monopolisation of neural network applications.

This thesis investigates the implementation of the RBF neural network architecture in electronic hardware.

1.3. The Need for Neural Hardware

Since ANNs are essentially simple computing machines connected in parallel, their behaviour can be replicated on von Neumann computers using an appropriate high-level programming language. However, in order to exploit the inherent parallelism of the architectures, dedicated hardware is required. Fortunately, ANNs are amenable to both analog and digital hardware implementations and numerous examples of each type are available in the literature. A summary of the advantages and disadvantages of both is presented in Table 1.1.

Analogue		Digital	
Advantages	Disadvantages	Advantages	Disadvantages
Small Area Low Power	Low Precision Corruptible	High Precision Robust	Large Area High Power

Table 1.1 - Summary of the advantages and disadvantages of analogue and digital neural network implementations

As the field of neural network research has matured, so the focus of the research community has developed and evolved. The main emphasis of ANN research has now shifted towards generic software solutions, where different architectures can be evaluated simply, efficiently and with the minimum of effort. Hardware implementations now fall into two main categories:

- digital co-processor and accelerator boards
- application specific analog and hybrid hardware solutions.

Whilst co-processor and accelerator boards are used for speeding up software neural network simulations, dedicated analog and hybrid hardware is expected to fill a niche market where there is an identified need for low power, parallel processing chips having both a small area and a high data throughput.

The Integrated Systems Group at the University of Edinburgh has been engaged in neural network research since 1986. Notable work performed by the group over the past decade includes the development of the Pulse Stream suite of neural evaluation techniques [9], the development of a generic neural network chip (EPSILON) [10-12], an investigation into the benefits gained from training MLP networks using inherent analogue noise [13] and, more recently, the successful application of a neural architecture to robotic control [14]. The work contained in this thesis both complements and extends this work.

1.4. Aim of Thesis

As mentioned in Section 1.2, the complementary MLP and RBF architectures have dominated the resurgent ANN field. However, whilst there have been many successful implementations of hardware MLPs, there have been, in contrast, surprisingly few implementations of RBF chips. This has been due partly to the shift in emphasis in neural research to generic software development platforms and dedicated application specific hardware solutions and partly due to the perceived difficulty in implementing the requirements of RBFs - especially the basis functions - in VLSI.

The aim of this work was to study the circuit, system and operational issues relating to implementing the Radial Basis Function neural network architecture in analogue VLSI, using the pulse width modulation (PWM) neural technique, and to explore the constraints imposed on the algorithm by the chosen implementation medium.

1.5. Thesis Overview

To address the thesis aim, suitable circuits for realising the paradigm were designed, developed, fabricated, tested and assessed. The goal of the work was the production of a functioning pulsed RBF demonstrator that could be applied to classification problems.

In addition to producing and testing the RBF hardware, software simulations were also carried out to assess the suitability of the developed circuitry for solving classification problems and investigate the precision constraints that analogue VLSI could impose on the implementation.

This project is therefore concerned with circuit and system level aspects of the design of feedforward RBF chips. No attempt was made to produce an optimal or generic neural chip, nor were new learning algorithms or on-chip learning implementations

considered.

The remainder of this thesis is organised as follows.

Chapter 2 reviews the theoretical background to the work through considering the RBF and MLP neural architectures as discriminant function classifiers. The operation of both architectures is discussed and training methods presented for both.

Chapter 3 reviews the Complementary MOSFET (CMOS) transistor technology - the medium for which the analogue circuitry was developed. This chapter considers alternative methods for implementing RBFs in CMOS VLSI before introducing the actual method used: the hybrid *Pulse Stream* technique. After considering the general principles of pulsed methods, the motivations and scope for the project are presented.

Chapter 4 discusses the design and development of novel circuitry to realise the output layer of the RBF architecture. In addition to explaining the theory and motivations behind the DYMPLE synapse design, hardware measurements from a fabricated test chip are presented, along with an assessment of circuit operation and suggestions for design improvements.

Chapter 5 discusses the circuits developed for the basis function test chip: the RHO chip. Again the theoretical operation of each circuit is presented along with results from the second test chip. On the basis of these results, conclusions are drawn as to the best circuit to implement on the final demonstrator chip.

Chapter 6 presents the PAR chip - the final, pulsed, RBF demonstrator produced for the project. The improvements made to the circuit designs from Chapters 4 and 5 are discussed, along with the system level considerations necessary for the demonstrator chip. Hardware measurements from the PAR chip are presented and the operation of the final circuits assessed.

Chapter 7 summarises the results from a set of simulations carried out to investigate whether the designed circuitry could successfully solve a variety of classification problems. Further, the effect of using limited precision parameter storage is also considered. In addition to presenting and discussing the results of these experiments, descriptions of the classification problems and the operation of the software simulator are also given.

Chapter 8 presents the results from the classification experiments performed with the PAR chip. After detailing the constraints imposed on the hardware experiments, the results and observations from the demonstrator chip are summarised and the conclusions drawn from this work are listed and discussed.

Finally, Chapter 9 summarises the project and presents the conclusions from the work.

Chapter 2

MLP and RBF Theory

As stated in the last chapter, the goal of this work was to produce a small RBF demonstrator chip and demonstrate its functionality by applying it to a classification problem. Classification problems occur frequently in industry and business and currently represent one of the main application areas for artificial neural network simulations and hence one of the expected areas for using dedicated hardware.

In the context of this thesis, it is also instructive to consider the problem of pattern classification because it allows some of the fundamental concepts underpinning MLPs and RBFs to be introduced.

This chapter uses pattern classification as a means to introduce discriminant functions, before going on to consider the MLP network and especially the RBF network in more detail. The aim of the chapter is to provide the necessary theoretical background for the work in this thesis from a consideration of a popular application area for feedforward neural networks.

2.1. Pattern Classification

Pattern classification is the task of assigning data to one of several categories, or *classes*, based on the information contained in a *feature vector* representing the data. The feature vectors to be classified, or identified, can either contain raw data or data which has been preprocessed to extract more salient features [15]. Whatever form the vectors take, though, they must contain relevant information that allows the data to be identified as belonging to a specific class and allows vectors belonging to different classes to be differentiated. The processing of the vectors to identify their relevant features and allocate them to specific categories is usually performed by a dedicated pattern classifier.

One way to achieve class differentiation is to use *discriminant functions.* A discriminant function is a mathematical relationship that can be used to determine whether or not an input vector belongs to a certain class [16].

2.2. Discriminant Function Classifiers

If each feature vector is assumed to define a point in some *feature space,* then a pattern classifier can be considered as a means of partitioning this space into a number of regions, Figure 2.1. The different symbols in this figure represent vectors from the different classes, whilst the lines mark the class boundaries and correspond to the crossing points of the discriminant functions used.

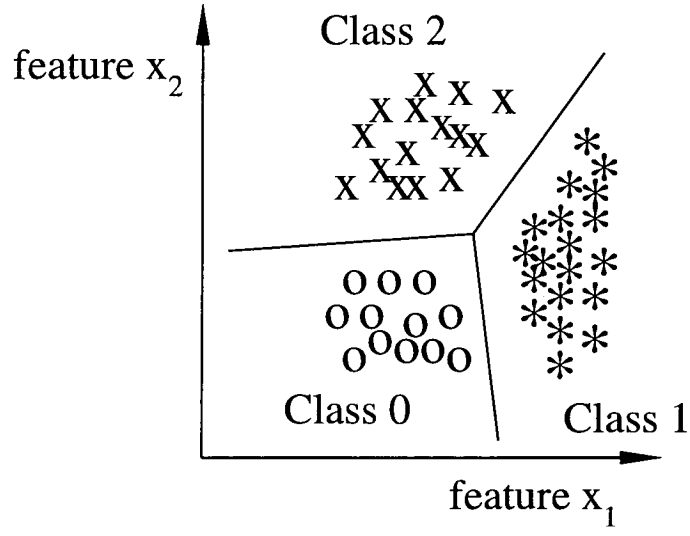


Figure 2.1 - Feature space divided up using discriminant functions

The classification problems considered in this thesis are *1-out-of-N* coded problems. This means that for an N-class problem, each class, n, will have its own unique discriminant function. Thus the classification problem can be considered as computing N discriminant functions, with the classification being determined by finding the class whose discriminant function produces the highest output [16]. Usually, the ideal output for class n, $O(n)$, will be 1.0 if input vector $\mathbf{x} \in n$ and will be 0.0 otherwise. Thus, for any input vector, one and only one output can be 1.0, with all other outputs ideally 0.0. However, a more likely scenario is that all outputs will lie in the range, $0.0 < O(n) < 1.0, n \in N$.

To consider pattern classification using discriminant functions in more detail, consider the two class problem in Figure 2.2 [15, 17]. Clearly these two classes can be separated with the dotted line - it is termed a *decision boundary* - and the problem is said to be *linearly separable*.

The decision boundary can be described mathematically using the general equation for a straight line - $y = mx + c$. In this case, the decision boundary can be written as:

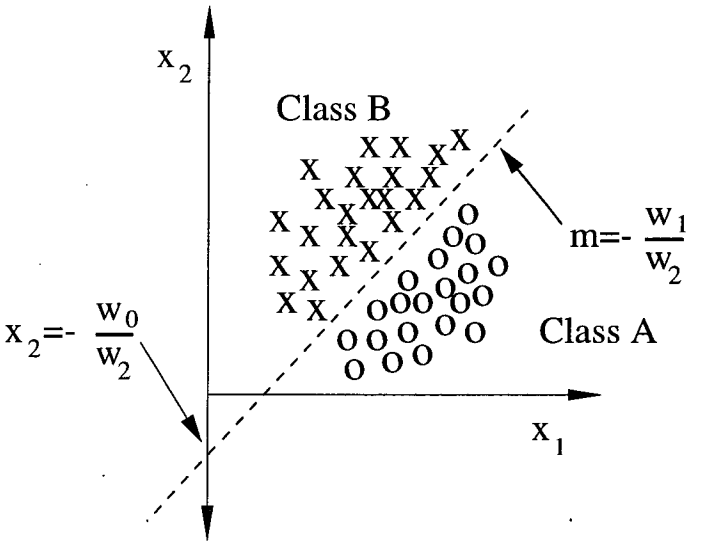


Figure 2.2 - A two-class linearly separable problem

$$x_2 = ax_1 + b \tag{2.1}$$

By defining $a = -\frac{w_1}{w_2}$ and $b = -\frac{w_0}{w_2}$, and assuming x_0 is always 1.0, equation 2.1 can be re-expressed as:

$$w_2\, x_2 + w_1\, x_1 + w_0 = 0 \tag{2.2}$$

or more generally

$$\mathbf{w}^T\mathbf{x} = 0 \tag{2.3}$$

where $\mathbf{w}$ is the weight vector and describes the orientation of the 1-dimensional decision boundary in the 2-dimensional x_1 - x_2 plane. In general, equation 2.3 describes the orientation of a $(d-1)$-dimensional boundary in d-dimensional space. The weight vector actually defines a d-dimensional vector normal to the decision boundary and pointing in the direction where $\mathbf{w}^T\mathbf{x} > 0$.

For this two class case, $\mathbf{w}^T\mathbf{x}$ is a *discriminant function* since it can be used to differentiate the two classes. For example, the input vector could be assigned to class A if $\mathbf{w}^T\mathbf{x} > 0$ and class B if $\mathbf{w}^T\mathbf{x} < 0$. If $\mathbf{w}^T\mathbf{x} = 0$, however, no unique decision can be reached since the vector lies on the decision boundary and could belong to either class.

As stated previously, most decision problems usually use one discriminant function per class and assign the input vector to the class whose discriminant function has the largest value. Many functions can be used as discriminant functions and if the dot

product of the weight vector and input vector, $\mathbf{w}^T\mathbf{x}$, is transformed using a monotonic function $f()$, then the decision process is unaltered [16]. Suitable forms for $f()$ include the Heaviside function $H(x)$, Figure 2.3(a), and the logistic sigmoid function as defined by equation 2.4 and shown in Figure 2.3(b).

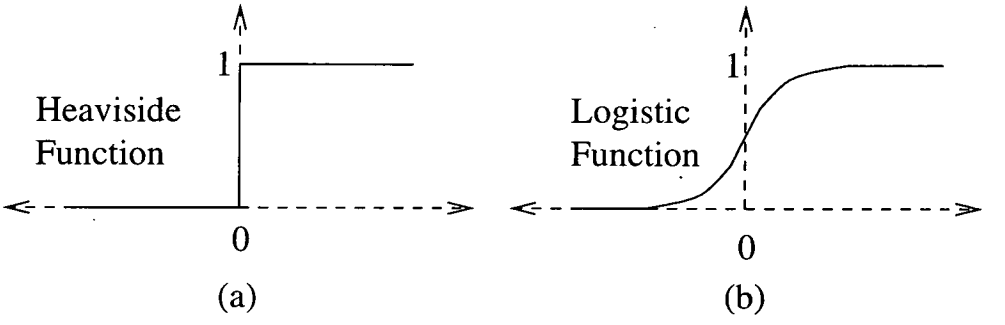


Figure 2.3 - (a) The Heaviside Function and (b) the Logistic Sigmoid Function

$$f(\theta) = \frac{1}{(1 + e^{-\theta})} \tag{2.4}$$

Using the logistic function for the 2-class problem in Figure 2.2, it is possible to define two discriminant functions $f(\mathbf{w}_1^T\mathbf{x})$ and $f(\mathbf{w}_2^T\mathbf{x})$ where $\mathbf{w}_1 = -\mathbf{w}_2$. This situation is depicted in Figure 2.4.

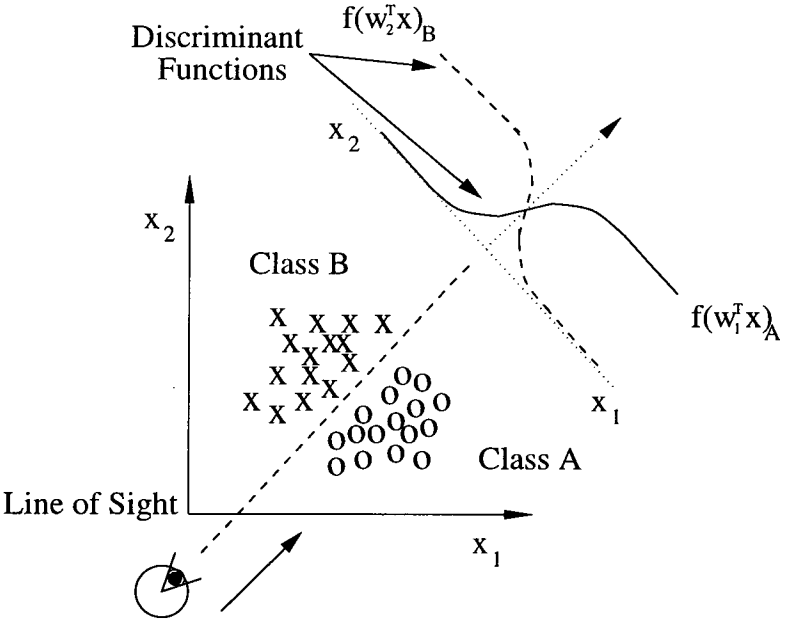


Figure 2.4 - Visualisation of the two-class problem using non-linear discriminant functions

Now the classication problem can be defined in terms of the two discriminant functions:

$$\text{if } f(\mathbf{w}_1^T\mathbf{x}) > f(\mathbf{w}_2^T\mathbf{x}), \mathbf{x} \in \text{ class A}$$

$$\text{if } f(\mathbf{w}_1^T\mathbf{x}) < f(\mathbf{w}_2^T\mathbf{x}), \mathbf{x} \in \text{ class B}$$

This consideration of non-linear discriminant functions for pattern classification leads naturally onto multilayer perceptron and radial basis function neural networks. These networks can be used to solve non-linear classification problems because they have simple computational units that form non-linear and linear discriminant functions respectively.

2.3. Feedforward Neural Networks for Classification Problems

Multilayer Perceptron and Radial Basis Function neural networks are feedforward architectures that have no recursive links within their topologies and information propagates from input to output through parallel processing units, Figure 2.5.

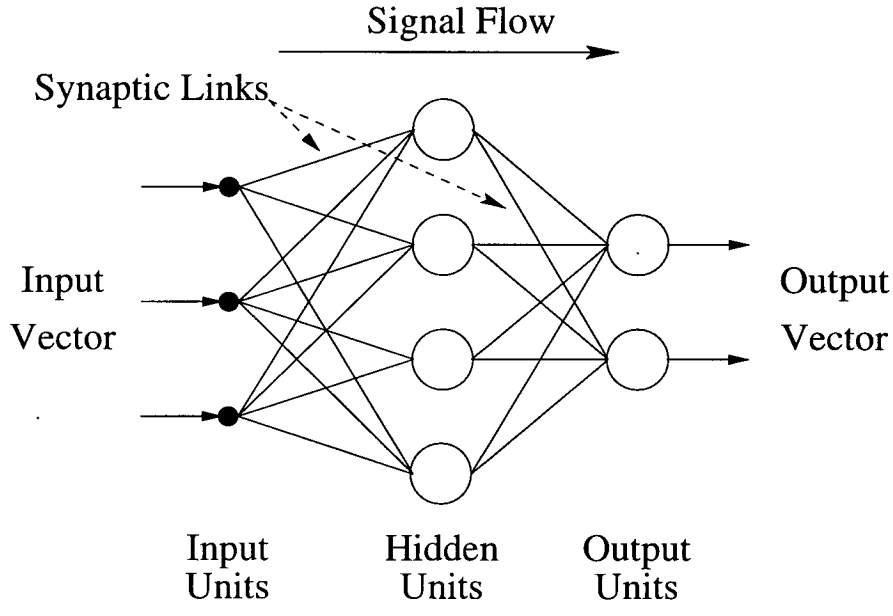


Figure 2.5 - A Typical Feedforward Neural Network Architecture

Both types of network implement uni-directional, non-linear functional mappings between multi-dimensional input space and multi-dimensional output space. Feedforward neural networks can solve non-linear classification problems, a special case of general non-linear multi-variate functional mappings, through the use of the non-linear activation functions in the computational units within the networks.

In this chapter, only the MLP and RBF neural network architectures are considered for solving pattern classification problems. These two topologies are not the only neural classifiers [18], nor are neural networks the soul means of solving classification problems [15, 16]. However, it is necessary to limit the scope of this discussion due to the available space.

Neural networks learn to produce the desired functional mappings through the adaption of the neural parameters, or *weights*, associated with the *synaptic links* connecting the processing units. No prior knowledge about the form of the mapping is required, only a **training set** of input and output vector pairs, which the network uses in the learning process, and a similar set of vector pairs, the **test set**. The test vectors are applied to the network after training, allowing its *generalisation* performance (ie its ability to classify previously unseen data) to be evaluated.

In essence, feedforward neural networks are non-parametric models, designed to capture the underlying trends and dynamics in a data set by gleaning information about the problem during learning. The primary aim for fitting such a model to a problem is to balance the conflicting requirements of providing enough free parameters to adequately reproduce the mapping, without fitting the model to the idiosyncrasies of the noise within the available data. Since any data set has its input and output dimensionalities pre-specified, then the number of weights in a neural network depends on the number of hidden units required to solve the problem. Knowing how many hidden units to use must usually be determined empirically, although algorithms do exist for both the MLP and RBF that allow the number of hidden units to be adapted as training proceeds [19, 20]. Normally the number of hidden units required depends on the complexity of the problem, with more difficult tasks needing more adaptable parameters.

Neural learning can either be **supervised** or **unsupervised**. Supervised learning involves adapting the neural weights to reduce a global error function; the error function usually being defined in terms of the adjustable network parameters. Supervised learning is therefore akin to teacher-based training. Alternatively, unsupervised learning requires no output vectors and has no error function to minimise. Unsupervised learning often involves a clustering of the input vectors into groups possessing similar properties.

Geometrically, neural learning can be thought of as finding an abstract global *solution surface* for the problem under consideration. The shape of the solution surface is determined by the neural weights and changes as the weights are adapted.

Similarly, generalisation can be thought of as interpolating new data onto the final solution surface, with each classification decision determined by where the data vector lies on the surface.

2.4. Multilayer Perceptrons

The early ANN researchers proposed neural models that used non-linear discriminant functions based on the Heaviside Function, Figure 2.6. Rosenblatt termed such computational units *perceptrons* [2].

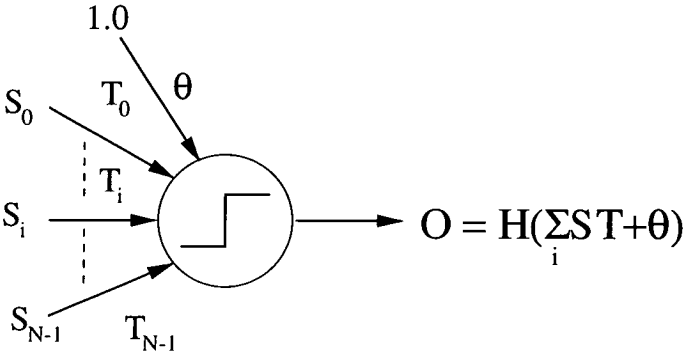


Figure 2.6 - A Perceptron with a Heaviside Activation Function

Minsky and Papert's book [3] proved conclusively that it was impossible to solve non-linear classification problems using only a single layer of perceptrons. However, it was realised that it was possible to solve non-linear problems by cascading layers of perceptrons together [21]. The main problem was how to adapt the weights of the network to partition up input space correctly. For example, consider a two-layer feed-forward network of perceptrons with hard-limiting Heaviside non-linearities, Figure 2.7. The training algorithms for single layer perceptron networks adapt the network weights depending upon how much they contribute to providing a right or wrong answer [17]. However, the outputs from the hidden layer of the network in Figure 2.7, are all either 0 or 1 and it is impossible to assign target values to these units and therefore tell by how much the hidden weights should be adapted to produce better performance. This is known as the *credit assignment* problem [15]. Neural network research thus stalled in the 1960s because of the lack of suitable learning algorithms for training multi-layer non-linear networks.

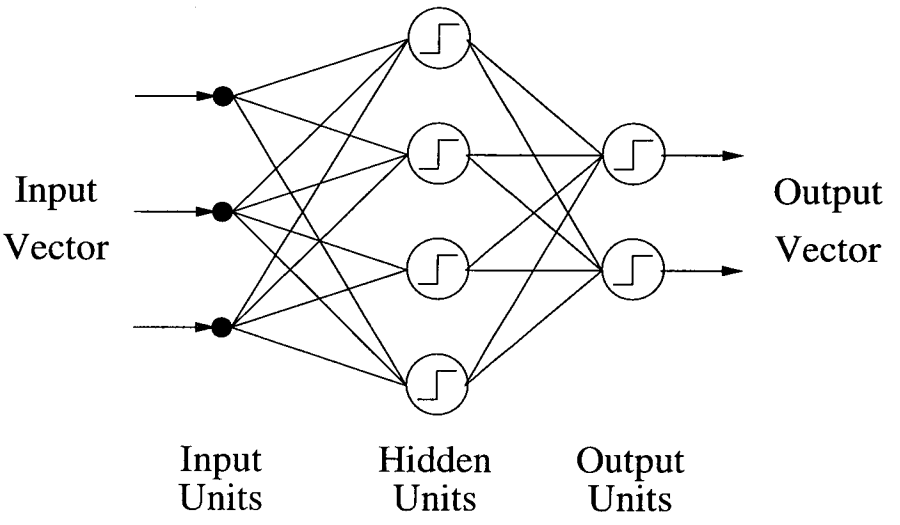


Figure 2.7 - A Multilayer Neural Network with Heaviside Activation Functions

However, work by several authors showed that the credit assignment problem could be solved, and the weight updates for all layers of the network calculated, if continuous monotonic activation functions were used instead of the Heaviside function [4-6]. These researchers devised the back-propagation algorithm and it gained widespread popularity due to a publication by Rumelhart and McClelland [22]. The development of this algorithm and the use of continuous, differentiable activation functions represented a major contribution to the neural network field and helped regenerate a stagnant research area: trainable networks consisting of multiple layers of perceptrons were now a reality.

2.4.1. MLP Operation

The basic architecture for an MLP is shown in Figure 2.8.

In general, the MLP consists of an input layer, a number of hidden layers and an output layer. Both the hidden and output layers consist of banks of perceptrons that use logistic function non-linearities. In principle, any number of hidden layers can be used, but it has been shown that an MLP with only a single hidden layer can approximate any continuous function to arbitrary accuracy provided enough hidden units are used [23]. Thus MLPs with a single hidden layer are currently used in most applications and are considered here.

The input units in the MLP distribute the components of the input vector to all the hidden layer perceptrons, with the output from each perceptron in the hidden layer

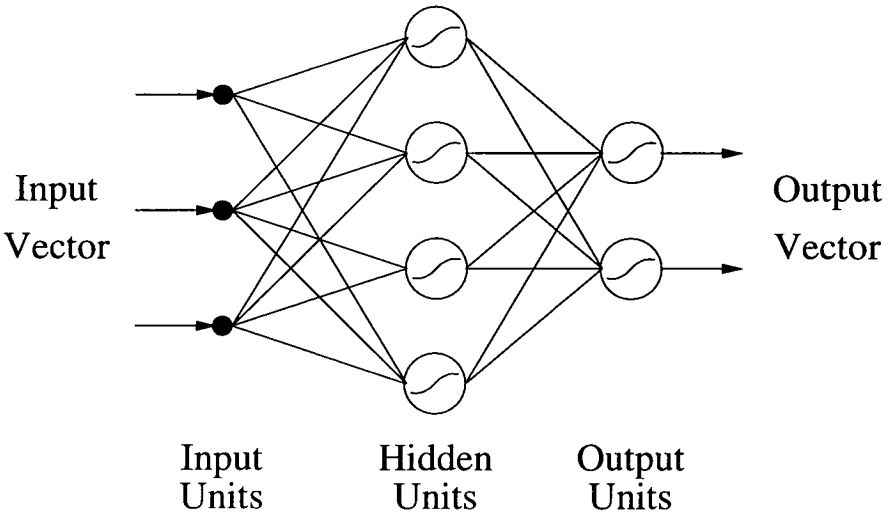


Figure 2.8 - The Basic MLP Architecture

distributed to all the output layer perceptrons.

The perceptrons in the hidden and output layers calculate the inner vector product (or dot product) of that unit's input neural state vector and the synaptic weights of the links feeding into that unit. A threshold value is then added to the resulting sum to form the *neural activation* value, equation 2.5:

$$x_{j_{act}} = \sum_i w_{ji} S_i + \theta_j \tag{2.5}$$

where $x_{j_{act}}$ is the neural activation value for neuron j, w_{ji} is the weight for the synaptic link connecting unit i in one layer to neuron j in the next layer, S_i is the input neural state from unit i and θ_j is the *threshold* value for neuron j, and is equivalent to w_0 in equation 2.2.

Each neural activation is then transformed into a neural state (between 0 and 1) by applying it to a non-linear function, usually the logistic sigmoid from Figure 2.3(b), equation 2.6.

$$S_j = \frac{1}{1 + e^{-\frac{x_{j_{act}}}{temp}}} \tag{2.6}$$

where S_j is the output neural state value, $x_{j_{act}}$ is the neural activation value from equation 2.5 and *temp* is a parameter which defines the slope of the sigmoid function.

Clearly each perceptron calculates a non-linear discriminant function as described in Section 2.2.

The new neural states, S_j, are then propagated to the next layer of perceptrons, or if the current layer is the output layer, the neural states form the network outputs.

Geometrically, the operation of an MLP for pattern classification can be considered as follows. The hidden units non-linearly transform the input space into a new space, the *classification space*, where the problem is linearly separable. The output perceptrons then calculate the discriminant functions for each class and the original input vector is assigned to the class whose discriminant function output is largest.

Since the transformation taking place in the hidden layer is done using a continuous monotonic function, $f()$, it is possible to project the straight decision boundaries in classification space back into input space by transforming them using the inverse of the non-linear function, f^{-1}. In this way, the overall operation of the network can simply be regarded as partitioning input space using non-linear decision boundaries, Figure 2.9.

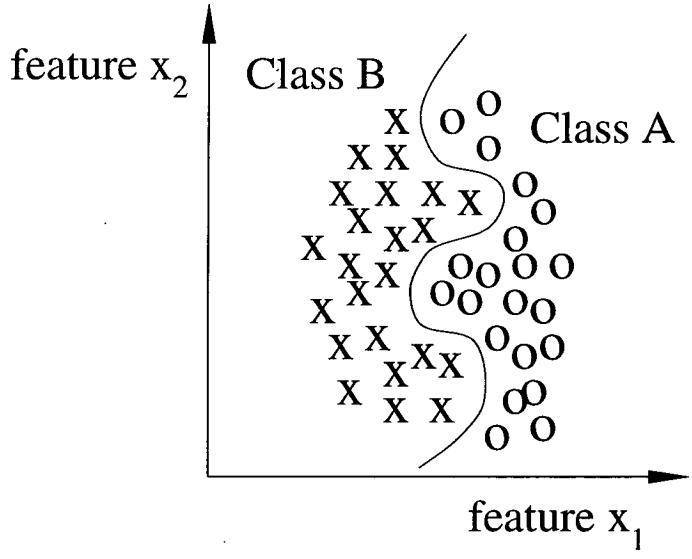


Figure 2.9 - A non-linearly separable two-class problem

2.4.2. MLP Training

The standard method for training an MLP is to use a two step supervised learning process [15, 17, 24]. Step 1 consists of presenting an input training vector to the network, propagating it through from input to output and generating an output vector. Step 2 consists of comparing this output vector to the desired output vector for the pattern, forming an error vector and using the error vector to adjust the weights and

reduce the *error cost function* for the network. This is repeated for all the training vectors until the error in the input to output mapping for the problem is minimised.

For non-linear networks such as the MLP, a weight set must evolve through the repeated application of an iterative training algorithm such as backpropagation. However, because MLP training is a non-linear optimisation task, it can be a long and laborious task.

2.4.2.1. The Backpropagation Algorithm

The backpropagation algorithm is a computationally efficient algorithm that allows the first derivative of the error cost function defined for the network to be calculated with respect to the current weight set [6]. Knowing the derivatives of the cost function with respect to the weights allows the weights to be changed so as to reduce the value of the cost function.

The cost function is normally expressed in terms of the sum of the squares of the differences between the desired and actual output vectors for all the patterns in the training set. This is expressed using equations 2.7 and 2.8:

$$E_{net} = \sum_{pats} E_{pat} \tag{2.7}$$

$$E_{pat} = \tfrac{1}{2} \sum_{k} (\tau_k - o_k)^2 \tag{2.8}$$

where τ_k is the desired target value for output k and o_k is the actual value for that output. The poorer the network performance, the higher the value of E_{net}.

Clearly the actual output can be re-expressed in terms of the the input vector components plus the synaptic weights and thresholds in the network, equation 2.9.

$$E_{net} = \tfrac{1}{2} \sum_{pats} \sum_{k} (\tau_k - f(\sum_{k} w_{kj} f(\sum_{j} w_{ji} x_{patsi} + \theta_j) + \theta_k))^2 \tag{2.9}$$

where w_{kj} is the weight of the link connecting output unit k to hidden unit j, w_{ji} is the weight of the link connecting hidden unit j to input i, θ_j is the threshold value for hidden unit j, θ_k is the threshold value for output unit k and $f(.)$ represents the continuously differentiable activation function.

By applying simple differential calculus to equation 2.9, the partial derivative of the cost function in terms of each network weight and threshold can be calculated, ie it is possible to discover the rate of change of the cost function with respect to all the network parameters.

A full derivation of the backpropagation algorithm is beyond the scope of this thesis, so only the main results will be summarised here [6].

Essentially the backpropagation algorithm allows the gradient of the current point in weight space to be expressed in terms of the inputs to the current layer, the errors between the target and actual output values and the first derivative of the non-linear activation function - hence the use of differentiable activation functions. In general, the expression for the change in the cost function with respect to the weights is given by:

$$\frac{\partial E_{pat}}{\partial w_{kj}} = \delta_{patk} x_{kj} \tag{2.10}$$

where δ_{patk} represents the error term for the kth node and pattern pat and x_{kj} is the jth input to that node. For output units, δ is expressed by equation 2.11 and for hidden units, it is expressed by equation 2.12.

$$\delta_{patk} = o_k(1 - o_k)(\tau_k - o_k) \tag{2.11}$$

$$\delta_{patj} = o_j(1 - o_j) \sum_k \delta_{patk} w_{kj} \tag{2.12}$$

In essence, equation 2.12, expresses the error for each hidden unit output in terms of the errors it produces in the perceptron outputs in the next layer.

During MLP training, the partial derivatives are calculated after each pattern presentation. However, the actual weight adaptations can be made either after each pattern presentation (stochastic or on-line learning) or the changes for each pattern can be summed and applied after **all** the pattern presentations (batch-mode learning). Some differences exist between stochastic and batch-mode learning and the choice of which one to use depends on the problem under consideration [25]. However, if stochastic training is used, changing the order of presentation for the training patterns after each epoch will usually produce a more robust network with better generalisation properties.

Geometrically, each partial derivative can be thought of as the positive (uphill) slope of the error surface, with respect to the weight or threshold, at the point on the error surface defined by the current weight vector. Taking partial derivatives with respect to all the weights produces a gradient vector which points in the direction of the maximum increase in E_{pat} for the current pattern. Thus moving in the opposite direction (ie downhill) will produce the maximum **decrease** in the cost function with respect to the current weights. Having found the gradient vector using the backpropagation

algorithm, most MLP training techniques use some form of *gradient descent* algorithm to adapt the weights in the network.

2.4.2.2. Simple Gradient Descent

Simple gradient descent is commonly used to train MLP networks. Since the back-prop algorithm allows the instantaneous gradient vector for each pattern to be calculated, simply moving in the opposite direction to this vector (or the sum of the gradient vector over all the patterns for batch-mode learning) the value of E_{net} can be reduced. Once the weights have been adapted they define a new point on the error surface that has its own gradient vector and the back-propagation algorithm must be re-applied.

Error surfaces for feedforward neural networks are usually very harsh, being characterised by steep slopes, flat plateaus and local minima solutions [26]. Given the severity of the error surfaces, it is not advisable to move too far on the surface for any one training iteration, so the actual change made to each weight is usually the gradient vector component for that weight attenuated by a small positive constant, equation 2.13.

$$\Delta w_{ji} = -\eta \frac{\partial E_{pat}}{\partial w_{ji}} \tag{2.13}$$

where η is the *learning rate*. Thus for each learning iteration (or *epoch*), the weight vector is adjusted slightly in the direction which causes the maximum instantaneous decrease in the error function.

The choice of learning rate can have a profound impact on the training time of the network. Choosing a small value for η tends to make the weights less susceptible to oscillations during training, but can dramatically increase the training time: weight updates are proportional to the gradient of the error surface, so the training proceeds very slowly across the plateaus. Alternatively, choosing a larger learning rate means that bigger steps can be taken, but this can lead to a lack of convergence or, if the learning rate is too big, to the weight values "exploding", their value incrementing out of control [24].

Several techniques exist for speeding up the convergence of the simple gradient descent algorithm. For example, solutions can be found more quickly if the learning rate is made adaptable [27, 28], or if a proportion of the previous weight changes are also included in the weight update equations - a process know as adding

momentum [6]. These techniques allow the local terrain of the error surface to influence the size of step the network takes towards a solution and can lead to faster convergence. However, as with all gradient descent techniques, they do not guarantee that the solution found will be the optimal one.

The presence of local minima, representing possibly good but non-optimal solutions, means that networks are frequently trained many times, from different starting points on the error surface, in order to find the network with the best generalisation performance. This further increases the time taken to find the best MLP architecture and weight set for a given problem.

2.4.2.3. Second Order Techniques

The simple gradient descent algorithm uses information about the first derivatives of the cost function, with respect to the weight set, to improve the fit of the model to the problem. Alternatively, second order techniques, which use information about the second derivatives of the same cost function, can be used to find solutions in fewer iterations. Examples of second order techniques include the conjugate gradient algorithm [25, 27] and quasi-Newton methods [29]. Second-order methods tend to be computationally intensive, although, given an initial starting point in weight space, second order techniques will normally find the nearest minimum more quickly than simple gradient descent [29].

2.5. Radial Basis Functions

Like the MLP, the RBF topology is a feedforward architecture which can universally approximate any function using only a single hidden layer, provided enough hidden units are chosen [30]. As will be explained, however, the RBF can often be trained faster and without the training pathologies of the MLP.

2.5.1. Principle of Operation

The RBF architecture, Figure 2.10, is similar in structure to the MLP, but operates in a slightly different way. Instead of having two or more layers of non-linear perceptrons, the RBF has a single hidden layer of non-linear basis function units, or *centres*, followed by an output layer of linear units. For RBF networks, each output is a linear combination of the basis function responses, with the output layer forming linear discriminant functions, equation 2.14.

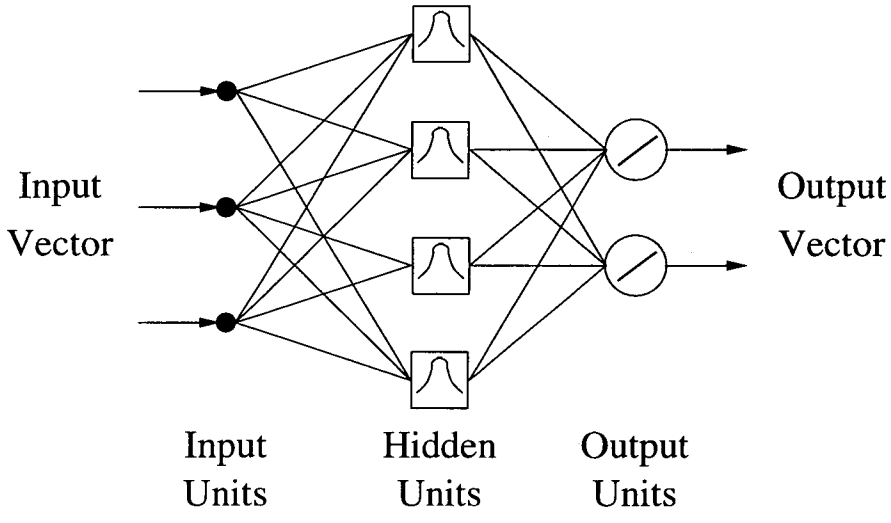


Figure 2.10 - The Radial Basis Function neural network architecture

$$y_k = \sum_{j=0}^{m-1} \lambda_{kj}\phi(\|\mathbf{x} - \mathbf{c}_j\|) + \lambda_{kbias} \tag{2.14}$$

In this equation, y_k is the output from output unit k, λ_{kj} is the weight associated with the connection between output unit k and centre j, $\mathbf{x}$ is the input vector, $\mathbf{c}_j$ is the *location vector* for centre j, $\phi(.)$ represents the basis function, λ_{kbias} is the bias term for output k and is similar to the threshold term in the MLP, and $\|.\|$ represents a distance metric. Usually the bias term, λ_{kbias}, is considered to be connected to an additional centre whose output is always 1.0, and it is absorbed into the summation, equation 2.15.

$$y_k = \sum_{j=0}^{m} \lambda_{kj}\phi(\|\mathbf{x} - \mathbf{c}_j\|) \tag{2.15}$$

Each hidden unit is assumed to have a position, $\mathbf{c}_j$, in input space and a corresponding region of influence, or *width*. Operationally, each hidden unit calculates the distance between its location vector and the input vector and outputs a response - again between 0 and 1 - that is a non-linear function of the proximity of the input vector to that centre, eg Figure 2.11. In other words, the response of each unit in the hidden layer of an RBF network is an indication of the **match** between the input vector and the position vector of that unit.

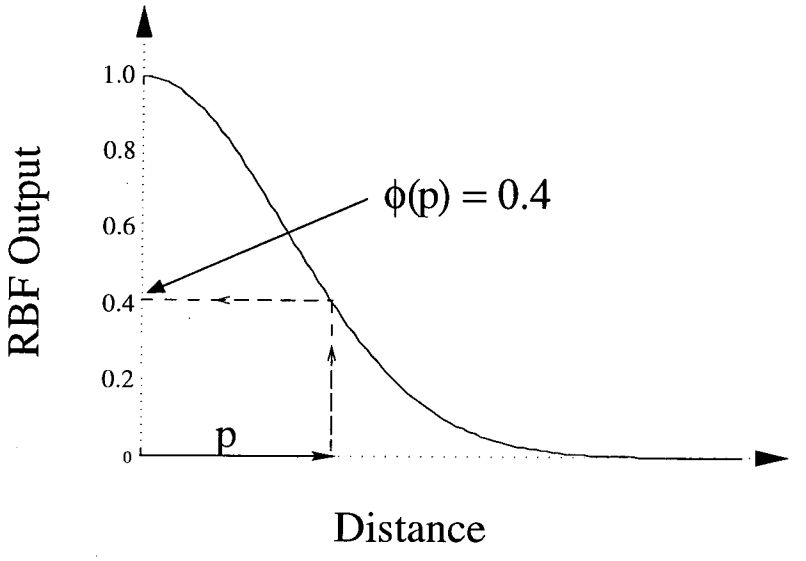


Figure 2.11 - A hidden unit in an RBF calculates the distance between the input vector and its location vector in d-dimensional space. It then uses this distance as the argument to a non-linear function as shown.

The hidden layer responses are then propagated to the linear output layer of the network, where each output unit calculates the weighted sum (dot product) of the hidden layer responses before adding a bias term (threshold), equation 2.15. This final sum is the network output for that unit. In pattern classification problems, the input data is assigned to the class corresponding to the output that produces the largest value.

The primary advantage of using RBFs over MLPs is that the two layers can be trained independently and, once the candidate centres have been chosen, the output layer weights can be determined using linear as opposed to non-linear optimisation techniques. This means that RBF training is faster than MLP training. Also, since the error surface for the output layer is a quadratic function of the output weights, a unique, optimal output weight set exists for the selected set of centres. Note that the solution will **not** be optimal for the given problem, rather the output weight set will produce the least squares solution **given** the selected set of centres.

Geometrically, an RBF network can be thought of as non-linearly expanding input space into a new space, usually of higher dimensionality [25, 31], where the problem is linearly separable. However, in contrast to the MLP, the classification space is partitioned using linear, instead of non-linear, discriminant functions.

2.5.2. Basis Function Considerations

Theoretically, many functions can be used as the non-linear basis functions, such as thin-plate splines

$$\phi(x) = x^2 \, log(x) \tag{2.16}$$

the multiquadric function

$$\phi(x) = (x^2 + r^2)^{\frac{1}{2}} \tag{2.17}$$

or the inverse multiquadric function

$$\phi(x) = (x^2 + r^2)^{-\frac{1}{2}} \tag{2.18}$$

where r is a real constant.

However, most RBF applications have concentrated on using the Gaussian non-linearity, equation 2.19, and Gaussians are used for all the illustrations in this thesis.

$$\phi(x) = e^{-\left(\frac{x^2}{2r^2}\right)} \tag{2.19}$$

As shown by equation 2.15, each basis function is required to calculate the distance between the input vector and the location vector for that basis function. The Euclidean distance measure, equation 2.20, is the most popular distance measure used, but the Manhattan distance, equation 2.21, often suffices.

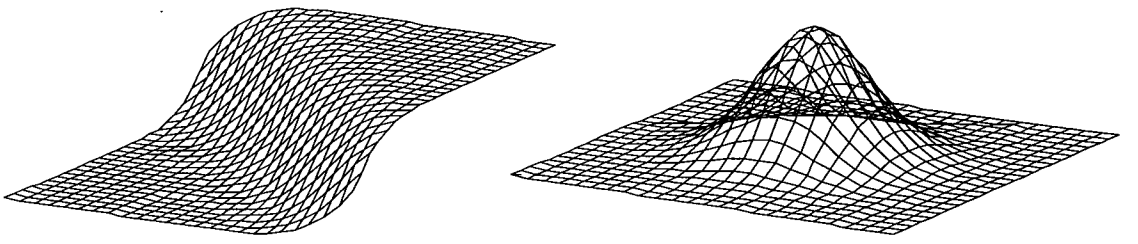
$$\| \mathbf{x} - \mathbf{c}_j \|_{euc} = \left((x_1 - c_{j1})^2 + (x_2 - c_{j2})^2 + \cdots + (x_d - c_{jd})^2 \right)^{\frac{1}{2}} \tag{2.20}$$

$$\| \mathbf{x} - \mathbf{c}_j \|_{man} = |x_1 - c_{j1}| + |x_2 - c_{j2}| + \cdots + |x_d - c_{jd}| \tag{2.21}$$

Using the Gaussian function and the Euclidean distance measure gives the following equation for an RBF:

$$y_k = \sum_{j=0}^{m} \lambda_{kj} e^{-\left(\frac{\| \mathbf{x} - \mathbf{c}_j \|^2}{2r^2}\right)} \tag{2.22}$$

Thus, in contrast, to the globally responsive perceptrons in the MLP, the centres in a Gaussian RBF are *locally responsive* and act only over a finite region of input space, Figure 2.12.



(a) (b)

Figure 2.12 - (a) An MLP paves input space with a number of globally responsive hyperplanes. (b) In contrast, an RBF paves input space with locally responsive hyperspheres.

The use of a single width parameter, r, for each dimension of the RBF is a simplification of the Mahalanobis distance, equation 2.23, a more general expression for the argument to the Gaussian in equation 2.22.

$$\left((\mathbf{x} - \mathbf{c}_j)^T \sum^{-1} (\mathbf{x} - \mathbf{c}_j)\right)^{\frac{1}{2}} \tag{2.23}$$

where

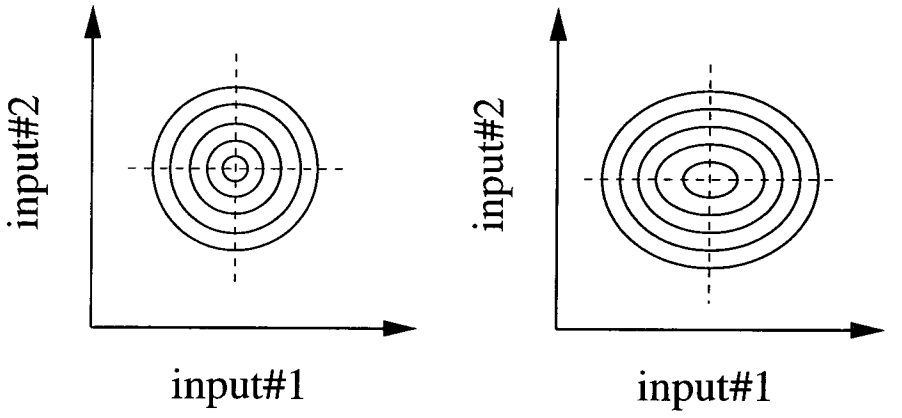
$$\sum = E[(\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T] \tag{2.24}$$

In equation 2.24, $E[.]$ is the expectation operator and $\mathbf{m}$ is the mean vector calculated from **all** the training vectors.

Using the Mahalanobis distance allows different shapes of basis functions to be defined [27]:

- If $\sum$ is a diagonal matrix with equal elements, then equation 2.22 describes the radially symmetric basis functions generated, Figure 2.13(a).
- If $\sum$ is diagonal with unequal elements, then hyperellipsoidal basis functions are produced, whose axes are parallel to the coordinate axes of input space, Figure 2.13(b).
- If $\sum$ is not diagonal, hyperellipsoids are formed in input space whose orientation to the co-ordinate axes is described by the covariance matrix $\sum$.

The use of the different width parameters for the separate dimensions can improve classification performance, eg [27, 32], although the number of parameters to be adjusted during learning is also increased.



(a) (b)

Figure 2.13 - (a) If a single value is used for the width, radial basis functions are produced. (b) If a diagonal co-variance matrix is used, the basis functions become elliptical.

For the work in this thesis, a single width is used for all the input dimensions since this keeps the number of adjustable parameters to a minimum and ensures that **radially symmetric** basis functions are produced.

2.5.3. Practical RBF Implementations

The use of Radial Basis Functions is not a new concept, confined to neural networks. Radial Basis Functions were originally applied to the problem of strict interpolation in high multi-dimensional spaces [33, 34].

The problem of strict interpolation can be defined as choosing a function $f(.)$ for a set of m n-dimensional data vectors, $\mathbf{x}_i$ ($1 \leq i \leq m$), and m real numbers $y \in R$, such that $f(\mathbf{x}_i) = y_i$, $\forall i \in m$. For strict interpolation, the function $f(.)$ is forced to pass through all the data points used in the curve fitting process [7].

Work by Poggio and Girosi also showed that radial basis functions provide a natural solution to certain *regularisation* problems [35]. For solving regularisation problems, the aim is to find a function $f()$ which minimises the cost function defined by

$$E_{cost} = E_{sq\_err}(f()) + \lambda E_{reg}(f()) \tag{2.25}$$

where the first term represents the usual error cost function summed over all the patterns and all the outputs of the network, the second term represents the regularisation *penalty* term for the cost function and the parameter λ is a positive number that

determines the contribution that the regularisation term makes to the cost function. The second term in equation 2.25 contains a stabiliser, P, that stabilises the solution and makes it smooth by employing some *a priori* information about the problem.

Poggio and Girosi show that if P is both rotationally and translationally invariant, then the solution for $f()$ is a sum of basis functions as defined by equation 2.15. Again, though, for a unique solution to the regularisation problem, all the data points in the training set must be used [35].

The use of all the training data for finding the exact, unique solutions for various problems may be conceptually pleasing mathematically, but for practical RBF implementations it is neither necessary nor desirable to fit a function to the vagaries of all the outputs.

The types of problems that require to be solved in the real world tend to be ill-posed, (meaning there is insufficient data from which to define a unique mapping from input space to output space) and the data vectors tend to be corrupted by noise. Further, if an abundant supply of training data is available, the use of all the data vectors can lead to the storage and use of vast amounts of possibly redundant data [7].

If fewer basis functions than data vectors are used, the problem becomes over-determined and no unique solution exists. In this case, the solution is constrained to lie on a vector sub-space of the space spanned by all the data vectors and the output weights must be found using a linear optimisation technique such as minimising a sum of squares error cost function.

2.5.4. RBF Training

The following review presents a small selection of the possible ways to implement RBF training, but is far from exhaustive.

2.5.4.1. Centres Chosen Directly from the Training Data

One of the easiest ways to train an RBF network is to simply choose N vectors at random from the training data and assign each input vector directly to a centre. This assumes that the chosen vectors adequately represent the problem and that enough have been picked to adequately cover input space.

The widths of the Gaussian centres are determined using a simple heuristic which sets each width equal to some proportion of the distance between that centre and its nearest neighbour or, alternatively, to some portion of the average distance between

the centre and its k nearest neighbours. The actual heuristic chosen does not really matter. What matters is the coverage of input space that the width parameter affords each centre. If more centres are used then the widths can be decreased, but if only a few centres are used, then large widths should be used in order to adequately cover input space.

Once the centre positions and widths have been determined, then the output layer weights can be calculated using either the least mean squares (LMS) algorithm [36] or some form of pseudo-inverse matrix technique [15].

The advantage of choosing the centres directly from the training set is its speed: no iterative learning is required within the hidden layer and the output layer can be trained very quickly. However, the learning scheme does have a significant disadvantage in that since the N centres are chosen at random, there is no way of knowing whether they truly represent the underlying problem. Thus the solution could be poor.

2.5.4.2. Adaptive k-Means Training

With this form of learning, an on-line version of the adaptive k-means algorithm [8] is applied to the hidden layer of an RBF network whose initial N centre values have been selected at random from the training vectors.

Once the candidate centres have been chosen, each vector in the training set is presented in turn to the network. The Euclidean distance between each training vector and each centre is calculated and the centre nearest to the current training vector is moved towards the training vector using equation 2.26:

$$\Delta c_{ji} = \alpha[x_i - c_{ji}] \tag{2.26}$$

where Δc_{ji} is the change made to component i of centre $\mathbf{c_j}$, x_i is the ith component of training vector $\mathbf{x}$ and α is a constant learning parameter.

The training data is usually presented in random order for a fixed number of epochs and the location of the candidate centres evolve to represent the distribution of the training data within input space. Once the centre positions have been adapted, the widths can again be calculated using some form of heuristic and the output weights again found by a linear optimisation technique.

2.5.4.3. Fully Supervised Learning

Instead of decoupling the two layers of an RBF network and training both separately, fully supervised learning can be used to train all the network parameters simultaneously [15, 37].

The sum-of-squared-errors cost function for an RBF network can be expressed in terms of all the network parameters if equation 2.22 is substituted into the right hand side of the cost function, equation 2.27.

$$E_{net} = \sum_{pat} E_{pat} = \tfrac{1}{2} \sum_{pat} \sum_{k} (\tau_{patk} - \sum_{j=0}^{m} \lambda_{kj} e^{-\frac{\| \mathbf{x}_{pat} - \mathbf{c}_j \|^2}{2r^2}})^2 \tag{2.27}$$

Again, by the application of simple differential calculus to equation 2.27, similar to that carried out for the MLP cost function, it is possible to express the rate of change of the network error in terms of all the adjustable parameters. Thus, the rate of change of the error with respect to output weight, λ_{kj} is given by

$$\frac{\partial E_{net}}{\partial \lambda_{kj}} = - \sum_{pat} \sum_{k} e^{-\frac{\| \mathbf{x}_{pat} - \mathbf{c}_j \|^2}{2r^2}} \left(\tau_{patk} - \sum_{j=0}^{m} \lambda_{kj} e^{-\frac{\| \mathbf{x}_{pat} - \mathbf{c}_j \|^2}{2r^2}} \right) \tag{2.28}$$

while the partial derivative of the error with respect to centre position component c_{ji} is given by

$$\frac{\partial E_{net}}{\partial c_{ji}} = - \sum_{pat} \sum_{k} \frac{x_j - c_{ji}}{r^2} \lambda_{kj} e^{-\frac{\| \mathbf{x}_{pat} - \mathbf{c}_j \|^2}{2r^2}} \left(\tau_{patk} - \sum_{j=0}^{m} \lambda_{kj} e^{-\frac{\| \mathbf{x}_{pat} - \mathbf{c}_j \|^2}{2r^2}} \right) \tag{2.29}$$

and the partial derivative of the error cost function with respect to width r_j is given as

$$\frac{\partial E_{net}}{\partial r} = - \sum_{pat} \sum_{k} \frac{\| \mathbf{x}_{pat} - \mathbf{c}_j \|^2}{r^3} \lambda_{kj} e^{-\frac{\| \mathbf{x}_{pat} - \mathbf{c}_j \|^2}{2r^2}} \left(\tau_{patk} - \sum_{j=0}^{m} \lambda_{kj} e^{-\frac{\| \mathbf{x}_{pat} - \mathbf{c}_j \|^2}{2r^2}} \right) \tag{2.30}$$

These partial derivatives can be used along with the simple gradient descent algorithm (or a more complex second order technique) to find a possible solution to the problem under consideration.

The main disadvantage with using such fully supervised learning, however, is that the problem becomes a non-linear optimisation, with the same pathologies and problems, such as local minima solutions, as an MLP. Indeed, when Moody and Darken used fully supervised learning to train their RBFs, they found that some of the local responsiveness of the network was lost, especially if the width parameters of the centres became large [8]. Tarassenko and Roberts suggest using either fixed or adapted

centre positions, as would be obtained from the first two learning strategies considered in this section, to get good initial guesses for the positions and widths of the centres. They believe that doing so will help to circumvent the the problems witnessed when the centres and widths are selected as small random numbers [37]. Any supervised learning will then be applied to these *educated guesses.*

Training an RBF network using a fully supervised method would not seem to provide many benefits over training an MLP network, but it may prove useful for fine-tuning a solution found by a faster method and does represent another possible method of training RBFs.

2.5.4.4. Resource Allocating Network

The three training mechanisms considered thus far have all required N centres be chosen from the training data. However, the pertinent question to ask is: how many centres should be selected in order to find a good solution to the problem ?

For the previous algorithms, N must be determined empirically, requiring that many simulations be run, using different values for N, in order to determine its optimal value and find the best weight set for the problem. A more efficient solution would be to allocate a new centre during training whenever a training vector with sufficient novelty was presented to the network. This is the motivation behind Platt's Resource Allocating Network (RAN) [20].

Training vectors are presented to the network in turn and if the current vector lies far enough away from all the current centres **and** the output error from the network exceeds some pre-defined threshold, then the current input vector is selected as a new centre. The position of this new centre is defined by its vector components, its initial width is the distance between it and the closest of the previously selected centres and the output weights are initialised as the network output errors for the current "novel" pattern.

If both novelty criteria are not met, then stochastic gradient descent is used to improve the fit of the network to the problem through further reducing the network error.

This algorithm is particularly efficient since it requires only a single pass through the data and also allows the complexity of the problem to determine how many centres are chosen. Indeed, more than one pass is often used to allow the solution to be improved through repeated application of the gradient descent algorithm.

However, since the first training vector will always be selected as the first centre, it may be advantageous to run the algorithm several times, with the input vectors presented in a random order each time, and select the network which gives the best performance on the problem test set of vectors.

2.5.4.5. Orthogonal Least Squares

The Orthogonal Least Squares (OLS) algorithm [38, 39], like the RAN algorithm discussed in the previous section, builds up the hidden layer of the RBF through time. The OLS technique selects candidate centres from the training set in a systematic way, determined by the contribution they make to producing the correct input to output mapping: those centres that contribute most to the mapping are selected in descending order. Once a centre has been selected, its directional component is removed from all remaining unchosen vectors using techniques such as Gram-Schmitt orthogonalisation [38]. Thus each candidate centre is selected based to its contribution to the reduction of the input to output mapping error in a direction *orthogonal* to the previously selected centres.

The algorithm can be stopped when either the requisite number of centres have been selected or when the approximation to the input to output mapping is good enough. The OLS algorithm does not provide a globally optimal solution, however, as this can only be determined after considering all possible subset models. A further drawback with this method is that all the Gaussian widths must be determined before training commences and the highest tolerable error on the training set must also be defined.

Thus, the OLS algorithm may also need to be applied to a given problem several times, with different values of Gaussian width and maximum allowable network error, before the network which produces the best performance on the test vectors is found.

Once again, after the hidden layer has been trained, the output weights can be found using a linear matrix inversion technique.

2.6. Summary

This chapter has considered the problem of pattern classification using discriminant functions, using it as a means to introduce the MLP and RBF feedforward neural network architectures. Both the operation and training of the two topologies has been reviewed and the use of both types of network for pattern classification problems

highlighted.

As stated at the beginning of the chapter, the aim has been to introduce the fundamentals of the MLP and RBF networks through a consideration of their use as pattern classifiers.

Chapter 3

RBF Hardware

Having discussed the theory behind RBF networks, this chapter considers the implementation of these networks in CMOS VLSI technology. CMOS VLSI has emerged as the most popular technology for implementing neural networks in hardware and a myriad designs exist for the different neural algorithms, eg see [40-42]. It is impossible to cover every example here, so this chapter only covers the circuitry capable of reproducing the mathematical operations necessary for RBF networks, viz. distance calculation, non-linear transformation, plus linear multiplication and addition.

The chapter begins with an explanation of the different operating modes of MOS transistors (or MOSFETs) before reviewing the implementation of complete RBF networks, or their constituent parts, using transistors biased into these modes. The hybrid analogue-digital pulse stream method is then discussed as an alternative implementation technique and some consideration is given as to how pulsed RBF circuits could be produced.

3.1. MOSFET Transistors.

The MOS transistor [43] is the basic building block of the modern electronics industry. Capable of denser implementation and dissipating less power than the older bipolar junction technologies, it is the technological advances associated with MOSFETs, especially the CMOS technology, that have allowed electronics products to become smaller, cheaper and have higher functionality.

CMOS transistors can be operated as digital switches or as analogue devices in either strong or weak inversion. This section discusses the three operating modes of the MOSFET, using the NMOS transistor for simplicity when describing strong and weak inversion analogue operation. Subsequent sections will review the neural circuitry designed for each mode.

3.1.1. Digital Regime

When operated digitally, CMOS transistors are simple complementary switches. Logic HIGHs turn NMOS devices ON and PMOS devices OFF, whilst logic LOWs have the opposite effect. Digital CMOS circuits, eg Figure 3.1, propagate and process logic levels through the charging and discharging of the input capacitances of the next logic gates in the data path. By combining MOSFETs in different ways, it is possible to implement a large number of simple logic gate, eg NAND, NOT, NOR, EX-OR, AND etc, and more complicated digital circuits, such as multipliers, adders and shifters, can be constructed from the simpler gates.

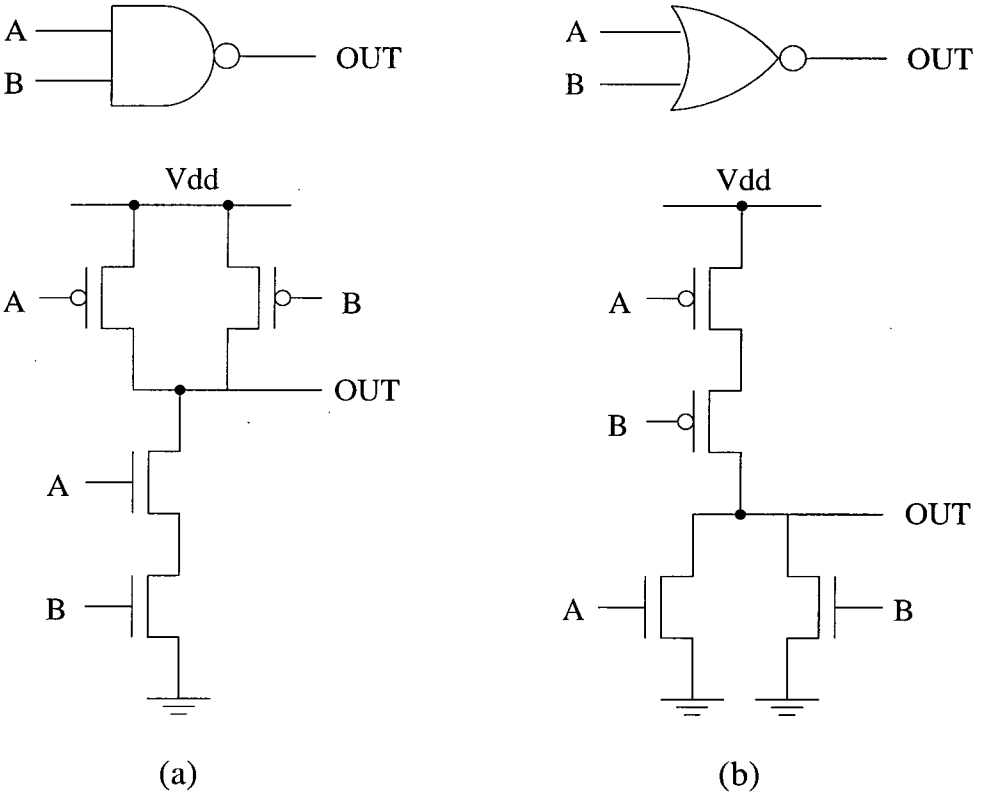


Figure 3.1 - Logic level (Top) and Transistor level representations of (a) a two input digital NAND gate and (b) a two input digital NOR gate

However, because each transistor only acts as a switch, huge numbers are required to implement high functionality logic blocks, with the number of required transistors scaling linearly with the required precision of the block. Thus as the precision of digital circuitry increases, so to does the area and power requirement.

3.1.2. Analogue Strong Inversion

Operating a MOS transistor, from a standard 5V process, with gate voltages between the usual logic levels of 0V and 5V lets the inherent physics of the device be exploited. Complex circuits can be designed using a fraction of the transistors required in the digital equivalent, with a corresponding drop in the area and power requirement of the circuit.

When biased into either weak or strong inversion, MOS transistors operate in the *linear* region or the *saturation* region depending on the values of V_{gs} and V_{ds}. For strong inversion operation, $V_{gs} \geq V_T$, and the transistor has an I-V characteristic as shown in Figure 3.2.

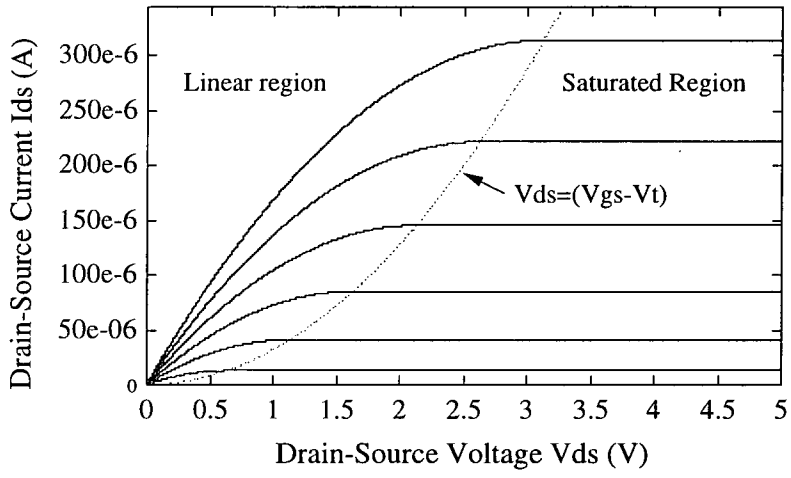


Figure 3.2 - The I_{ds} vs. V_{ds} first-order characteristic for an NMOS transistor operating in strong inversion

When $(V_{gs} - V_T) \geq V_{ds}$ the transistor is biased into its linear region and the first order drain-source current vs. drain-source voltage relationship is given by:

$$I_{ds} = \beta \left[(V_{gs} - V_T)V_{ds} - \frac{{V_{ds}}^2}{2} \right] \tag{3.1}$$

Meanwhile, when $(V_{gs} - V_T) \leq V_{ds}$, the transistor is in saturation and the first order current-voltage relationship is:

$$I_{ds} = \frac{\beta}{2}(V_{gs} - V_T)^2 \tag{3.2}$$

For these equations, I_{ds} is the drain-source current flowing through the transistor, V_{gs} is the gate to source voltage of the device, V_T is the threshold voltage and V_{ds} is the

voltage between the drain and the source. The transconductance parameter, β, is defined as [43]

$$\beta = \mu_0 C_{TOX} \frac{W}{L} \tag{3.3}$$

where μ_0 is the surface mobility of the carriers in the device, C_{TOX} is the capacitance per unit area of the gate oxide, W is the transistor width and L is the transistor length. By altering the $\frac{W}{L}$ ratio of a MOSFET, a circuit designer can tailor the operation of each device.

3.1.3. Analogue Weak Inversion

When a MOS transistor is operated with a gate to source voltage in the range, $0 < V_{gs} < V_T$, the transistor operates in the weak inversion, or subthreshold, region.

The characteristic equation for a transistor operating in weak inversion is given by [44, 43]

$$I_{ds} = \frac{W}{L} I_{dso} e^{\frac{qV_g}{nkT}} [e^{\frac{-qV_s}{kT}} - e^{\frac{-qV_d}{kT}}] \tag{3.4}$$

and this can be approximated to

$$I_{ds} \approx \frac{W}{L} I_{dso} e^{\frac{qV_{gs}}{nkT}} \tag{3.5}$$

when $\frac{qV_d}{kT} \gg 1.0$ and $V_s = 0$. The exponential current to voltage relationship for subthreshold transistors exists for several orders of magnitude [25]. In these equations, the terms retain the same definitions as described previously with the addition of the thermal voltage term $\frac{kT}{q}$ ($\approx$ 25mV at room temperature), n, the subthreshold slope factor and I_{dso}, a process dependent current [44].

The currents in the subthreshold region are in the pA to μA range, far smaller than the currents flowing in transistors biased into strong inversion. Subsequently the power dissipation in subthreshold transistors is very small, making them popular in applications where low power and high functionality is required [45]. The primary disadvantages with subthreshold circuits are their potential susceptibility to noise and poor transistor matching.

In the context of neural networks, subthreshold circuits have been used by various researchers [46-48], especially Mead [49, 50], mainly to implement locally

connected, fixed function circuitry capable of modelling biological "sensors" such as the retina or cochlea [49].

3.2. Digital Neural Implementations

As already discussed in Chapter 1, ANN architectures can be implemented serially on general purpose computers using a suitable high-level programming language. However, general purpose serial computers are not optimised for executing the set of operations required by parallel neural networks and are usually too slow for *real-time* applications. Thus, in order to speed-up the through-put of neural networks, several special purpose digital configurations can be used [51].

Standard neural architectures such as the MLP or RBF consist of straight-forward mathematical operations that are usually simple to cast into digital hardware. By employing the specialised digital arrangements as discussed below, the through-put of the neural network can be increased by exploiting the specialised instruction set, reduced precision computation and streamlined data-flow that these digital solutions can provide [52].

3.2.1. General Purpose Parallel Computers

This type of digital solution consists of several autonomous general purpose micro-processors connected together and operating in parallel. Each processor has its own memory and data paths and can operate using its own instruction set. Some examples are presented in a review by Atlas and Suzuki [51].

Although some researchers have investigated this technique for implementing neural networks, difficulties exist with the interconnectivity required for neural networks and the programming of the parallel processors [53]. Thus, this type of digital implementation is only mentioned for the sake of completeness.

3.2.2. Reconfigurable Digital Neurocomputers

A popular form of implementing ANNs in digital VLSI is with special purpose, reconfigurable boards. These usually consist of several interconnected DSP or FPGA chips on a PC board or bus-based system. Such solutions are generally capable of supporting several neural algorithms through the programmability inherent in the multi-functional, reprogrammable chips.

Reconfigurable neurocomputers usually provide at least an order of magnitude increase in data through-put compared to serial computers and can often utilise reduced precision data representations [52]. Examples of digital neural accelerators include the Adaptive Solutions CNAPS chip [54], the Virtual Image Processor of Cloutier *et al* [55], based on the Altera EPF81500 FPGA, and the L-Neuro 2.3 neurocomputer of Duranton [56], based on an array of 12 DSP chips.

Reconfigurable accelerator boards are available commercially and are an invaluable experimental tool for neural researchers allowing high functionality at low cost, with the chance to try different networks quickly and with the minimum of fuss.

3.2.3. Dedicated Digital VLSI

Although reconfigurable neurocomputers can provide faster throughput compared to general purpose microprocessors, speed of operation must be sacrificed for the flexibility to support several algorithms. In order to optimise speed, custom VLSI circuits are required. Again, as with the board-based accelerators, significant savings in area and speed can be achieved using reduced precision arithmetic and tailored architectures.

For example, Watkins and Chau [57] have investigated using reduced complexity VLSI to implement RBF neural networks that allow faster operation at lower power and which require a smaller silicon area compared to higher precision circuitry. Their 10-bit custom VLSI solution realised an 88% reduction in the power and area requirement compared to a 32-bit custom approach, for little loss in performance.

Another dedicated digital RBF solution that utilises reduced precision arithmetic is the Ni1000 chip from Nestor Inc. [58]. It can implement a 256 input, 1024 centre, 64 output RBF network and is capable of supporting other Gaussian-based topologies such as a Restricted Coulomb Energy (P-RCE) network [27], Parzen windows classifiers [59] and the Probabilistic Neural Network (PNN) [60].

The Ni1000 has a resolution of 5 bits for weight storage, uses the Manhattan distance measure in the hidden layer and is capable of classifying 40,000 patterns per second. It is fabricated in 0.8 μm Flash EEPROM technology on a 15.8mm by 13.7mm die and dissipates 5W at 5V.

In order to reproduce the Gaussian non-linearity, the Ni1000 uses a look-up table to exponentiate the distance measure. Maffezzoni and Gubin [61] have produced a custom RBF circuit that implements an approximation to a Gaussian using only digital

circuitry. Their method derives from considering the definition of a exponential in the limit

$$e^{-z} = \lim_{n\to\infty}\left(1-\frac{z}{n}\right)^n \tag{3.6}$$

which can be approximated by

$$e^{-z} = \begin{cases}\left(1-\frac{z}{2}\right)^2 & \text{if} x < 2 \\ 0 & x \geq 2\end{cases} \tag{3.7}$$

Clearly the RHS of equation 3.7 is far simpler to cast into digital VLSI than the LHS and reproduces an approximation to the Gaussian without the need for look-up tables. The authors discuss implementing the design in VLSI and indicate that for a 100 centre chip, they would expect to increase the through-put by two orders of magnitude over a similar design that uses a single sequential look-up table.

3.3. Analogue Implementations

The large area and power requirement of the components for multi-functional or fixed-function digital ANNs means only a relatively small number of processing elements can be provided on a given chip. For larger, more compact and lower power VLSI solutions, ANNs must be designed using analogue techniques.

The parallel structure of neural networks makes them amenable to arrays of regularly structured, repeatable circuit blocks [62], whilst the simplicity of the computational units allows them to be produced using a small number of transistors.

An implicit assumption with standard neural architectures is the individual computational elements all have exactly the same characteristics, with identical transfer functions. Whilst this is true for the processing elements within digital solutions, across-chip variations affect the operation of arrays of analogue circuits, causing different elements within the array to have slightly different characteristics. Variations in doping levels, ion implants and the photolithographic etching process, for example, lead to differences in the threshold voltage V_T, transconductance parameter β, and the drawn values of transistor widths W and lengths L. These variations, in turn, mean that different synapses can have different characteristics. In other words, arrays of analogue circuits are not matched. Further, the desire for compact circuitry in hardware neural nets means that precision is usually sacrificed for area reduction, with

the design engineer making the trade-off between area and acceptable accuracy.

However, learning schemes such as *chip-in-the-loop* learning have been shown to account for circuit inaccuracies and process variations, compensating for the loss of precision introduced by designing the neural networks in analogue VLSI [12, 63-65]. Thus analogue VLSI currently represents the best medium for the production of dense, compact realisations of artificial neural networks.

The structure of neural networks requires that the same input be applied to many processing elements within the array, while the outputs from these elements must be summed. Parallel signal distribution into high impedance nodes can be readily achieved using voltage as the signal medium, whilst the outputs can be summed as currents through Kirchoff's Current Law. For these reasons, most hardware neural processing units are based on *transconductance* circuits.

The transconductance parameter, β (equation 3.3), is used to define the range of currents in a MOSFET through appropriate choice of the width to length ratio, $\frac{W}{L}$. Tailoring β via the transistor width and length parameters defines the **static** transconductance properties of the transistor, ie those fixed at fabrication. However, the current through an operational transistor also depends on V_{gs}, V_{ds} and V_T as shown in Figure 3.2. Therefore, altering these quantities during normal circuit operation allows the **dynamic** transconductance properties of the transistor to be varied. The most common ways of achieving this dynamic variation in neural networks is through altering the gate or threshold voltages.

The dependency of I_{ds} on V_{gs}, combined with its high input impedance, makes a MOSFET an ideal building block for transconductance circuits. Typical examples of developed circuits will be considered later, once the various methods for storing weights within an analogue array have been introduced.

3.4. Weight Storage in Analogue Neural Networks

When using arrays of analogue circuits to build ANN solutions, the most efficient realisations store the neural weights locally. When designing feedforward neural networks, the important issues pertaining to weight storage are as follows.

- **Adaptability**. Each weight should be relatively easy to adapt so any non-idealities in the circuit can be trained out and the network can be retrained, if required, at some future date.

- **Storage Time**. Ideally, when the weights are not being adapted, their value should remain fixed indefinitely.
- **Resolution**. The range of implemented weights must have a high enough resolution to allow the problem to be adequately solved.
- **Implementation Area**. The area required to store the weight should be as small as possible to ensure that the area of each neural circuit can be as small as possible.
- **Process Compatibility**. The method chosen for weight storage should be compatible with existing processing steps in the chosen implementation technology.

Bearing these criteria in mind, the most suitable forms for the local storage of neural weights in analogue VLSI are local digital storage, capacitive storage with refresh and non-volatile analogue storage.

3.4.1. Local Digital Storage

It is possible to store neural weights for analogue networks as digital words in appropriate storage registers. This is commonly done in MDACs (Multiplying Digital to Analogue Converters). Examples of MDACs have been suggested by Hollis and Paulos [66], Tawel [64] and Jabri *et al* [67] amongst others.

The basic principle of an MDAC can be explained with reference to Figure 3.3. In this circuit, currents I_0 to I_3 represent binary weighted versions of biasing current I_{ref}. These weighted currents are selectively switched onto the common output line depending on the bit pattern stored in the registers D_0 to D_3. Thus

$$I_{out} = \sum_n 2^n I_{ref} d_n \tag{3.8}$$

where I_{ref} is the reference current and d_n is the value (0 or 1) of weight bit n.

In the example shown here, the output current from the DAC, I_{dac}, is used to bias the differential pair comprising transistors M_1 and M_2. The resulting differential output current, $I_+ - I_-$, then represents the desired product of the DAC current and the differential voltage $V_+ - V_-$. Two quadrant multiplication is achieved by using the MSB of the digital weight as a sign bit, steering the output current onto the appropriate output line.

Heim and Jabri [68] have also developed a digital static storage cell for analogue neural chips with *on-chip* learning. The circuit quantises a "learned" weight current to the nearest digital level using successive approximation, then regenerates an

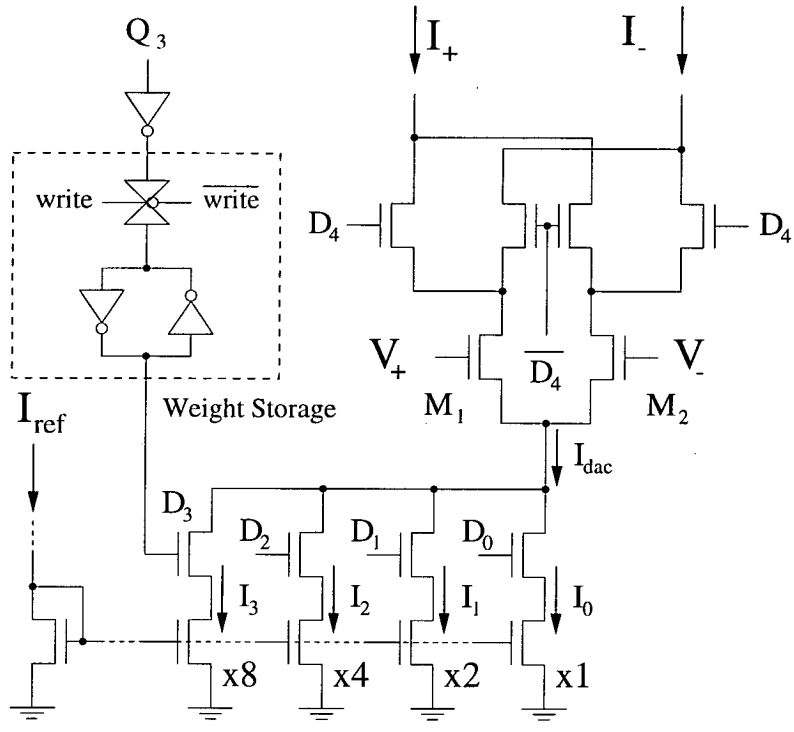


Figure 3.3 - MDAC with Local Digital Weight Storage

approximation to the current by using the stored digital word to switch binary weighted current sources onto a common output line.

Murray and Smith also used digital storage in the first pulse-stream chip [69]. The n bit digital synaptic weights were constrained to lie in the effective analogue range $-1 \leq T_{ij} \leq 1$ and the digital weight bits were used to selectively mask an input stream of pulses. The resulting masked pulse trains were then recombined on a common output line to yield the required product term.

The advantage of local digital storage is the ease of programming the weights (each weight storage circuit is essentially a write-only register), lack of corruption and indefinite storage times. The main disadvantages are the area requirement for storing each bit (higher resolution requires proportionally more area) and, for current-based MDACs, the difficulty in accurately weighting the currents as the resolution of the stored weights is increased. Nonetheless, local digital storage is an attractive solution for neural circuits requiring a weight precision (unsigned) of 5 bits or less.

3.4.2. Capacitive Storage with Refresh

A very popular method for the local storage of analogue weights is as the charge on a MOS capacitor implemented either between the two polysilicon layers on a double poly CMOS process or, more usually, using the gate to substrate capacitance of a MOSFET biased into strong inversion [70].

Storing charge on a capacitor causes a voltage $V_{wgt} = \frac{Q_{stored}}{C}$ to be developed between its plates and this voltage represents the analogue weight. The storage capacitor is accessed via a MOS transistor switch, Figure 3.4, and the system forms a crude sample and hold circuit.

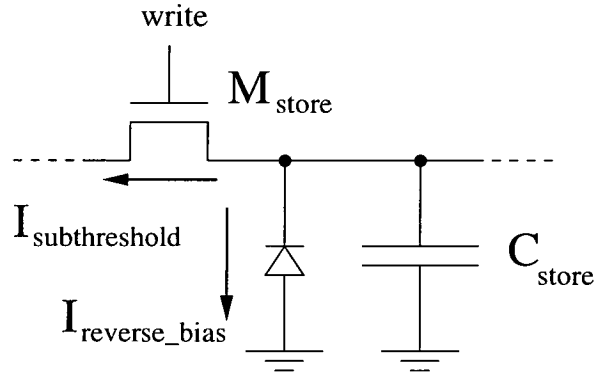


Figure 3.4 - Schematic diagram of a capacitive storage refresh node showing the subthreshold and reverse-biased diode leakage currents

The voltage on the capacitor will not be stored indefinitely, though. Charge leakage effects due to subthreshold conduction through the switch and the reverse biased diode current associated with the drain terminal of the MOS switch cause charge to leak from C_{store} and the stored voltage decays accordingly.

Refresh schemes are therefore required to periodically "top-up" the charge on the capacitor and ensure that the stored value remains as stable as possible. The refresh schemes are either global or local and the resolution of the stored weight depends on the size of the storing capacitor, the leakage currents and the time between refreshes. For neural applications, weight storage of over 8 bits has been reported using capacitive storage [71].

3.4.2.1. Global Capacitative Refresh

With global refresh, each neural circuit is accessed sequentially and the weight for that circuit is written from a global source, usually the combination of an external RAM chip and a voltage DAC, Figure 3.5.

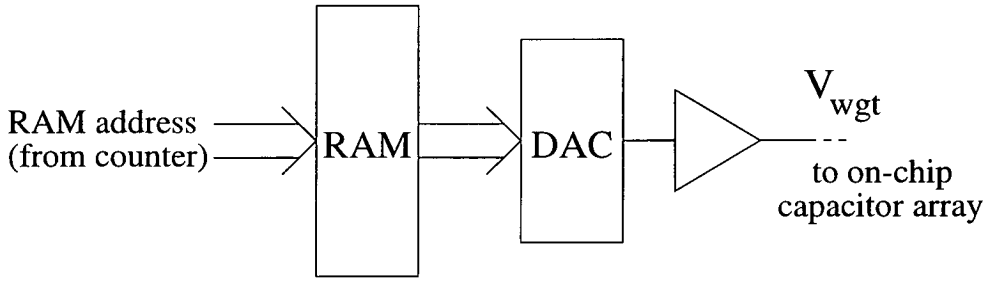


Figure 3.5 - Schematic Diagram of a typical Off-Chip Global Refresh System

The advantage of this form of refresh is that each weight is periodically refreshed with the trained value held in the global source. It is also easy to load the weights into the global source initially and adapt them iteratively during learning. Also, only a single refresh circuit is necessary for the whole array.

The main disadvantage with the system is that the refresh rate depends on the size of the network and may need to be customised for neural chips with different numbers of hidden and output units. Since the refresh rate depends on capacitor size and leakage currents, it must be chosen such that the time between refreshes **for the whole array** is less than the time required for any stored weight to decay by a voltage equivalent to half the LSB. Thus as the network size increases, the refresh rate and/or capacitor size must also increase to retain the precision of weight storage. This could prove problematic for very large networks.

3.4.2.2. Local Capacitative Refresh

Alternatively, rather than control the weight refresh globally, it can be done locally, eg Figure 3.6.

The principle of operation behind this system is straightforward [70]. Stored voltage, V_{wgt} is continuously compared to ramped voltage, V_{ramp}. When V_{ramp} exceeds V_{wgt}, the comparator switches, activating the pulse generating circuitry and causing a short pulse to turn on M_{switch} momentarily. This action restores the value on C_{store} to the instantaneous, quantised value of V_{ramp}. The pulse generating circuitry then remains SET until RESET, ready for the next refresh cycle, by the falling edge of V_{ramp}.

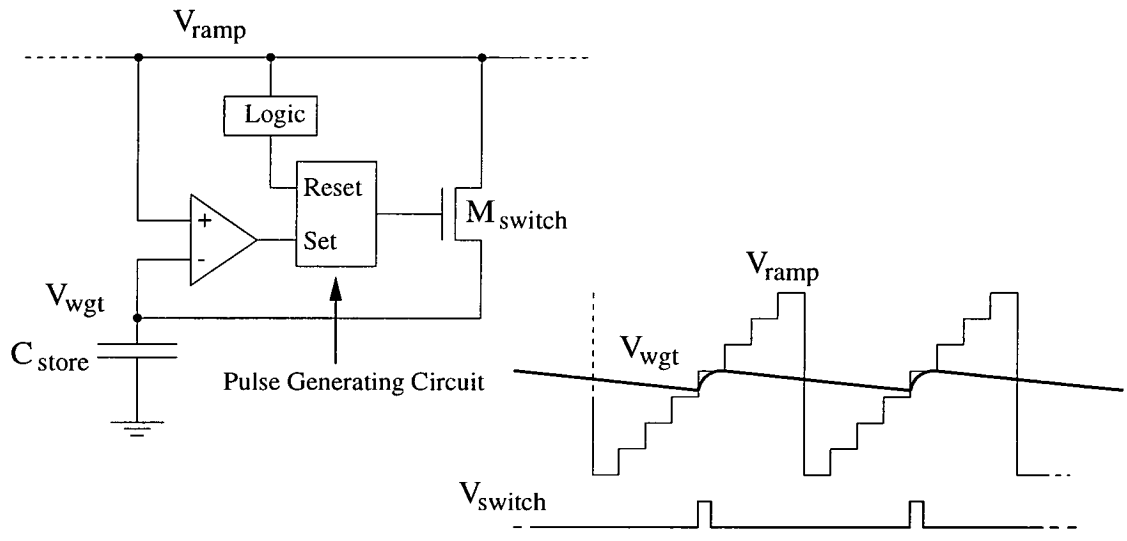


Figure 3.6 - Local Refresh Circuit Similar to that of Vittoz et al

Castello *et al* have designed a similar system, capable of a 5 bit resolution, using *current mode* circuits [72].

Hochet proposed a similar local refresh system, capable of 5 bit storage at 100kHz, based on ratioed clocks [73]. However, rather than distribute the ramp signal globally, Hochet's circuit generates it locally using globally distributed clock waveforms. The refresh rate of this circuit is determined by the periodic time of the slowest clock.

The main advantage of local refresh is that the refresh rate is independent of the size of the network, with only a ramp (or clock) signal needing to be distributed globally. With all these refresh circuits, though, the periodic time of the globally distributed signals and the resolution of the stored weight are again determined by the size of the storage capacitors and the leakage currents.

The main disadvantages of local capacitive refresh are:

- the refresh circuitry has to be included within every weight storage circuit, therefore it must be compact
- weight initialisation could prove troublesome
- any errors produced in the V_{wgt} values due to noise or interference could lead to the capacitors being incorrectly refreshed, irrecoverably, to the wrong quantised value.

In order to try and remove this last disadvantage, and potential source of error, Cauwenberghs and Yariv have developed a fault-tolerant local refresh scheme, capable of supporting at least 8-bit weights, where the analogue value is updated by a fixed increment in the direction of the closest quantised value, rather than being over written completely [74]. Again, however, the periodic time of the global ramp must not allow the voltage to decay by more than half the LSB.

3.4.3. Non-Volatile Analogue Weight Storage

The highest packing density for neural weight storage can be achieved using non-volatile analogue techniques. Examples include EEPROM technologies such as MNOS transistors and floating-gate transistors, as well as adaptable materials such as amorphous silicon. Not only are these devices compact, but because weight storage depends on altering a physical characteristic of the device, they retain their weight value even during power-downs.

3.4.3.1. Floating Gate Technology

The MNOS and floating gate technologies have both been used to implement EEPROMS via the alteration of transistor threshold voltages to one of two extremes, arbitrarily chosen to represent logic LOW and logic HIGH [75]. Both techniques rely on trapping charge in order to modulate the threshold voltage; the MNOS device traps charge between the nitride and oxide layers, whilst the floating gate transistor traps charge on an insulated polysilicon gate above the transistor channel, Figure 3.7. Both techniques can also produce intermediate changes in the threshold voltage, directly proportional to the trapped charge.

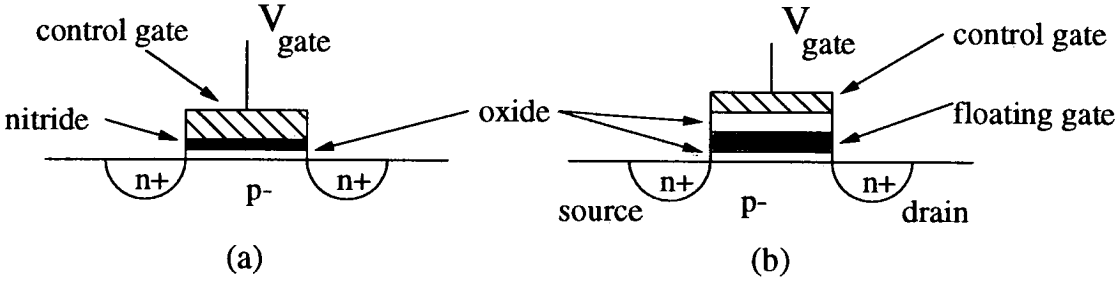


Figure 3.7 - (a) An MNOS Transistor and (b) a Floating Gate MOS Transistor

Whilst floating gates can be implemented on any standard double poly CMOS process, MNOS transistor technology is more exotic, requiring a special nitride layer.

Thus, although MNOS transistors have been used to implement neural weights [76], the use of floating gates is more widespread and the remainder of this discussion will be limited to them.

Floating gate transistors consist of an additional polysilicon gate, buried in the oxide and electrically isolated from the main gate terminal. Mechanisms for adding or removing charge from the floating gates include hot-electron injection [77], Fowler-Nordheim tunnelling [78] and the use of UV radiation [79].

The use of floating gates has been widely reported within the context of non-volatile weight storage in analogue neural networks, eg [70, 80-84], however, since different fabrication processes possess different characteristics, no universal, transferable programming standard exists for the effective implementation of floating gates in all VLSI processes. In summary, whilst the implementation of floating gates is straightforward, programming them is not and this constitutes the main disadvantage with their use.

Meanwhile, the principle advantage of using floating gates is that through modulating the charge on the isolated gate, it is possible to modulate the transconductance property of the floating gate transistor: although the floating gate is physically isolated from the control gate, it is capacitively coupled to it. Therefore, it is possible to produce neural circuits where the weight is actually stored within the circuit, contributing to its transfer characteristic, and the desired operation can be achieved by appropriately controlling the voltage on the control gate. This technique has been widely exploited in the distance and multiplier circuits discussed in later sections.

3.4.3.2. Amorphous Silicon

Amorphous silicon can also be used to implement programmable neural weights, not through the trapping and subsequent storage of charge, but through directly altering the resistive properties of the silicon.

Research into hydrogenated amorphous silicon, *a-Si:H,* has shown that, when sandwiched between vanadium (top) and chromium layers, Figure 3.8 [85], amorphous silicon can be programmed to have a resistance of between 1kΩ and 1MΩ. This could have significant implications for neural networks, with the possibility of the neural synaptic circuits and refresh circuits being replaced by programmable resistors. Indeed, neural networks have already been manufactured using fixed value amorphous silicon resistors, but these networks were fixed function and not adaptable

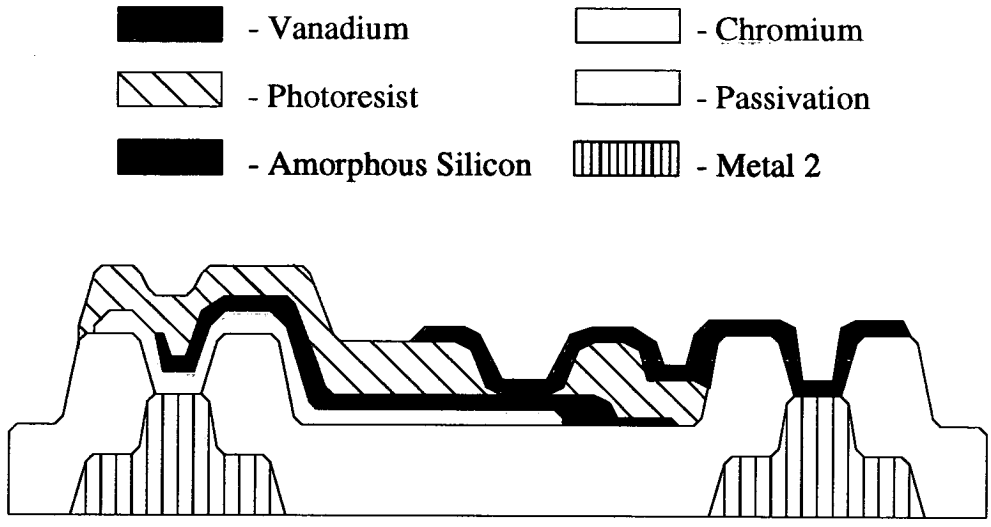


Figure 3.8 - Amorphous Silicon Cross-Section

[86].

Amorphous silicon is programmed as follows. After fabrication, short duration (300ns) voltage pulses incremented between 5V and 14V are applied to the vanadium electrode. This facilitates the formation of a vertical conducting channel in the amorphous silicon. This channel can then be iteratively programmed using shorter duration (120ns) pulses between 2V and 5V to implement the required resistance value in the range 1kΩ to 1MΩ[87].

Although this technology requires vanadium and chromium layers, these processing steps are required **after** the normal fabrication steps in a standard CMOS process. However, the resolution of the devices may be poor (4 bit resolution has been reported [88]) and a recent investigation concluded that a-Si:H resistors were not the optimal weight storage devices for small scale ANNs, as may be envisaged for application specific chips [85].

3.5. Strong Inversion Circuits for RBF Operations

For transistors operating in strong inversion, both the linear and saturation region equations yield transconductance relationships that are useful for feedforward neural networks. Here the use of strong inversion MOSFETs for reproducing RBF functions is considered.

3.5.1. Multiplication

To bias a transistor into its linear region, V_{gs} must be at least a threshold voltage above V_{ds}. Thus big gate voltages imply linear operation, while small gate voltages imply saturated operation.

For small V_{ds}, equation 3.1 becomes

$$I_{ds} \approx \beta (V_{gs} - V_T) V_{ds} \tag{3.9}$$

and I_{ds} is approximately linear in both V_{gs} and V_{ds}. Some researchers have suggested using single transistors to implement linear two-quadrant multiplication for neural networks, eg [89], Figure 3.9. However the source voltage must be kept fixed for this implementation.

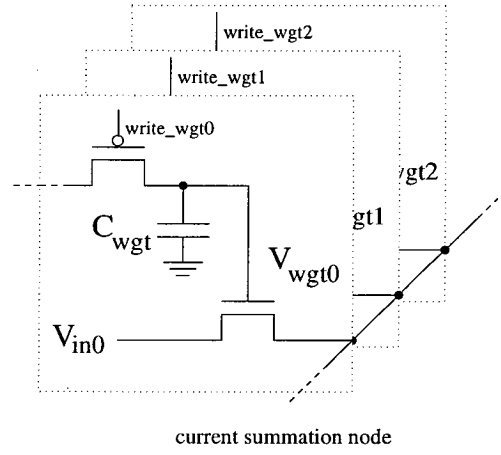


Figure 3.9 - An Array of Single Transistor Multipliers

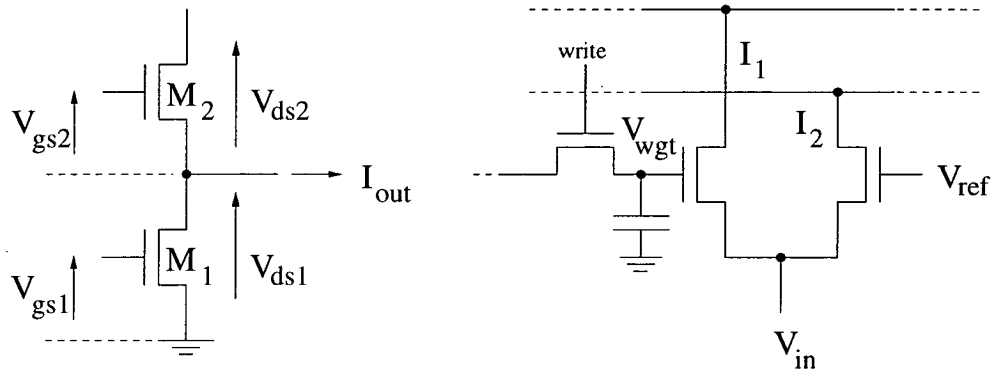
An alternative approach uses two matched transistors connected so that the non-linear terms cancel. Denyer and Mavor [90] suggested the circuit shown in Figure 3.10(a), for use in array-based monolithic filter applications. By assuming that transistors M_1 and M_2 are matched, biased into their linear region and have equal drain-source voltages,

$$I_{out} = \beta (V_{gs2} - V_{gs1}) V_{ds} \tag{3.10}$$

This circuit has been adapted for pulse-stream neural implementations and will be discussed in more detail later.

Another synapse circuit based on two linear MOSFETs was presented by Kub *et al* [91], Figure 3.10(b). Their circuit outputs a differential current

$$I_1 - I_2 = \beta (V_{wgt} - V_{ref}) V_{in} \tag{3.11}$$



(a) (b)

Figure 3.10 - Linear Transconductance Multiplier Circuits

exactly as required for two-quadrant operation. The disadvantage with this arrangement, however, is that low impedance amplifiers are required to convert the differential current to a voltage.

Differential pairs [91, 92] can also be used to implement multiplication in ANN chips. A differential pair circuit is shown in Figure 3.11(a) and has the following strong inversion transfer characteristic

$$I_1 - I_2 = \beta \Delta V_{diff} \left(\frac{2I_{bias}}{\beta} - \Delta V_{diff}^2 \right)^{\frac{1}{2}} \tag{3.12}$$

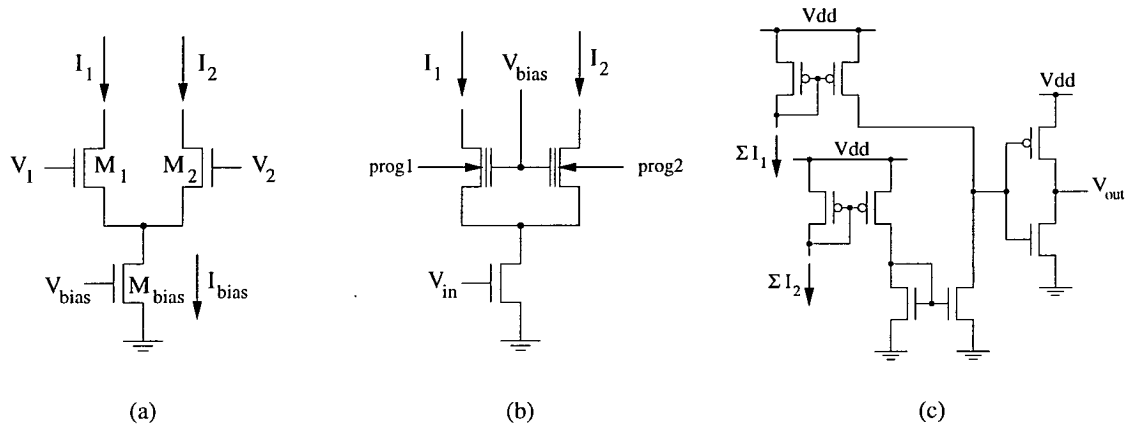
where ΔV_{diff} = V_1 - V_2 and I_{bias} is the bias current generated by M_{bias}. Clearly the output of this circuit is non-linear in both I_{bias} and ΔV_{diff}. However, for small ΔV_{diff}, the equation reduces to

$$I_1 - I_2 \approx \beta \Delta V_{diff} \left(\frac{2I_{bias}}{\beta} \right)^{\frac{1}{2}} \tag{3.13}$$

and this can be re-written as

$$I_1 - I_2 \approx \beta \Delta V_{diff} (V_{bias} - V_T) \tag{3.14}$$

if M_{bias} is saturated. Thus for small differential voltage inputs, the circuit in figure 3.11(a), can implement two-quadrant multiplication. In fact, the naturally saturating characteristic of the differential pair makes them suitable for implementing *multiply-and-add* arrays requiring asymptotic sigmoidal non-linearities.



(a) (b) (c)

Figure 3.11 - Schematic of (a) A Differential Pair, (b) Borgstrom's Floating Gate Differential Pair and (c) Borgstrom's Current Comparator Neuron

The simple differential pair and modifications to it, for example the Gilbert Multiplier [93], modified Gilbert Multiplier [83] and folded Gilbert Multiplier [94], have found widespread use in hardware neural networks [92, 49, 83, 95, 96, 80], including in the work of Borgstrom *et al* [97], who implemented a cascadable two-quadrant multiplier as shown in Figure 3.11(b). This circuit multiplies the difference between two stored floating gate voltages by V_{in} and highlights how compact analogue circuits can be produced using analogue techniques and a very small number of transistors. For Borgstrom's implementation, currents I_1 and I_2 from all the synapses feeding into a neuron are summed onto common lines and passed to a current comparator with a sigmoidal characteristic, Figure 3.11(c).

3.5.2. Euclidean Distance Based Circuits

Saturated MOSFETs have been used in compact circuits that calculate the Euclidean distance [98, 99], or squared Euclidean distance [84], between two vectors of voltages.

Landolt, Vittoz and Heim [98] have produced a self-biased circuit based on the squarer circuit of Bult and Wallingá [100]. Their circuit can be used to calculate the Euclidean distance between two n-dimensional vectors. However, as pointed out by Collins *et al* [84], to form the squared Euclidean distance between two n-dimensional vectors, the original circuit of Bult and Wallinga can simply be replicated n times and the n outputs connected to a common node. The main disadvantage with these two circuits is that they operate in current mode, requiring that, for

sensor fusion applications, the input voltages are initially converted into accurate currents that must be distributed to the relevant circuits. As already discussed, input signals are most easily distributed as voltages.

In comparison, the circuit produced by Tuttle *et al* [99], Figure 3.12, operates with input voltages and only uses small geometry transistors. This circuit block self-calibrates and memorises differential code-book vector components during its initialisation phase. It subsequently produces an output current equal to the Euclidean distance between the input and stored vectors, using a single, saturated transistor, during the evaluation phase. However, this circuit was not produced for implementing RBF networks, so no non-linear circuit has been designed for this implementation. As with the circuit of Landolt, Vittoz and Heim, however, the squared Euclidean distance between the two vectors, represented as a current, can be obtained from this circuit if the diode-connected output transistor is replaced by a current mirror.

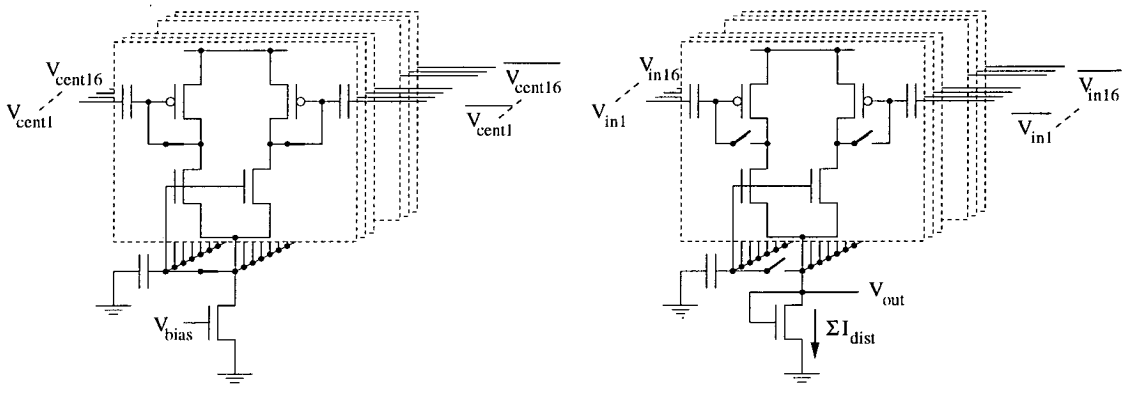


Figure 3.12 - The Euclidean Distance Cell developed by Tuttle *et al* showing (a) the Initialisation Phase and (b) the Evaluation Phase

Collins *et al* have developed a novel, compact, squared Euclidean distance cell designed for an RBF network [84], Figure 3.13(a). This circuit uses floating gates to "memorise" each centre position in a manner analogous to that used by Tuttle *et al,* except that the Collins centre circuit permanently stores the centre positions. Once the centres positions have been stored, complementary input vectors V_{in} and V_{in_bar} can be presented to the control gates. (Only one transistor conducts at any time, so complementary inputs are used to account for $V_{in} > V_{centre}$ and $V_{in} < V_{centre}$.) The input voltages capacitively couple to the floating gates, modulating the voltages on them and thus altering the current flowing in the device. Since the devices operate in

saturation, the output current is proportional to $(V_{in} - V_{centre})^2$.

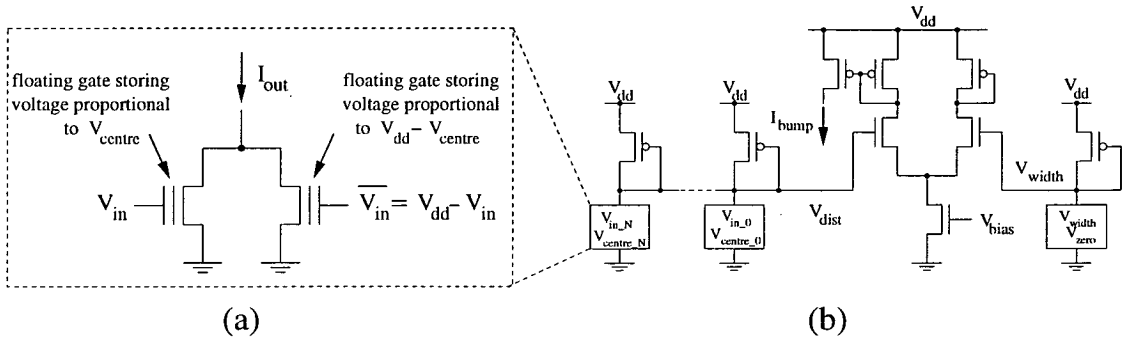


Figure 3.13 - (a) The Two Transistor Squared Euclidean Distance Cell Developed by Collins *et al* and (b) its Implementation in an N-Dimensional Centre Circuit

The output currents from the Collins cells are summed and converted to a voltage using a distributed, diode-connected load transistor, before being applied to a bump circuit [101], Figure 3.13(b). The bump circuit is based on a differential pair and I_{bump} depends on the proximity of V_{dist} to V_{width}, where V_{width} controls the spread of the bump. Subsequently, the I_{bump} current from each centre is passed to the output layer where it forms an input to the output layer's synaptic multipliers. The synaptic multipliers of this implementation are based on a modified Gilbert multiplier [101]. Collins and co-workers have designed a 3 input, 3 centre, 2 output chip to test the functionality of their circuits and predict that a 32 input, 160 centre, 16 output RBF network could be fabricated on a 1cm by 1cm microchip [102].

The centre circuit designed by Collins *et al* was an improved version of an earlier one produced by Anderson *et al* [103]. The operation of the Anderson circuit, Figure 3.14(a), depends on the fact that an inverter passes a non-linear current near its switching voltage. Anderson *et al* use this non-linear current to approximate the squared Euclidean distance between an input vector and a reference vector. Again the reference vector position is programmed onto the floating gate inputs to the inverters. The currents through all the inverter circuits in a given centre are summed and converted to a voltage using a sense amplifier tailored to produce the desired bump function.

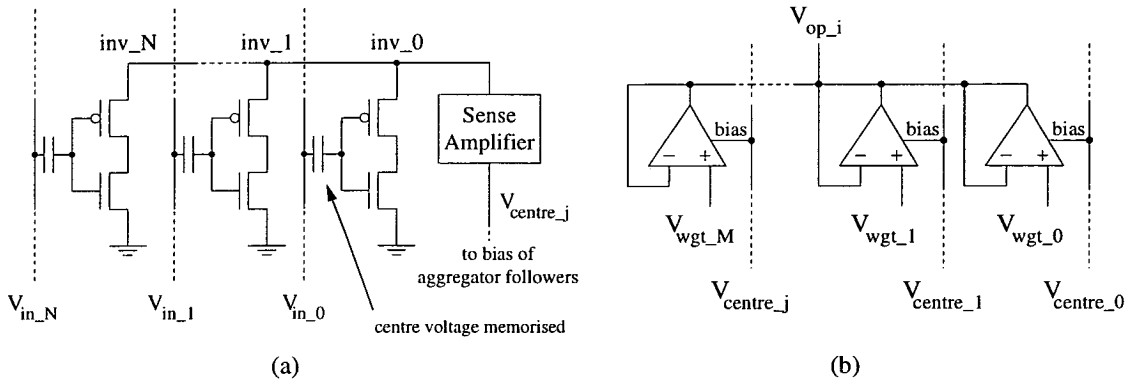


Figure 3.14 - (a) The Centre Circuit Developed by Anderson *et al* and (b) the Output Layer for their Chip

The outputs from the centre layer are combined using an *aggregator follower* [49] based on the one shown in Figure 3.14(b). The aggregator follower blends the hidden layer responses to implement equation 3.15, the equation for a normalised RBF network using the partition of unity[1].

$$y_j = \frac{\sum_i \lambda_{ji}\phi_i}{\sum_i \phi_i} \tag{3.15}$$

The chip produced by these researchers incorporates an 8 input, 159 centre, 4 output RBF network in 2.2mm by 9.6mm of silicon (2μm process) and has a static power dissipation of 2mW.

Finally, another distance circuit designed for a non-RBF application formed the basic element in the Euclidean distance block-matching array of Kramer [62]. The circuit fabricated here was developed by Seevinck and Wassenaar [105] and produces an output current that is the square of the difference between its input voltages.

3.5.3. Manhattan Distance Based Networks

As an alternative to calculating the Euclidean distance between two vectors, the Manhattan distance can be calculated instead.

[1] Normalised RBFs are valid RBF implementations, but have slightly different properties compared to normal RBFs [104].

Verleysen *et al* have designed and manufactured a classification chip which supports both vector quantisation and kernel-based classification [106, 107]. Their Manhattan distance circuitry is shown in Figure 3.15. Each component cell, Figure 3.15(a), stores a centre position current using a dynamic current mirror, loaded via M_{in}. Both M_{in} and M_{centre} are biased into their linear region by transistor pairs M_1/M_2 and M_3/M_4. The bidirectional current produced by the cell is either sourced from or sunk to the I_{neg} or I_{pos} lines respectively. By rectifying I_{neg} and adding it to I_{pos}, the Manhattan distance between two vectors can be realised.

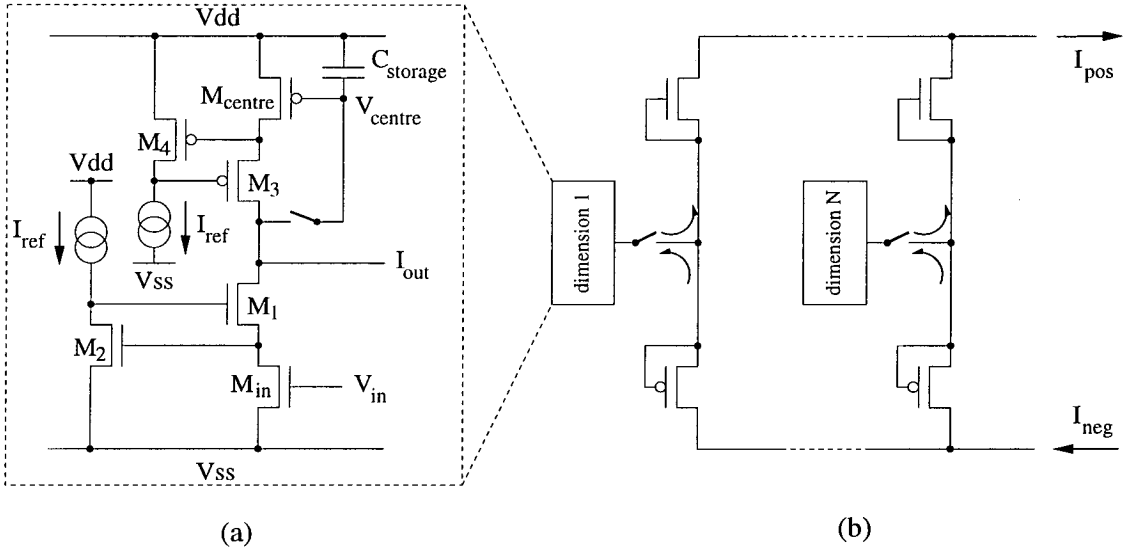


Figure 3.15 - (a) The Single Dimension Manhattan Distance Cell of Verleysen *et al.* (b) *N* Cells are Cascaded to Form the Total Manhattan Distance

Kernel circuits have also been developed by Verleysen and co-workers. One uses the non-linear properties of the differential pair to produce an approximation to the Gaussian, whilst the other uses transistors biased into subthreshold, exploiting the natural exponential transfer function in that region of operation [106].

The use of the Manhattan distance has also been advocated by Dogaru *et al* [108] after they obtained impressive results from software simulations of a network using a modified RBF non-linearity and the Manhattan distance metric.

Finally Cauwenberghs and Pedroni have designed a novel, compact Manhattan distance cell [109], although again it has been designed for use in a vector quantiser chip rather than a hardware RBF implementation.

3.6. Weak Inversion RBFs

Some implementations of RBF networks using subthreshold currents have also been proposed.

Watkins and co-workers [48, 95] have implemented a hybrid RBF network where the hidden layer functions are all executed using analogue subthreshold circuits and the output layer is implemented using a fast DSP chip. In the hidden layer, the distance calculation is calculated using a folded Gilbert Multiplier, figure 3.16(a), while the exponential transconductance function is produced using the circuit in Figure 3.16(b).

These researchers considered implementing a fully analog RBF network, but declined to do so, believing analogue inaccuracies and limitations in their circuitry would be compensated for by the increased precision in the digital output layer [95].

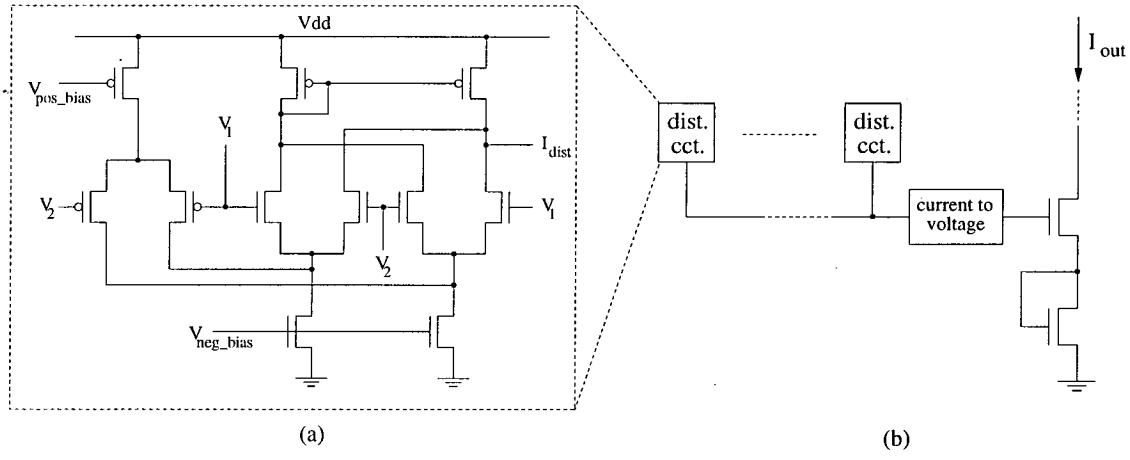


Figure 3.16 - (a) The Folded Gilbert Multiplier Used for Distance Calculation by Watkins et al and (b) the Variable Width Neuron they developed

Another subthreshold implementation has been proposed by Harris [110]. He suggests using circuits based on Delbruck's bump circuit [111] to implement RBFs, again using the partition of unity by implementing the output layer as an aggregator follower. Indeed, the implementation proposed by Harris is similar to that of Anderson *et al,* with the output currents from the bump circuits controlling the conductance of the transconductance amplifiers - via the *bias* input - in the aggregator follower in Figure 3.14(b) [112].

3.7. Pulse Stream Neural Networks

Hybrid neural systems are systems that combine analogue and digital circuit techniques, eg [113, 114].

One hybrid technique which has been successful in the neural network arena is the pulse stream neural method pioneered at the University of Edinburgh [9]. The pulse stream technique encodes the neural states within an architecture, in analogue format, along the time axis of a stream of digital pulses [115]. The neural states are usually represented as either the repetition frequency of a stream of fixed width pulses - pulse frequency modulation (PFM), or the width of a stream of fixed frequency pulses - pulse width modulation (PWM), Figure 3.17.

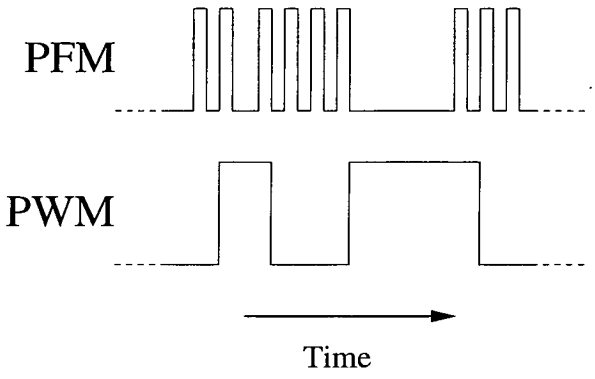


Figure 3.17 - Representation of PFM (top) and PWM (bottom) where analogue neural state information is recorded in time as the repetition rate of fixed width pulses and the width of fixed frequency pulses respectively

The original motivations for the use of pulses were the analogy with the biological exemplar - pulses are known to be used in natural neural systems - and the desire to implement analogue circuitry on economical, digital CMOS processes. While the first motivation may seem somewhat ambitious given the sheer magnitude of the difference in complexity between biological networks, believed to have evolved over millions of years, and the current artificial variety, which have existed for a little over fifty years, the second motivation has important consequences and implications for the design and fabrication of neural network chips.

Analogue VLSI allows the design of compact, fast, low power, asynchronous circuits that, unfortunately, are not robust to noise or interference and are susceptible to process variations. On the other hand, digital VLSI technologies are more readily available and cheaper, but have been tailored to produce circuits whose primary aim is to

process logic signals quickly; signals that are easily generated, transmitted, multiplexed and regenerated as required.

Pulse stream techniques therefore utilise the known advantages of both analogue and digital VLSI technologies, combining the use of analogue circuitry in the computational core of neural architectures, with the use of digital encoding technology for the transmission of analogue neural information as robust digital pulses. Since the digital pulses can be used to switch analogue circuitry **on** or **off**, the resulting pulsed circuits are compact and operate at all times in known and well defined ways.

3.7.1. Pulse Stream Circuits

The best way to illustrate the applicability and power of the pulse stream technique for neural VLSI, is through a review of some of the circuitry which has been developed over the years. Many varieties of pulsed circuits have been developed during the past decade, eg [69, 115-120], too many to describe in detail. For this reason, this review considers circuits from the EPSILON Cell Library [12], which evolved from early pulsed designs at the University of Edinburgh.

The EPSILON chip is a large generic analogue neural network chip, fabricated in the early 1990s, and designed to act as both an analogue neural accelerator and as a stand alone neural processor [10-12]. Subsequent experiments with the chip indicated that analogue circuitry is unlikely to be advantageous for producing reconfigurable neural co-processors - digital VLSI is much better - but that analogue solutions for application specific problems are likely to find a niche market. The production of the EPSILON chip was a tremendous achievement, marking the coming of age of the pulse stream technique, and the on-board circuitry serves as a good example of the power of the methodology.

3.7.1.1. Synaptic Multiplication

In order to implement the weighted sums, $\sum T_{ij}V_j$, the EPSILON chip used the circuit shown in Figure 3.18(a) [10].

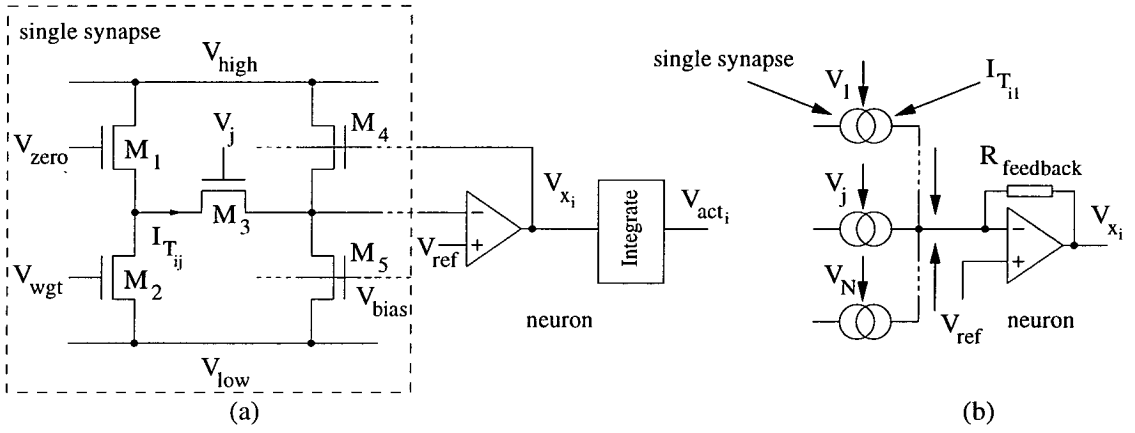


Figure 3.18 - (a) The Pulsed Synaptic Multiplier Developed by Baxter and (b) the equivalent circuit describing its operation

This 5 transistor circuit is cascadable and is based on the linear transconductance multiplier of Denyer and Mavor [90] described earlier in this chapter. The circuit is electrically equivalent to the one in Figure 3.18(b), with each synapse in (a) implementing a pulse-controlled, variable current source.

Transistors $M1$ and $M2$ are operated in their linear region and, assuming they are well-matched, current $I_{T_{ij}}$ can be expressed as

$$I_{T_{ij}} = \beta(V_{gs1} - V_{gs2})V_{ds} \tag{3.16}$$

where V_{gs1} is the gate-source voltage of transistor $M1$, V_{gs2} is the gate-source voltage of $M2$ and $V_{ds} = V_{ref}$, maintained at $\frac{V_{high} + V_{low}}{2}$.

Transistor $M3$, controlled by the input pulse stream V_j, is used to gate pulses of current, $I_{T_{ij}}$, between transistor pair $M1$-$M2$ and transistor pair $M4$-$M5$. For this circuit, V_j can be encoded as either PFM or PWM. Transistors $M4$ and $M5$, in conjunction with the Op Amp, implement negative feedback with the result that the Op Amp output voltage, V_{x_i}, represents a snap-shot of the instantaneous neural activity of all the synapses feeding into that particular neuron. By integrating V_{x_i} over time, the synaptic activation V_{act_i} can be calculated.

3.7.1.2. Pulse Frequency Neuron

Since pulse frequency modulation encodes the neural state, V_j, as the frequency of a stream of fixed width pulses, PFM neurons are simply voltage controlled oscillators (VCOs). The output frequency of these VCOs, ranging from f_{out} = 0Hz for V_j= 0.0 to $f_{out} = f_{\max}$ for V_j = 1.0, depends non-linearly on the instantaneous synaptic activation, V_{act}. The PFM neuron from the EPSILON Cell Library is shown in Figure 3.19 [12].

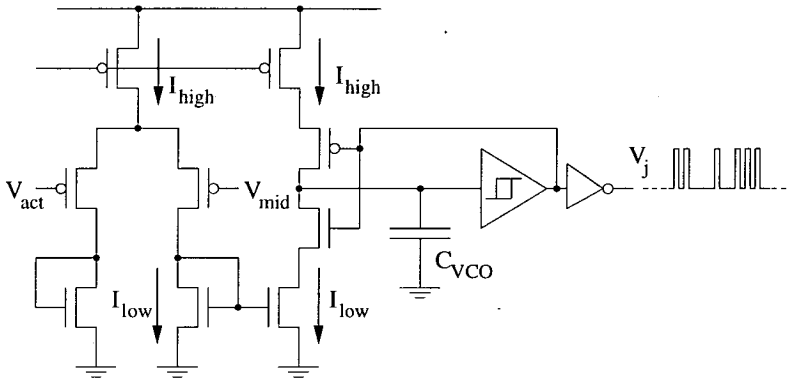


Figure 3.19 - The EPSILON PFM Neuron

The output pulse stream, V_j, is generated by the hysteretic charging and discharging of capacitor, C_{VCO}, using currents I_{high} and I_{low}. I_{high} is derived from a global reference current and is used to define the fixed width of the pulses; I_{low} defines the spacing of the pulses and is generated by the differential pair in Figure 3.19. Due to the non-linear (sigmoidal) characteristic of the differential pair, I_{low}, which lies in the range $0 \leq I_{low} \leq I_{high}$, is the correct non-linear transform of $V_{act} - V_{mid}$. The frequency of the output pulse stream is therefore the correct non-linear function of the input synaptic activation voltage, V_{act}.

The asynchronous nature of PFM makes it ideal for implementing asynchronous, recurrent architectures such as the Hopfield network [121]. However, the data-dependent evaluation time means that PFM is less suited for feedforward networks since the neural evaluation time cannot be guaranteed.

3.7.1.3. Pulse Width Neuron

The early hardware neural research focused on implementing simple recurrent architectures and most early pulsed circuits operated using PFM. The desire to develop circuits and chips capable of reproducing feedforward neural architectures, such as

MLPs, led to the design of circuits operating using PWM.

With pulse width modulation, the frequency of the pulses is constant and the neural states are encoded as the widths of *individual* pulses. Thus a separate neural calculation is performed per pulse and the evaluation time of the circuit is synchronised to the frequency of the pulses [122].

A simple PWM neuron is shown in Figure 3.20. The neuron is a comparator with the synaptic activation voltage, V_{act}, applied to one input and a ramp voltage waveform, V_{ramp}, applied to the other. The ramp waveform has been traditionally generated off-chip using an external RAM with a voltage DAC, allowing the non-linearity to be arbitrarily defined.

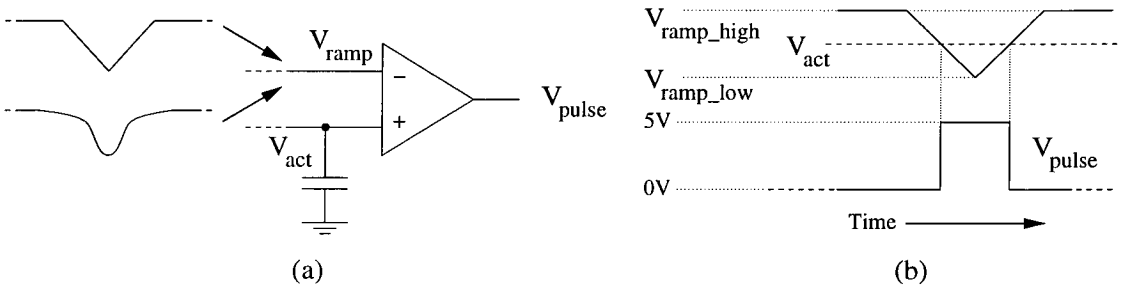


Figure 3.20 - A PWM Neuron

The principle of operation of the neuron is simple. The constant neural activation voltage V_{act} lies somewhere between the maximum and minimum extremes of V_{ramp}. When V_{ramp} dips below V_{act}, the comparator switches, switching again when V_{ramp} exceeds V_{act} once more. This double switching action produces an output pulse whose width in time depends jointly on V_{act} and V_{ramp}. The maximum evaluation time for the PWM circuit in Figure 3.20, is the length of time the ramped waveform deviates from its maximum value.

In order to correctly implement the non-linear neural transformation from the voltage domain to the time domain, V_{ramp} should be stored as the inverse function, $f^{-1}()$, of the desired non-linearity, $f()$. This is because the activation voltage is effectively being transformed from the y-axis variable into the x-axis variable. Also, symmetrical ramps are often used to ensure that all the comparators do not switch simultaneously as this can exert a large transient demand on the power rail. Symmetrical ramps produce symmetrical pulses centred on the mid-point of each time frame and are advisable for large networks with many neurons, but are not absolutely necessary for networks with only a few neurons.

3.8. Pulsed RBF Networks

Having described and demonstrated the elegance of using pulse streams for neural networks implementations, it is time to consider the motivations and scope for the implementation of Radial Basis Functions using the pulse stream technique.

3.8.1. Motivations

The reasons for implementing RBFs using pulse streams are as follows.

i) RBFs have not yet been implemented using pulses, while other network architectures have been successfully demonstrated. This makes any investigation into developing pulsed RBFs academically interesting.

ii) The potential advantages and limitations of using pulsed RBF networks for application specific problems can be investigated through developing the final demonstrator chip and applying it to a small-scale problem.

iii) The pulse stream technique can be developed and extended by building on previous work. The pulsed circuits described in Section 3.7 were developed from earlier designs, taking into account the lessons learned from their production and use. Thus the knowledge gained from testing and using EPSILON and subsequent pulsed circuits augments all the earlier work and all this practical information can be used in the development of the next generation of pulsed circuits.

3.8.2. Scope

Before describing the development of the pulsed circuitry in Chapters 4 to 6, the initial considerations regarding the nature and extent of the project will be summarised in order to define limits for the scope of this work.

- Primarily this project was a concept proving exercise. The aim was to develop and refine circuitry that demonstrated the possibility, or otherwise, of producing pulsed RBF circuitry. Thus large geometry processes were used because they were more economical, allowing for the production of two test chips as well as a final demonstrator.
- The desire to test whether theoretically viable circuits would work when manufactured meant that simple circuits were designed whenever possible. The transistors and capacitors were over-sized to minimise transistor mismatch and parasitic coupling respectively, and the MOS switches were designed to be small to

minimise charge injection. All the circuitry was also laid out very conservatively. These measures meant that area was consumed to give greater accuracy, but this was deemed to be a warranted trade-off for a concept proving project.

- Since PWM is a synchronous technique that guarantees the evaluation time for a forward pass through the circuit, it is more suited to implementing the RBF algorithm than PFM.
- From a consideration of the requirements for the RBF algorithm, the most natural partition is between the two layers. Thus it was decided to develop separate circuits for the hidden and output layers, with PWM neurons, Figure 3.20, converting the voltages produced by the hidden layer into pulses for the output layer, Figure 3.21.

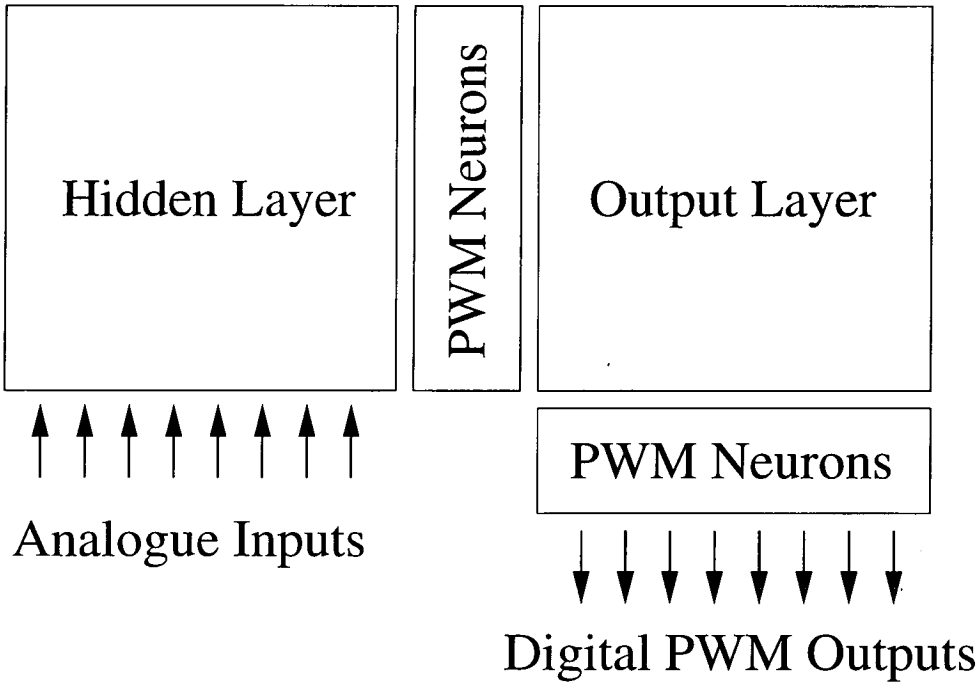


Figure 3.21 - Floorplan of Proposed RBF Demonstrator Chip

- The aim was to design pulsed circuits for implementing the forward pass through an RBF architecture. No attempt was made to produce circuits capable of on-chip learning and it was envisaged that chip-in-the-loop learning would be used to account for process variations and offsets.
- From the review of analogue weight storage techniques presented earlier, it is clear that floating gate technology probably represents the best method, in the long term, for implementing the local storage of neural weights. However,

programming these devices is not trivial and the most appropriate method for storing the weights in these development chips was as dynamic voltages, on MOS transistor capacitors, globally refreshed from off-chip RAM. In order to ensure that weight refresh was continuous, without affecting the through-put of the chip, the refresh system was designed to be transparent to the operation of the hidden layer and output layer circuits.

- All the developed analogue circuitry was designed to operate in strong inversion. None of the pulsed circuits developed thus far have used weak inversion and it is not clear what effect the noise generated by the rising and falling edges of the pulses would have on the subthreshold voltages or currents.
- Finally the centre circuits and output circuits were designed to be cascadable. This allows networks of any size to be designed without significant re-design of any circuit blocks, although the refresh rate may have to be increased as discussed in Section 3.4.2.

3.8.3. Related Work

Having defined the scope of the work contained in this thesis, it is worth mentioning the related work that was carried out in parallel, but independently, of this work by Reyneri and co-workers at the Politecnico di Torino in Italy.

This group use Coherent Pulse Width Modulation (CPWM) [123], a technique very similar to PWM, for implementing their neural chips.

Reyneri and co-workers are currently trying to implement the Weighted Radial Basis Function algorithm (WRBF) [124] in VLSI, using current mode circuits. Their aim is to produce a WRBF chip [125], based on an earlier design [126], which will be applied to various control problems. Although not directly comparable to the work in this thesis, their research is nonetheless relevant to this work, providing a useful alternative contribution to pulse stream neural network research.

3.9. Summary

This chapter has considered the different ways in which the RBF neural algorithm could be implemented in VLSI hardware. The use of both analogue and digital techniques was considered, and the actual implementation method, the hybrid pulse stream technique was described in detail. Finally the motivations for the project were discussed, along with a review of the initial considerations that defined the scope of

the project.

Now that the implementation method has been decided, it is time to discover how the algorithm was actually cast into hardware, beginning with the output layer circuitry.

Chapter 4

The DYMPLES Chip

As illustrated in Chapter 2, both the MLP and the RBF neural architectures require two-quadrant multipliers for the calculation of the vector dot (or inner) product of the weight and the neural state vectors. This requirement has led to the design and development of a variety of different circuit implementations to carry out this function, some of which have been reviewed in Chapter 3.

This chapter presents the DYMPLE synapse[2], the production of which was motivated by the desire to produce a simple, cascadable, pulsed synapse that was easy to set-up and operate and that could be easily transferred between different fabrication processes. To satisfy these requirements, the synapse was developed using the *current mode* approach [127-129]. Further, in order to test the viability of the new design and demonstrate its functionality, an array of DYMPLES multipliers were fabricated on a suitable test chip.

4.1. Voltage Mode vs. Current Mode Operation

The difference between *voltage mode* and *current mode* operation can be succinctly summarised as:

Voltage mode circuits use voltages as the main signal medium,
current mode circuits use currents.

Analogue IC design has been predominantly voltage mode for two main reasons.

- All circuit design was traditionally analogue voltage mode design.
- After the advent of the IC, a greater emphasis has been placed on digital technology and there has been a lack of investment in alternative analogue IC design techniques.

Before the widespread utilisation of cheap and reliable ICs, the majority of electronic circuit design was analogue in nature and used discrete components. Circuits tended

[2] DYMPLES is an acronym for DYnamic Mirror PuLsed Experimental Synapse.

to favour processing voltages rather than currents (eg operational amplifier circuits and discrete BJT transistor based circuits) due to good voltage processing performance. The inherently poor matching achievable with discrete components rendered them unsuitable for the current mode approach. Thus, before the advent of the IC, analogue voltage mode circuit design was a mature subject.

The advent of the IC saw the digital circuit replace the analogue circuit as the primary tool of the electronics industry. Compared to analogue circuits, digital ones are reliable and precise and the trends in technological advances, especially in computing, have favoured digital ICs. Nowadays, fabrication processes continue to be optimised for digital rather than analogue circuit performance, more investment is directed towards improving digital techniques and the development of analogue IC design has suffered as a result.

This situation is changing, however. The demand for increased functionality, increased performance and *system-on-a-chip* integration has led to renewed interest in analogue design techniques. Digital technology remains the dominant force, though, and as the level of digital integration increases, power supply voltages must decrease as a direct result. Reducing the supply voltage has serious implications, in terms of dynamic range and noise, for analogue voltage mode operation [127].

In comparison, analogue current mode operation has a high immunity to the level of the power supply voltage. In addition, current mode circuits tend to be simpler than voltage mode circuits, are more transferable between different processes and, due to their reliance on voltage **differences** rather than absolute values, are more amenable to fabrication on the existing digital processes. Thus current mode design has increased in popularity as a direct result.

4.2. Pulsed Two-Quadrant Multiplication

Two-quadrant multiplication is a straight-forward mathematical operation and the set of ideal characteristics for neural versions of such multipliers is given in Table 4.1.

However, as with all engineering designs, trade-offs must be made between these different competing priorities, always striving to make the implementation as small and as simple as possible.

Figure 4.1 shows the schematic diagram for a simple implementation of a pulsed two-quadrant multiplier whose operation depends on the principle of conservation of charge.

The Ideal Synapse.	
Multiplier	Circuit
Accurate	Small Area
Cascadable	Low Power
Linear	Robust

Table 4.1 - The Characteristics of an Ideal Synapse

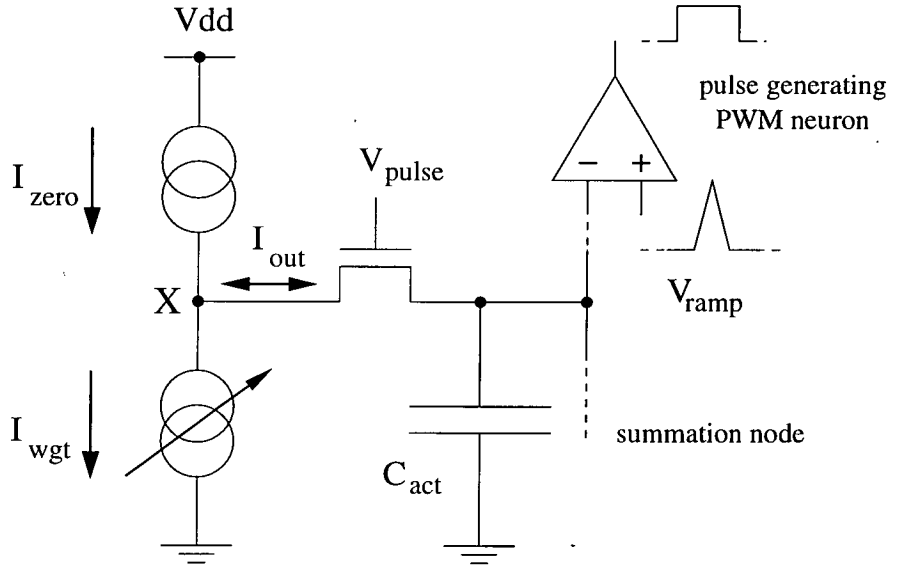


Figure 4.1 - A Pulsed Two-Quadrant Multiplier

In this design, a fixed current source and a variable current sink are connected together and joined, via a simple pass transistor, to capacitor C_{act}. By applying Kirchoff's Current Law at node X, the value and direction of the output current, I_{out}, is determined by the difference between I_{wgt} and I_{zero}. This net current is subsequently used to selectively charge or discharge C_{act} under the control of the voltage on the gate of the pass transistor. If I_{out} is chosen to represent the synaptic weight, λ_{ji}, and the relative on-time of the pass transistor, Δt_{on}, is used to represent the neural state, S_i, then, by the principle of conservation of charge, the change in voltage on the capacitor is proportional to the product of the synaptic weight with the neural state, equation 4.1.

$$\Delta V_{out} \cdot C_{act} = I_{out} \cdot \Delta t_{on} = \lambda_{ji} \cdot S_i \tag{4.1}$$

This arrangement represents a simple and parsimonious implementation of a two-quadrant multiplier. To turn this schematic into a circuit, suitable implementations for the current source and sink must be used.

4.3. MOS Transistors as Real Current Sources and Sinks

Ideal current sources and sinks conduct the same constant current irrespective of the voltage across their terminals [43]. They therefore have infinite output resistance. Ideal sources and sinks cannot be implemented, however, but accurate approximations can be implemented using appropriately biased MOS transistors [128].

As discussed in Section 3.1.2, the current through a MOS transistor operated in **strong inversion** is a function of $V_{gs} - V_T$ and V_{ds}. Clearly, as shown in Figure 3.2, a MOSFET provides a good first order approximation to a current source when saturated. Further, if the drain voltage can be fixed, then a transistor operated in its linear region can also sink a constant current, although it is less common to implement a current sink in this way. In both cases the current conducted through the device is dependent on the voltage stored on its gate.

Thus, to produce a pulsed two-quadrant multiplier, a PMOS and NMOS transistor can implement the constant current source and variable current sink respectively, Figure 4.2. This design is similar to those proposed by Tombs [122], Churcher [11] and Baxter [10].

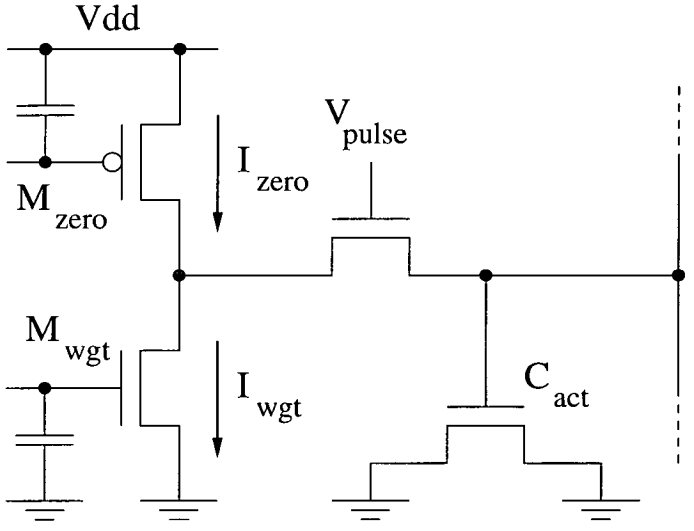


Figure 4.2 - A traditional CMOS Pulsed Two Quadrant Multiplier

Traditionally, pulsed multipliers of this type have operated in voltage mode, using transistors biased into their linear region. Global gate voltages, refreshed from off-chip voltage RAM, have been used to define local currents, with the appropriate voltages for each synapse stored locally on capacitors attached to the transistor gates. However, as discussed in Section 3.3, a problem exists with this approach. The global gate voltages will remain the same for all the synapses, but the threshold voltage for individual synapses will vary across-chip. Thus, since the current through a transistor varies as a function of $V_{gs} - V_T$, synapses in opposite corners of the chip will, in all likelihood, have different characteristics.

Since it is the currents I_{wgt} and I_{zero} which determine the accuracy and linearity of the multiplier, far higher performance should result if global currents are used to program the synapses. This approach was adopted for the DYMPLE synapse, with the current loading implemented using dynamic current mirrors (DCMs).

4.4. Dynamic Current Mirrors

One of the most popular and widely used building blocks in analog IC design is the current mirror, Figure 4.3. Current mirrors are current mode circuits that allow currents to be replicated, multiplied, divided and distributed across-chip. Their operation exploits the principle that if two identical, matched transistors are biased with the same gate-source voltage, then the currents through them will be identical. Therefore, ideally, $I_{out} = I_{in}$ in Figure 4.3.

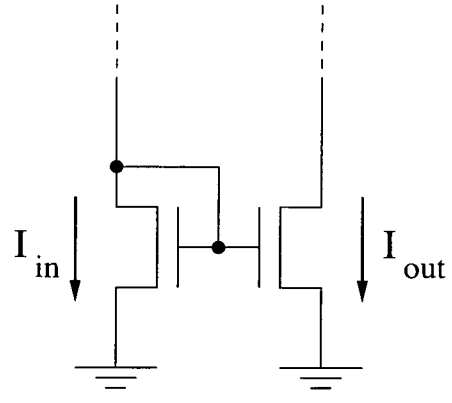


Figure 4.3 - A Two Transistor Conventional Current Mirror

However, fabrication process variations mean that it is almost impossible to produce two transistors with *exactly* the same operating characteristics. Thus, even two transistors that are physically close on the silicon substrate, and are biased with the same

gate-source voltage, cannot be guaranteed to have exactly the same operating characteristics. These variations, in turn, mean that, for the current mirror in Figure 4.3, $I_{out} = kI_{in}$, where k is close to, but not exactly equal to, unity. Although the precision of analogue devices such as current mirrors can always be improved by increasing the size of the transistor gate area or operating the transistors with higher gate voltages, the circuits performance can never be better than the effective mismatch between the two devices.

Improved precision in matching I_{in} and I_{out} can be obtained by using dynamic current mirrors [130, 131], Figure 4.4. Dynamic current mirrors are two-phase sampled data circuits that remove transistor mismatch by using a single transistor instead of the "identical" pair. In the first phase, the DCM is loaded with a current, which, in the second phase, is distributed, by switches, to another node. Thus, assuming the biasing conditions remain the same between the loading and copying phases, the loaded current will be precisely copied.

4.4.1. DCM Principle of Operation

The two-phase operation of the DCM is straight-forward [130, 131] and is illustrated in Figure 4.4. When the toggle switch TS1 is closed in position A, transistor $M_{dynamic}$ tries to conduct current I_{in}. When switch S1 is closed, the voltage required by $M_{dynamic}$ to conduct I_{in} is established on the gate capacitor, C_{gate}. Once equilibrium has been reached, S1 can be opened and C_{gate} isolated. The dynamic mirror has thus been programmed to sink I_{in} due to the voltage on C_{gate}. Further, if toggle switch TS1 is moved to position B, I_{in} is sourced from the power rail, through the load resistor.

The critical point with DCMs is that, although a voltage is still stored on a capacitor and used to generate a current, knowledge of the precise voltage stored is not important. The voltage stored is that voltage which causes the loaded current to flow in that specific transistor. Gate voltages are generated and stored locally by global currents, so assuming that the currents are well controlled, the stored gate voltages implicitly account for any across-chip transistor variations, leading to more accurate current reproduction.

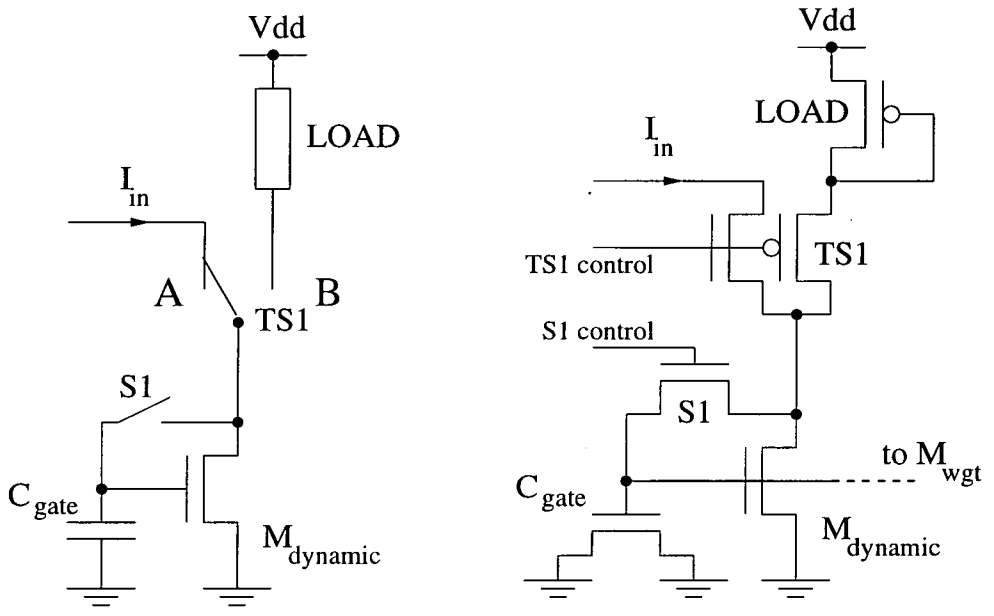


Figure 4.4 - (a) Schematic for a DCM and (b) its CMOS implementation

4.4.2. Factors Affecting Ideal DCM Operation

Although DCMs employ a single transistor to remove the mismatch problems that occur in conventional current mirrors, they are affected by channel length modulation and, because they are sampled data systems, DCMs are also affected by charge injection.

4.4.2.1. Channel Length Modulation

Channel length modulation [43] is a second order effect that causes a change in I_{ds} with increasing V_{ds} when a transistor is in saturation. Channel length modulation occurs because the effective channel length of a saturated MOS transistor varies with V_{ds}. Basically, as the gate to drain voltage of the MOS transistor increases, the pinch-off point of the substrate moves closer to the source and the effective length of the channel is reduced. Since the current through a MOS device is inversely proportional to the channel length, the reduction in length causes a corresponding increase in the current through the device. This is modelled by the parameter λ in the second order equations for a saturated MOS transistor and results in a slight positive slope for the flat saturated region characteristics in Figure 3.2.

Channel length modulation occurs in both conventional and dynamic current mirrors. It arises when different drain-source voltages bias the two legs of the circuit in the

conventional case and when different drain-source voltages bias the single transistor during the two phases of DCM operation.

It can be reduced by increasing the output impedance of the circuits by either using longer transistors, or by using a cascode biasing technique [130, 132-135]. The former solution results in bigger transistors, whilst cascode methods increase circuit complexity. Both solutions increase the area of the circuit.

4.4.2.2. Charge Injection

Charge injection in DCMs occurs due to two phenomena, both related to the use of MOS transistor switches to sample the current.

Charge Injection from the Channel

In order to conduct current in its **on** state, a MOS transistor needs to attract sufficient carriers into its channel: an **on** MOSFET thus acts like a crude electrostatic magnet. To turn switch S1 **on**, its gate voltage must be large enough to attract many carriers into its channel, Figure 4.5. Similarly, when it switches **off**, these surplus charges are released and flow away from the gate.

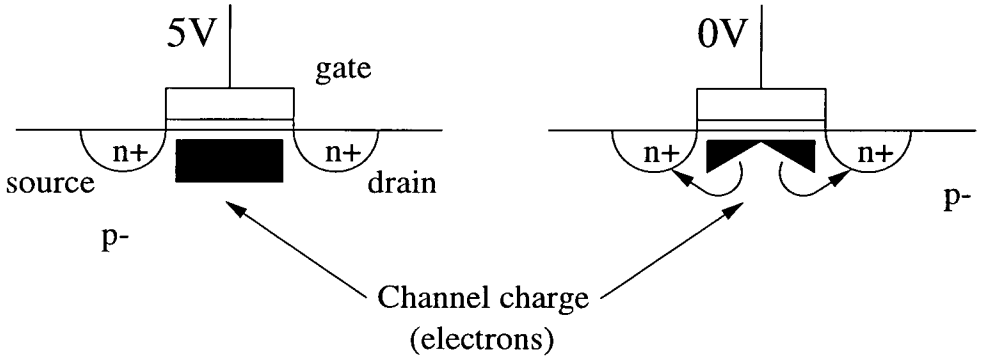


Figure 4.5 - The MOS Transistor as an electrostatic magnet

Some of this released charge flows to ground and some flows onto the C_{gate} capacitance [136-139]. The charge which flows onto the gate affects the voltage stored, as described by equation 4.2.

$$\Delta V_{switch} = \frac{\Delta Q_{switch}}{C_{gate}} \tag{4.2}$$

Charge Injection due to Clock Feedthrough

MOSFET switches require one voltage to turn them **on** and another to turn them **off**. Typically, an NMOS switch is turned on by applying 5V to its gate and is turned off by applying 0V to the gate. However, due to the parasitic capacitances which exist between the gate and drain/source of a MOS transistor (caused by the gate overlap of the source/drain regions) a portion of the gate voltage is fed onto the source and drain nodes, Figure 4.6.

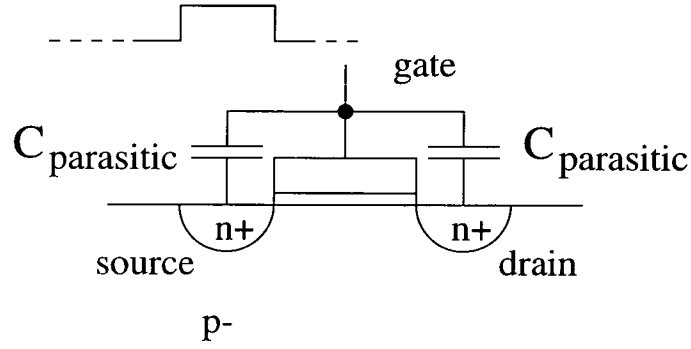


Figure 4.6 - Charge Injection from Clock Feedthrough

Overlap capacitors cause a fixed portion of charge to be dumped onto the C_{gate} capacitor of the DCM and the stored voltage is affected as described by equation 4.3.

$$\Delta V_{feedthrough} = \frac{C_{parasitic}}{C_{parasitic} + C_{gate}} \Delta V_{gate} \tag{4.3}$$

Several techniques exist for overcoming the limitations due to both types of charge injection. These include increasing the size of the gate capacitor and making the switch as small as possible or altering the effective switch-on voltage of the device and thereby decreasing the channel charge [129, 131]. Again, however, these techniques either increase the size of the circuit or they increase its complexity.

4.5. The DYMPLE Synapse

The DYMPLES circuit was designed such that dynamic current mirrors, programmed from global **current** sources, replaced the gate capacitors, programmed by global voltage sources, used in previous designs. The schematic diagram for the complete DYMPLES design is shown in Figure 4.7.

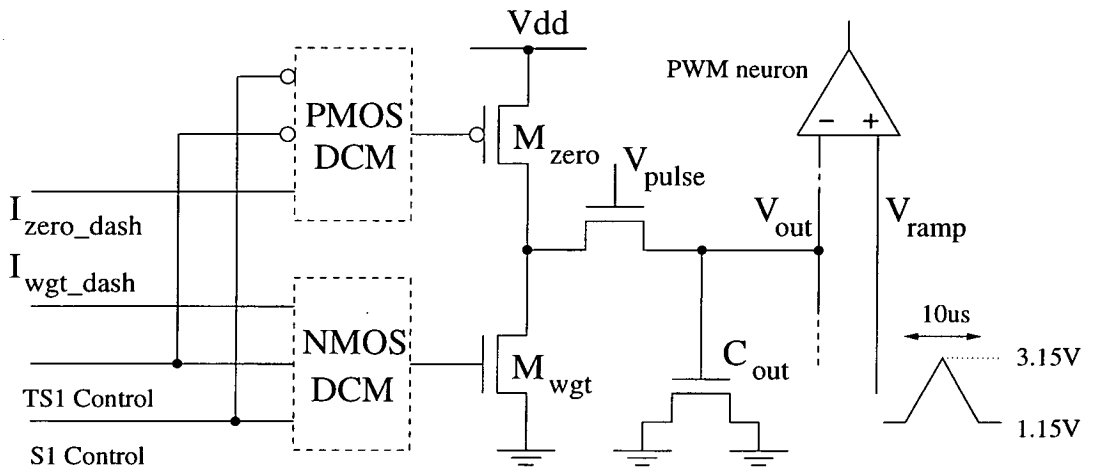


Figure 4.7 - A complete DYMPLE synapse circuit

Because DCMs are used to load and store the global currents, M_{zero} and M_{wgt} operate in saturation. By operating M_{zero} and M_{wgt} in saturation, and ensuring their output impedance accounts for channel length modulation, I_{zero} and I_{wgt} are independent of V_{out}, and no additional compensation circuitry is required to fix the common drain voltage of M_{zero} and M_{wgt}.

The synapse operates as follows. Currents I_{wgt_dash} and I_{zero_dash} are loaded and stored simultaneously in the NMOS and PMOS DCMs via *S1 Control* and *TS1 Control.* These currents generate I_{wgt} and I_{zero} in M_{wgt} and M_{zero}, and it is these latter currents that selectively charge or discharge C_{out} under the control of V_{pulse}.

The circuit was designed for I_{zero} = 0.5μA, I_{wgt} ranging between 0.0μA and 1.0μA and with C_{out} = 5pF. Bigger currents are produced and distributed by the global source to allow faster loading of the DCMs and to remove any imprecision effects that may result from distributing small currents. Thus I_{zero_dash} was designed to be 2.5μA, with I_{wgt_dash} ranging between 0.0μA and 5.0μA.

By defining a 0μs pulse to represent a neural state of 0.0, a 5μs pulse to represent a neural state of 0.5 and a 10μs pulse to represent a neural state of 1.0 etc, this synapse can, at most, produce a ±1V change in V_{out}. Prior to each calculation, C_{out} is precharged to 2.15V and, after the input pulse has been applied, a triangular V_{ramp} waveform, ranging from 1.15V to 3.15V and back to 1.15V in 10μs, is used to generate the output pulse in the output PWM neuron (also shown in Figure 4.7).

It is worth reiterating that, by accurately controlling the I_{wgt_dash} and I_{zero_dash} currents, it is possible to exploit the ability of DCMs to derive and store gate voltages

locally, allowing globally distributed currents to be accurately stored in specific DCMs, therefore helping to account for across-chip variations in individual synapses.

However, it is important to realise that this circuit does not fully exploit the potential offered by DCMs. Here each DCM is used to load and store a gate voltage that generates a copy of the loaded current in another, similar, transistor and it is this copy that alters the charge on the output capacitor. The true potential offered by DCMs can only be realised by using the actual DCM transistors to discharge the output capacitor. The implementation in Figure 4.7 is actually based on a conventional current mirror; using the DCM principle simply allows more accurate programming of individual synapses. Therefore, not only will the DYMPLES circuit be affected by the limitations associated with conventional current mirrors, it will also have to account for the additional discrepancies, such as charge injection and channel length modulation, introduced by the DCMs.

Despite these known limitations, the synapse was implemented as described for two main reasons:

i) the currents I_{zero_dash} and I_{wgt_dash} were required to be scaled down before being used to charge or discharge the output capacitance in the desired time period

ii) by employing the DCMs for weight storage only, a circuit was produced that was transparent to the refresh system and whose data through-put was therefore independent of the refresh rate of the circuit.

Both reasons were deemed to be critical to the effective operation of the synapse and took precedence over other design considerations.

The current division in each DCM was implemented using five matched transistors, connected in parallel, to load and store the current. A sixth transistor, M_{zero} or M_{wgt}, identical to the other five, was then used to replicate it. Thus the currents I_{wgt_dash} and I_{zero_dash} are approximately five times bigger then I_{wgt} and I_{zero} respectively [140].

To maximise the circuit performance, the transistors implementing the DCMs and current sources/sinks were laid out to be physically close on silicon to minimise any mismatch between them. They are also designed to be long and thin as this increases their output impedance and so reduces the effects of channel length modulation as V_{out} varies. Furthermore, to minimise the effects of charge injection, the C_{gate} capacitances are large and the switch transistors in each DCM have the minimum dimensions allowed by the process.

4.6. DYMPLES Simulations

DYMPLES was designed for ES2's 1.5μm double metal, single poly, digital CMOS process. *Typical*, *fast* and *slow* Level 6 model cards for this process were used to model the circuit's operation.

4.6.1. Variation of V_{gs} with Current in the DCMs

The first experiment performed with the simulated synapse illustrates that different gate voltages are required to conduct the same current through transistors with different threshold voltages. Sixteen values of current (equally distributed between 0μA and 5μA) were loaded into the NMOS DCM and the corresponding voltage stored on C_{gate} was recorded. This process was repeated for all three model cards and the results are shown in Figure 4.8.

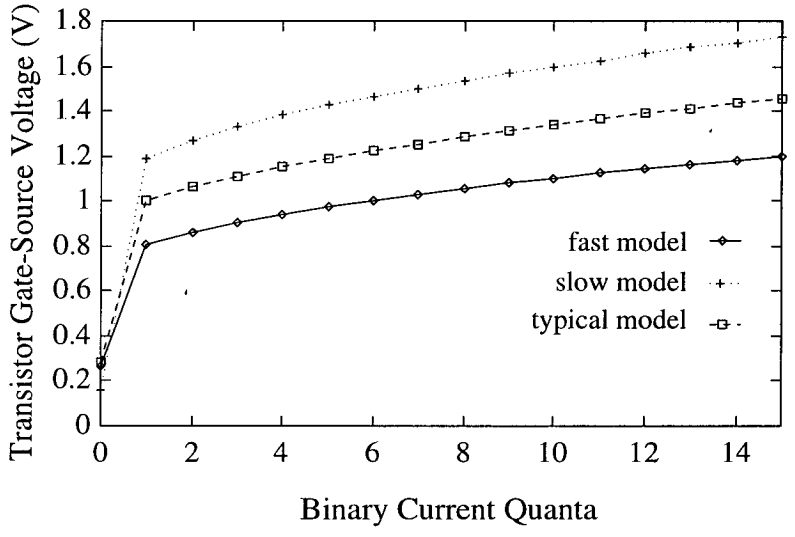


Figure 4.8 - Variation of V_{gs} with Synaptic Weight for transistors with different threshold voltages

This result confirms that, if global voltages are used to derive local currents, variations in the threshold voltage of the transistors will cause variations in the generated currents.

4.6.2. Simulated Multiplier Performance

As already discussed, the DYMPLES circuit was designed to use currents to program the synapses in the belief that simpler multipliers, with a higher degree of process tolerance, would result. Thus the next experiment investigated the similarity of the multiplier performance for synapses constructed from *fast, slow* and *typical* transistors. Again the circuit was modelled with the three model cards and the multiplier performance of each synapse was recorded. For the purpose of these experiments, the synapses were used to alter the charge on a 5pF capacitor implemented using a *typical* NMOS transistor. The same capacitor was used for all the experiments since it was essential to compare the operation of the synapses rather than the combination of the synapse and transistor capacitor.

For the experiments, each NMOS DCM was loaded with currents of 0μA, 1.667μA, 3.333μA and 5μA and the PMOS DCM was loaded with 2.5μA. The net current was then allowed to charge or discharge the output capacitor for several time periods (as determined by the on-time of the pass transistor connecting the DCMs to the capacitance) and the final output voltage on C_{out} was noted at the end of each trial. The results from these simulations are shown in Figure 4.9, which shows the final Output Voltage (on C_{out}) vs. Input Pulse Width for the three synapse types and the four different values of I_{wgt_dash}. The characteristics for all 3 synapses overlap.

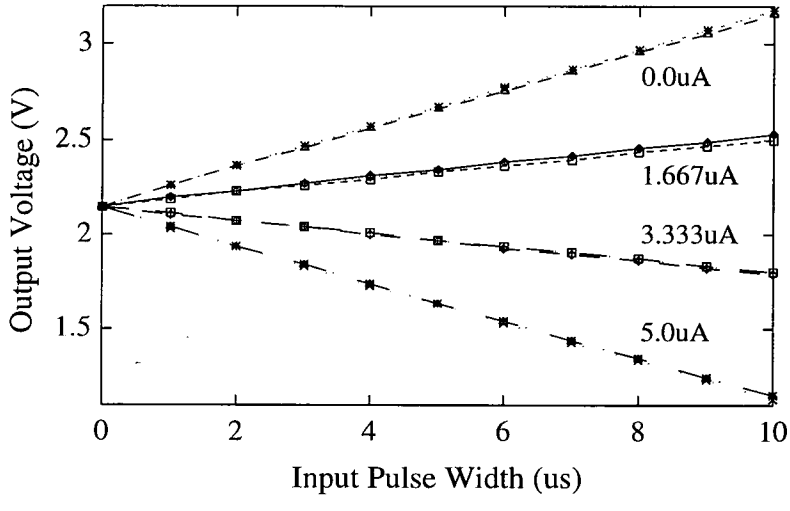


Figure 4.9 - Simulated Multiplier Performance

The excellent results obtained from this experiment confirm that the DYMPLES implementation should work as required, despite the threshold voltage variations, and indicates that the current mode approach appears to account for the type of transistor

variations that can occur across a silicon wafer. Encouraged by these findings, a hardware implementation of an array of DYMPLES devices was designed, fabricated and tested.

4.7. The DYMPLES Chip

The DYMPLES chip was manufactured to:

i) prove the viability of hardware implementations of the synapse design

ii) highlight the potential of current mode circuit implementations for ANNs, including the benefits to be gained from greater on-chip integration.

4.7.1. DYMPLES Chip Floorplan

The floorplan of the DYMPLES chip is shown in Figure 4.10 and the layout of the chip is shown in Figure 4.11. The DYMPLES chip was implemented in 4mm by 4mm of silicon, is pulse-width input : pulse-width output and consists of:

- an 8 by 8 array of DYMPLES multipliers (arranged as 8 synaptic columns with 8 synapses per column)
- a 4-bit current DAC for generating I_{wgt_dash}
- a *zero current mirror* (ZCM) for generating I_{zero_dash}
- 8 PWM output neurons (one per column)
- row and column decoders to allow each synapse to be addressed and loaded.

The synapse was laid out by Jean E Louvet, an undergraduate student from Napier University, as part of his "industrial" experience.

4.7.2. The On-Chip ZCM and Current DAC

Since the correct operation of the DCMs rely on accurate control of the global currents, it was necessary to design an on-chip current DAC for generating the I_{wgt_dash} currents and a suitably biased conventional current mirror for generating I_{zero_dash}. By biasing these circuits with small off-chip currents, it was possible to allow the DAC and ZCM output currents to be set up independently.

The ZCM was easily designed: it is simply a 2:5 multiplying conventional current mirror that outputs a current 2.5 times larger than its biasing current, Figure 4.12. Current multiplication was implemented by using two transistors on the input and five on the output. These transistors were interleaved on the chip to account for any

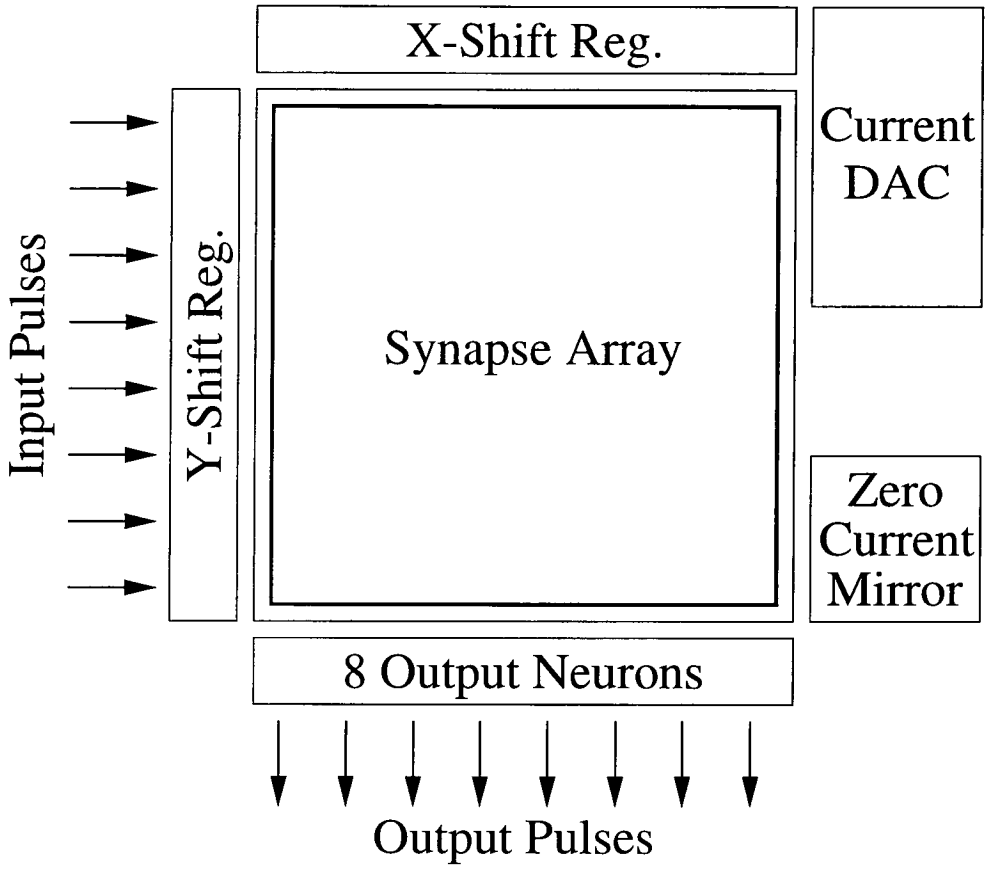


Figure 4.10 - A block diagram of the DYMPLES Chip

on-chip gradients.

The 4-bit current DAC on the DYMPLES chip was designed to switch binary weighted values of the biasing current onto the common output line [141, 142]. Thus it was necessary to balance the desire for the weightings to scale as linearly as possible with the requirement that the DAC be as simple as possible to design and lay out. Thus a compromise was reached whereby multiple banks of transistors were used to generate the bias current weightings, Figure 4.13, and the transistors were laid out such that the DAC had a common axis of rotation [142]. Again this was to implicitly account for across-chip gradients to ensure that the DAC would function as intended.

For correct operation, it is important that the transistors in the DAC and ZCM are resistant to noise and output constant currents regardless of their drain-source voltages. Thus the transistors in these devices were designed to be long and thin to ensure the output currents were unaffected by changes in V_{ds} and ensure the currents in the transistors were resistant to noise coupling onto the gates[3].

[3] Decreasing the W:L ratio of a transistor increases the range of gate voltages needed to conduct a specified range of currents in it and thereby reduces the effect capacitively-coupled noise has on the current in the transistor.

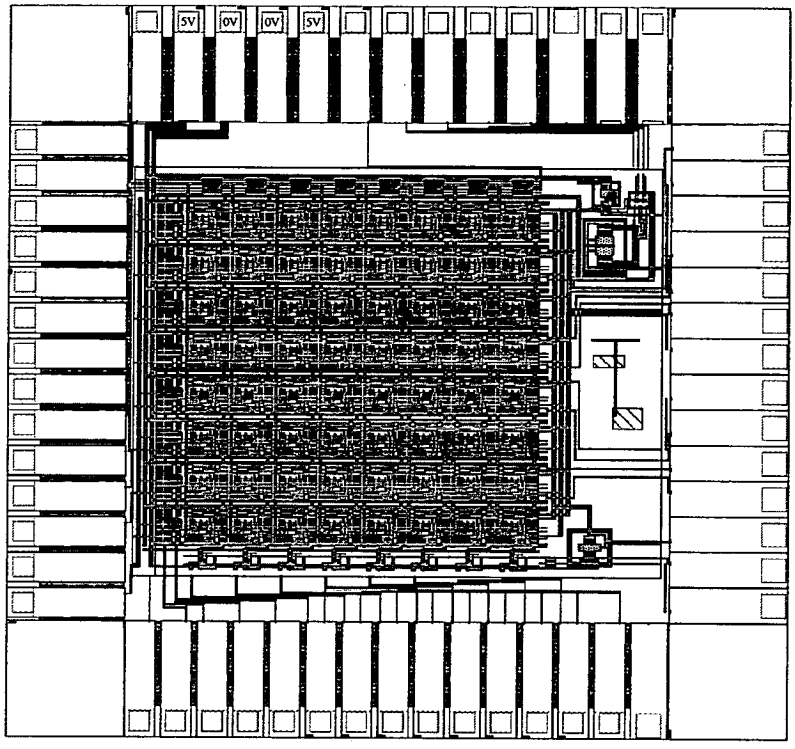


Figure 4.11 - The DYMPLES chip

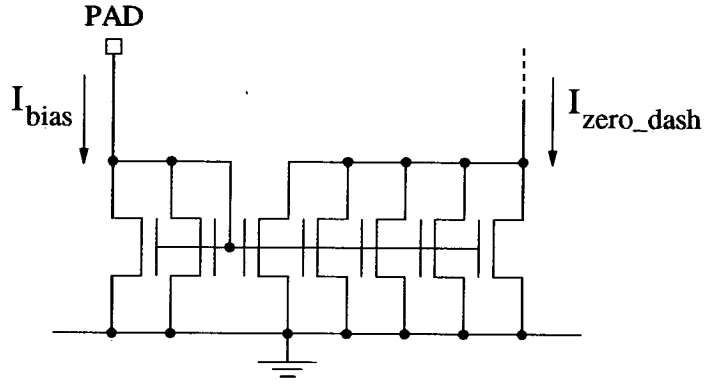


Figure 4.12 - Schematic Diagram of the Zero Current Mirror

4.7.3. Design for Testability

One of the aims of the chip was to ensure it was as simple as possible to set up, operate and test. Thus all the necessary biasing signals were designed to be easily derived and several additional features were included on the chip to allow straight-forward measurements to be taken.

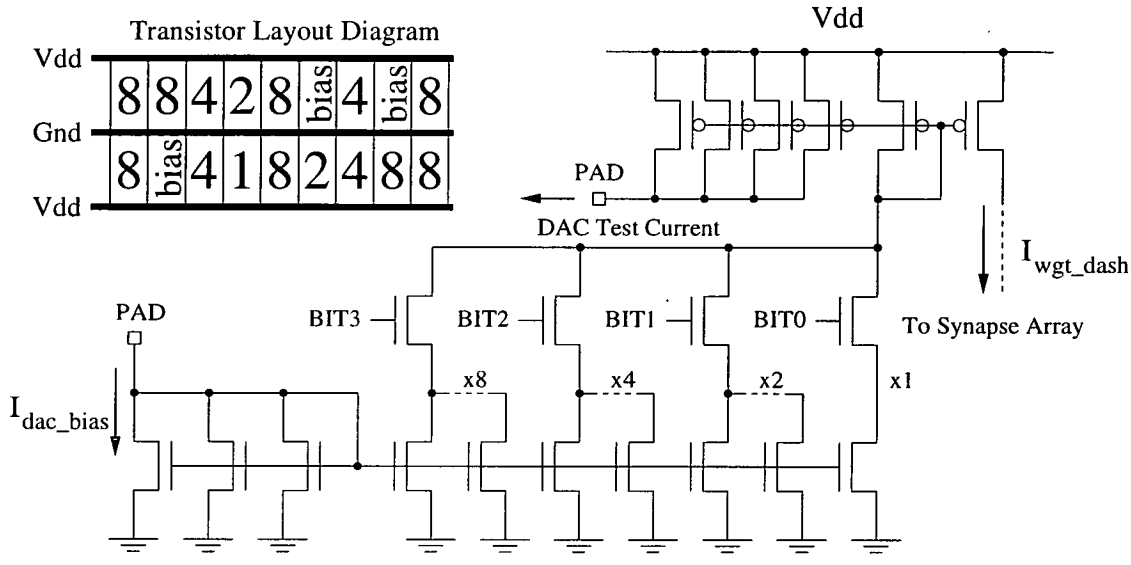


Figure 4.13 - Schematic Diagram of the Current DAC

In case the chip did not function as expected (even to the point of complete failure), sufficient test points were provided to allow potential discrepancies to be traced and identified. Therefore the chip was designed so that individual units (row and column decoders, the DAC and some DCMs) could be tested in addition to the neurons.

In order to allow on-chip currents to be measured off-chip, transistors were included within the core of the design, the drains of which were connected to pads. Scaled versions of the on-chip currents are generated in these transistors and can be used to produce measurable voltages off-chip, Appendix A. Therefore, on-chip **current** variations can be characterised by corresponding variations in off-chip **voltages**.

4.8. Chip Results

A total of eight chips were returned by the fabrication house, ensuring that enough devices were available to allow the functionality of the designs to be assessed.

The DYMPLES Development Board (Appendix A) was designed and built to allow the chips to be biased correctly and allow measurements to be taken easily and automatically.

The performance of the DYMPLES chips can be characterised by several factors viz. the performance of the current DAC, the performance of the NMOS and PMOS dynamic current mirrors and the multiplication characteristic of the chips. A description of how the experimental results were recorded is also given in Appendix A.

Since the pulse widths used in these experiments are generated or recorded in RAM chips on the development board, the following graphs of measured results represent the Input and Output pulse widths in terms of *RAM Locations* rather than a time period. This was done purely for simplicity and the corresponding pulse times can be obtained by dividing the pulse widths by the frequency of the development board clock (24MHz).

4.8.1. DAC Characteristic

The performance of each current DAC was measured using the off-chip Op-Amps. Each DAC was loaded with all sixteen 4-bit binary combinations and the Op-Amp output voltage recorded. The currents necessary to generate the voltages were then calculated to allow the DAC characteristic to be visualised. All the DACs produced similar characteristics, and the average of the results for all the DACs, along with errorbars representing ±3 standard deviations of the results, are shown in Figure 4.14. Clearly there is a linear relationship between applied weight and measured on-chip current and from these results, it was concluded that the DACs functioned as required.

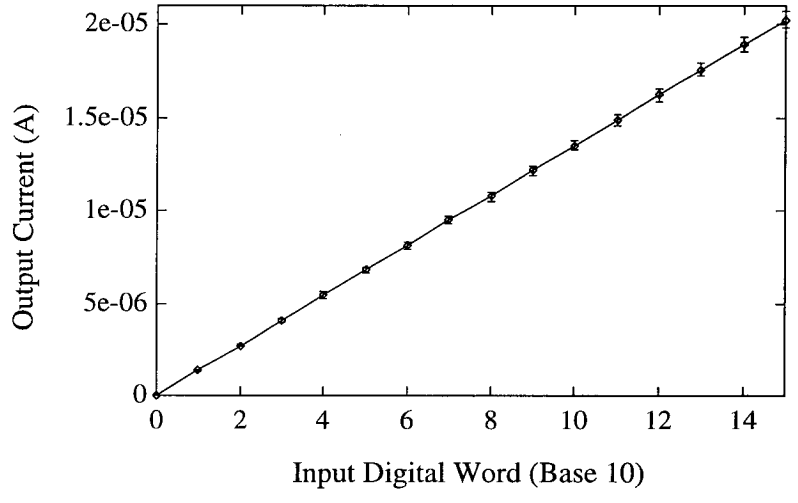


Figure 4.14 - The DAC Characteristic of the DYMPLES chips

4.8.2. Dynamic Current Mirror Characteristics

Having shown that the DACs worked, it was concluded that accurate on-chip current control should be possible and that the performance of the NMOS and PMOS DCMs could be reliably assessed. A total of eight current mirrors were available for

characterisation per chip, four NMOS versions and four PMOS versions.

For assessing the DCMs, all possible 4-bit words were applied to the DAC in turn and every synapse on the chip was then loaded with I_{wgt_dash} and I_{zero_dash}. The currents stored in each testable synapse were then measured using the same type of Op-Amp arrangement as for the DAC characteristic. However, in order to minimise the differences between the measurements for each output, an 8-way PC-controlled analogue multiplexor was used to steer the currents from the individual outputs to the Op-Amp arrangement. The results from most of the chips were in good agreement, although two chips had NMOS DCMs which exhibited large across-chip variability. A typical set of NMOS and PMOS DCM characteristics are shown in Figure 4.15.

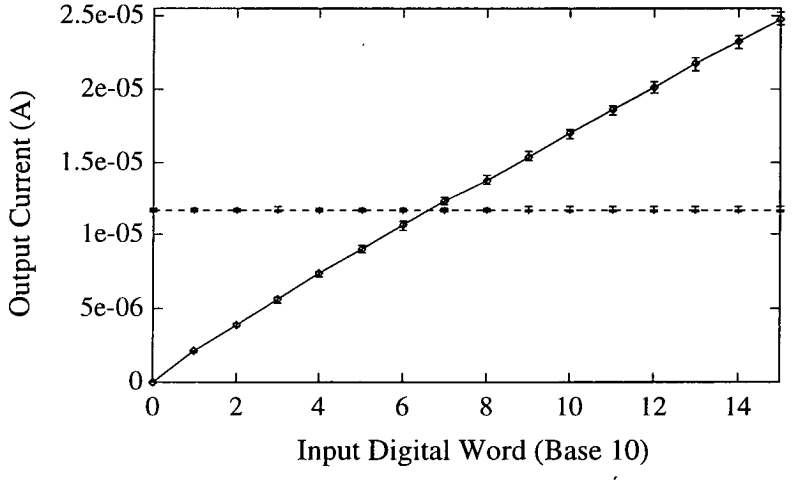


Figure 4.15 - The DCM Characteristic of a DYMPLES Chip

This graph shows the average current measured in the DCMs on a single chip against the weight applied to the DAC of that chip; the error bars represent ±3 standard deviations of the measured results.

The results obtained indicate that the DCMs on most of the chips function correctly. The currents in the PMOS devices are constant with respect to the applied weight, whereas the current in the NMOS devices vary linearly with the applied weight. The small spread in the results from all but two of the chips also suggests that the DCMs possess the inherent ability to account for across-chip variations, although no firm conclusions can be drawn from experiments on a few chips from a single wafer.

4.8.3. Multiplication Performance

With both the DACs and the DCMs functioning, the multiplication performance of the chips was measured. Two sets of measurements were used to determine the capability of the circuits to implement the desired two-quadrant multiplication operation:

i) **output pulse width vs. input pulse width** - to indicate the linearity of the multiplier with the applied neural state

ii) **output pulse vs. loaded weight** - to highlight the linearity of the multiplier with the loaded synaptic weight.

By loading the entire chip with all the neural weights and applying every possible neural state it was possible to measure the output pulse widths for every combination of weight and state for all eight neurons per chip. From these results it was possible to assess the performance of the multipliers with respect to both input variables.

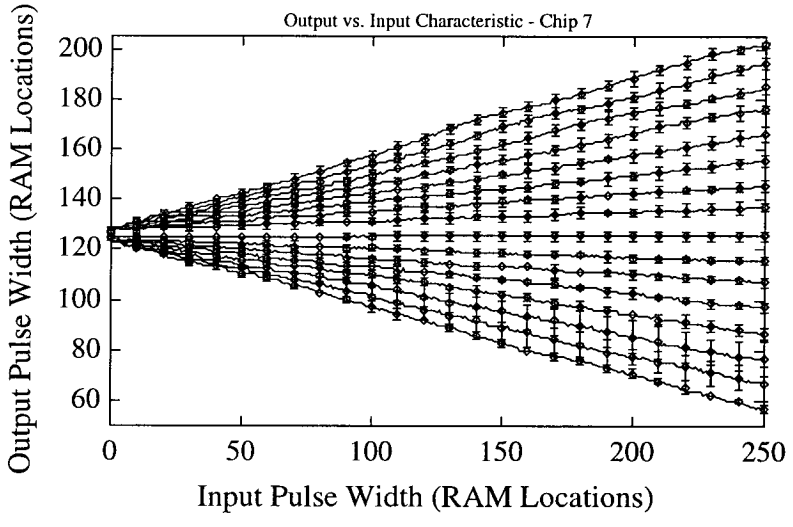


Figure 4.16 - The Output Pulse vs. Input Pulse Characteristic of a DYMPLES Chip

Figure 4.16, shows the Output Pulse Width against Input Pulse Width characteristic averaged over all 8 output neurons on a single chip. These results clearly indicate the chip is functioning as intended and show a linear relationship exists between the input pulse width and the output pulse width for all 16 neural weights. However, the linearity of the multiplier can be affected by the off-chip ramp[4].

[4] The ramp is generated by firing values from off-chip RAM through an off-chip DAC (Appendix A) and so any slight perturbations on the generated ramp (due to noise etc.) permeate through to the multiplication characteristic graph, as shown by the slight "wobbles" in Figure 4.16.

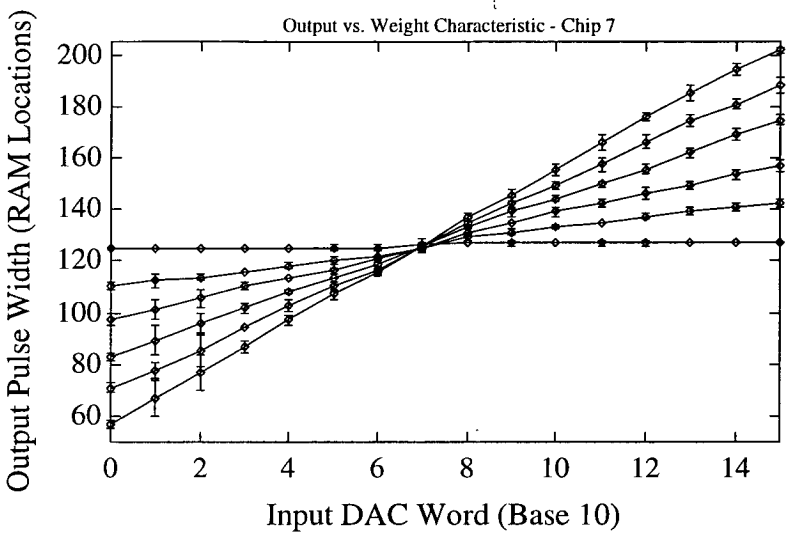


Figure 4.17 - The Output Pulse vs. Synaptic Weight Characteristic of a DYMPLES Chip

Further, Figure 4.17, shows the variation of Output Pulse Width vs. Synaptic Weight, averaged over all 8 output neurons on the same chip, for input pulse widths of 0, 50, 100, 150, 200 and 250 RAM locations. Again it is clear that the chip is functioning correctly and that the multiplier is linear with respect to the neural weight.

4.8.4. Discrepancies with the DYMPLES Multipliers

Unfortunately, not all the DYMPLES multipliers produced the excellent results as in Figures 4.16 and 4.17. Half of the chips appeared to have a weight dependent off-set error as shown in Figure 4.18.

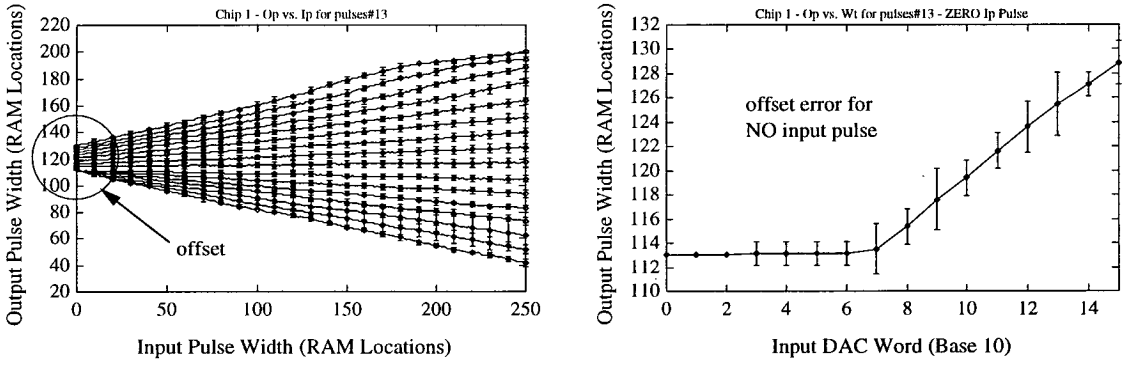


Figure 4.18 - Effect of the weight dependent offset error on the averaged multiplication characteristic of a DYMPLES multiplier

From these results, it is apparent that as the neural weight increases, so does the initial offset. This can be more clearly seen in Figure 4.18(b). The reason for the offset is not known, although it may be caused by the pass transistor used to charge and discharge the output capacitor. Apart from this offset, though, the characteristic is still linear and it is not known if the phenomenon would affect the operation of the multiplier in a neural application.

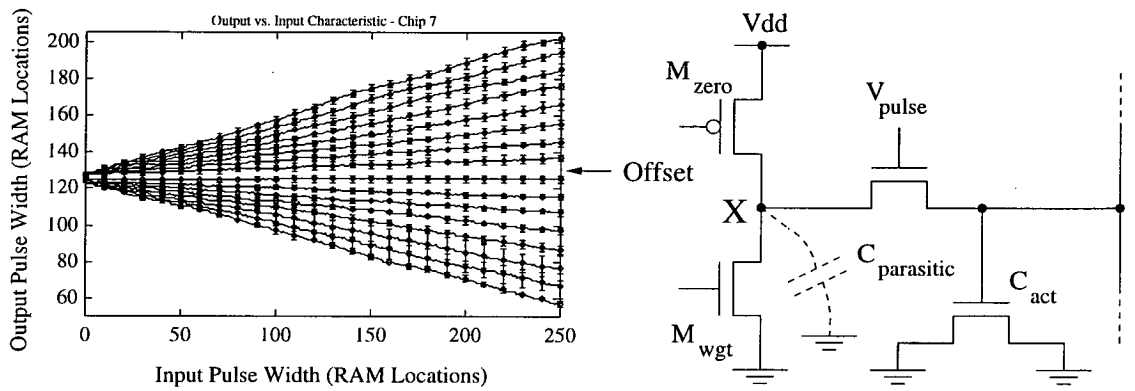


Figure 4.19 - The parasitic induced offset

Another minor effect seen in the measured results was the small offset produced between the positive and negative weights, Figure 4.19(a). This effect can be easily explained. When the synaptic weight, I_{wgt}, is greater than I_{zero}, the parasitic capacitance of node X in Figure 4.19(b) will be discharged. When the switch is opened, some charge sharing will occur, resulting in a slight decrease in the voltage on the output capacitor, which translates into a slight increase in the observed output pulse width. Similarly, if I_{zero} is greater than I_{wgt}, the capacitor gains a slight increase in its voltage that translates, in turn, into a small reduction in the output pulse width. As can be seen from the results presented, the offset has an almost negligible effect on the multiplication characteristic and it was not thought to be critical to the operation of the circuit. In any case, the effect can be minimised by reducing the size of the parasitic drain capacitance, $C_{parasitic}$ of the synapse output.

4.9. Design Improvements

The original DYMPLES design required a large silicon area for a number of reasons:

i) conservative layout techniques were used since this was a concept-proving chip and not a prototype for a commercial product

ii) large capacitors were used for the C_{gate} capacitors and C_{out} to minimise the effects of noise, charge injection and capacitive coupling

iii) guard-rings were used whenever possible to help isolate and protect sensitive transistors.

All these techniques increase the likely accuracy of the design at the expense of silicon area and in order to be commercially feasible, the overall area of the synapse will need to be reduced to increase its implementation density. A number of techniques can be employed to achieve this.

i) **Cascode DCM transistors**. By using cascode techniques it will be possible to increase the output impedance of the $M_{dynamic}$ transistors and reduce the voltage feedthrough to the C_{gate} capacitors from the common drain node of transistors M_{zero} and M_{wgt}. This modification will allow a reduction in the size of the C_{gate} capacitances and a reduction in the length of M_{wgt} and M_{zero}. However, as already discussed, cascode techniques increase the complexity of the circuit and the cascode configuration chosen must allow the synapse to be realised in a smaller silicon area.

ii) **Ratioed Transistors**. By using single $M_{dynamic}$ transistors which are N times wider than the corresponding M_{wgt} and M_{zero} transistors, I_{wgt} and I_{zero} will be approximately equal to $\frac{I_{wgt\_dash}}{N}$ and $\frac{I_{zero\_dash}}{N}$ respectively. Using ratioed transistor widths to scale currents in this way is not nearly as accurate as the multi-transistor method used in the DYMPLES multiplier. However, the DYMPLES design relies on the charge/discharge of a fixed value capacitor that can only be fabricated to within 10% of its value [43], with the correct operating point obtained by adjusting the biasing currents for the DAC and ZCM. Thus the new inaccuracy in the mirrors will simply be combined with the existing capacitor inaccuracy and both can be accounted for by adjusting the off-chip biasing currents.

iii) **Smaller C_{out} Capacitors**. By using ratioed transistors as in ii), smaller values of I_{wgt} and I_{zero} can be generated. Therefore, by equation 4.1, this allows the size (and therefore the area) of C_{out} to be reduced. Also, by using smaller Δt_{on} times, the value of C_{out} can be reduced too.

All these techniques should allow the synapse to be implemented in a smaller area at the expense of circuit accuracy (possibly) and circuit complexity.

4.10. Summary

This chapter has concentrated on the design and development of the DYMPLES current-mode pulsed synaptic multiplier. The DYMPLES circuit was designed to be easy to implement, set-up and operate and theoretical considerations indicated that it should provide improved process-tolerance to digital fabrication processes by using currents, rather than voltages, to implement the synaptic programming.

HSPICE simulation results and hardware measurements highlighted that the design has the following capabilities.

- It is a valid implementation of a two-quadrant multiplier since the output is linear for both operands (synaptic weight and input neural state).
- The use of DCMs appears to account for across-chip variations implicitly.
- The use of the current mode approach makes it possible to improve the level of on-chip integration for hardware ANNs.

The hardware measurements also indicated some of the potential disadvantages with the design and these were subsequently discussed, along with suggestions for possible improvements.

It is clear from this chapter, however, that the DYMPLES design fulfils all its initial requirements and is a valid implementation of a two-quadrant multiplier. Thus it can be used to implement the output layer of pulsed RBF neural networks.

Chapter 5

The RHO Chip

The previous chapter detailed a suitable current mode implementation of a pulsed two-quadrant multiplier and it was concluded that an array of these multipliers could be used in the output layer of an RBF network. This chapter focuses on the design and testing of circuits for reproducing the operation of the basis functions in the hidden layer.

As detailed in Chapter 2, each hidden unit in an RBF calculates the Euclidean (or Manhattan) distance between an input vector and a reference, or *centre,* vector and produces a non-linear transformation of this distance as its output. Separate circuits have therefore been developed to calculate an approximation to the squared Euclidean distance between two voltages and implement the non-linear transformation.

Another test chip, the RHO chip[5], was fabricated to demonstrate the functionality of the circuits and this chapter presents simulation and measured hardware results from the developed designs.

5.1. Centre Circuit Aims

In addition to reproducing the correct operation of an RBF hidden unit, the centre circuits were designed to fulfil the following requirements:

i) they should be easily set-up, operated and tested

ii) they should easily interface to the "real world"

iii) they should be cascadable

iv) they should be designed to operate using pulse width modulation and should easily interface to the DYMPLES circuits.

The target process for these circuits was the MIETEC 2.4μm, double metal, double poly, analog CMOS process and all the circuit simulations were carried out using the

[5] RHO is an acronym for RBF Hardware Options.

Level 3 transistor models for this process.

5.2. The Distance Circuit

The schematic diagram for the distance circuit is shown in Figure 5.1. It calculates an approximation to the squared Euclidean distance between two voltages, nominally V_{in} and V_{centre}, which represent single components of the input and centre vectors respectively.

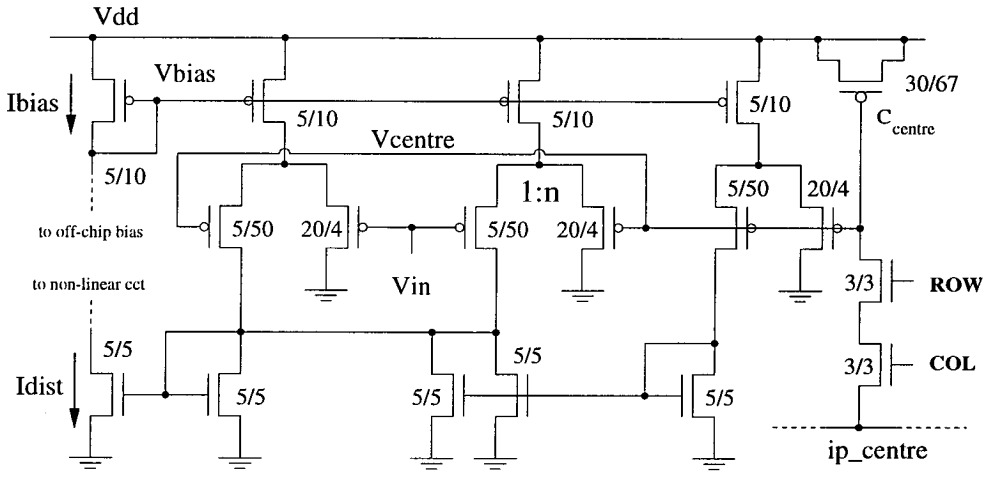


Figure 5.1 - Distance Circuit

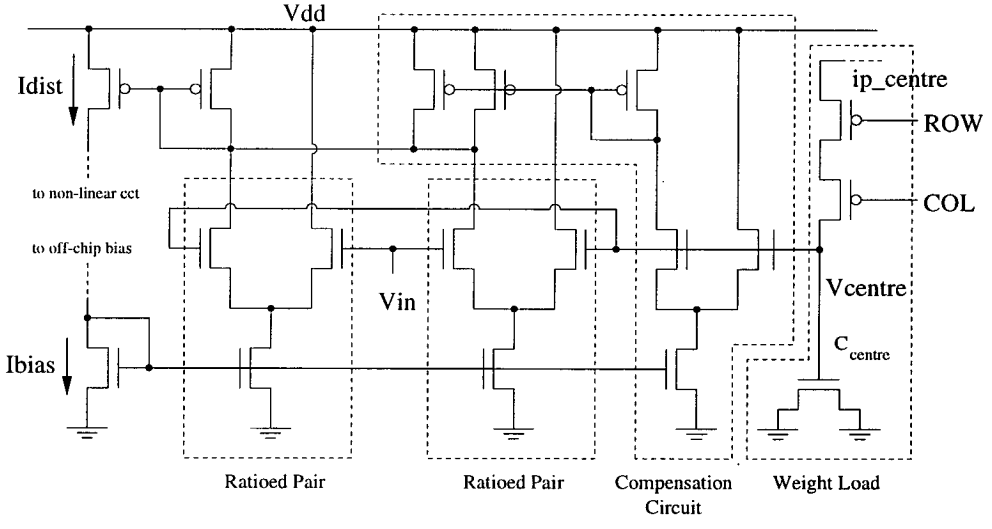


Figure 5.2 - NMOS version of the Distance Circuit in Figure 5.1

The implementation chosen [143, 144] exploits the natural square-law relationship between I_{ds} and V_{gs} in a saturated MOS transistor. This implementation for the

distance circuit had already been proposed by a fellow researcher in this group and published in the open literature [143]. The circuit was originally designed for the ES2 1.5μm process, but had never been laid out nor fabricated. The ES2 1.5μm process was withdrawn just after the DYMPLES chip was fabricated and the RHO chip was designed for the MIETEC 2.4μm process. Thus the original distance circuit had to be re-developed for the new process, and it was possible to extend the input range of the circuit in doing so.

The operation of the distance circuit can be easily explained by considering the NMOS version, Figure 5.2, and decomposing it into its constituent parts.

5.2.1. Ratioed Transistor Pairs

Consider that the long thin transistor M_{narrow} in Figure 5.3, operates in saturation.

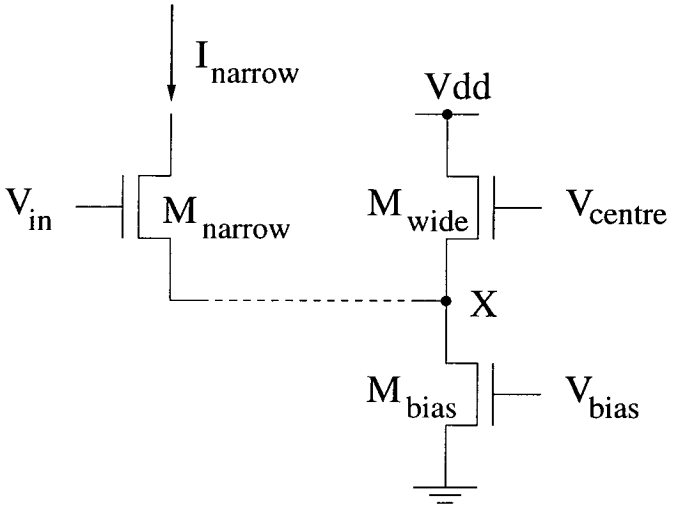


Figure 5.3 - Single Ratioed Pair Schematic Diagram

To a first order, the relationship between its drain-source current and gate-source voltage is given by equation 5.1.

$$I_{ds} = \frac{\beta}{2}\left(V_{gs} - V_T\right)^2 \tag{5.1}$$

Fixing the source voltage of M_{narrow} to a known, constant value, produces a current that varies as (the transistor gate voltage minus a constant) all squared. The source voltage of the transistor can be fixed by using a simple source follower, Figure 5.3.

The source follower consists of two series connected transistors M_{wide} and M_{bias}. M_{wide} has a high conductivity (large W/L ratio) to ensure that its source is clamped to around a threshold voltage below its gate, whilst M_{bias} controls the maximum current flowing through the circuit. By connecting a source follower to M_{narrow} as shown, a circuit is created where the common source, node X, of transistors M_{wide} and M_{narrow} is clamped to approximately $(V_{centre} - V_{T_{wide}})$. By considering the currents at node X, three equations are obtained (assuming all three transistors operate in saturation):

$$I_{bias} = I_{narrow} + I_{wide} \tag{5.2}$$

$$I_{wide} = \frac{\beta_{wide}}{2}\left(V_{centre} - V_x - V_{T_{wide}}\right)^2 \tag{5.3}$$

$$I_{narrow} = \frac{\beta_{narrow}}{2}\left(V_{in} - V_x - V_{T_{narrow}}\right)^2 \tag{5.4}$$

If M_{wide} and M_{narrow} are physically close in silicon, then it can be assumed that $V_{T_{narrow}} = V_{T_{wide}}$. Combining equations 5.3 and 5.4 therefore yields the following equation for the output current from the circuit:

$$I_{narrow} = \frac{\beta_{narrow}}{2}\left(V_{in} - V_{centre} + \left(\frac{2I_{wide}}{\beta_{wide}}\right)^{\frac{1}{2}}\right)^2 \tag{5.5}$$

This result is valid for $V_{in} > V_{centre}$. Using a second identical *ratioed pair,* with V_{in} and V_{centre} applied to the complementary transistors, Figure 5.4, a circuit is obtained whose output current (to a first order) is proportional to a quadratic function of the difference between V_{in} and V_{centre}.

The current I_{dist} output from the circuit in Figure 5.4 is given by equation 5.6.

$$I_{dist} = \frac{\beta_{narrow}}{2}\left(|V_{in} - V_{centre}| + \left(\frac{2I_{wide}}{\beta_{wide}}\right)^{\frac{1}{2}}\right)^2 \tag{5.6}$$

The valid range of equation 5.6 (in terms of the voltage difference $V_{in} - V_{centre}$) is determined by $(W/L)_{narrow}$ and I_{bias}. If $(W/L)_{narrow}$ is large for a given bias current, then M_{narrow} does not require too substantial a gate voltage before it can sink all I_{bias}. When this happens, I_{dist} levels off, the quadratic relationship between $(V_{in}\text{-}V_{centre})$ and I_{dist} is lost and the input dynamic range of the circuit is small. Similarly, M_{narrow} must be strong enough to sink I_{bias} at the maximum value of $|V_{in} - V_{centre}|$ in the desired operating range, otherwise power will be wasted unnecessarily, since

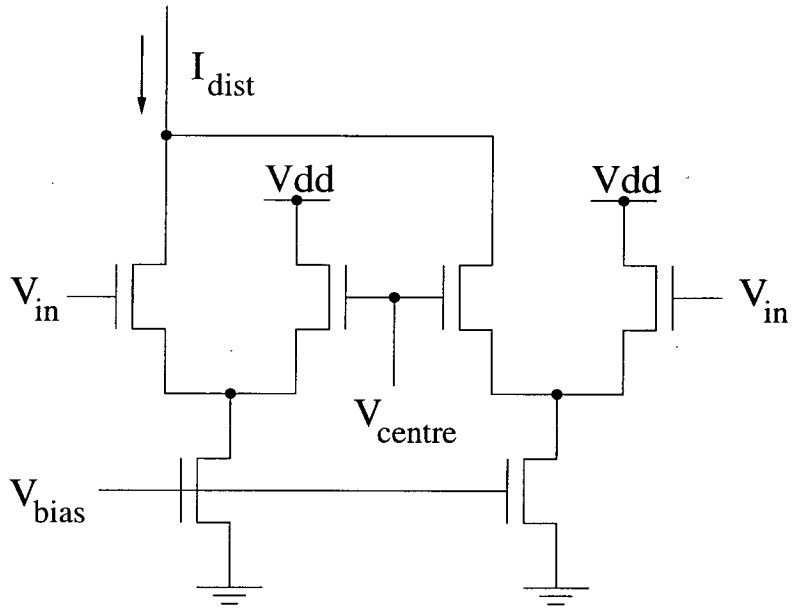


Figure 5.4 - Double Ratioed Pair Circuit

$I_{wide} = I_{bias} - I_{narrow}$. Thus the M_{wide} and M_{narrow} transistors must be carefully sized to produce a circuit which satisfies these criteria.

For the distance design fabricated on the RHO chip, PMOS rather than NMOS transistors have been used to implement the circuit. PMOS devices were used because the intrinsic mobility of *holes* in a silicon substrate is lower than that of *electrons,* especially in an N-well process. Thus, the inherently smaller transconductance of PMOS devices is a distinct advantage in this case, allowing a higher input dynamic range to be obtained, for a given I_{bias}, compared with the NMOS version of the circuit.

5.2.2. Compensation Circuit

Both ratioed pairs in Figure 5.4, output a current when (V_{in} - V_{centre}) is zero. The value of this current is small, approximately constant, and can be removed using the circuitry in Figure 5.5.

This compensation circuit is identical to the ratioed pairs, except the gates of both transistors are tied to V_{centre}. Thus $I_{comp} = (\beta_{narrow} / \beta_{wide})I_{wide}$. By using a 1:2 conventional current mirror, I_{comp} can be doubled and subtracted from I_{dist}, producing an (ideally) offset-free current from the final distance circuit, equation 5.7.

$$I_{dist} = \frac{\beta_{narrow}}{2}(|V_{in} - V_{centre}|)^2 + \beta_{narrow}\left(\frac{2I_{wide}}{\beta_{wide}}\right)^{\frac{1}{2}}(|V_{in} - V_{centre}|) \qquad (5.7)$$

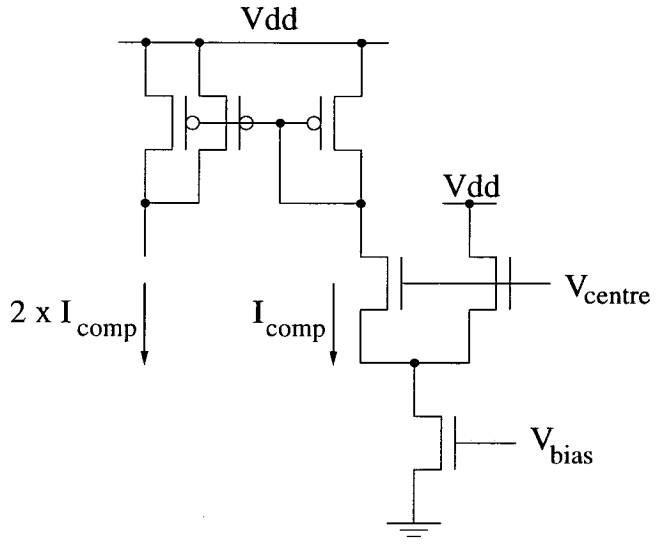


Figure 5.5 - Distance Circuit Compensation Circuit

Figure 5.6, shows the simulated distance circuit current from an HSPICE simulation of the circuit in Figure 5.1, along with a scaled version of the ideal squared Euclidean distance current between V_{in} and V_{centre}. For these simulations, V_{centre} was tied to 0V. The simulated characteristic indicates that the fabricated circuit should produce a good approximation to the desired distance metric.

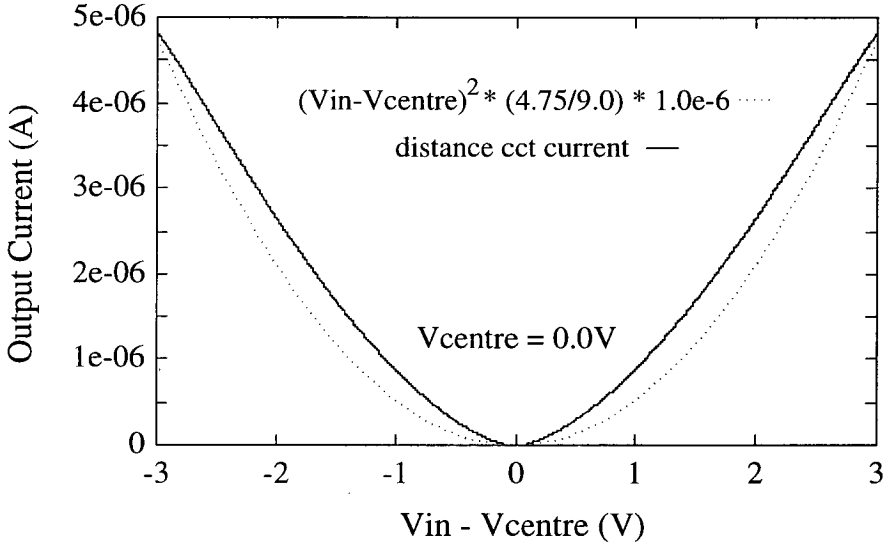


Figure 5.6 - Simulated Distance Circuit Output and Scaled Squared Euclidean Distance Measure

The current generated by the compensation circuit, $2I_{comp}$, was estimated from an HSPICE simulation, Figure 5.7. The generated current is only approximately 100nA over the 0V to 3V range of V_{centre}, 2% of the maximum I_{dist} value.

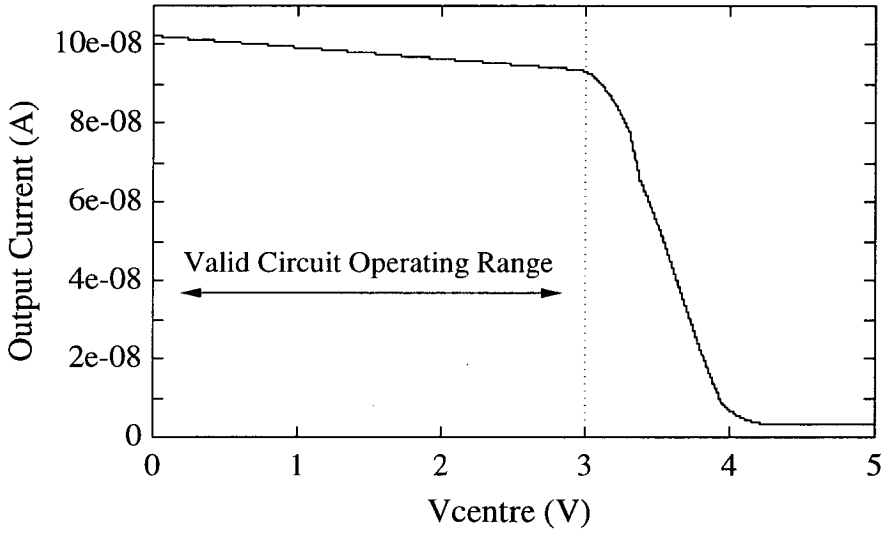


Figure 5.7 - Simulated Output from the PMOS Compensation Circuit

Further, the change in $2I_{comp}$ with V_{centre} appeared to be negligible over the valid range of V_{centre}. Therefore, in what appeared to be a pragmatic engineering solution, which would save silicon area, *local* compensation, where each distance circuit has its own dedicated compensation circuit, was sacrificed in favour of *global* compensation, where a single ratioed pair was used to generate a compensation current. This compensation current was then copied to several cells.

Also, by tying both gates of the compensation circuit ratioed pairs to 0V, a parsimonious solution, which required fewer transistors than *local* compensation, appeared to have been found for removing the constant term in equation 5.6. The effect of this solution on the simulated performance of the circuit was negligible.

5.2.3. The Body Effect

Correct symmetrical operation of the distance circuit requires that each ratioed pair of M_{wide} and M_{narrow} transistors are placed in separate wells so that the voltage between the common source of the transistors and the bulk silicon, V_{bs}, does not affect the operation of the circuit. If V_{bs} is not equal to zero, then the Body Effect [145], or substrate bias [146], affects the symmetry of the output current variation with $|V_{in} - V_{centre}|$, Figure 5.8(a).

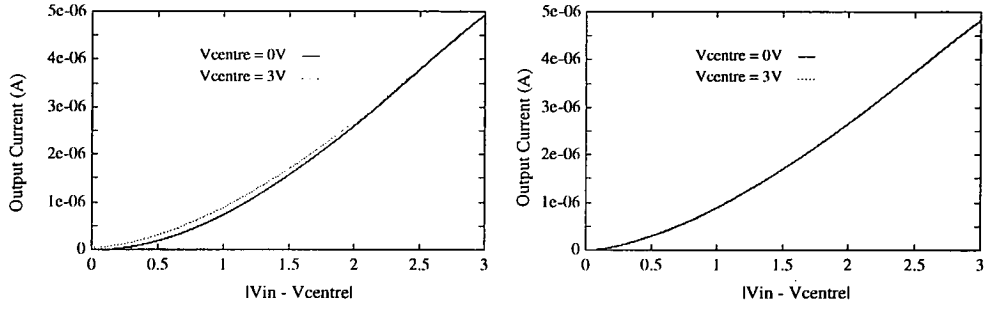


Figure 5.8 - The output of the PMOS Distance Circuit (a) Without N-Wells and (b) With N-Wells

The reason for this effect is clear. A non-zero bulk-source voltage increases the magnitude of the threshold voltage for the transistors, making them less conductive: the higher the bulk-source voltage, the lower the conductivity of the transistor channel. The common source voltage in each ratioed pair is determined by the gate voltage of M_{wide}. Therefore, V_{bs} varies with the gate voltage of M_{wide}, affecting the operation of the circuit and leading to the asymmetry of the output current.

The use of separate wells for each ratioed pair ensures that V_{bs}=0 and removes the asymmetry from the characteristic, Figure 5.8(b). Unfortunately using separate wells necessarily increases the area of silicon required. For the RHO test chip, second order effects such as the Body Effect were removed wherever possible, normally at the expense of increased silicon area. Separate wells were therefore used for each ratioed pair in order to establish that the theoretical operation of the circuit was correct.

5.2.4. Weight Load Circuitry

The distance circuit uses two pass transistors (switched by the ROW and COL signals) to facilitate the loading of the C_{centre} storage capacitance. Thus the loading of the neural parameters is a voltage mode operation that relies on transferring the correct amount of charge from the common *ip_centre* node, Figure 5.2, onto C_{centre} to establish the correct value of V_{centre}. With both the ROW and COL pass transistors **off**, the V_{centre} voltage is stored dynamically on C_{centre}. This voltage will decay over time due to the constant leakage currents associated with the transistor switch (Section 3.4.2), and so must be periodically refreshed. As with the C_{gate} capacitances in

the previous chapter, charge injection from the two switches can affect the stored value of V_{centre}. Again, though, the effects of charge injection can be minimised by using minimum area switches and a large C_{centre} capacitance.

5.3. Non-linearity Circuits

The non-linear transformation of the Euclidean distance approximation into the hidden layer output state was implemented using two different PWM approaches:

i) linear discharge of a capacitor with I_{dist} followed by use of a non-linear ramp to create the output pulse [143]

ii) use of the inherent non-linearities of MOS transistors to perform a non-linear current to voltage transformation, followed by use of a linear ramp to create the output pulse.

Both these methods have programmable width parameters, allowing a series of non-linear curves to be generated.

5.3.1. Capacitor-Based Non-linearity Circuit

By using the output current from the distance circuit, I_{dist}, to selectively discharge a capacitor, it was possible to implement a system (similar to other PWM implementations) which exploited the principle of conservation of charge in its operation, Figure 5.9.

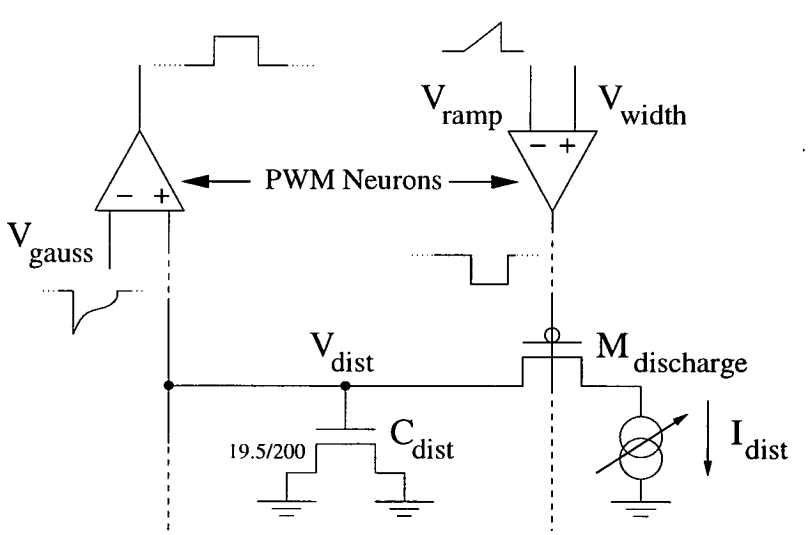


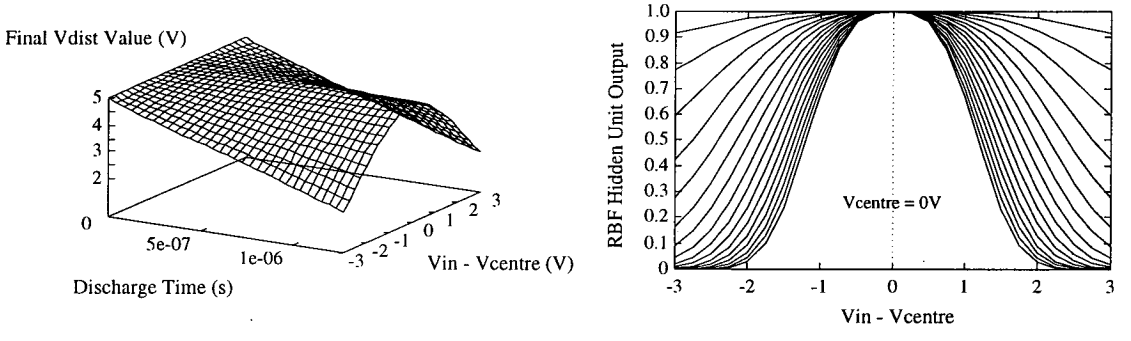
Figure 5.9 - Schematic Diagram for the Capacitor-Based Non-linearity Circuit

Capacitor C_{dist} is initially precharged to V_{limit} and I_{dist} is generated by the distance circuit described in Section 5.2, shown here as a variable current sink. Whilst $M_{discharge}$ is **on**, I_{dist} linearly discharges C_{dist} producing a change in the output voltage V_{dist}. Once $M_{discharge}$ switches **off**, C_{dist} is isolated, and V_{dist} is held dynamically on the output node. With C_{dist} isolated, V_{dist} remains constant (subject to leakage currents) and can be applied to one input of a pulse generating comparator. By applying a suitable inverse Gaussian to the other input, V_{gauss}, an output pulse is produced whose width is a Gaussian function of time.

The linear operation of this circuit can be described using equation 5.8.

$$\Delta V_{dist} = \frac{I_{dist}\Delta t_{width}}{C_{dist}} \tag{5.8}$$

From this equation, it is clear that the discharge of the capacitor can be varied using both I_{dist} and Δt_{width}, the on-time of $M_{discharge}$. By varying the discharge time of the capacitor as well as I_{dist}, it is possible to obtain a family of V_{dist} vs. I_{dist} curves, Figure 5.10(a).



(a) (b)

Figure 5.10 - (a) Variation in V_{dist} with Δt_{width} and $V_{in} - V_{centre}$ and (b) the Gaussian-like curves produced by non-linearly transforming V_{dist}

The results shown here were obtained from a suitable HSPICE simulation of the circuit. These curves illustrate that the discharge of the capacitance is quadratic in $(V_{in}\text{-}V_{centre})$ and linear in Δt_{width}. Each individual point from the family of curves is an approximation to the squared Euclidean distance between two voltages and each curve can be transformed, by a suitable exponential function, into an equivalent Gaussian curve, Figure 5.10(b). Therefore, not only does this circuit allow the reproduction of a Gaussian non-linearity, it also allows the width of the curve to be adjusted.

For this circuit, the distance circuit can discharge C_{dist} by 3.2V in 1.28μs using a maximum value for I_{dist} of 5μA.

5.3.2. Transistor-Based Non-linearity Circuit

The basic transistor non-linearity circuit is shown in Figure 5.11. Its operation is based on the fact that, when a transistor is biased into its linear region of operation, it behaves like a non-linear voltage controlled resistor: the voltage dropped between the source and drain terminals being related to both the drain-source current in the device and the voltage on its gate.

The operation of the circuit is illustrated by the results of the HSPICE simulation of a circuit comprised of a distance circuit, as in Figure 5.1, and the M_{width} transistor from Figure 5.11. For this simulation, different values of V_{width} were applied to M_{width}, V_{centre} was fixed at 0V and V_{in} was increased linearly from 0V to 3V for each value of V_{width}. Figure 5.12(a) shows the variation of both V_{dist} and I_{dist} with $(V_{in} - V_{centre})$ for several values of V_{width}.

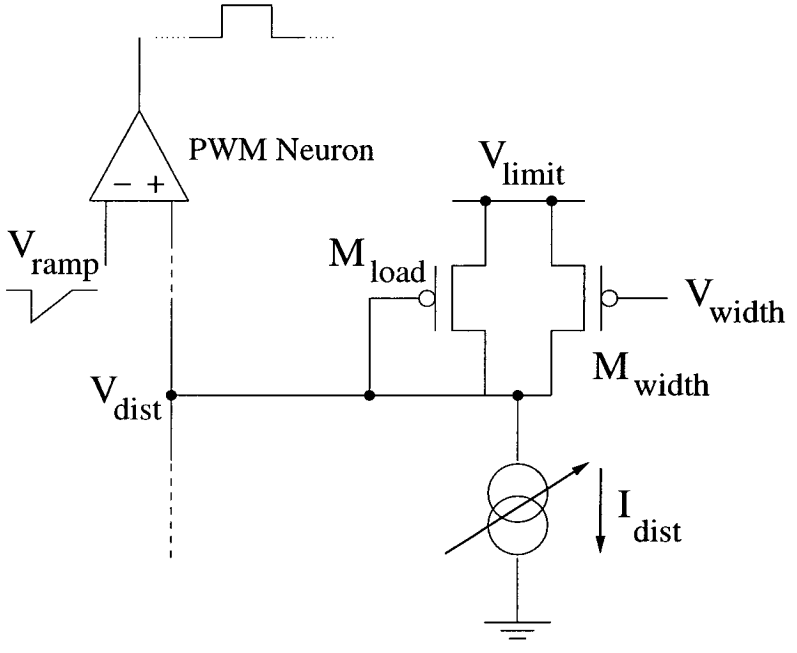


Figure 5.11 - Schematic Diagram of the Transistor-Based Non-linearity Circuit

Whilst these graphs indicate the possibility of using single transistors biased into their linear operating regime to produce a family of "bump-like" non-linearities, correct operation breaks down as soon as M_{width} saturates. Once M_{width} saturates, V_{dist}

falls quickly to 0V and I_{dist} levels off, no longer a quadratic function of $(V_{in} - V_{centre})$.

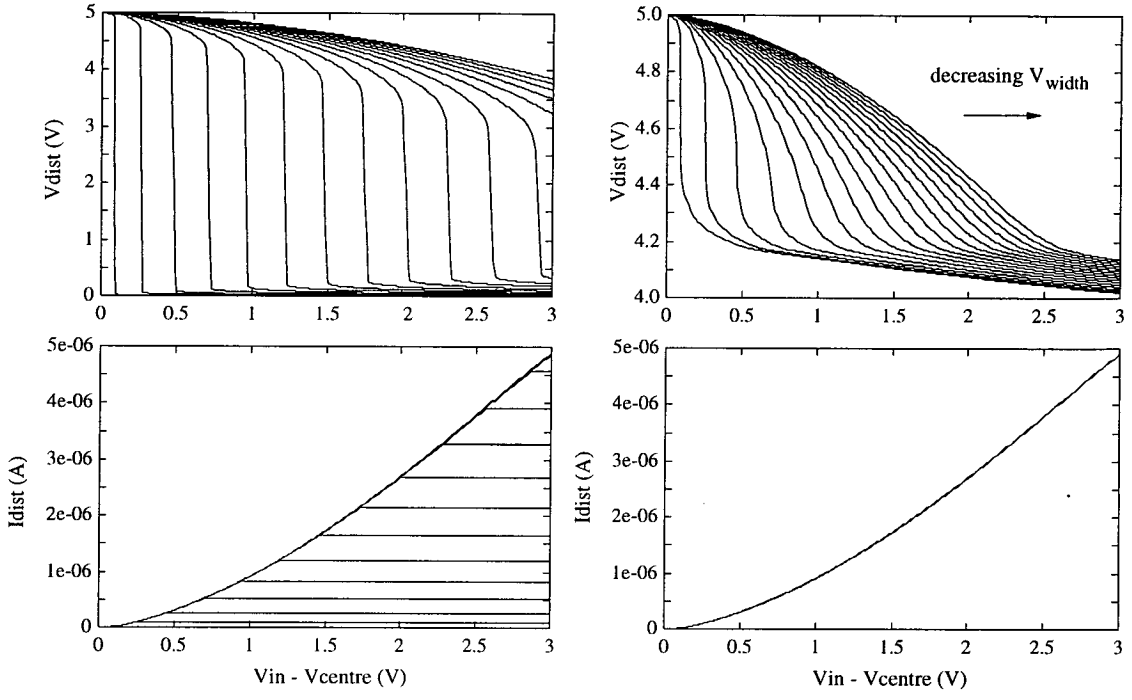


Figure 5.12 - Transistor Circuit Curves showing (a) V_{dist} and I_{dist} with M_{load} disconnected from the circuit in Figure 5.11 and (b) V_{dist} and I_{dist} with M_{load} connected

However, if M_{load} is connected in parallel with M_{width} and the simulation repeated, it can be seen that the disadvantages of the previous circuit are overcome, Figure 5.12(b). M_{load} conducts any excess current drawn as M_{width} saturates and clamps the output voltage to around a threshold drop below V_{limit}. The load transistor also rounds the corners of the V_{ds} vs. I_{ds} characteristic as M_{width} enters saturation, smoothing the shape of the non-linearity and providing a tapered roll-off as I_{dist} increases.

The shape and spacing of the series of non-linear curves generated for different values of V_{width} are determined by the W/L ratios of M_{width} and M_{load} respectively. Varying $(W/L)_{load}$ for constant $(W/L)_{width}$ affects the fall-off of V_{dist} after M_{width} saturates - wider M_{load} transistors produce flatter tapers. Meanwhile, varying $(W/L)_{width}$ for constant $(W/L)_{load}$ alters the shape and range of the curves - longer M_{width} transistors produce less spacing between the curves. Thus, different shapes of non-linearity can be created by varying the W/L ratios for the two transistors.

Whilst the two-transistor circuit of Figure 5.11 provides a parsimonious implementation of a transistor-based non-linearity, it is not the only option. The *load* connected in parallel with M_{width} can be made of several diode connected transistors. These other arrangements allow for countless other non-linear curves to be produced and by series-connecting two or three devices, it is possible to increase the dynamic range of the circuit. The transistor-based non-linearity circuit fabricated on the RHO chip was the four transistor version in Figure 5.13, shown along with its output characteristic.

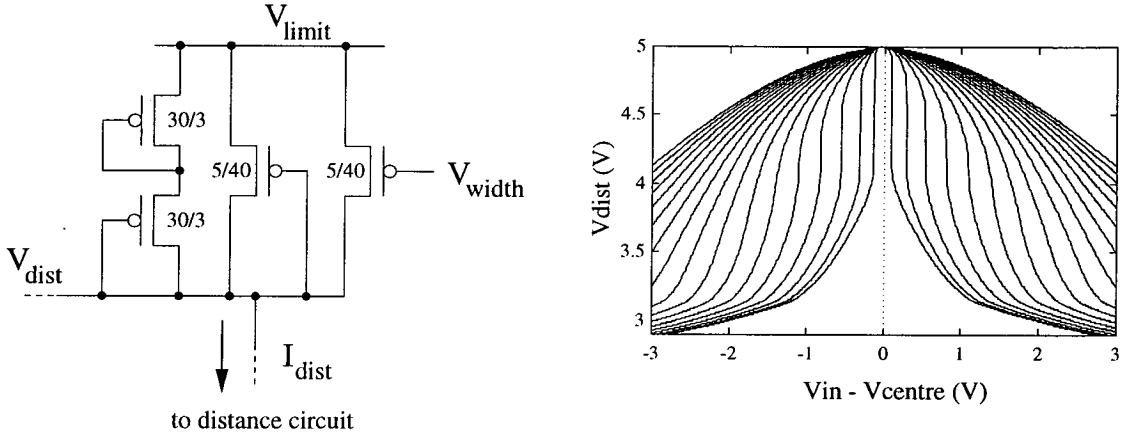


Figure 5.13 - The Transistor-Based Non-linearity Circuit implemented on the RHO chip along with its output characteristic

5.3.3. Circuit Cascadability

The different problems that can be solved using an ANN require specific numbers of input units, hidden units and output units. In order to prevent re-design of the circuits and cells for each application specific neural chip solution, it is imperative that a neural cell library of cascadable cells is designed. Then, only the relative number of the hidden and output cells will change between different neural chip designs, greatly reducing the design effort required for the chips. The circuits for the hidden layer were all designed to be cascadable components of a neural cell library.

5.3.3.1. Cascaded Distance Circuits

The distance circuit in Section 5.2 calculates a quadratic approximation to the squared Euclidean distance between two voltages, each voltage representing single components of multi-dimensional vectors. In order to calculate an approximation to the distance between multi-dimensional vectors, several instances of these circuits must be cascaded together. This cascading is achieved by connecting all the output

nodes of the circuits together. Then, by Kirchoff's Current Law, the total current sunk by the cascaded circuits is equal to the linear summation of the currents sunk by the individual circuits - as required by equation 2.15.

5.3.3.2. Cascaded Capacitor Circuits

The capacitor non-linearity circuit is also easily cascaded. By assuming each distance circuit has a capacitor of fixed value attached to its output, and that the same PWM width signal is applied to all the $M_{discharge}$ transistors, the size of the total distance capacitance, C_{dist_tot}, increases linearly with the number of cascaded distance circuits. The operation of the cascaded capacitor circuitry is therefore described by equation 5.9.

$$\Delta V_{out} = \frac{\Delta t_{width} \sum_{i=1}^{N} I_{out\_i}}{NC_i} \tag{5.9}$$

Since N lumped capacitances, C_i combine to form the single distributed capacitance, C_{dist_tot}, this arrangement results in the formation of an averaged capacitance distributed across the chip. Distributing the capacitance in this way helps to implicitly account for across-chip variations in one direction.

5.3.3.3. Cascaded Transistor Circuits

In essence, the operation of the transistor-based non-linearity circuit is described by equation 5.10, where R_{tran} is the resistance of the transistor combination.

$$V_{dist} = V_{limit} - R_{tran}\, I_{dist} \tag{5.10}$$

Ideally, I_{dist} should be drawn through a circuit whose non-linear resistive characteristic remains constant no matter how many distance circuits are cascaded together. Therefore, R_{tran} must decrease linearly with the number of distance circuits. Correct operation is obtained by cascading the transistor-based non-linearity circuits in exactly the same way as the capacitor-based circuits, although this may not be so intuitively obvious.

By way of explanation, consider the equivalent circuit for three cascaded transistor-based centre circuits, Figure 5.14.

In Figure 5.14,

$$V_{dist} = V_{limit} - R_{TOT}\, I_{TOT} \tag{5.11}$$

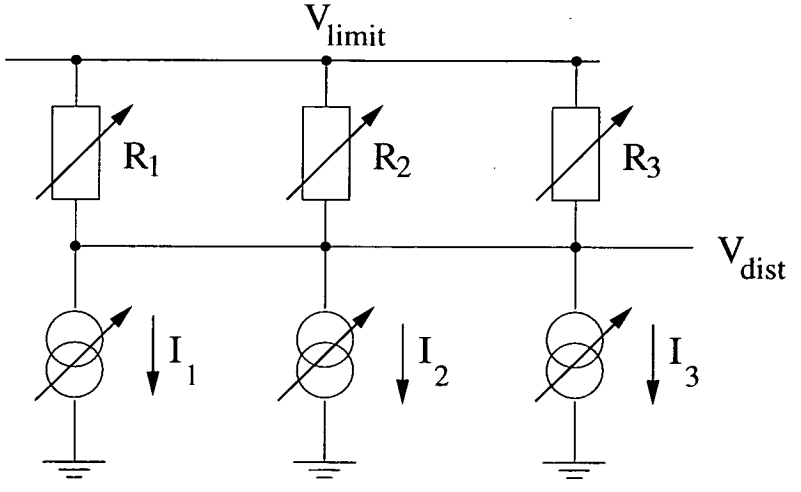


Figure 5.14 - Equivalent Circuit of 3 Cascaded Transistor-Based Circuits

$$I_{TOT} = I_1 + I_2 + I_3 = \sum_{i=1}^{3} I_i \tag{5.12}$$

$$R_{TOT} = \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} \right)^{-1} = \frac{R_{tran}}{3} \tag{5.13}$$

$$\textit{iff } R_1 = R_2 = R_3 = R_{tran} \tag{5.14}$$

Assuming the resistance of each circuit is the same, then, to a first order, as the number of circuits increases, the total resistance, R_{tot}, decreases linearly. The non-linear characteristic of each circuit is determined by the gate voltage of M_{width}, thus if the same gate voltage is applied to each circuit, it will, ideally, generate the same resistance in all the cascaded circuits.

5.3.4. Comparison of Capacitor and Transistor Non-linearities

Although both the capacitor and transistor-based non-linearity circuits are capable of producing families of non-linear curves, each implementation has its respective advantages and disadvantages.

The Capacitor Circuit

The capacitor circuit is simple to implement and understand and is also easily adaptable: the actual shape and width of the non-linear curves are determined by the arbitrary ramped waveforms applied to the PWM Neurons. There is therefore no restriction to using Gaussian non-linearities with the circuit.

However, due to the pulsed nature of the width control signal, the circuits have a finite evaluation time which restricts the maximum rate of data through-put. Furthermore, the dynamic storage of V_{dist} allows this voltage to be corrupted by leakage currents, analogue noise and capacitive coupling. To reduce the effects of these phenomena, C_{dist} should be large, but this increases the area of silicon needed to implement the circuit and directly reduces the number of centre circuits that can be fabricated on a given die size.

The Transistor Circuit

The transistor circuit takes up a far smaller area than the capacitor circuit, holds the final value of V_{dist} statically on the input to the PWM Neuron, and the only evaluation time overhead required to produce V_{dist} is the settling time of the circuit should V_{width} or I_{dist} change. Also, since the transistors generate the non-linearity, only linear ramps are required for the voltage to pulse width conversion. Linear ramps are easier to generate both on-chip and off-chip.

However, the non-linear characteristic of the transistor-based circuits are fabrication dependent, so the shape of the non-linear curves is likely to vary both across-chip and between chips.

5.4. The RHO Test Chip

Having developed a distance circuit and non-linear capacitor and transistor-based PWM circuits for implementing the hidden layer of an RBF network, a test chip was manufactured to test and assess their functionality and operational performance.

The aims of the chip were to:

i) confirm the functionality of multiple instances of all three circuit implementations

ii) review the level of across-chip and inter-chip variations of the circuits

iii) compare and contrast the operation and characteristics of arrays of both the capacitor-based and transistor-based non-linearity circuits

iv) consider the operational requirements for both methods of generating the RBF non-linearity.

A block diagram of the chip is shown in Figure 5.15.

Each chip consists of a *CAP* array of capacitor-based centre circuits, a *TRAN* array of transistor-based centre circuits and the necessary support circuitry for both arrays. Both the *CAP* and *TRAN* arrays consist of 64 cells arranged as 8 centres with 8 inputs per centre. Each centre cell is cascadable and consists of a single distance circuit and one capacitor or one transistor-based non-linearity circuit.

The Row and Column decoders are simply multiple instances of 3 input NOR gates and allow each cell in both arrays to be uniquely addressed.

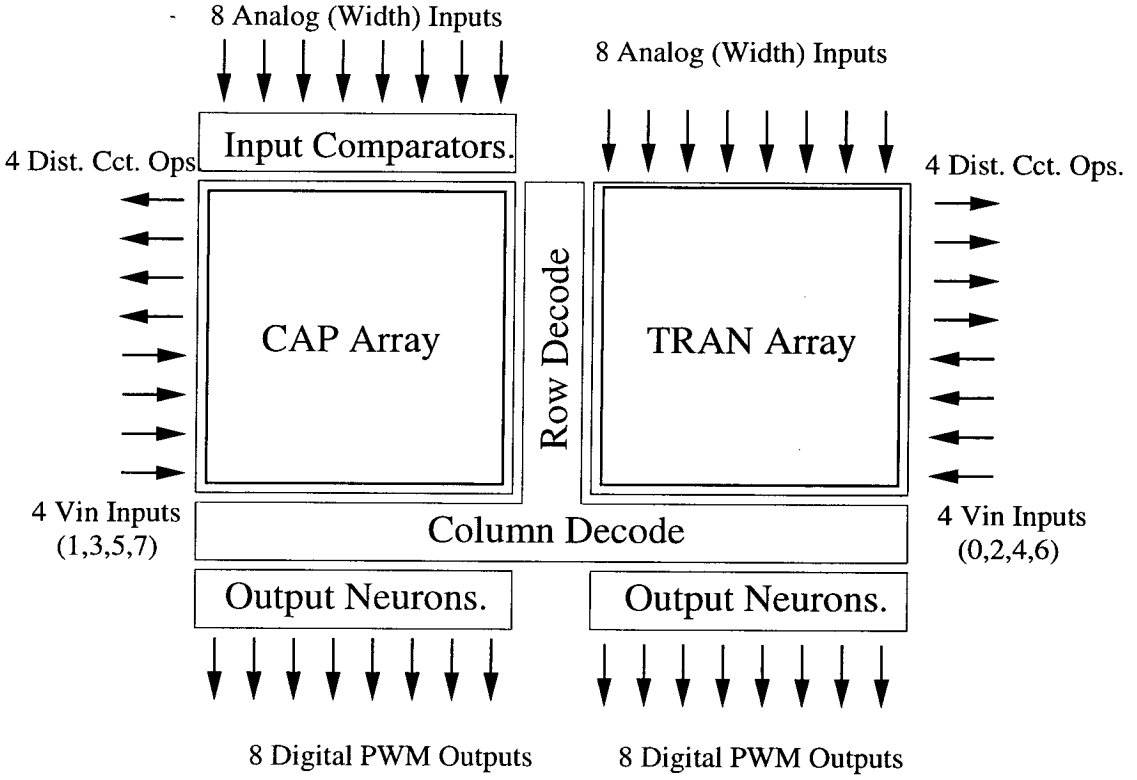


Figure 5.15 - RHO Chip Block Diagram

The output neurons are the PWM comparators - they generate the output pulses from the centres in each array - while the input PWM comparators in the *CAP* array generate the pulses that control the selective discharge of the C_{dist} capacitances.

The layout for the chip is shown in Figure 5.16, and it was fabricated on a 6mm by 5mm die.

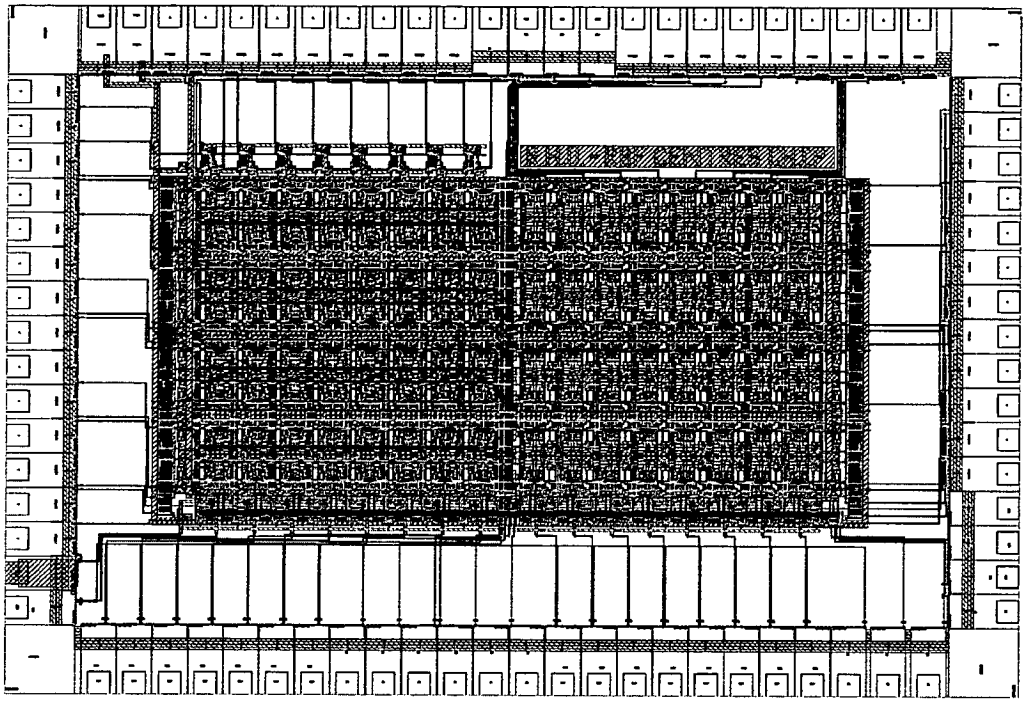


Figure 5.16 - The Layout for the RHO chip

5.5. Experimental Results

A dedicated development board was built for the RHO chip to allow it to be correctly biased and automatically tested. A description of the RHO development board is given in Appendix A.

For the RHO Development Board, the V_{in} and V_{centre} voltages were generated using off-chip 8-bit voltage DACs, calibrated to produce voltages between 0V and 3V. For this reason, both voltages, or their difference, are expressed in terms of the 8-bit digital words presented to the respective DACs. Also, as per Chapter 4, the neural outputs from the chip are expressed in *RAM Locations*.

5.5.1. Distance Circuit Results

Experiments were carried out to assess the functionality of several distance circuits on the RHO chips. By providing on-chip measurement transistors, similar to those used for characterising the DAC and DCMs on the DYMPLES chip, it was possible to allow 8 distance circuits to be characterised per chip, 4 from each array. Again on-chip current variations were tracked as off-chip voltage variations.

5.5.1.1. Functional Operation

The variation of the output voltage from the off-chip Op Amp with (V_{in}-V_{centre}) was measured for the 8 distance circuits on 10 RHO chips and Figure 5.17 shows the results from two chips. All the distance circuits characterised produced results consistent with the existence of a quadratic relationship between I_{dist} and (V_{in}-V_{centre}), however variations exist between the curves from different circuit implementations. This is no more than expected, though, since no explicit steps were taken to account for process variations during the design process.

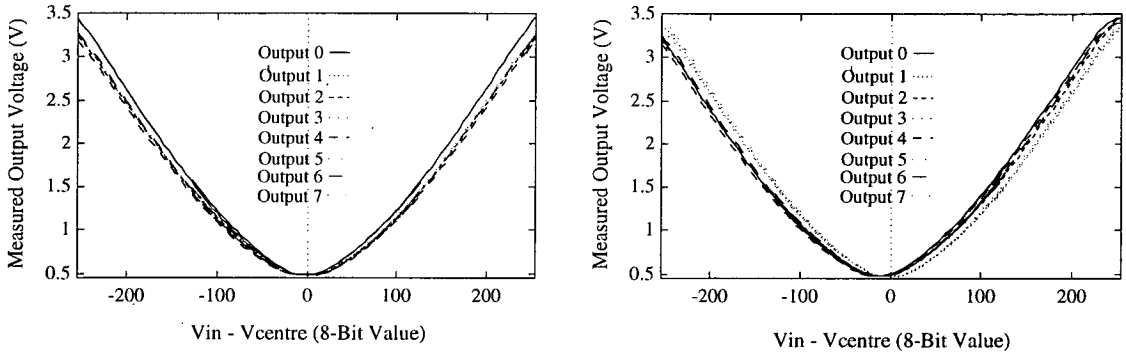


Figure 5.17 - Results from 8 Distance Circuits on two RHO chips

Figure 17(b) highlights a more worrying observation. Here the minima of half the curves are offset along the x-axis. This phenomenon was witnessed to some extent on over half of the characterised chips. After an exhaustive test procedure, the following observations were noted regarding this offset error:

i) the results were consistent - the same outputs on any given chip were always offset

ii) pairs of curves were always offset and both came from the same row of cells, one from the distance circuit in the *CAP* array, the other from the distance circuit in the *TRAN* array

iii) pairs of curves from the even, but never the odd, numbered rows were always offset.

The source of the error was traced to a design decision taken when the cells forming the *CAP* and *TRAN* arrays were fitted together. In an attempt to reduce the silicon area required, every second row of cells in both the *CAP* and *TRAN* arrays were flipped over to allow the power and ground lines of alternate rows to butt together.

However, this "mirroring" increases the likelihood of differences existing between the normal and mirrored rows and this is the most likely explanation for the offset characteristics witnessed in the results from the distance circuit experiments.

5.5.2. Capacitor Circuit

The operational performance of the capacitor-based non-linearity circuit was thoroughly investigated and the observed results are now considered in terms of the functional operation of the circuit and the effect that the discharge time, Δt_{width}, and the offset distance characteristics, have on this operation. Details of how the experimental results were obtained are given in Appendix A.

5.5.2.1. Functional Operation

Initial results from the RHO chips confirmed that the capacitor-based non-linearity circuits functioned correctly, Figure 5.18. This graph shows the measured output pulse width vs. V_{in} characteristic averaged over the 8 outputs on a single RHO chip. V_{centre} was fixed at 128 for this experiment. The errorbars shown for every 8th value of (V_{in} - V_{centre}) represent ±3 standard deviations of the measured results. The family of curves shown were produced using different values of V_{width}, and confirms that the width of the capacitor-based circuit can be altered using traditional PWM techniques.

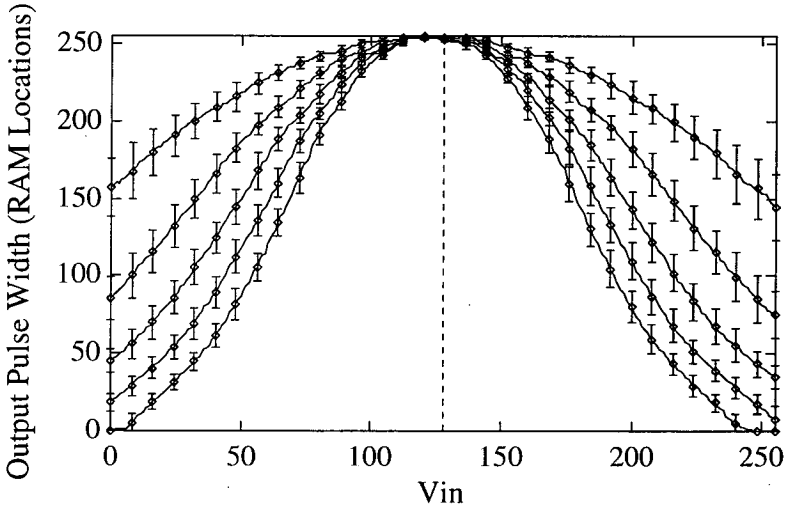


Figure 5.18 - Measured Output Pulse Widths for a Single RHO chip

Further experiments using different maximum times for the width pulse indicated that by further increasing the evaluation time, Δt_{width} for the circuit, it was possible to

produce narrower Gaussian-like curves. Sample results from a single RHO chip are displayed in Figure 5.19.

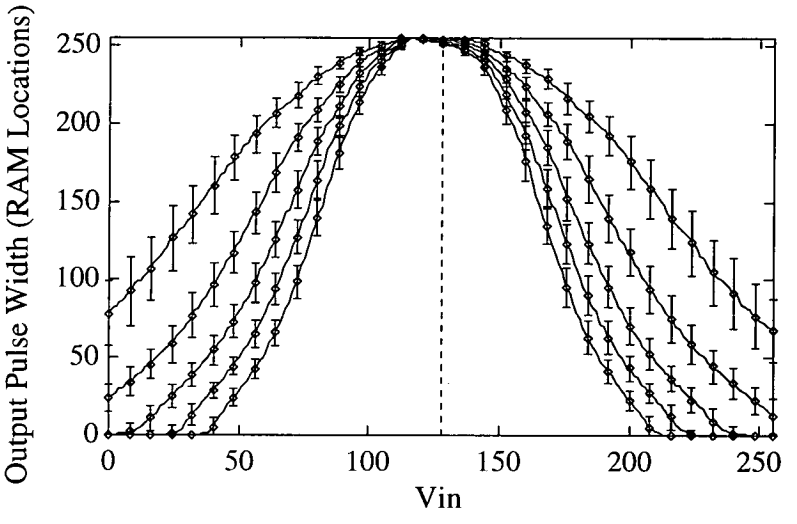


Figure 5.19 - Narrower Measured Output Pulse Widths for a Single RHO chip, generated using a longer evaluation time, Δt_{width}

The following observations were also recorded from the functionality experiments:

i) multiple results from single centres using the same V_{width} and V_{centre} values showed the circuit performance to be consistent under consistent operating conditions, although the dynamic storage of V_{dist} meant that it was susceptible to noise, leading to occasional corruption of the output pulses

ii) the performance of the capacitor circuits and the width generating comparators varied across chip

iii) the use of PWM circuitry to determine the discharge time of the C_{dist} capacitances increased the complexity of the circuitry and the effort required to set it up and operate it correctly

iv) although the circuit performance was impressive, the Gaussians were fairly wide in comparison to the range of each input dimension - even for a full $10\mu s$ discharge pulse.

5.5.2.2. Circuit Performance and Δt_{width} Evaluation Time

Concern at the implications of the last observation in the previous section warranted investigation of the circuit operation at slower development board clock speeds. The theoretical considerations already discussed in Section 5.3.1 suggested that operating the circuit with a slower clock should produce narrower Gaussian-like curves. Therefore some experiments were performed using 1 MHz and 12 MHz development board clocks.

These experiments consisted of producing output curves from the *CAP* arrays for six different values of V_{centre}. The combined results from three trials, on two RHO chips, are shown in Figure 5.20.

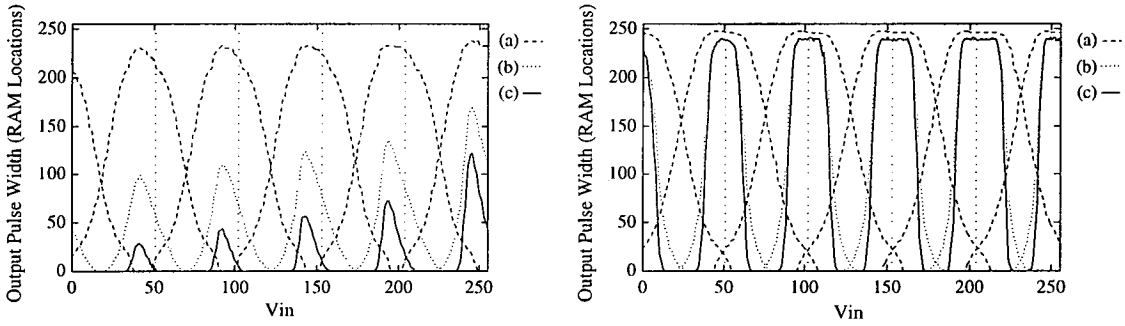


Figure 5.20 - Gaussian Shape Variation with Board Speed for Two Different RHO Chips

These results are from a single capacitor-based centre circuit on two chips and correspond to the measured output pulse widths obtained for:

a) a maximum discharge pulse, generated using a 12MHz clock (approx. 21μs)

b) a discharge pulse one quarter of the maximum generated using a 1MHz clock (approx. 64μs)

c) a maximum discharge pulse generated by a 1MHz clock (approx. 250μs).

Clearly, for these curves, not only is there a variation in curve amplitude with board clock speed for some chips, there is also an amplitude variation with V_{centre}.

5.5.2.3. Common Mode Circuit Performance

Since the last experiments indicated that the performance of some of the *CAP* arrays varied with V_{centre}, further experiments were performed to investigate the common mode operation of the circuit, ie when $V_{in} = V_{centre}$. The maximum discharge pulse

width was applied to $M_{discharge}$ for these observations since this allows any offset current to produce the maximum change in V_{dist}.

The trials were conducted several times, with different clock speeds, and using 3 different chips. The collective results are shown in Figure 5.21. They indicate:

i) the results of these experiments are chip dependent

ii) using the 12MHz and 24MHz crystals to drive the RHO Development Board produces negligible differences in the amplitude of the measured output pulse width with either the clock or with V_{centre}

iii) using the 1MHz crystal produces a marked reduction in the amplitude of the measured output pulse width on some chips and the amplitude of the output pulse increases as the values of V_{in} and V_{centre} increase.

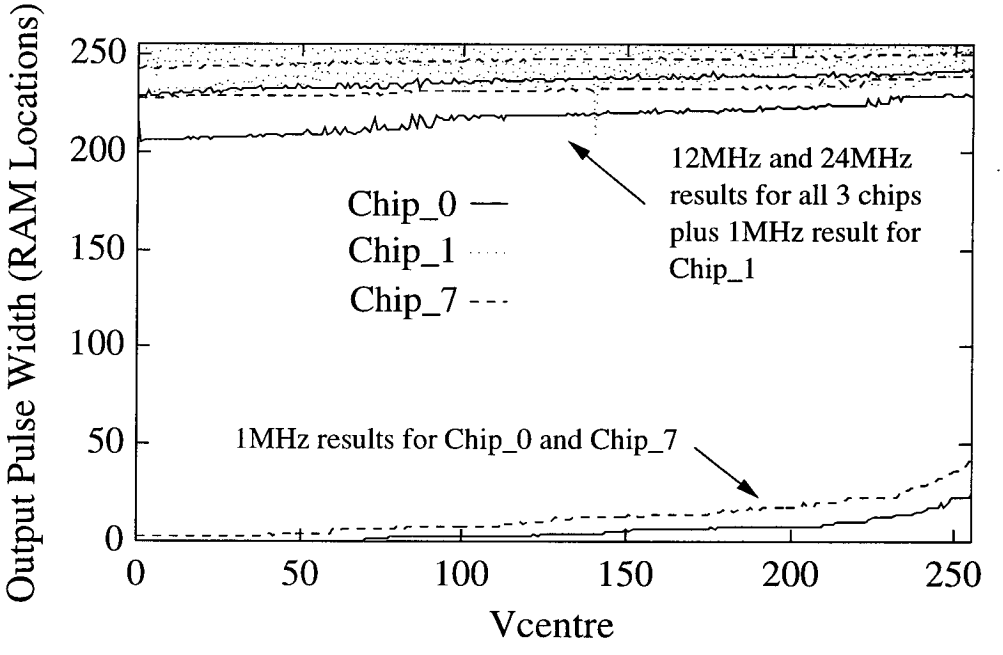
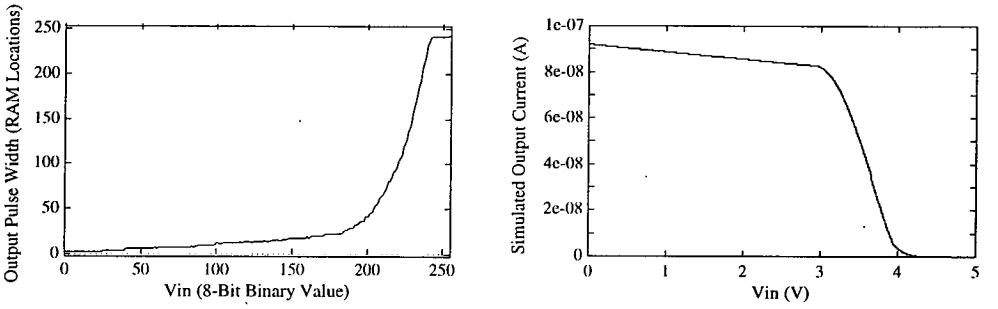


Figure 5.21 - Common Mode Variation: Different Clock Speeds and Different Chips

By extending the valid voltage range of V_{in} and V_{centre} from 0V - 3V to 0V - 4V and repeating the experiment using a single chip and the 1MHz board clock, the graph in Figure 5.22(a) was obtained. This graph mirrors the output current vs. V_{centre} characteristic obtained from an HSPICE simulation of the distance circuit *without* a compensation circuit, Figure 5.22(b).



(a) (b)

Figure 5.22 - Graphs showing (a) the common mode output pulse width variation with V_{in} measured from the circuit and (b) the simulated output current against V_{in} variation of a distance circuit with no compensation circuit

Thus it was concluded from these results that the compensation circuit on certain chips was not working correctly and was causing small offset currents to be produced by the distance circuits even when the same voltage was applied to both inputs. These currents were too small to be of significance at board speeds of 12MHz and 24MHz but were critical when the 1MHz clock was used. Indeed, it was calculated that a current of only 30nA from a distance circuit could discharge the C_{dist} capacitance by half its range using a maximum width discharge pulse generated using a 1MHz board clock. This current represents only 0.6% of the maximum discharge current the circuit was designed to produce.

The most important consequence of these results, however, is that increasing the discharge time is equivalent to reducing the size of C_{dist} - deemed a necessary modification for reducing the area of future versions of the circuit. Therefore, using this circuit with a smaller output capacitance could introduce significant discharge problems when the board is operated at 12MHz or 24MHz.

5.5.2.4. Non-linearity Reproduction Ability

Since some chips appeared to have malfunctioning compensation circuits, the ability of all 10 RHO chips to reproduce the Gaussian-like non-linearity at 1,12 and 24 MHz board speeds was investigated. A total of nine chips had working *CAP* arrays and results from one output from two different chips are shown in Figure 5.23. The measurements showed that whilst some of the *CAP* arrays were unable to reproduce the non-linearity at low clock speeds, others could reproduce it extremely well.

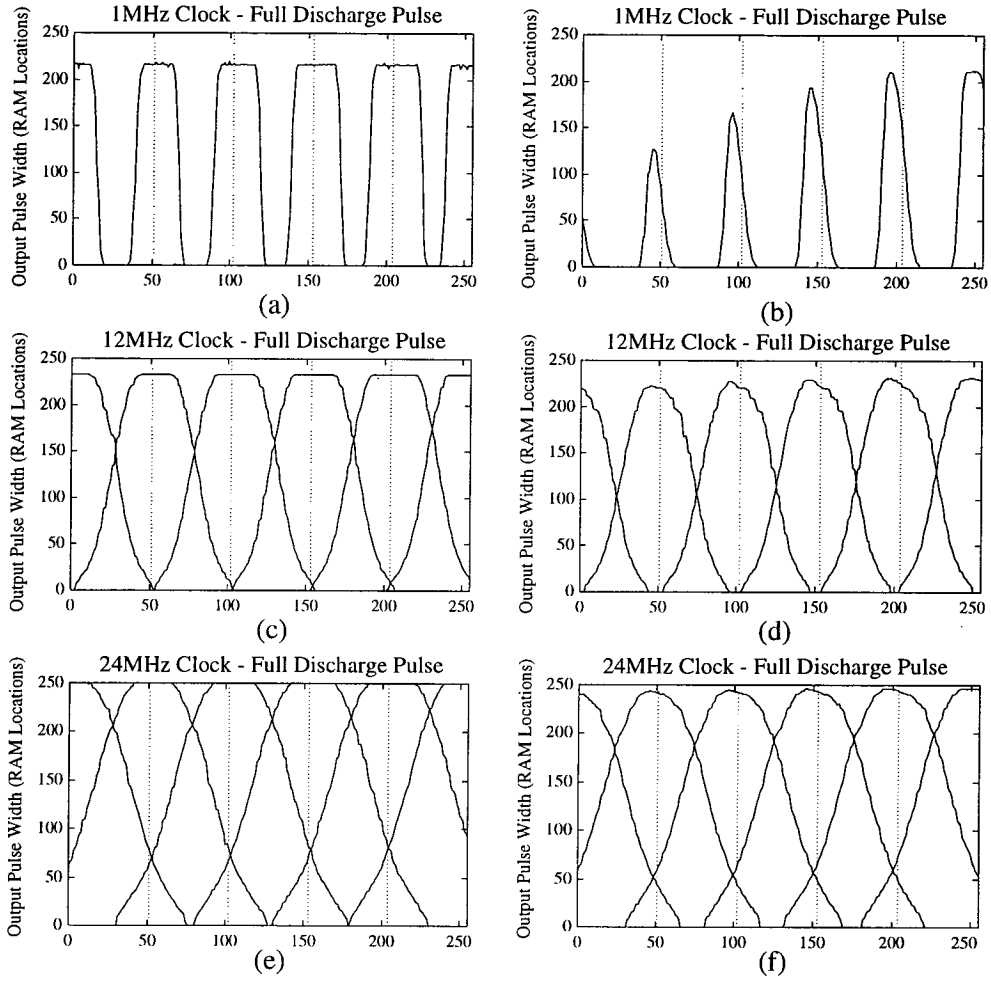


Figure 5.23 - Comparison of the ability of two chips to reproduce Gaussians at different clock speeds

Significantly, those chips that had difficulty reproducing the non-linearity also had offset distance characteristics from either or both even rows of measurable distance circuits. Thus it appears that the failure of some chips to reproduce the correct shape of the non-linearity is a direct result of the inappropriate layout decision discussed earlier.

5.5.2.5. Other Observations

Further observations from the experiments carried out on the *CAP* array were:

i) the grounded transistor gates in the global compensation circuits caused the "flat tops" on the non-linearities obtained from working chips, Figure 5.23

ii) most of the Gaussian-like curves appeared to be asymmetrical, indicating that the use of separate N-wells for the ratioed pair transistors in the distance circuits may not be necessary.

5.5.3. Transistor Circuit Experiments

As with the *CAP* arrays, the performance of the *TRAN* arrays on the RHO chips was thoroughly investigated.

Before considering the results from the transistor circuit experiments, it is worth highlighting that the small currents generated by the distance circuit when V_{in} = V_{centre} will not adversely affect the operation of the transistor-based circuit. Any small currents generated by the distance circuits in the *TRAN* array will simply produce small $I_{dist}R_{tran}$ voltage drops from V_{limit}. These small voltage drops will have a negligible affect on the operation of the circuit.

5.5.3.1. Functional Operation

Experiments again determined the functionality of the circuitry within the *TRAN* array and were used to investigate the across-chip variations in the shape of the transistor-based non-linear curve.

The experimental results shown in Figure 5.24(a) to (d) confirmed the theoretical properties of the circuit, namely:

i) each centre produced a consistent non-linearity, Figure 5.24(a)

ii) a family of non-linear curves was produced by varying V_{width}, Figure 5.24(b)

iii) the shape of the non-linear curves varied both across-chip and between chips, Figures 5.24(c) and (d).

In all, 8 *TRAN* arrays had 8 functioning centres, all of which were affected to some degree by across-chip variations in the fabricated circuits. Again such discrepancies were expected since the circuits operate in voltage mode.

Further trials on a single chip for a single value of V_{centre} and several values of V_{width} indicated that the measured output pulse width was unaffected by the operating speed of the RHO development board.

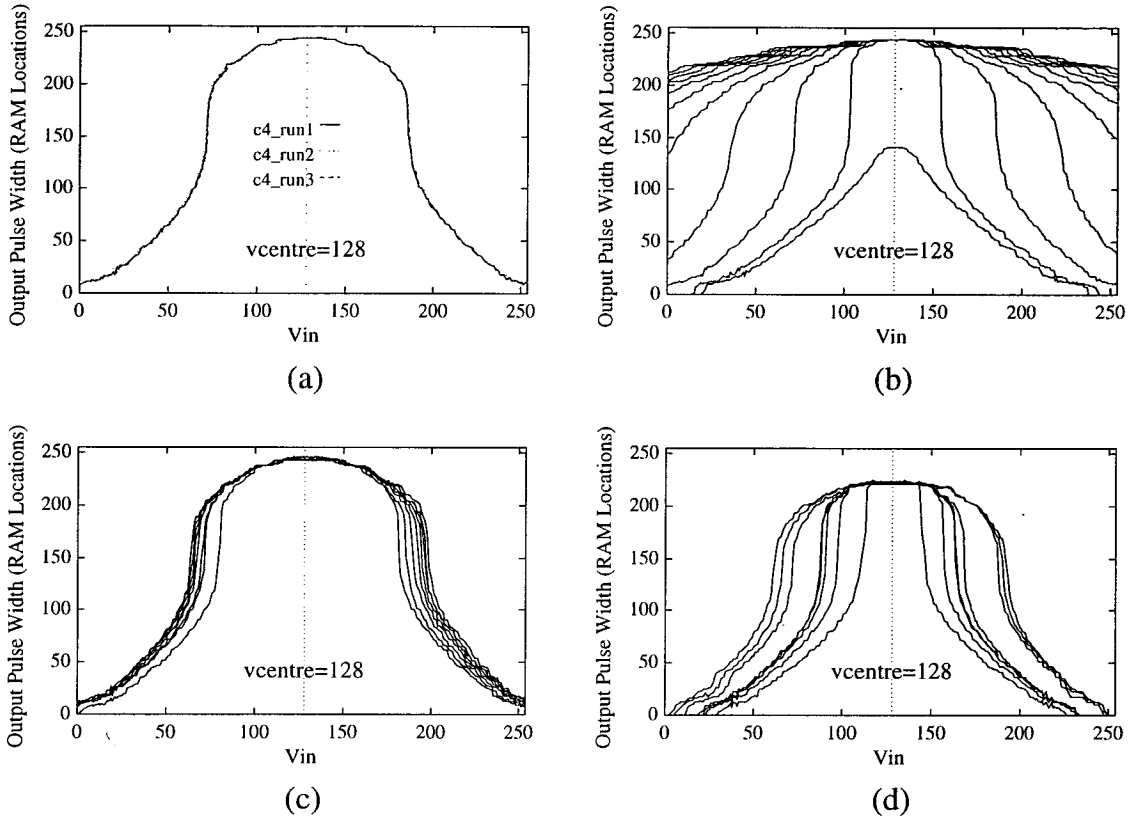


Figure 5.24 - Results from the TRAN array of the RHO chip

5.5.3.2. Transistor Centre "Block Tests"

Some of the initial experiments on the *TRAN* arrays indicated that some centres on some chips were narrower than the others. Since the RHO chip was designed so that rows of distance and non-linearity cells could be switched in and out of the centres, allowing the dimensionality of the centres to be altered by 2, 2 and 4, this discrepancy was investigated further.

The transistor-based non-linearity circuit depends on the total I_{dist} current being drawn through what is effectively a distributed resistance. Thus, it was postulated that the observed width variation in the centres could be due to the malfunctioning of one or more of the cascaded circuits within the centre, with a single circuit malfunction affecting the averaged operation of the centre.

All 8 combinations of circuit blocks on several chips were characterised and the results from two centres on a single chip are shown in Figure 5.25. It was observed that although slight differences exist between the different combinations of blocks for both centres, clearly all the curves are almost identical. Therefore, the difference in

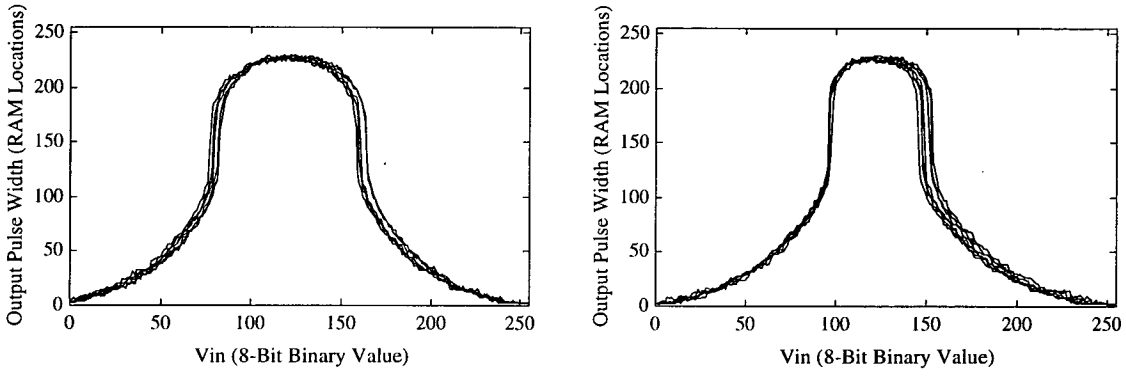


Figure 5.25 - Block Test Results showing the transistor non-linearities obtained for different combinations of the cascaded cells on a single RHO chip - (a) shows the curves obtained from centre 0 and (b) shows the curves obtained from centre 4 of RHO chip 0

width between these two centres cannot be due to the malfunction of a single circuit, or a combination of centre circuits. Rather the malfunctioning must be due to something that is common to each centre. The most likely culprit is the V_{width} voltage since it determines the shape of the non-linearity.

5.6. Discussion

A study of the characteristics and operation of the two centre arrays on the RHO chip has led to a number of interesting observations, as described in the previous sections. Now the probable effect of these observations for RBF implementations must be considered.

5.6.1. Distance Circuit

The experiments on the testable distance circuits on the RHO chips showed:

- the circuits functioned as required, producing a quadratic approximation to the squared Euclidean distance between two voltages
- offsets occurred in the characteristics of some distance circuits because every second row of cells were mirrored in the y-axis.

Since the characteristics of all the distance circuit cells on an RBF chip should be identical, the cells must have the same orientation on the silicon substrate [147]. This will reduce the likelihood of offsets, such as those seen in Figure 5.17(b),

occurring.

Further, the following observations were attributed to malfunctioning compensation circuits:

- the flat portions on the tops of the non-linear curves from some of the *CAP* arrays, Figures 5.23(a),(c) and (e)
- the variation of the amplitude of the *CAP* array non-linearities with V_{centre}, especially at low clock speeds, Figure 5.21
- the non-linearities with reduced height obtained from the *TRAN* arrays, Figure 5.24(b).

Future implementations of the circuit should therefore use *local*, rather than *global*, compensation circuitry, with the gates of the compensation circuit ratioed pair tied to V_{centre}.

Whilst the adoption of the identical cell orientation and local compensation cannot guarantee identical, matched operation for all instances of the circuit on an RBF chip (they will still be subject to the vagaries of process variations), these techniques will help minimise the effect of the experimentally observed corruption mechanisms and so will help minimise the discrepancies between the characteristics, potentially leading to improved circuit performance.

5.6.2. Capacitor Circuit

The main observations from the experiments on the *CAP* arrays can be summarised as:

- the expected functional performance has been demonstrated, but the performance varied dramatically between different chips
- the non-linear curves were wide compared to the domain of input space when a 24MHz board clock was used
- circuit performance, for some chips, is dependent on the Δt_{width} evaluation time
- the common-mode output characteristic, for some chips, varies with the input and these chips also have offset distance curves.

Whilst it was clear that the operation of the circuits in the *CAP* arrays were affected by the discrepancies in the distance circuits, it was concluded that any reduction in the size of C_{dist} would magnify the effects of any offset currents and this could corrupt the operation of the circuit unacceptably.

Furthermore, the observations from the measured results are compounded by the circuit's large area requirement, finite evaluation time, reliance on dynamic voltage storage and its need for additional circuitry to determine the Δt_{width} evaluation time.

5.6.3. Transistor Circuit

The experiments on the *TRAN* arrays showed:

- the circuitry functioned as expected
- the performance of the centres were fabrication dependent, with the width varying across-chip
- the affect of the discrepancies in the distance circuit were far less severe than they were in the *CAP* arrays.

Indeed, the main concern with circuitry in the *TRAN* arrays was the fabrication dependent nature of the non-linearities. However, since the width of these curves is dependent on V_{width}, it should be possible to account for fabrication variations by independently setting the value of V_{width} for each centre.

5.6.4. Conclusion

Based on the experimental results obtained from the RHO chips, the experience gained from setting up and operating the arrays and the known properties of the circuits, it was concluded that a transistor-based non-linear circuit offered the greatest potential for the final pulsed RBF chip.

5.7. Summary

This chapter has concentrated on the RHO chip. RHO was developed, built and tested to investigate the operational characteristics of pulsed circuits for implementing the hidden layer of an RBF neural network. The aims and requirements of the circuits and chip were stated before the operation of each circuit was described in detail. The experimental observations, and their likely implications, were then discussed and a conclusion reached about which non-linearity circuit showed the greatest potential for further development.

Chapter 6

The PAR Chip

The previous two chapters have detailed the development of pulsed analogue VLSI circuitry capable of implementing the functions required by an RBF neural network. This chapter considers, in terms of the circuit and system level aspects, how this collection of individual circuits was modified and combined on a single piece of silicon to produce the Pulsed Analogue RBF (PAR) chip - a pulsed analogue VLSI implementation of a complete RBF neural network.

6.1. Circuit Level Considerations

The experiments on the DYMPLES and RHO chips had proven the functionality of all the developed pulsed RBF circuits. Furthermore, observations from these test chips suggested that possible alterations could be made to the original circuit designs and/or cell layout in order to improve performance. However, due to a pressing fabrication deadline, there was insufficient time to fully investigate and implement all the suggested modifications. In any case, since the PAR chip was the final demonstrator chip for the project, there was little merit in radically modifying or redesigning the circuits, as making such changes could introduce design errors or degrade the performance of the circuits.

Several modifications to the circuits were necessary, though, since, for example, the cell libraries for the DYMPLES and RHO chips were incompatible and the size of the centre cell needed to be significantly reduced to implement more centres on-chip. The necessary modifications made for the PAR chip are considered in the following sub-sections.

6.1.1. Centre Circuit Modifications

Three main modifications were made to the centre cell design.

i) **Local Compensation in the Distance Circuit.**

The experiments on the RHO chip had indicated that *local,* as opposed to *global,* compensation circuitry was required for the distance circuits to function correctly.

ii) **Removal of the N-wells for the Ratioed Transistor Pairs.**

The requirement for separate N-wells for each ratioed pair of transistors was questioned. Indeed, since local compensation was also to be adopted, three pairs of ratioed transistors would now be required per cell and the removal of the wells would save silicon real estate.

iii) **Use of a Two-Transistor Non-Linearity.**

In Chapter 5, the non-linearity circuit was initially presented as a two-transistor device, although a four transistor variation was fabricated on the RHO chip to double the dynamic range of the circuit output. Inspection of the four transistor characteristic (both measured and simulated) showed it covered all of input space for many choices of V_{width}. Since these RBFs should be *locally responsive,* it is unlikely that such large widths will be desirable. Furthermore, the four-transistor non-linearity was deemed to be too complicated since a suitable non-linear circuit could be designed using only two transistors.

Thus a series of HSPICE simulations were performed to investigate:

- the effect of the N-wells and local compensation circuit on the distance circuit performance
- the effect of the local compensation circuit on the shape of the non-linearity produced by a two-transistor circuit.

6.1.1.1. Distance Circuit HSPICE Simulations

The purpose of the compensation circuity is to remove offset currents in the distance circuits, ensuring I_{dist} = 0A when $V_{in} = V_{centre}$. Thus several HSPICE simulations were performed to investigate the effect of different compensation circuit configurations on the common mode current in the distance circuit. For these simulations, distance circuits with and without separate N-wells for the ratioed pairs were simulated for the following configurations:

i) a distance circuit with no compensation circuitry

ii) a distance circuit with a compensation circuit whose ratioed pair transistor gates were connected to ground

iii) a distance circuit with a compensation circuit whose ratioed pair transistor gates connected to V_{centre}.

The results of these simulations are shown in Figures 6.1(a)-(d) and from these graphs, it is clear that very small common-mode currents result when **no** separate N-wells are used and the compensation circuit has the gates of its ratioed pair transistors tied to V_{centre}.

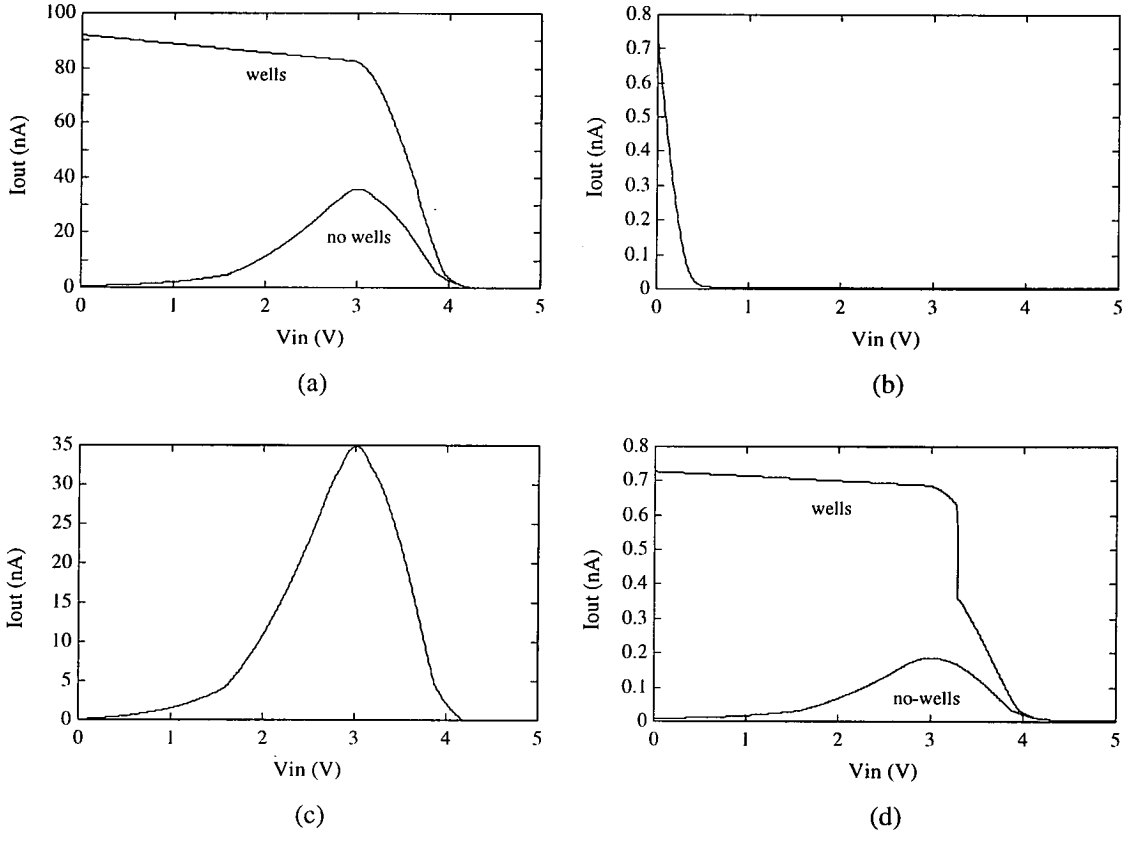


Figure 6.1 - HSPICE simulation results showing the common-mode output current for a distance circuit with (a) no compensation circuit, (b) separate N-wells and a compensation circuit whose gates are tied to 0V, (c) no separate N-wells and a compensation circuit whose gates are tied to 0V and (d) a compensation circuit whose gates are tied to V_{centre}

As can be seen from Figure 6.1(d)[6], though, when the compensation circuit is used with a distance circuit having no N-wells, the common-mode current varies with the

[6] The discontinuity in Figure 6.1(d) for the circuit with separate N-wells is believed to be due to the simulator having difficulty modelling the transition region between strong inversion and weak inversion MOSFET operation.

applied common-mode voltage. This produces slight variations in both the output current from the circuit, Figure 6.2(a), and the centre circuit output voltage, Figure 6.2(b). These differences are due to the non-zero, bulk-source voltages produced when V_{centre} = 0V and V_{centre} = 3V, as discussed in Section 5.2.3.

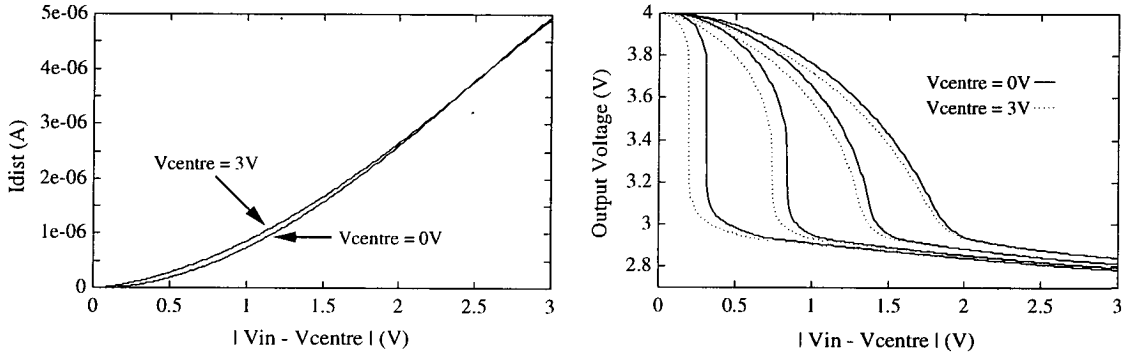


Figure 6.2 - The asymmetry in the distance circuit (a) I_{dist} current and (b) output voltage for compensation circuit configuration iii) without N-wells

Having established that separate N-wells were not essential to the operation of the distance circuit, they were not used for the centre circuits on the PAR chip.

6.1.1.2. Transistor Non-Linearity Circuit Simulations

Next the requirement for a compensation circuit was questioned: as shown in Figures 6.3(a) and (b), very little difference exists between the I_{dist} versus $|V_{in} - V_{centre}|$ curves for distance circuits with and without compensation, when no N-wells are used.

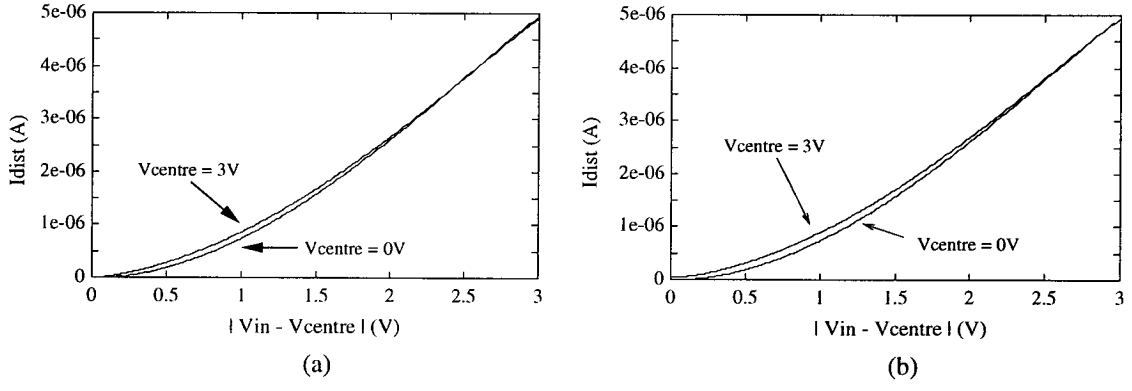


Figure 6.3 - Comparison of the output current, I_{dist}, from the distance circuit both (a) with and (b) without a compensation circuit

However, simulations of the complete centre circuit (distance circuit plus two-transistor non-linearity) indicated that a compensation circuit is definitely required if the non-linearity is to be reproduced for the valid range of V_{centre}, Figures 6.4-6.6.

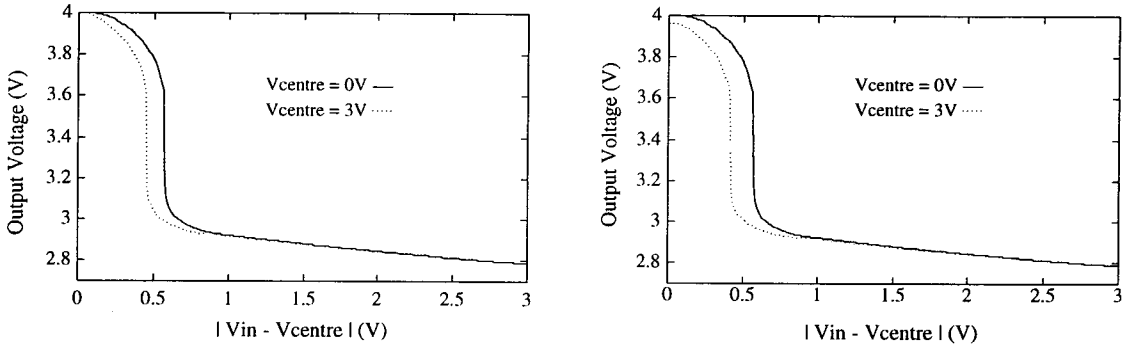


Figure 6.4 - Comparison of distance circuit output voltage, V_{out}, both (a) with and (b) without a compensation circuit. V_{width} = 2.25V

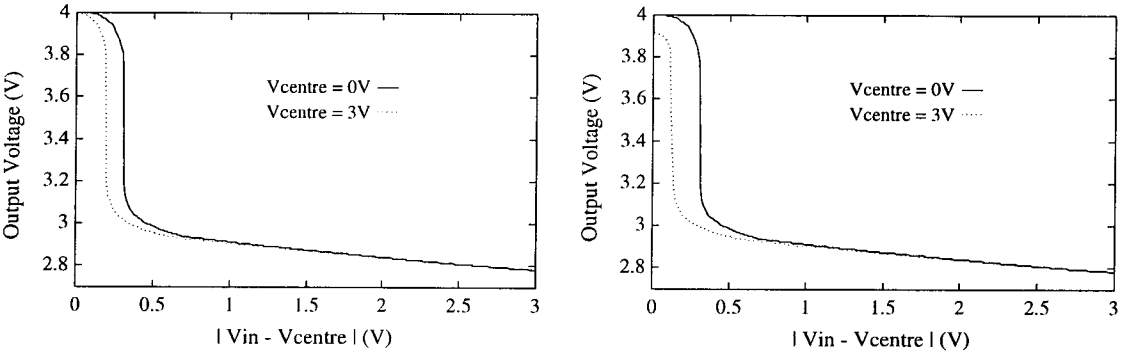


Figure 6.5 - Comparison of distance circuit output voltage, V_{out}, both (a) with and (b) without a compensation circuit. V_{width} = 2.5V

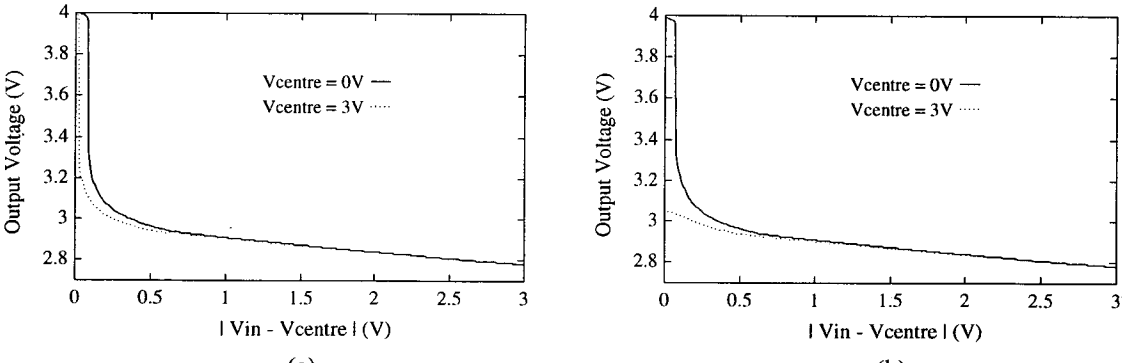


Figure 6.6 - Comparison of distance circuit output voltage, V_{out}, both (a) with and (b) without a compensation circuit. V_{width} = 2.75V

6.1.1.3. Centre Circuit Layout

It was concluded from the HSPICE simulations that the N-wells could be removed from the distance circuit if, and only if, a local compensation circuit was included. Thus new layout was created for the centre circuit, with the new cell occupying 248μm by 168μm of silicon real estate. The schematic diagram and layout for the circuit are shown in Figure 6.7 and Figure 6.8, respectively.

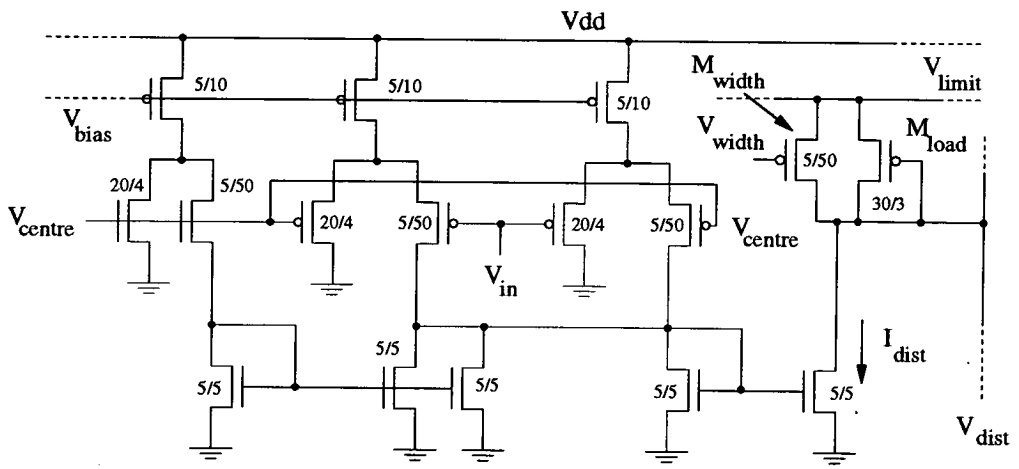


Figure 6.7 - The Centre Circuit as fabricated on the PAR chip

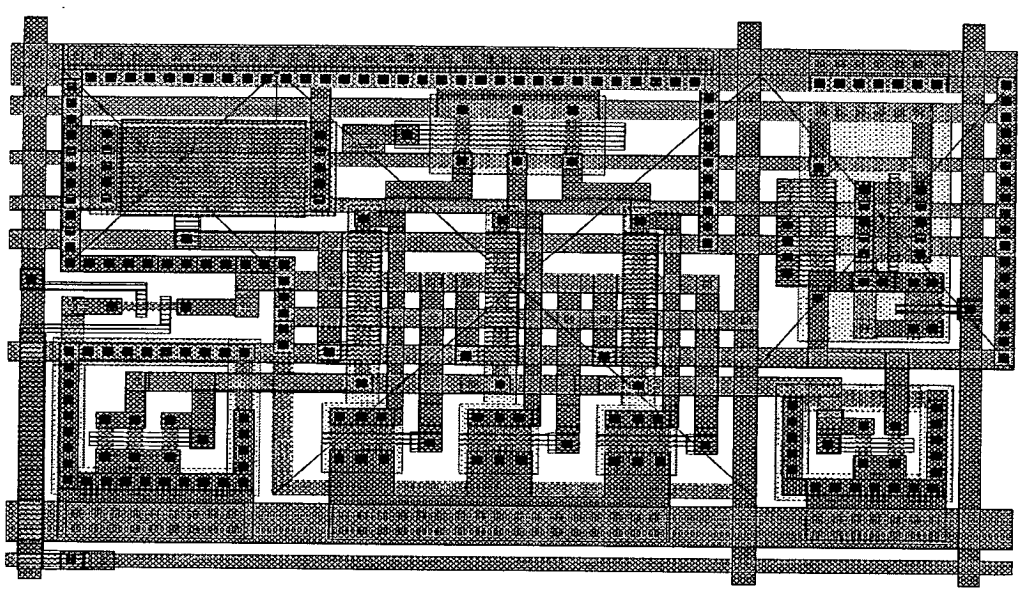


Figure 6.8 - The Centre Circuit Layout for the PAR chip

6.1.2. Two-Transistor Centre Circuit Properties

Previously, the two transistor non-linearity circuit has simply been considered as a circuit that produces a non-linear monotonic response when a current is drawn through it. However, by analysing the circuit, some desirable first-order properties emerge.

As already stated in Chapter 5, the two-transistor circuit, Figure 6.9(a), relies on transistor M_{width} operating in its linear region, with M_{load} used to smooth off the tail of the curve and clamp V_{out} as M_{width} saturates. For the purposes of this analysis, consider that M_{load} is **off** and M_{width} is biased into its linear region. The circuit to be analysed is therefore shown in Figure 6.9(b).

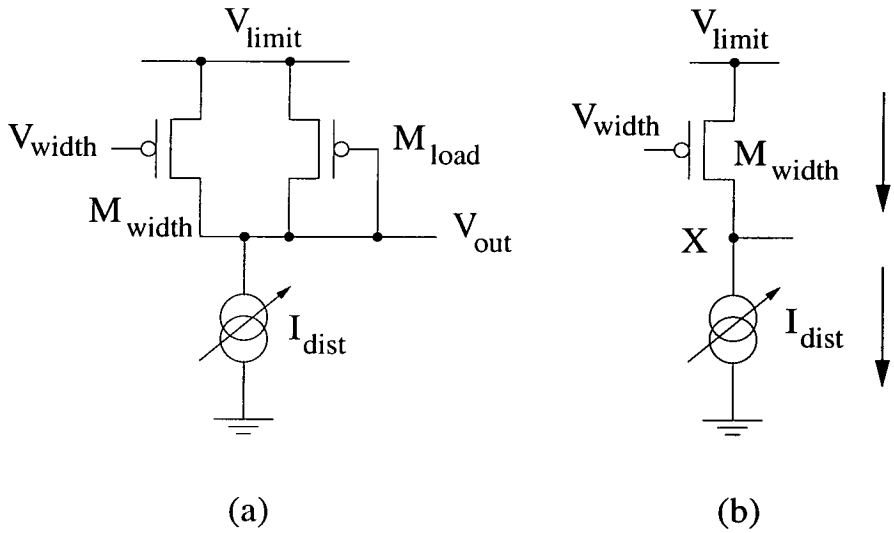


Figure 6.9 - Schematic diagrams of (a) the complete centre circuit and (b) the circuit analysed in this section

Applying Kirchoff's Current Law to node X yields equation 6.1.

$$I_{width} = I_{dist} \tag{6.1}$$

where I_{width} is the current flowing from source to drain in M_{width} and I_{dist} is the current produced by the distance circuit.

Currents I_{width} and I_{dist} are defined by equations 6.2 and 6.3 respectively.

$$\begin{aligned} I_{width} &= \beta\left((V_{gs} - V_T)V_{ds} - \frac{V_{ds}^2}{2}\right) \\ &= \beta_{width}\left((V_{width} - V_{limit} - V_T)(V_{out} - V_{limit}) - \frac{(V_{out} - V_{limit})^2}{2}\right) \end{aligned} \tag{6.2}$$

$$I_{dist} = \frac{\beta_{narrow}}{2}(|V_{in} - V_{centre}|)^2 + \beta_{narrow}\left(\frac{2I_{wide}}{\beta_{wide}}\right)^{\frac{1}{2}}(|V_{in} - V_{centre}|) \quad (6.3)$$

Rearranging equation 6.2 gives:

$$I_{width} = \beta_{width}\left[V_{out}V_{width} - V_T V_{out} - V_{limit}V_{width} + V_T V_{limit}\right] + \frac{\beta_{width}}{2}(V_{limit}^2 - V_{out}^2) \quad (6.4)$$

Expanding this equation and using the relationship given by equation 6.1 yields:

$$2I_{dist} = 2\beta_{width}V_{out}(V_{width} - V_T) - 2\beta_{width}V_{limit}(V_{width} - V_T) + \beta_{width}V_{limit}^2 - \beta_{width}V_{out}^2 \quad (6.5)$$

Dividing equation 6.5 by β_{width} and rearranging gives:

$$V_{out}^2 - 2V_{out}(V_{width} - V_T) + 2V_{limit}(V_{width} - V_T) - V_{limit}^2 + 2\left(\frac{I_{dist}}{\beta_{width}}\right) = 0 \quad (6.6)$$

Combining equations 6.3 and 6.6 produces the following quadratic expression in terms of V_{out}.

$$V_{out}^2 - 2V_{out}(V_{width} - V_T) + 2V_{limit}(V_{width} - V_T) - V_{limit}^2 + \left(\frac{\beta_{narrow}}{\beta_{width}}\right)(|V_{in} - V_{centre}|)^2 + 2\left(\frac{\beta_{narrow}}{\beta_{width}}\right)\left(\frac{2I_{wide}}{\beta_{wide}}\right)^{\frac{1}{2}}(|V_{in} - V_{centre}|) = 0 \quad (6.7)$$

Therefore, to a first order, it is possible to relate the output voltage from the centre circuit to the other circuit parameters **before** M_{width} saturates.

The final equation can be solved for V_{out} using the standard equation for finding the roots of a general quadratic equation of the form $ax^2 + bx + c = 0$, equation 6.8.

$$x = \frac{-b \pm (b^2 - 4ac)^{\frac{1}{2}}}{2a} \quad (6.8)$$

where

$$a = 1$$

$$b = -2\,(V_{width} - V_T)$$

$$c = 2V_{limit}(V_{width} - V_T) - V_{limit}^2 + \frac{\beta_{narrow}}{\beta_{width}}(|V_{in} - V_{centre}|)^2 + 2\frac{\beta_{narrow}}{\beta_{width}}\left(\frac{2I_{wide}}{\beta_{wide}}\right)^{\frac{1}{2}}(|V_{in} - V_{centre}|)$$

Whilst it is unlikely that this equation will ever be used to find specific values of V_{out} (HSPICE simulations will give more accurate estimations if high level models are used), two important properties emerge from equation 6.7. These are:

- the $\frac{\beta_{narrow}}{\beta_{width}}$ β-ratios which help to make the circuit more tolerant to process variations
- the $(V_{width} - V_T)$ terms, that allow variations in the threshold voltages of M_{width} to be compensated for by tuning the V_{width} values for individual centres, ie a variation in the threshold voltage of ΔV_T can be compensated for by a change in V_{width} of ΔV_{width}.

Thus, as this analysis shows, the two-transistor non-linearity circuit has more desirable properties than first realised.

6.1.3. DYMPLES Circuit Re-Design

As already mentioned, the original DYMPLES circuit was designed for ES2's 1.5μm digital CMOS process. However, this process was withdrawn during the course of this project and the RHO and PAR chips were designed for MIETEC's 2.4μm analogue CMOS process. In order to implement DYMPLES multipliers on the PAR chip, the DYMPLES circuit had to be re-designed, and new layout created.

Fortunately, the DYMPLES circuit was designed to be conceptually simple to ensure it was easy to transfer to new processes. Since the synapse only consists of current mirrors, switches, capacitors and digital logic, subsequent re-designs were intended to be straight-forward.

For the initial simulations of the circuit with the HSPICE transistor models for the MIETEC 2.4μm process, the original width to length ratio of the current mirror transistors was retained, the transistor sizes were scaled accordingly and the original specifications of the circuit (voltage change, capacitor size, maximum discharge time etc.) were preserved. The results from the simulations of the circuit for the *fast,*

slow and *typical* transistor models are shown in Figure 6.10 and illustrate the excellent linearity obtained from the re-sized synapse. Again all 3 characteristics overlap. Thus the re-design of the DYMPLES circuitry proved to be trivial.

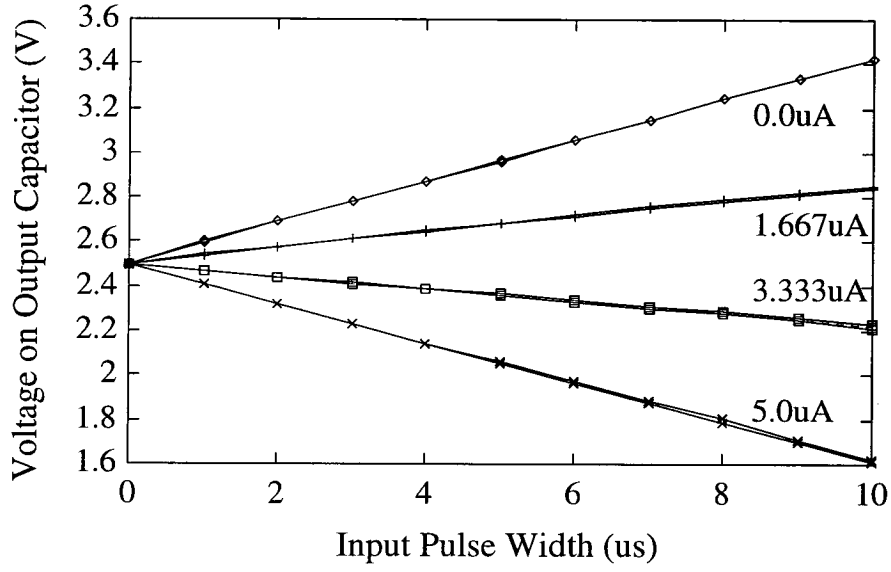


Figure 6.10 - Response of DYMPLES Circuit for the MIETEC 2.4um Process Using *Fast*, *Slow* and *Typical* Transistor Models for 4 different Synaptic Weights

One consequence of moving to the larger geometry process, however, was the inevitable increase in the area of the synapse - the new synapse occupies 404μm by 168μm in the new process, as opposed to 220μm by 220μm in the ES2 1.5μm process. Although this area increase is unavoidable (unless the circuit is re-designed) it is an insignificant detail in the development of the PAR chip, which is, after all, simply the final concept-proving demonstrator.

The schematic diagram and layout for the circuit fabricated on the PAR chip are shown in Figure 6.11 and Figure 6.12, respectively.

6.2. System Level Considerations

Although they were complex designs, the DYMPLES and RHO chips were test chips of limited functionality, fabricated primarily to test circuit prototypes and prove novel ideas as opposed to forming part of a complete RBF chip. Since the PAR chip was intended to be a fully functioning, pulsed RBF demonstrator, more consideration had to be devoted to the system level aspects of the design. These issues are discussed in the following sections.

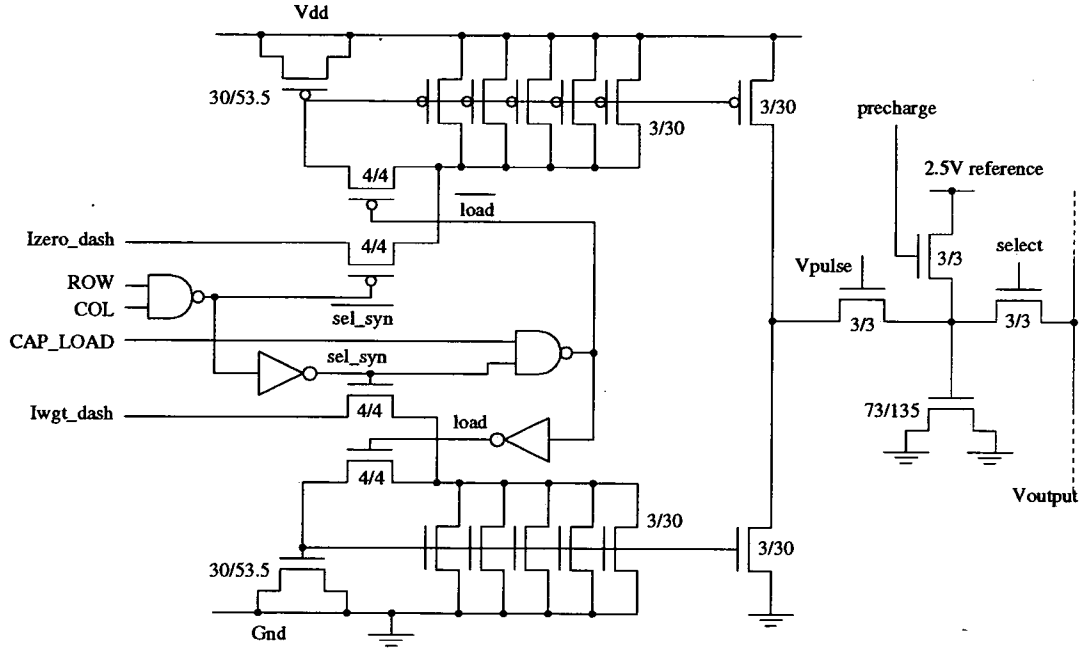


Figure 6.11 - Schematic Diagram of the PAR chip's DYMPLES circuit

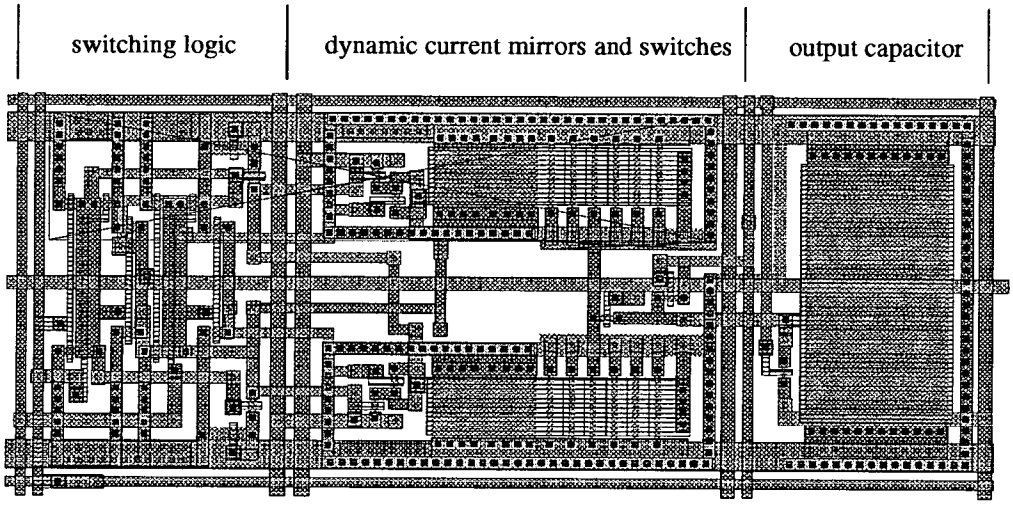


Figure 6.12 - Layout for the PAR chip's DYMPLES circuit

6.2.1. Network Size - How big should the PAR chip be ?

The size of a neural network is determined by the number of input, hidden and output units it possesses. Although there is no merit in producing large, generic neural network chips simply for the sake of it, the ideal demonstrator should be non-trivial and should be capable of indicating the potential of a larger system. Thus enough inputs, centres and outputs had to be chosen so that the chip had the flexibility to be applied

to a range of non-trivial problems, whilst not exceeding the available silicon area.

For the PAR chip, it was therefore necessary to compromise between circuit size, network complexity and the available silicon area. From a consideration of these three criteria, PAR was designed with 8 analog inputs, 16 centres and 4 digital PWM outputs.

PAR was designed with this specification for a number of reasons.

- Designing a chip whose inputs, centres and outputs are binary powers eases the interfacing of the chip to standard logic blocks (eg the 74000 TTL series), hence simplifying the design of the test board.
- Twice as many inputs as outputs were used because the hidden layer cell was smaller than the output layer cell and it was not envisaged that the chip would be applied to problems with more than a few classes.
- The chip was designed with 16 centres simply because there was insufficient silicon to implement 32 centres without significantly re-designing the centre cell - a task not undertaken for the reasons discussed in Section 6.1.

In fact, whilst the chip has been designed with 16 centres, it was necessary to use one to generate the bias term pulse for the RBF, equation 2.15. Thus, the centre cells at the top of the design were adapted such that V_{in} was applied to both gates in all three ratioed pairs, ensuring that the output pulse from that centre is always of maximum width. Therefore, in reality, the chip has 15 centres whose position and width can be altered with the sixteenth centre being used to produce a fixed width pulse.

6.2.2. On-chip DAC Precision

As described in Chapter 4, the DYMPLES circuit had shown good performance with a 4-bit on-chip current DAC. However, given that it is possible to achieve at least 8-bit precision in analog VLSI and that recent results have indicated that MLPs need at least 6-bit to 8-bit precision in the forward pass [148], it was decided to implement a simple 8-bit DAC on the PAR chip.

No quantitative analysis has been conducted for the precision required in RBF networks as yet, and it can be argued that, assuming that the RBF hidden layer's non-linear expansion sufficiently separates the classes in classification space, it is unlikely that the linear output layer units would require the same precision to define the hyperplanes as the non-linear output units of a corresponding MLP applied to the same problem. Implementing an 8-bit DAC on the PAR chip provided it with the

flexibility to have the same precision as an MLP, if required, whilst also allowing the precision requirement of the output layer to be investigated in hardware.

6.2.2.1. PAR Chip DAC Design

The time constraint for producing the PAR chip meant that the on-chip current DAC had to be simple and straightforward to design, easy to lay out and have a creditable level of performance. To fulfil these requirements, the DAC was implemented as two 4-bit DACs connected by a 16:1 attenuating current mirror, Figure 6.13.

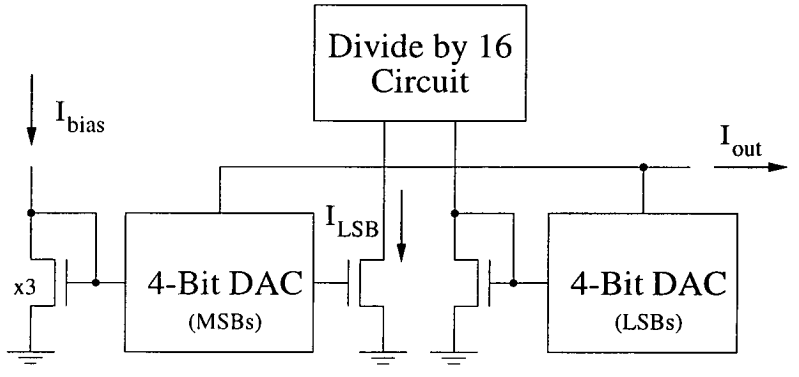


Figure 6.13 - PAR Chip 8-Bit DAC Schematic

By laying out the DAC so that its mirroring transistors were again rotationally symmetric, it was hoped to reduce the effects of on-chip gradients and variations, thus preventing degradation of the DAC's performance. In addition, long, thin mirroring transistors were again used within the DAC to increase the output impedance and make the currents less susceptible to corruption.

Whilst it is acknowledged that this design is far from optimal, HSPICE simulations indicated that it was approximately linear and monotonic, Figure 6.14, and thus sufficient to fulfil the requirements for the DAC required on the PAR chip.

In addition to an on-chip current DAC, the PAR chip also required a Zero Current Mirror (ZCM), Section 4.7.2, to generate the I_{zero} currents for the PMOS transistors in each DYMPLES multiplier. Again the ZCM was designed as a 2:5 amplifying current mirror biased with an off-chip current.

The final layout for the DAC and ZCM is shown in Figure 6.15 and the cell shown occupies a silicon area of 925.2μm by 499.2μm.

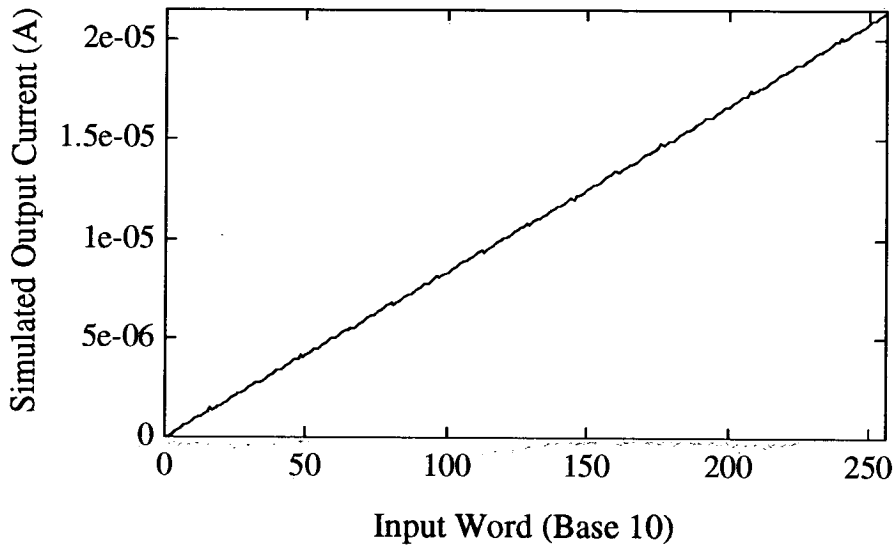


Figure 6.14 - Simulated 8-Bit DAC Characteristic

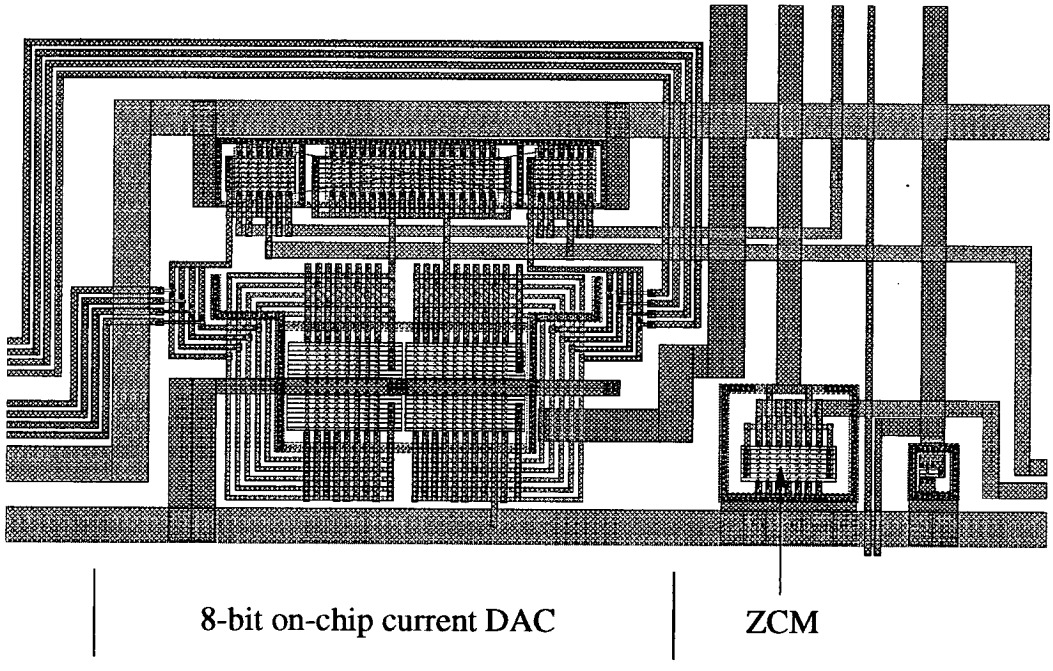


Figure 6.15 - PAR Chip 8-Bit DAC and ZCM Layout

6.2.3. PAR Chip Refresh Scheme

Since the neural parameters on the PAR chip are dynamically stored on capacitors, they are subject to decay from leakage currents and so must be periodically refreshed. This section describes the PAR chip's refresh scheme, concentrating on

how the refresh rate was determined and how the system was actually implemented.

6.2.3.1. Refresh Rate Determination

The refresh rate of a chip is determined from a compromise between the tolerable parameter corruption due to leakage currents and the tolerable performance corruption due to capacitively coupled noise.

Assuming that the neural circuitry is expected to obtain a precision of N bits, then the refresh rate must be sufficient to ensure that the leakage currents do not discharge the storage capacitor by a voltage equivalent to half of the storage requirement for the least significant bit. Thus fast refresh rates are desirable. However, slower refresh rates mean fewer clock edges, lower levels of capacitive coupling and hence lower levels of clock-induced noise within the chip.

As can be seen from the graphs in Figure 6.16, the range of V_{gate} voltages in the DYMPLES circuits is significantly smaller than the range of V_{centre} voltages in the centre circuits. Further, since the globally distributed currents are used to charge the C_{gate} capacitances, current mode loading can be significantly slower than voltage mode loading if small currents are used. Thus the DYMPLES circuit was used to determine the refresh rate of the PAR chip.

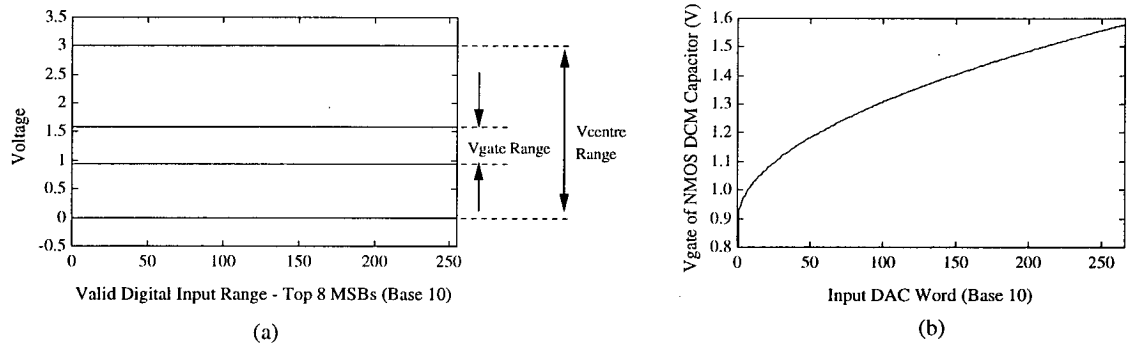


Figure 6.16 - Voltage Ranges for the Hidden and Output Circuits

From an HSPICE simulation of the NMOS DCM, it was established that the stored voltage was a non-linear function of the loaded current and the minimum voltage difference between two consecutive 8-bit weight values was 1mV, Figure 6.16(b).(The minimum voltage difference (at 8-bit precision) between V_{in} and V_{centre} for this chip is approx. 11.7mV). Thus the refresh rate must be sufficiently high to ensure the leakage currents do not discharge C_{gate} by more than 0.5mV between refresh cycles.

By assuming that the leakage currents are constant, the periodic refresh time can be determined using equation 6.9.

$$I_{leakage}\Delta t = C_{gate}\Delta V \tag{6.9}$$

where $I_{leakage}$ is the total constant leakage current due to subthreshold conduction and reverse-biased diodes, and Δt is the time for a voltage drop of ΔV to occur on C_{gate}, the capacitor storing voltage V_{gate}.

In order to calculate an approximate value for Δt for the minimum refresh period, the value of the leakage currents in the MIETEC process had to be determined. This was achieved using the measurement transistors on the RHO chip to monitor the variation, over time, of the V_{centre} voltages dynamically stored on the C_{centre} capacitors of the RHO chip. By setting V_{centre} to 3V and V_{in} to 0V, it was possible to determine the rate of decay of V_{centre} from the variation in the measured Op Amp output voltage, V_{out}, Figure 6.17.

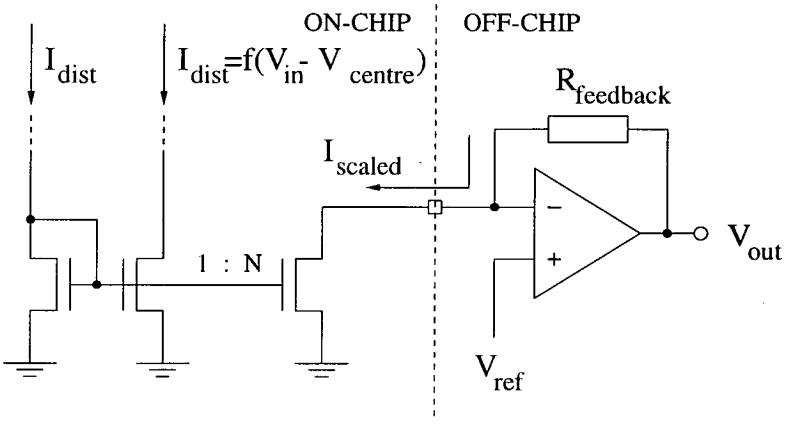


Figure 6.17 - Schematic Diagram of $I_{leakage}$ Determination Circuit

Figures 6.18(a) and (b) show the change in V_{out} with time for two circuits on two different RHO chips. As can be seen, the variation is quadratic and identical to the left hand side of the measured distance circuit curves shown in Figure 5.17. So, since V_{out} - and hence I_{dist} - is a quadratic function of $|V_{in} - V_{centre}|$, it was concluded that V_{centre} is decaying *linearly* over time, showing $I_{leakage}$ is indeed constant for the MIETEC process.

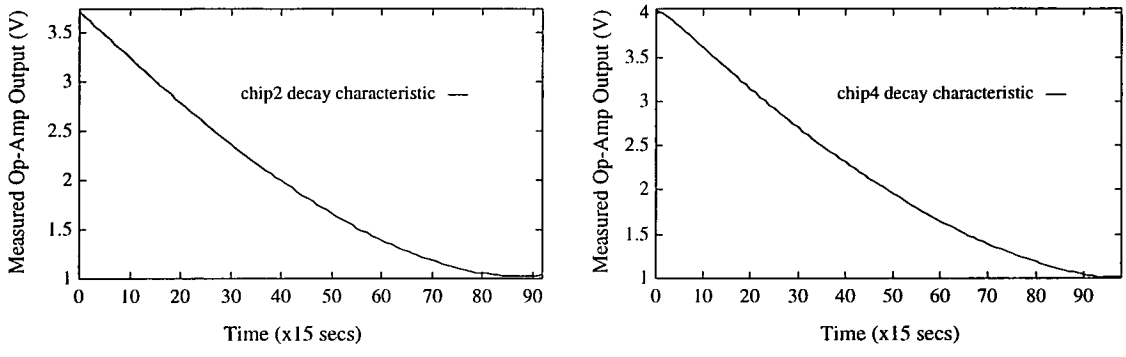


Figure 6.18 - Measured Decay Characteristic for 2 Distance Circuits on 2 RHO Chips

By substituting the values for the known and determined parameters into equation 6.9, the estimated leakage currents for the two chips were determined, Table 6.1. The leakage currents for both chips were calculated to be less than 5fA.

Chip	C_{centre}	ΔV	$\Delta t_{discharge}$	$I_{leakage}$
2	2.01pF	3V	1305s	4.621fA
4	2.01pF	3V	1425s	4.232fA

Table 6.1 - Parameter Values for $I_{leakage}$ Determination.

Having empirically determined $I_{leakage}$, it was now possible to determine the maximum allowable period between refreshes such that the voltages on the C_{gate} capacitors of the NMOS DYMPLES DCMs did not fall by more than 0.5mV. For the purposes of this calculation, leakage currents of **10fA** were assumed, allowing a safety margin of over 100%, and the C_{gate} capacitances were assumed to be 1.0pF.

$I_{leakage}$	C_{gate}	$\Delta V_{min\_discharge}$	Δt_{max_period}
10fA	1.0pF	0.5mV	50ms

Table 6.2 - Parameter Values used for Minimum Refresh Rate Determination.

By again substituting the appropriate parameter values into equation 6.1, Δt_{max_period} was calculated as 50ms, Table 6.2. Thus the refresh rate for the PAR chip can be as low as 20Hz for 8-bit resolution. Again, however, it was decided to have a large safety margin in the design and the initial refresh period was selected as 1.28ms, or approximately 800Hz. One benefit of having such a short refresh period is that the precision of V_{in} and V_{centre} can be increased to 12-bits, **with respect to the leakage**

currents alone, since the maximum refresh period required for 1.0pF C_{centre} capacitors at 12-bit resolution is 36ms[7].

6.2.3.2. Refresh Scheme Implementation

The off-chip refresh scheme for the PAR chip is shown in Figure 6.19.

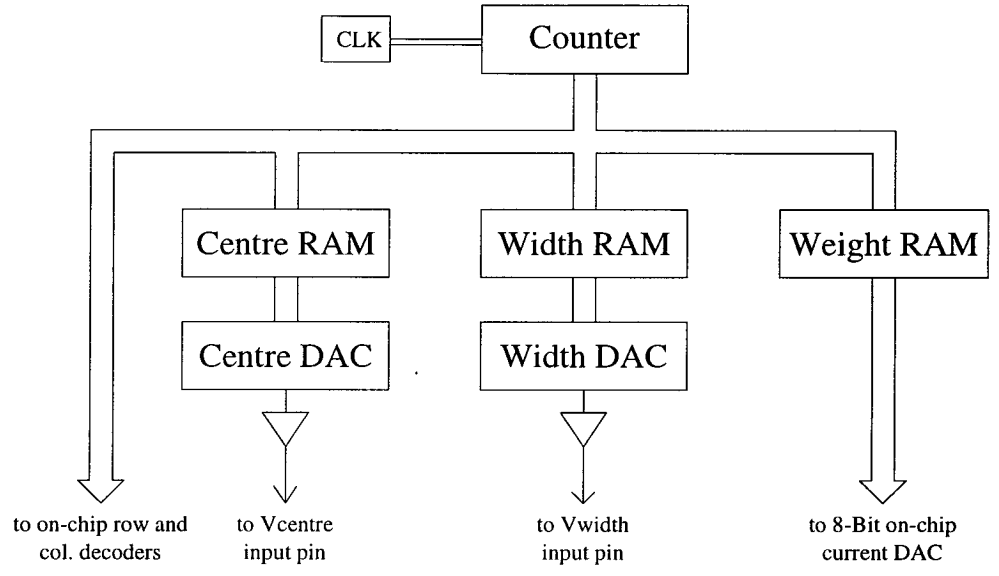


Figure 6.19 - Off-Chip Refresh Scheme for the PAR Chip

Centre, width and synaptic weight values are generated and loaded simultaneously as determined by an off-chip counter. Since there are approximately twice as many centre cells as output cells, the centre cells are addressed for half as long as the output cells.

The cells on the PAR chip are refreshed sequentially using on-chip row and column decoders (composed of NOR gates) to address each one in turn, while additional off-chip circuitry produces the appropriate neural parameter for that cell. This additional off-chip circuitry consists of the counter, which simultaneously addresses the neural parameter RAM chips in addition to driving the row and column decoders and, in the case of the V_{centre} and V_{width} values, voltage DACs to convert the digital word to an analogue voltage. Thus, the appropriate centre, width and synaptic weight values are generated whilst the cells to which they will be loaded are being addressed.

[7] This calculation does not take into account other possible corruption mechanisms within the hardware, such as charge injection, noise, mismatch, temperature variations etc., which will conspire to reduce the actual achievable precision to less than the expected 12-bits.

A potential problem exists with such a refresh mechanism, though: delays through the off-chip circuitry and on-chip clock skew are likely to produce over-lapping clock edges, Figure 6.20, possibly causing two adjacent cells to be addressed concurrently **if the cell addressing signals alone are used to initiate cell loading.** Concurrent loading could result in cells being loaded with the wrong value, or correctly loaded values being partially corrupted, causing significant degradations in the performance of the hardware.

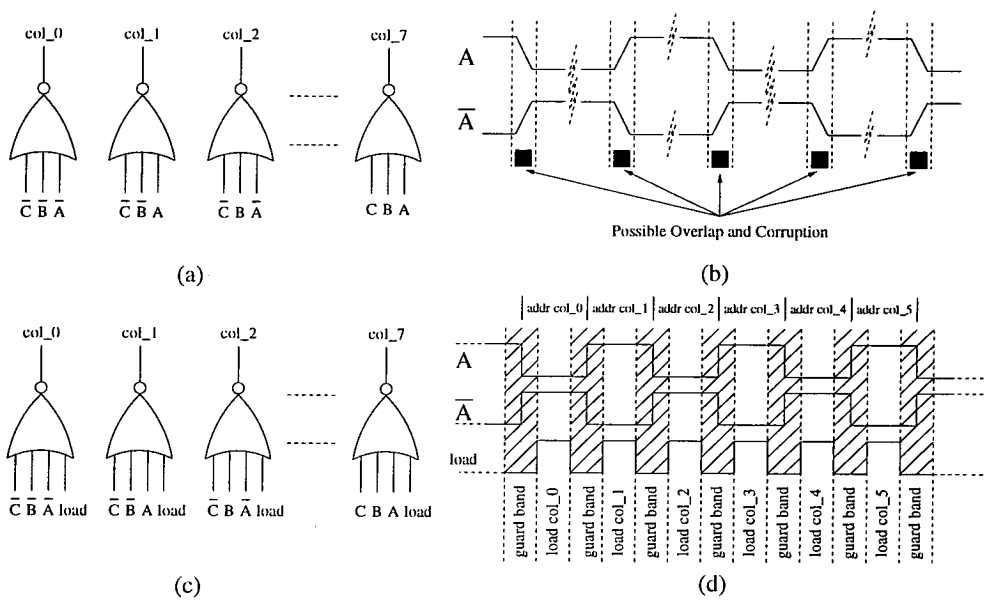


Figure 6.20 - (a) A possible implementation of a NOR gate decoder along with (b) an indication of the overlapping clock edges which are likely to affect this decoder. Also shown is an improved decoder (c) and (d) an illustration of the guard bands that the new signalling scheme introduces

To eradicate this problem, signals derived by exclusive-ORing some of the higher frequency counter bits were used to initiate the loading of the centre, width and synaptic weight values. The use of these loading signals creates *guard bands*, allowing the circuitry in Figure 6.19 to settle after being addressed and before the cell capacitors are loaded, thereby removing any possibility of concurrent cell loading. The price for this added feature is one more input to each column decoder NOR gate; the benefit is the knowledge that time delays and clock skews should not now pose a problem assuming the guard bands are long enough. The addressing time, loading time and guard band widths for each cell in the hidden and output layers, with the 1.28ms chip refresh time, are presented in Table 6.3.

Layer	Addressing	Pre-Load Guard Band	Loading	Post-Load Guard Band
Hidden	10 μs	2.5 μs	5 μs	2.5 μs
Output	20 μs	5 μs	10 μs	5 μs

Table 6.3 - Cell Addressing Times, Loading Times and Guard Band Times for the PAR Chip Refresh Scheme

6.2.4. Width Storage Scheme

Results from the RHO chip had indicated that the width of the centre non-linearity could vary from centre to centre and from chip to chip. After conducting several tests, this problem was deemed to be due to some on-chip alteration or corruption of the V_{width} values. Thus, to give the PAR chip added flexibility, a width storage capacitor was allocated to each centre. Each centre can therefore have an individually tuned width, allowing any on-chip variations to be accounted for during chip-in-the-loop training.

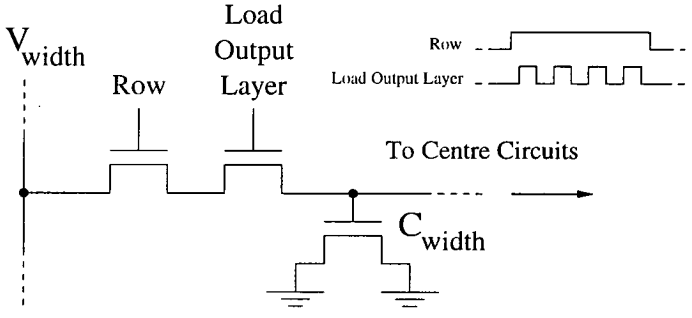


Figure 6.21 - Width Storage Scheme Schematic

From the implementation aspect, each C_{width} capacitor is addressed using the row decoder and loaded using the load signal for the output layer, Figure 6.21. As a result, the width storage capacitor is actually loaded four times every time it is addressed.

6.3. PAR Chip Floorplan

The individual cells for the PAR chip were laid-out, netlisted and their functional operation checked before they were combined to form the final pulsed RBF demonstrator chip. The floor plan for the chip is shown in Figure 6.22, whilst Figure 6.23 shows the chip plot of the fabricated design. The PAR chip was implemented on a silicon die measuring 6.5mm by 4.8mm.

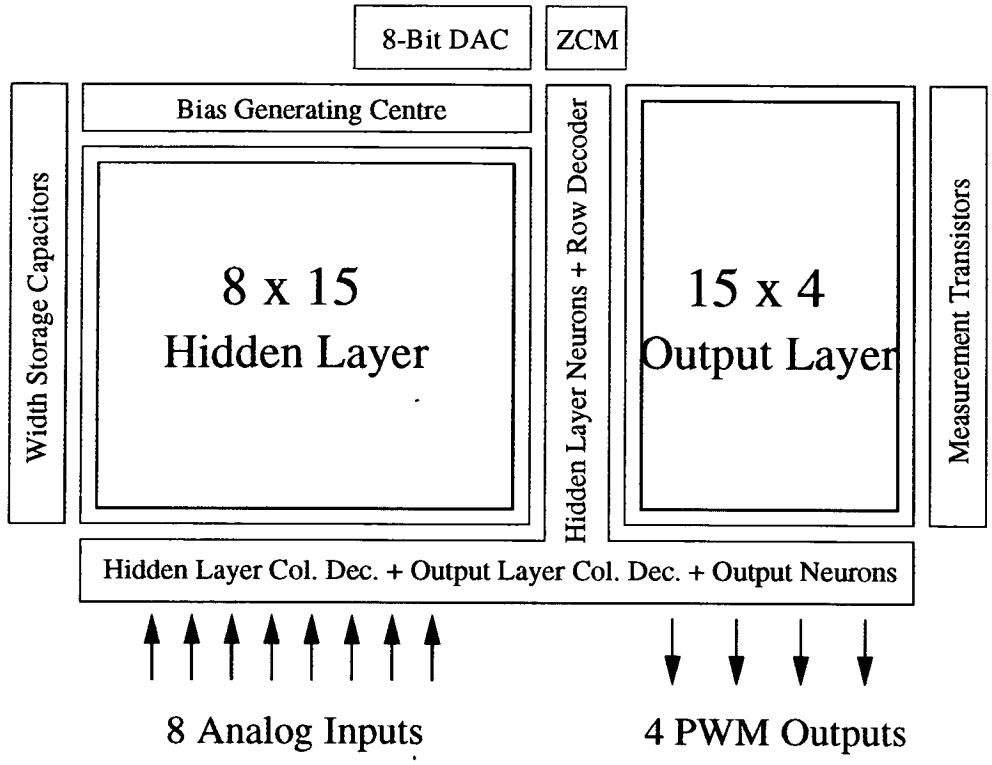


Figure 6.22 - PAR Chip Floorplan

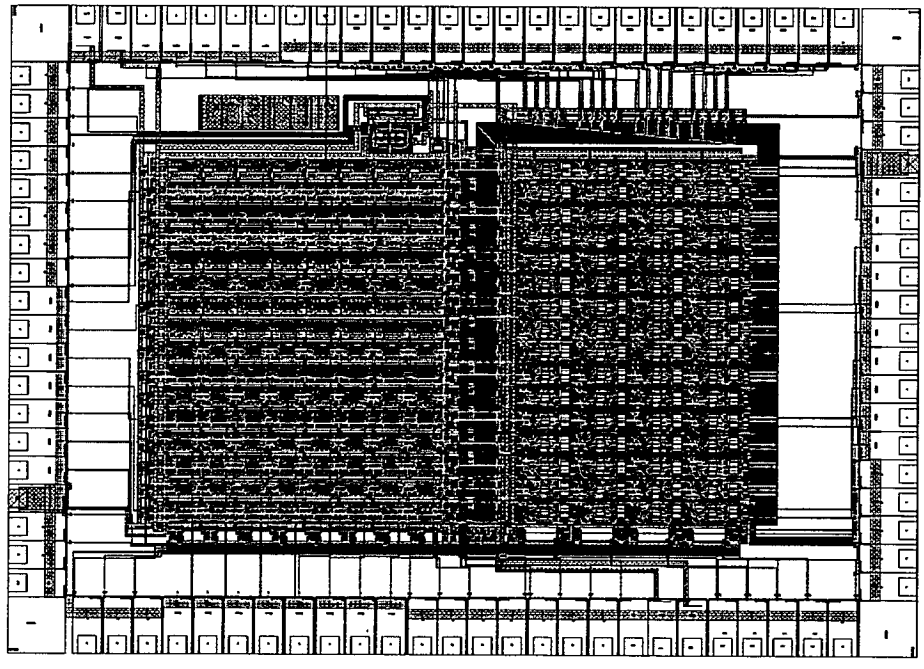


Figure 6.23 - PAR Chip Layout

6.4. Chip Testing

A total of ten chips were returned from the silicon foundry and the PAR Development Board (Appendix A) was built to allow them to be correctly biased and tested automatically.

In order to assess the operation of the fabricated circuits, including the modifications to the centre circuits, sufficient provision had been made to allow the separate chip components to be tested independently. In addition to including outputs to test the operation of the on-chip digital inverters, measurement transistors were again included for testing and assessing the functionality of the DAC, DCMs and distance circuits. As well as the DAC output, four distance circuits, four PMOS DCMs and four NMOS DCMs could be characterised per chip.

Further, it was intended that the chip be configured to allow the hidden and output layers to be tested and characterised separately, in addition to operating the chip with the hidden layer directly connected to the output layer. To achieve this, the switching arrangement in Figure 6.24 was devised, allowing the chip to be configured as desired using only two off-chip control bits (**Connect** and **Enable**).

As with the previous chips, the experimental procedures for recording the results in the following sections are detailed in Appendix A.

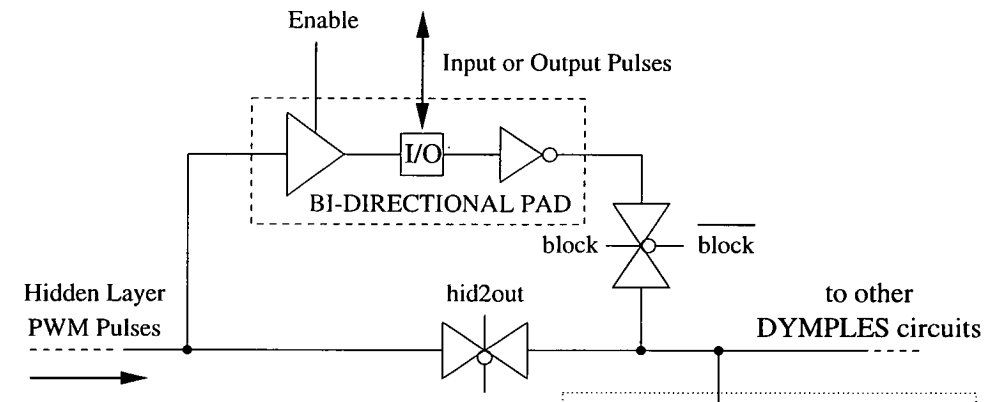
6.4.1. Test Procedure

Since the aim of this work was to develop a pulsed RBF demonstrator, it was necessary to check the operation of the circuits to verify their operation and investigate what differences, if any, existed between different circuit instances both across-chip and between chips.

However, the DACs, DCMs and distance circuits plus the hidden and output layers on all the chips were only rigourously tested once or twice for two main reasons:

i) the results obtained from the tests proved the circuits worked correctly and consistently, highlighted the expected deficiencies in performance and displayed no unexpected ones

ii) the time available was limited and there was judged to be little point, given the good quality of the results obtained, in testing each set of circuits, on every chip, five or ten times.



	Control Signals		On-chip Signals	
Mode	Connect	Enable	hid2out	block
Test Op. Layer	0	0	0	1
Test Hid. Layer	0	1	0	0
Normal RBF	1	0	1	0
DC Tests	1	1	0	0

Figure 6.24 - Switching Arrangement to Facilitate the Testing of the Hidden and Output Layers

Therefore the results presented in the remainder of this chapter are typical of the performance for all the chips, except where stated otherwise, but were generated from a single measurement run on one of the chips. The one exception are the results for the current DACs: the DAC responses for all 10 chips were measured and the presented results indicate the mean performance and ±3 standard deviation error bounds for **all** the DACs.

No numerical values for the statistical quantities are quoted, though, since they are almost meaningless for such a small number of results and this thesis is more concerned with the performance of the circuits **within** a trained RBF network, rather than how well matched the circuits are. Thus a qualitative rather than a quantitative presentation of the measured results suffices.

It is also believed that chip-in-the-loop training will account for across-chip and inter-chip differences between the circuits, rendering any differences insignificant to the operation of the trained networks. However, for the sake of completeness, each results section describes the differences noted in circuit operation.

As in Chapters 4 and 5, the input and output pulse widths in the following results graphs are presented in terms of *RAM Locations* instead of time periods and the V_{in}

and V_{centre} voltages are represented by the input digital words to the 12-bit off-chip voltage DACs.

6.4.2. PAR DAC Characteristic

The performance of the DAC on all ten PAR chips was measured. Each DAC was loaded with all 256 valid digital words and the output voltage from the off-chip Op-Amp recorded. From these recorded voltages, the test currents flowing out the DAC were calculated. All the DACs produced very similar characteristics and Figure 6.25 shows the Output Test Current vs. Input DAC Word characteristic averaged over all ten DACs. Also shown in the figure are the error bounds representing ±3 standard deviations of the results. The measured results have been processed such that all the DAC characteristics pass through the origin.

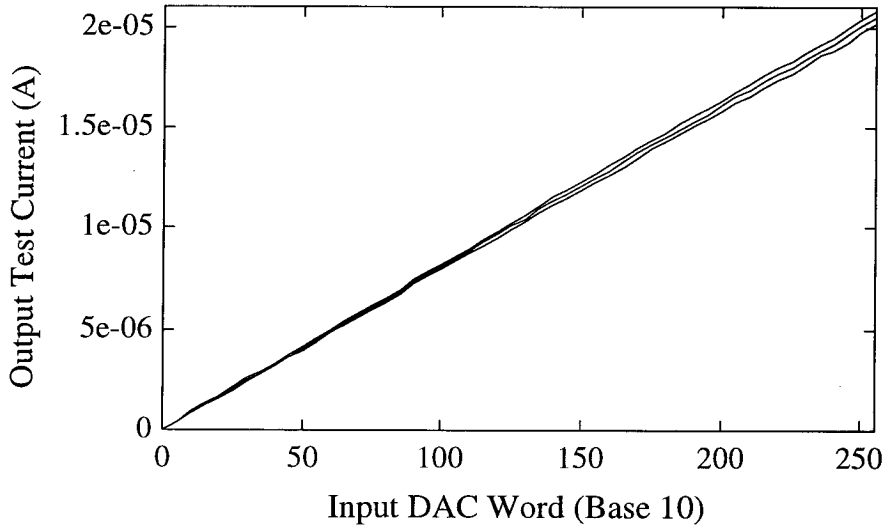


Figure 6.25 - Average Test Current vs. Input Word Characteristic for the DACs on all ten PAR Chips

From analysing the obtained DAC characteristics, it was concluded that the DACs were approximately linear, but were not monotonic. The non-monotonicity in the DAC is due to the inaccuracy in the "divide by sixteen circuit" between the two 4-bit DACs in the circuit, Figure 6.13. Since the division is implemented using a 16:1 current mirror, it is highly unlikely this division will be exact and from the results, it is apparent that the maximum output from the DAC representing the lower nybble is greater than the LSB of the upper nybble DAC.

A non-monotonic DAC should not cripple the operation of the chip, but if any problems are encountered, it is envisaged that a software "fix" can be used to swap the offending codes, thereby ensuring monotonicity.

The average results presented here show that the *gain* of the DACs vary from chip to chip, although only slightly. Such gain variations should be due to different degrees of transistor mismatch between the chips.

6.4.3. PAR DCM Characteristics

As with the DYMPLES Chip, provision was made to allow four NMOS and four PMOS DCMs to be tested on each PAR chip. Again, in order to minimise the output differences, each output DCM current was steered through a common Op-Amp using an 8-way analogue multiplexor. From the measured voltage results, the output current from the chip was calculated. All the chips had working DCMs and a typical set of results are shown in Figure 6.26.

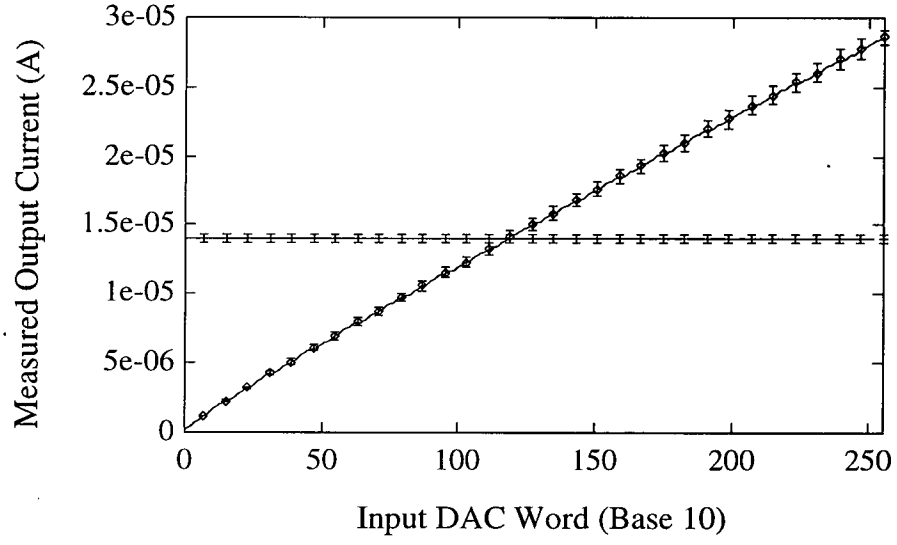


Figure 6.26 - The DCM Characteristic of a PAR Chip

These results were obtained by averaging the output currents from the four PMOS and four NMOS DCMs on a single PAR chip for one experimental run. They show the average characteristic obtained from the NMOS and PMOS DCMs, plus errorbars, for every 8th word, representing ±3 standard deviations of the results.

From the recorded results, it was obvious that the DCMs on the PAR chip were operating correctly: the currents stored in the NMOS DCMs varied linearly with the

applied DAC word (although the non-monotonicity of the DAC permeated through to the DCMs) and the PMOS DCMs stored and conducted constant currents.

The range of currents from the NMOS DCMs was found to vary very slightly both across chip and between chips and the level of the constant PMOS DCM current also varied slightly. Such across-chip variations will be due to mismatches between the on-chip measurement transistors since the currents are fed to the same Op Amp measurement circuit and, for the PMOS DCMs, the recorded currents are constant for each DCM. On a chip to chip basis, different biasing conditions will also affect the DAC gain and PMOS current level. Thus a combination of both phenomena will account for the gain variations witnessed in the results.

6.4.4. PAR Distance Circuit Characteristics

The distance circuit characteristics for all the PAR chips were measured and a typical set of results are shown in Figure 6.27.

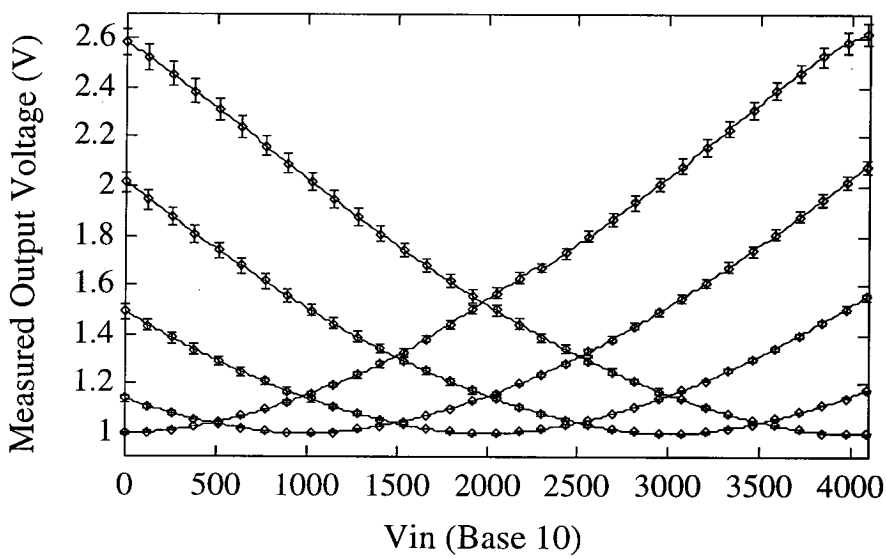


Figure 6.27 - Averaged Distance Circuit Characteristics from a Single PAR Chip

These results were obtained by averaging the results obtained from the four measurable distance circuits on a single PAR chip.

The results obtained from the distance circuits indicated they function as required. Furthermore, the results obtained from the characterisable circuits on all the chips indicated the offset error noted on the measurements from the RHO chip had been eradicated. This is because all the distance cells have the same orientation on the

silicon substrate.

All the distance circuits produced the correct input-output characteristic, although one circuit on chip 7 did have a larger current range than all the other circuits. The reason for this was not clear and was attributed to a chip anomaly. Again slight differences in the output current range were noted both across chip and between chips and these were again attributed to on-chip transistor mismatches.

6.4.5. PAR Non-linearity Characteristic

Several experiments were carried out to investigate the performance of the two-transistor non-linearity circuit. Again the measured results confirmed the theoretical properties of the circuit. Figure 6.28 shows the non-linear characteristics of the 15 centres on a single PAR chip, with the horizontal line at approximately 245 RAM locations representing the output of the centre generating the constant, full width, "bias" pulse.

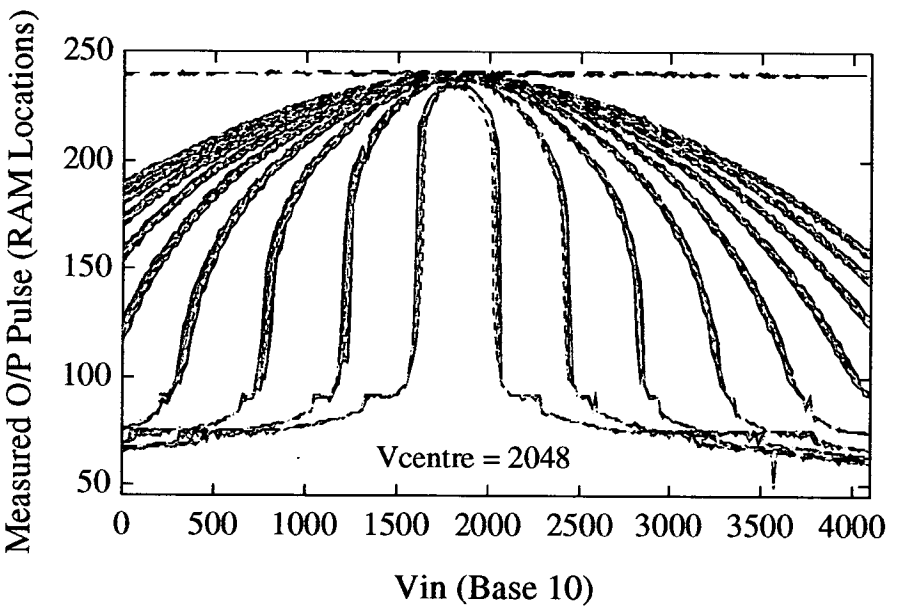


Figure 6.28 - Measured Results from the PAR Chip Centre Circuits

This level of matching between centres was noted on nearly all the chips (chip 0 had two centres that did not function correctly and chip 6 failed completely during testing), although differences existed in the non-linearities produced by the same centre on different chips, Figure 6.29. Again this was no more than expected since the centre circuit operates in voltage mode.

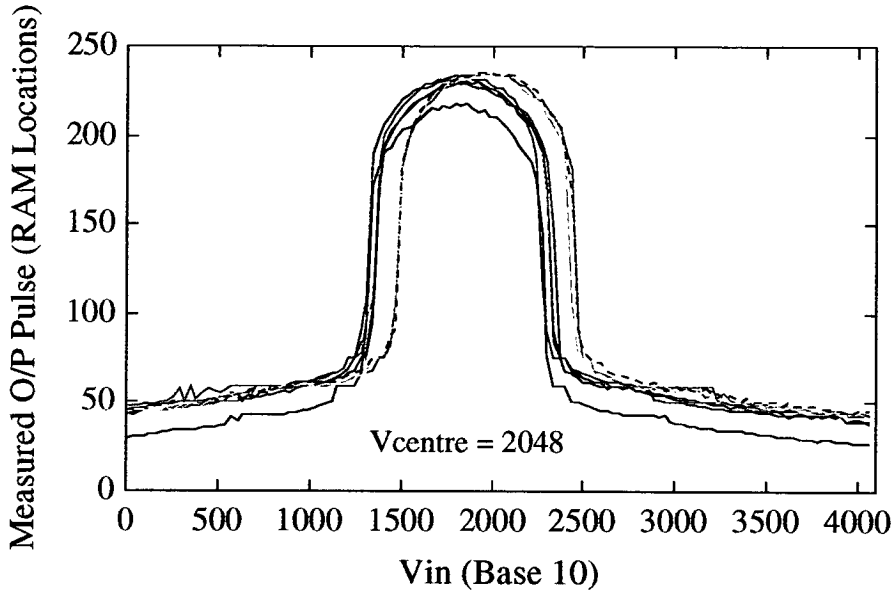


Figure 6.29 - Variation of a Single Centre Between 9 PAR Chips for a Single Value of V_{width}

Further trials carried out on the hidden layer circuits showed that it was possible to reproduce the non-linear shape of the circuit, for the same value of V_{width}, consistently across-chip, Figure 6.30(a) and (b), although it was noted that the centre of the non-linearity was offset from the ideal value loaded into the V_{centre} off-chip DAC, Figure 6.30(a). Further, it was noted that this offset was constant as V_{centre} varied, Figure 6.30(b).

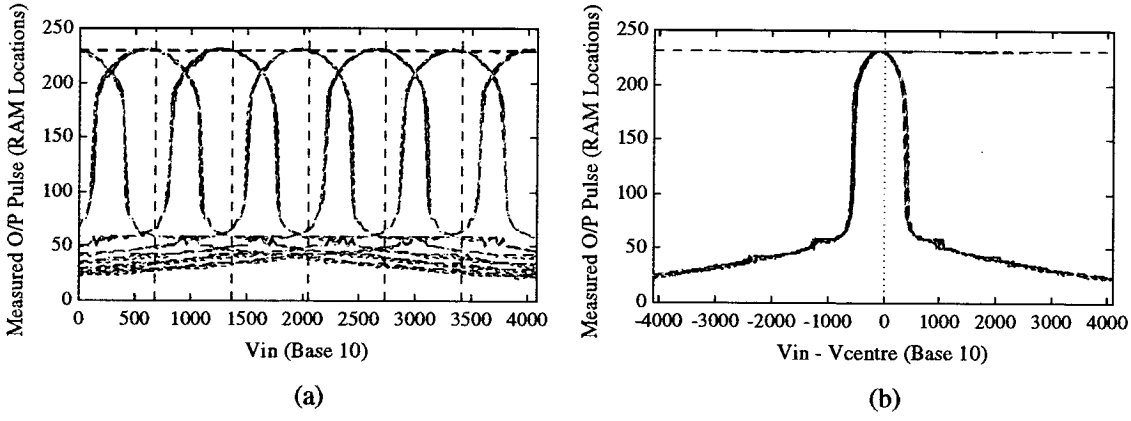


Figure 6.30 - Non-linearity Reproduction Variation with V_{in} and V_{centre}

The offset is not consistent with a non-zero V_{bs} voltage and instead corresponds to the change in V_{centre} resulting from charge injection from the access transistors to C_{centre}. Given the results displayed in Figure 6.29, it was concluded that this offset was likely to vary from chip to chip.

These results show the non-linear characteristic translated from the baseline. However, it should be remembered that both the baseline and height of the non-linearities can be altered by adapting the off-chip voltage ramp to the PWM Neurons.

6.4.6. PAR Multiplication Characteristic

The correct operation of both the on-chip DAC and the DCMs indicated that the two-quadrant multiplication circuits in the output layer should also function correctly. This proved to be the case and typical results from a single chip are shown in Figures 6.31 and 6.32.

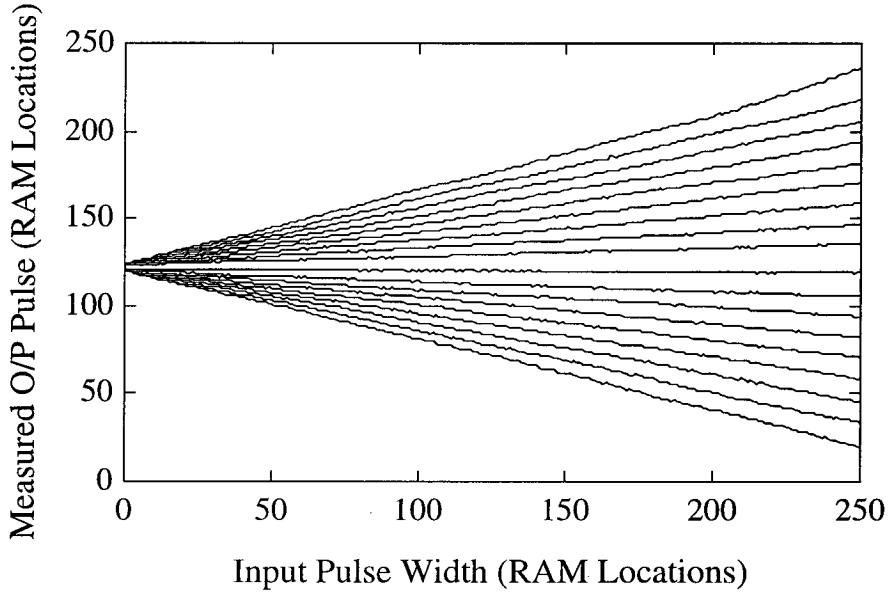


Figure 6.31 - The Output Pulse vs Input Pulse Two Quadrant Multiplication Characteristic for a Single PAR Chip Averaged Over All 4 Outputs

Figure 6.31 shows the measured output pulse width vs. input pulse width results averaged over the four output neurons on one chip and Figure 6.32 shows the output pulse width vs. loaded DAC weight results for the same chip. Figure 6.33 shows the characteristics of the neurons that produced the maximum deviations from the average characteristic.

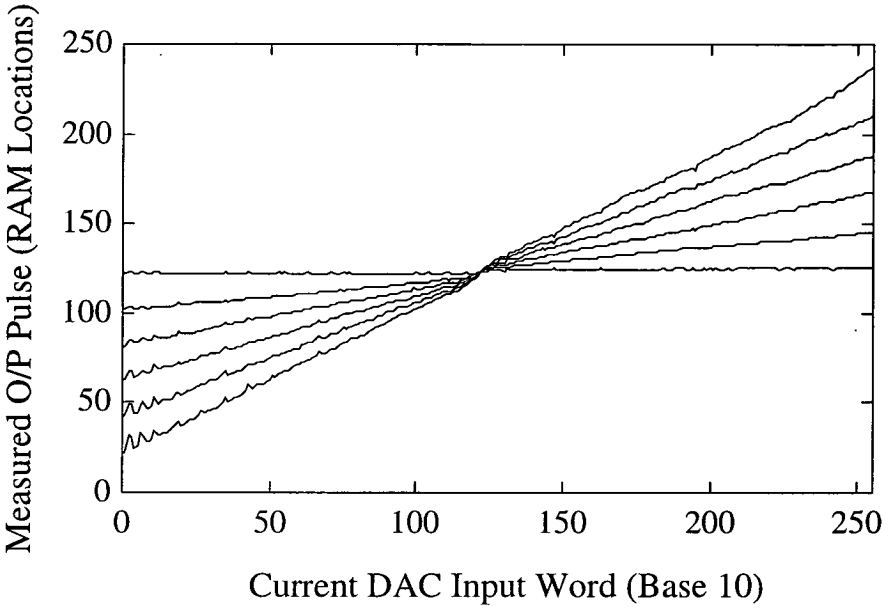
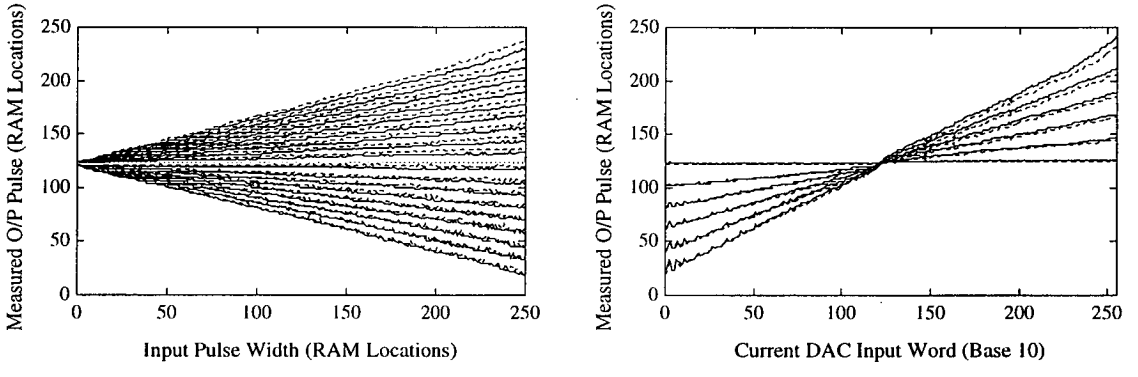


Figure 6.32 - The Output Pulse vs Input Weight Two Quadrant Multiplication Characteristic for a Single PAR Chip Averaged Over All 4 Outputs

From the characterisation experiments on the output layer circuits, it was noted that the weight dependent offset error which had been seen on the multipliers on the DYMPLES chip was no longer present, although a slight discrepancy still existed between the positive and negative synaptic weights, Section 4.8.4.



(a) (b)

Figure 6.33 - (a) The Worst Case Output Pulse Widths Recorded and Used for Producing the Averaged Results in Figure 6.31 and (b) the Worst Case Values Recorded for the Measurements Used in Figure 6.32

All the output layers worked, however output neuron 3 (the end neuron in the array) always produced narrower output pulse widths for the positive weights than the other three. This effect was noticed on all the chips and after investigating it, it was deduced to be caused by a chip anomaly.

For the two quadrant multipliers, the width of the zero activation pulse can be adjusted by altering the voltage to which C_{out} is precharged, the range of the output pulse widths can be adjusted by altering the baseline and amplitude of the output ramp, and the symmetry of the positive and negative weights can be altered by varying the on-chip DAC biasing current with respect to the ZCM biasing current, or vice versa. Thus, as with the hidden layer, the response of the output layer can be adjusted off-chip.

6.4.7. PAR Chip Summary

From the experiments carried out to characterise the operation of the DACs, DCMs, distance circuits, hidden layer and output layer on the PAR chip, it was concluded that the different circuits now functioned as required, that the observed discrepancies were principally due to transistor mismatch (and hence only to be expected) and that the discrepancies should not affect the chip's ability to implement an RBF classifier.

6.5. Summary

This chapter has detailed the development of the PAR chip, the final neural demonstrator for this project. After discussing the modifications made to previous incarnations of the RBF circuits, some consideration was given to the system level aspects of the chip, before the design was described and actual hardware results presented.

The measured results from the PAR chip indicate that all the circuit designs now function as required. Furthermore, the presented results also indicate that different instances of the circuits, on the same chip, produce very similar characteristics; this trend was noticed on all the chips. Thus it was concluded that the PAR chip **as a whole** should function as required, making it capable of implementing an RBF neural network. This hypothesis is investigated in Chapter 8.

Chapter 7

Software Simulation Results

It has been shown in previous chapters that the two-transistor non-linearity circuit produces a "bump" response that falls off monotonically as V_{in} and V_{centre} become disparate, exactly as required by the hidden layer of an RBF. However, this non-linearity has never been used in RBFs and, to investigate the capabilities of the bump function, several classification problems were solved using software RBFs having this hidden layer response. By modelling the designed circuits (albeit ideally) in software, it was possible to investigate the functionality and properties of the developed centre circuit without resorting to the design and manufacture of complicated, multi-functional hardware.

This chapter reviews the classification problems used in the software experiments, explains the operation of the simulator incorporating the circuit models, describes the software experiments performed and discusses the conclusions drawn as a result of the work. In addition to presenting the results of experiments carried out using full 64-bit floating point precision calculations, the findings from *weight quantisation* experiments are also discussed. The quantisation experiments were performed to investigate the likely effect of reducing the precision of the neural parameters in the hardware model **after** training was complete.

In addition to performing circuit feasibility trials, the simulator was used to produce solution sets of centres, weights and thresholds for downloading to the PAR chip. The generation of the neural parameters and the performance issues pertaining to using the actual RBF hardware to solve a problem are discussed in Chapter 8.

7.1. Classification Problems

Four *1-out-of-N* encoded classification problems were chosen for these investigations. The number of inputs and outputs plus the size of the training and test sets for each problem are summarised in Table 7.1.

Although many vectors were available for training the network, only small training sets (containing 100 vectors from each class) were used. This was due to the training

Problem	Inputs	Outputs	Training Vectors	Test Vectors
Gaussian Distributions	2	2	200	9800
Speaker Recognition	8	3	300	150
Sleep State	10	3	300	7200
Robot Location	8	6	600	5000

Table 7.1 - Summary of the Properties for each Classification Problem

method selected and will be discussed in greater detail in Section 7.2.3. Additionally, the components of the input vectors for all eight data files were translated and scaled to lie between 0.0 and 3.0, corresponding to the range of voltages that the PAR chip was designed to accept.

The Gaussian Distributions and Robot Location problems are both artificial, ie the data has been synthetically generated, whilst the Speaker Recognition and Sleep State problems use input vectors obtained after processing real measurements.

7.1.1. Two Class Gaussian Problem

This problem consists of data drawn from two separate, but overlapping Gaussian distributions. One distribution was centred at (1.224,1.224) with a standard deviation of $\sigma = 0.127$, whilst the other was centred at (1.478,1.478), with $\sigma = 0.380$, Figure 7.1. For this problem, the RBF network is required to identify from which distribution any given input vector is drawn.

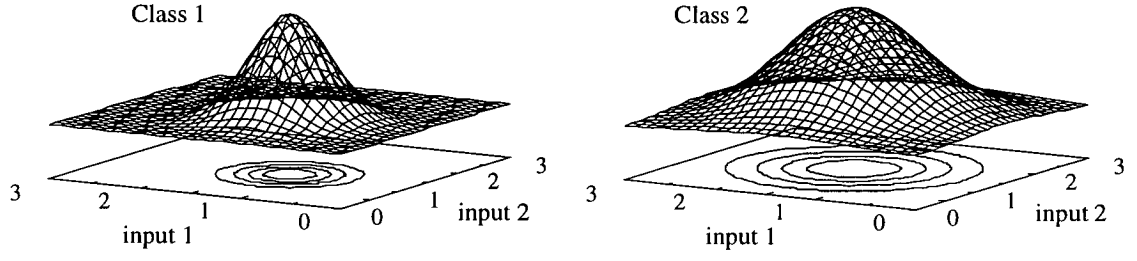


Figure 7.1 - Statistical Distributions for the 2 Class, 2 Input Gaussian Problem

This classification problem is an artificial one and it is impossible to achieve perfect classification. The maximum performance can be calculated by using Bayes Rule [16] and the optimal classification performance on the full 9800 vector test set is 88.4%.

7.1.2. Speaker Recognition Problem

To solve the speaker recognition problem, the neural network must learn to recognise which of three speakers produced a given sample of speech. The data base of classified exemplars has been produced by sampling and processing raw speech data, from 3 male speakers, to generate the cepstrum [149]. The first 8 cepstral coefficients form the input vector for each exemplar. This data set, and those for the two remaining problems, were obtained from the Engineering Science Department at the University of Oxford [148].

7.1.3. Medical Sleep Data Problem

For this problem, the RBF is required to identify whether the input vector represents a patient in the wakefulness, the dreaming sleep or the deep sleep state. The data vectors have been obtained from a single channel of recorded human EEG (electroencephalogram). By sampling analogue EEG recordings (from 5 patients) at 128 Hz and computing Kalman filter coefficients from the samples, a database was constructed which contains equal numbers of exemplars from all 3 sleep states [37]. Each input vector comprises the first 10 Kalman filter coefficients, averaged over one second blocks.

7.1.4. Robot Location Problem

This is the most difficult of the problems considered. A mobile robot is assumed to be in one of six regions in an L-shaped room that contains 6 corners and two objects, Figure 7.2.

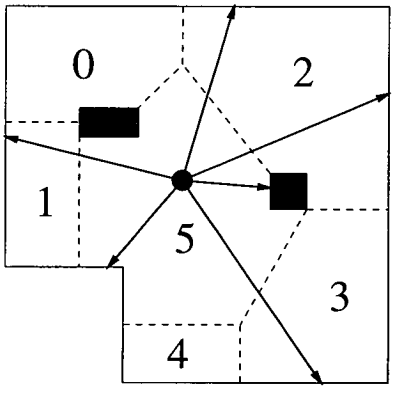


Figure 7.2 - Diagrammatic representation of the Robot Location Problem

To generate the data vectors, the robot was assumed to be positioned at various locations in the room and 360° range scans were generated artificially using a simulation program. From these range scans, the information was extracted to construct 8-dimensional input vectors [150]. Whilst it is assumed that each input vector can only be generated in one region of the room, this problem is intrinsically difficult because locations close in input space (ie positions within the room) can be far apart in classification space [148].

7.2. Software Simulator

In order to investigate the properties of the two-transistor circuit, a special software simulator was written in C. The purpose of the simulator was to model the operation of the developed centre circuit and indicate:

i) whether the developed circuit could usefully solve any of the classification problems

ii) how the classification results from a model of the developed circuits compared to those obtained from Gaussian RBF networks with the same centres and trained in exactly the same way.

In the simulator, the operation of several non-linearities, corresponding to different values of V_{width}, were compared to two Gaussian networks. One Gaussian network used basis functions all having the same width (equal to the maximum distance between any two centres), while the other used non-linearities with widths equal to the distance between that centre and its nearest neighbour. All the basis functions used single width values for all the dimensions.

In the simulator, the output layer was assumed to be ideal, thus it was "modelled" in software using the normal CPU multiplication and addition operators.

It is important to realise that for the simulations carried out in this chapter, the absolute value of the classification rate is not critical. Of greater significance here is the performance of the modelled hardware compared to a software RBF that uses Gaussian non-linearities.

7.2.1. Overall Operation

The overall operation of the simulation program, which uses the hybrid training scheme of Moody and Darken (as described in Section 2.5.4.2), is shown by the flow diagram in Figure 7.3.

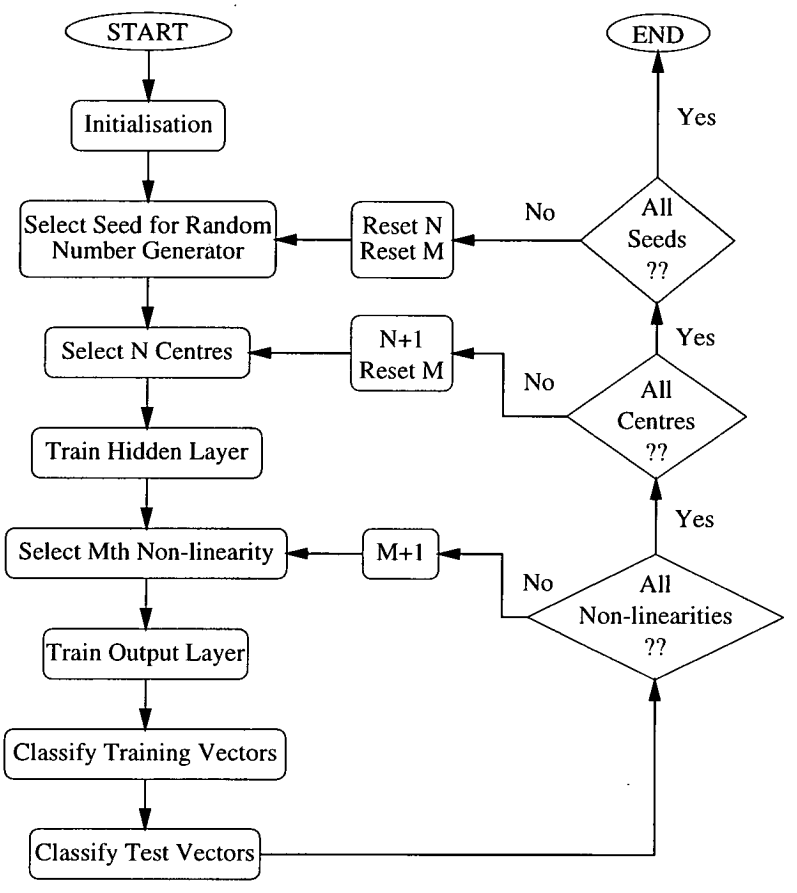


Figure 7.3 - Flow Diagram description of the operation of the Software Simulator Program

After the program has been initialised, a random number generator seed is selected and the hidden layer of the network trained using the adaptive k-means algorithm: each input vector, $\mathbf{x}$, in the training set is presented to the network in turn and the closest centre, $\mathbf{c}_j$, to the input vector is adjusted using equation 2.26, with α set to 0.02. This procedure is repeated 100 times for the entire training set, with the order of vector presentation varied for each epoch to improve the robustness of the training.

The random number generator seed is used to select the location of the initial centres and determine the order of presentation of the training vectors. Thus the use of seeds allows a direct comparison to be made between the different non-linearities since each seed produces a unique training schedule.

Once the hidden layer has been trained, a non-linearity is placed on each centre location and the output weights calculated using a pseudo-inverse method, Section 7.2.3. On completion of output layer training, the network is used to classify the vectors in

both the training set and the test set for the problem under consideration and the correct classification rate is recorded. This process is repeated for all the non-linearities. However, since the weights required to reproduce the input to output mapping vary with the non-linearities, a new output weight set must be calculated for each non-linearity used.

After all the non-linearities have been used as the basis functions in a given hidden layer, the number of centres is increased by one, the hidden layer is re-trained using the same seed and the output weight calculation and classification processes are repeated for the new hidden layer. This process is continued until the maximum number of centres is reached, whereupon the next seed is loaded, the number of centres is re-set to the minimum allowed and the entire process is repeated.

By running this procedure using many seeds, enough results were generated to allow meaningful mean classification rates to be obtained.

7.2.2. Hidden Layer Operation

In an RBF network using the Euclidean distance measure and the Gaussian non-linearity, a suitable algorithm for calculating the output of each centre is:

```
loop over centres{
    tot=0.0;
    /* calculate squared euclidean distance */
    loop over vector dimensions{
        tot+=(input[i] - centre[j][i])^2;
    }
    /* take square root */
    dist=(tot)^(1/2);
    /* calculate basis function output */
    rbf_op[j]=f(dist;width[j]);
}
```

where $f()$ is the non-linearity used. With this algorithm, the Euclidean distance between the input and centre vectors is calculated initially and it is then applied as the argument to the non-linear function. The region of influence of the function is, of course, determined by its width parameter.

Unfortunately, although it is possible to use this algorithm to calculate the hidden layer responses for the Gaussians, it is not possible to use it for the non-linearities generated by the two-transistor circuit. This is due to the important operational differences that exist between using the circuit and the Gaussian/Euclidean distance combination in the hidden layer, and the difficulty of accurately modelling the two-transistor circuit with a simple formula.

The modelling of the two-transistor circuit operation will now be considered in more detail.

7.2.2.1. Distance Circuit Modelling

Since the response of each distance circuit is a quadratic approximation to the squared Euclidean distance between V_{in} and V_{centre}, it was deemed necessary to approximate the transfer function of the distance circuit more accurately than simply assuming the squared Euclidean distance would suffice.

As derived in Section 5.2, the output current from the distance circuit (to a first order) can be expressed as

$$I_{dist} = \frac{\beta_{narrow}}{2}\left(|V_{in} - V_{centre}|\right)^2 + \beta_{narrow}\left(\frac{2I_{wide}}{\beta_{wide}}\right)^{\frac{1}{2}}|V_{in} - V_{centre}| \qquad (7.1)$$

By using *typical* process parameters to evaluate the coefficients for the squared and linear terms in the above equations, it was possible to model the actual current for the circuit (as generated by a level 3 HSPICE simulation) by an empirical approximation, equation 7.2.

$$I_{dist(modelled)} = 5.376e-7 * \left(|V_{in} - V_{centre}|\right)^2 + 1.554e-4 * (I_{wide})^{\frac{1}{2}}|V_{in} - V_{centre}| \qquad (7.2)$$

The empirical formula described by equation 7.2 required a scaling factor of $\frac{1}{1.78}$ to scale the modelled current to the same range as the measured current when the *typical* process parameters for the MIETEC 2.4μ process were used. This scaling factor has been included in the given coefficients in equation 7.2

Since I_{wide}, the current sunk by the wide transistor in the conducting ratioed pair, depends non-linearly on $|V_{in} - V_{centre}|$, this current was modelled in the simulator using an HSPICE-extracted current look-up table.

Further, as acknowledged in Chapter 6, by not using separate wells for the ratioed pairs, a non-symmetrical current response, that depends on the value of V_{centre}, is obtained from the distance circuit. However, as shown in Figure 7.4, the empirically modelled current is a good approximation to the simulated cases for V_{centre} equal to 0V and 3V. Thus the formula was deemed to be a good representation of the actual current, irrespective of V_{centre}.

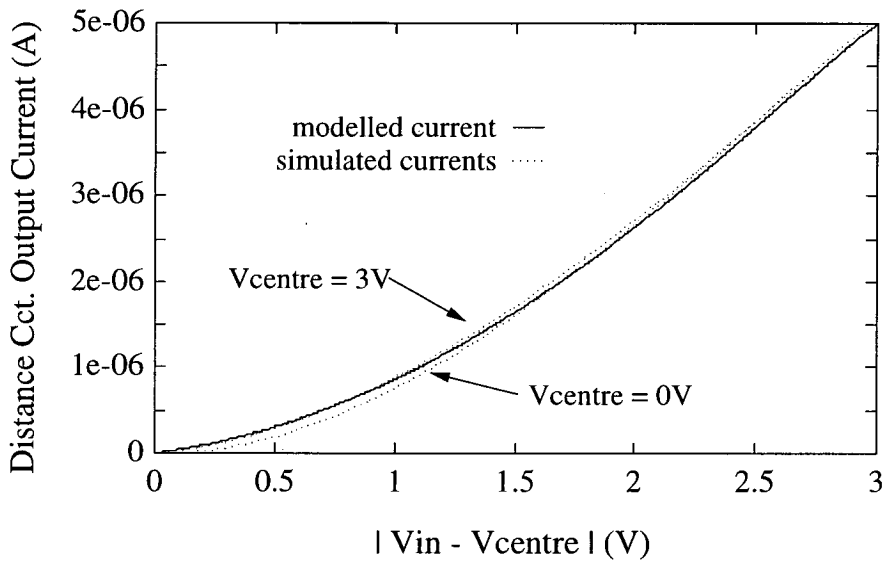


Figure 7.4 - Comparison of the Modelled and Simulated Distance Circuit Currents

7.2.2.2. Non-Linear Circuit Modelling

The non-linear response of the two-transistor circuit, Figure 5.11, relies on M_{width} and M_{load} competing for I_{dist}. Since the output voltage relies on exploiting the physical properties of the transistors, it will be very difficult to model their operation using simple formulae. Indeed, this problem is further exacerbated by the fact that both transistors operate in different modes depending on the values of V_{width} and I_{dist}.

However, the operation of the circuit can be adequately and easily modelled for simulation purposes by using look-up tables of either the V_{out} vs. $|V_{in} - V_{centre}|$ or the V_{out} vs. I_{dist} curves generated by HSPICE simulation.

The V_{out} vs. I_{dist} curves were used for this simulator since the output from each distance circuit (I_{dist}) has already been modelled by equation 7.2. The use of the voltage vs. current curves is also advantageous since the V_{out} vs. $|V_{in} - V_{centre}|$ curves will differ slightly depending on the value of V_{centre}, whereas the V_{out} vs. I_{dist} curves do not.

For the purpose of the software simulator, the look-up table consisted of 11 different non-linear curves, Figure 7.5, with 501 points per curve.

7.2.2.3. Complete Circuit Model

Having looked at how the distance circuit and non-linear circuit were modelled, the algorithm used for modelling the feedforward operation of the centre circuit can now be defined.

loop over centres{

tot=0.0;

/ calculate total current */*

loop over vector dimensions{

$cur=|V_{in\_i} - V_{centre\_ji}|$;

I_{wide}= *interpolation of "cur" onto* I_{wide} *look-up table;*

/ calculate current component using empirical formula */*

/ a = 5.376e-7, b = 1.554e-4 */*

$tot+=acur^2 + b(I_{wide})^{\frac{1}{2}}\ cur;$

}

/ scale and average total current */*

/ n is the number of vector pairs */*

$I_{dist} = \frac{tot}{n};$

/ calculate basis function output */*

rbf_op[j] = interpolation of I_{dist} *onto transistor curve look-up table;*

}

As can be seen, this algorithm differs from the one described earlier in this section and models the simulated operation of the distance circuit more realistically than simply assuming it can be approximated by Gaussians using the Euclidean distance metric. One important addition to this new recipe is the division of the total distance circuit current by n, the number of circuits: I_{dist} is now the **average** current in any circuit and is interpolated onto the transistor non-linearity curves for a **single** centre circuit. It is easy to prove mathematically, using equations 5.11 to 5.14, that this is

equivalent to interpolating the total current onto the combined non-linear curves for n centre circuits. This proof assumes that the same non-linear curves are generated by all n circuits.

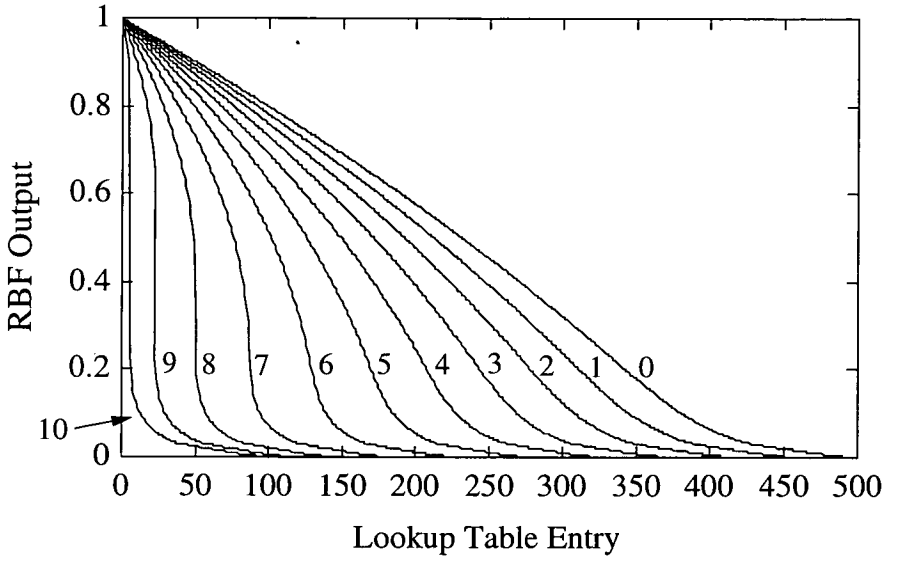


Figure 7.5 - Graph of the Scaled V_{out} vs. I_{dist} Non-linearity Look-up Table

7.2.3. Output Layer Training

Using the hybrid training method of Moody and Darken [8], the centre positions are fixed after the hidden layer has been trained and the training of the output layer can be achieved through the minimisation of a sum of squares error cost function similar to equation 2.7. Further, since only a single layer of linear units is being trained, the error surface for the output layer is a quadratic function in weight space, guaranteeing that a global minimum solution, corresponding to an optimal weight set, can be found. However, it should be noted that this weight set is only optimal, in a least squares sense, for the chosen centres and chosen non-linearity. It need not be the best solution for the actual problem itself.

The optimal output weight set for an RBF can be found by solving the following equation for Λ.

$$Y_{DxN} = \Phi_{DxM}\Lambda_{MxN} \tag{7.3}$$

In this equation, Y represents the D by N matrix of N network output unit responses for each of the D training vectors, Φ is the D by M matrix of the M hidden layer responses for each of the input vectors and Λ is the M by N matrix of output

weights. If $M < D$, the usual case for an RBF neural network, then the pseudo-inverse [15], $\Phi^{\dagger}$, of Φ must be found instead of the inverse matrix Φ^{-1}.

Assuming that $(\Phi_{MxD})^{\dagger}$ exists, then the output weight matrix can be found from

$$\Lambda_{MxN} = (\Phi_{DxM})^{\dagger} Y_{DxN} \tag{7.4}$$

One of the most powerful methods for matrix inversion is singular value decomposition (SVD) [151], and it was used in this hardware simulator program. One major advantage of using SVD is that the optimal output weight set for an RBF network can be found in a single execution of the algorithm, unlike training an MLP using backpropagation and gradient descent. Thus, using SVD allows the output weights for all the networks to be found quickly, without the need to optimise the learning rate, or the number of epochs, for an iterative algorithm.

The main disadvantage of using the SVD algorithm is the large memory requirement for storing the various matrices during normal program operation. The memory required scales with both the size of the training set and the size of the network and this was why only small networks and 100 vector training sets were used.

7.3. Full Precision Software Experiments

The main software experiments carried out with the simulator investigated the ability of the centre circuit, using transistor and Gaussian non-linearities, to solve the four classification problems. In addition, the simulator was also used to investigate possible explanations for some of the observed results.

7.3.1. Two-Transistor Non-linearity Performance

Initially the simulator was used to find the classification performance of RBF networks with different numbers of centres, for 13 non-linearities. The non-linearities were given different reference tags: the 11 transistor curves were numbered as shown in Figure 7.5, whilst the RBF using Gaussians of a single width was tagged with 11 and the RBF with Gaussians having individual widths was tagged with 12. Each simulation run used 25 seeds to train RBFs with between 10 and 32 centres using the 11 transistor and 2 Gaussian non-linearities[8]. The network size was restricted to between 10 and 32 centres because this was considered to be the range that would be

[8] The stated number of centres does not include the bias term. Separate provision was made for it within the software simulator.

initially implementable on a chip - the desire with RBFs is to adequately solve the problem using as few centres as possible - and the small size of the training sets meant that the maximum number of centres had to be limited otherwise the network could simply be trained to act as a look-up table.

Thus 25 sets of results were obtained for each problem for a total of 299 RBF networks. Mean classification results were calculated for both the training and the test data sets and these are tabulated in Appendix B. Figure 7.6 provides a visual display of the results. A summary of the best and worst classification performances are given in Tables 7.2 and 7.3. Table 7.4 compares the performance of the best network using a transistor non-linearity with the best network using a Gaussian non-linearity.

A number of features are apparent from the results of these classification trials.

- The performance of both the transistor and Gaussian non-linearities are clearly problem dependent.
- There is a general tendency for the classification rate for all the training sets to increase with the number of centres used, Figure 7.6. Indeed, the best classification results are usually obtained from networks with a large number of centres, whilst the worst classification performances tend to be produced by networks using a small number of narrow non-linearities, Tables 7.2 and 7.3.
- All the classification rate graphs have dips in them, occurring for some of the narrower curves, though not necessarily the narrowest. The reason behind this warranted further investigation and is explored in Section 7.3.2.
- In some cases, the performance of the transistor non-linearity look-up table curves surpassed that of the Gaussian networks. This too warranted further research and this is considered in Section 7.3.3.

Since the results from both the transistor and Gaussian non-linearities are similar, it can be concluded from these results that the designed centre circuit can be used as effectively as Gaussians within small RBF networks solving classification problems. It must be emphasised, though, that it is impossible to make general assumptions from these results with regard to the ability, or the lack of ability, of the new circuit to outperform Gaussians generally. These trials have been limited, only networks with a small number of units have been used, and were designed to investigate the performance capabilities of transistor non-linearity RBFs, using the same networks with Gaussian non-linearities as a benchmark. From these results, all that can be stated is that the performance achieved using both types of non-linearity are similar and that

Classification Results Summary - All Non-linearities						
Data Set	Max(%)	Curve	Centres	Min(%)	Curve	Centres
Gaussian Training	87.44	0	32	80.18	8	11
Gaussian Test	87.78	11	14	82.27	8	11
Speech Training	87.80	11	32	67.52	10	10
Speech Test	76.67	7	29	54.13	9	10
Sleep Training	86.65	11	32	72.21	8	10
Sleep Test	80.52	11	32	68.22	9	32
Robot Training	86.53	2	32	58.59	10	10
Robot Test	80.18	2	32	56.49	8	10

Table 7.2 - Summary of the Maximum and Minimum Classification Rates Obtained Using the Hardware Simulator Program and all the non-linearities

Classification Results Summary - Only the Transistor Non-linearities						
Data Set	Max(%)	Curve	Centres	Min(%)	Curve	Centres
Gaussian Training	87.44	0	32	80.18	8	11
Gaussian Test	87.45	10	27	82.27	8	11
Speech Training	84.41	0	32	67.52	10	10
Speech Test	76.67	7	29	54.13	9	10
Sleep Training	81.51	6	32	72.21	8	10
Sleep Test	75.10	6	32	68.22	9	32
Robot Training	86.53	2	32	58.59	10	10
Robot Test	80.18	2	32	56.49	8	10

Table 7.3 - Summary of the Maximum and Minimum Classification Rates Obtained Using the Hardware Simulator Program and excluding the Gaussian results

the classification performances are adequate.

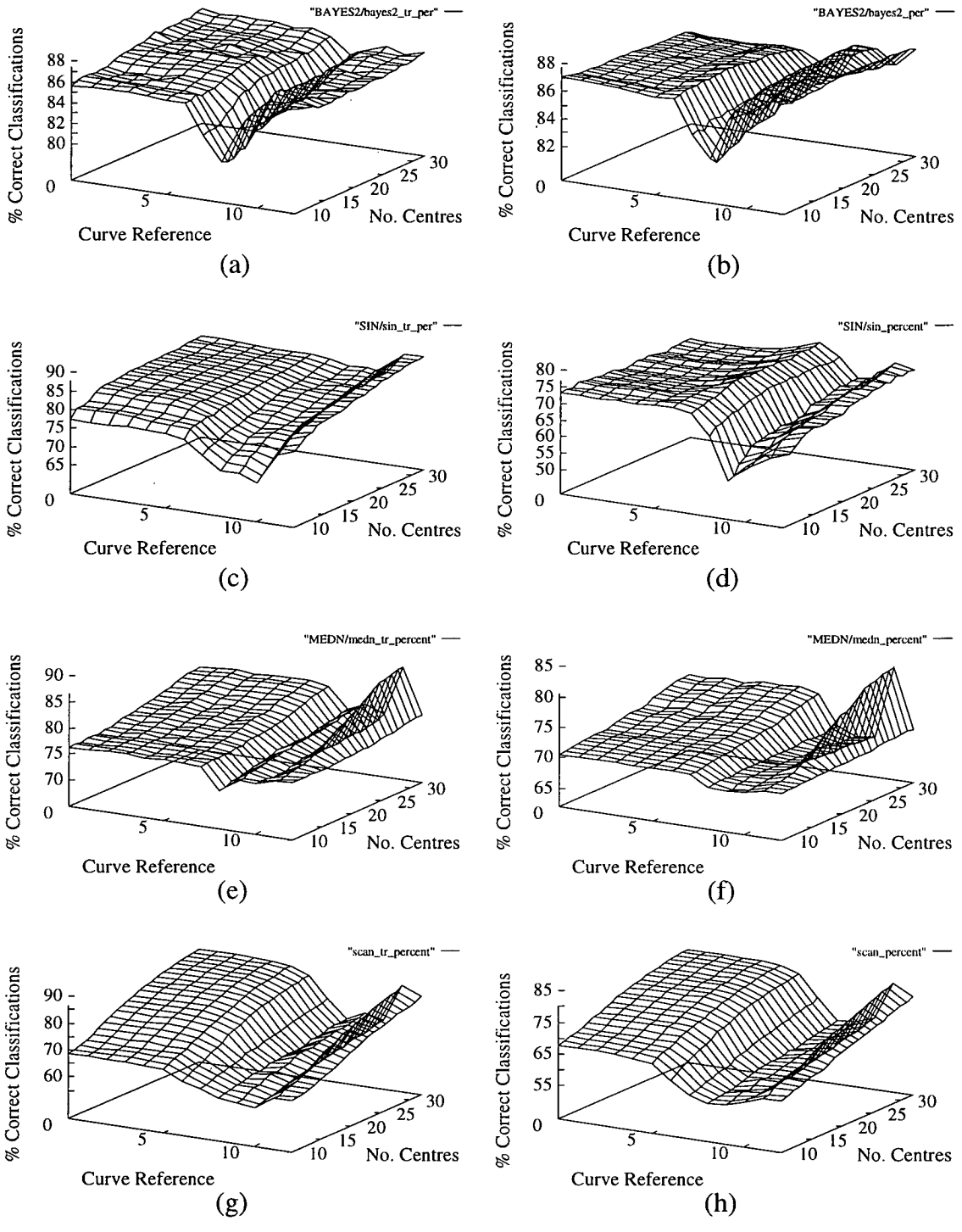


Figure 7.6 - Graphical Illustration of the mean classification performance of the software simulator for the training and test sets of all 4 problems

Comparison of Best Transistor with Best Gaussian Non-linearities		
Data Set	Best Transistor (%)	Best Gaussian (%)
Gaussian Training	87.44 ± 0.94	86.78 ± 0.53
Gaussian Test	87.45 ± 0.29	87.78 ± 0.10
Speech Training	84.41 ± 1.16	87.80 ± 2.24
Speech Test	76.67 ± 2.06	73.95 ± 1.87
Sleep Training	81.51 ± 1.59	86.65 ± 1.82
Sleep Test	75.10 ± 1.54	80.52 ± 1.76
Robot Training	86.53 ± 0.81	84.10 ± 1.87
Robot Test	80.18 ± 0.78	78.86 ± 1.59

Table 7.4 - Comparison of the Best Transistor Non-linearity Performance with the Best Gaussian Non-linearity Performance. Mean and ±1 standard deviation results are given.

7.3.2. Curve Interpolation Investigation

To investigate the cause of the dips in the classification surfaces displayed in Figure 7.6, the simulation program was altered so that the region where each scaled, normalised distance measure was interpolated onto the non-linearity look-up table was recorded for 15 centre and 31 centre RBF networks: these two networks being the most suitable (of the ones tested) for implementing on a chip (Section 6.2.1)[9]. The distances were stored for exemplar training and test vectors for all the problems and results from 25 seeds were obtained for the 4 training sets and the test set for the Speaker Recognition problem. However, due to file size limitations, only the results from one seed were recorded for the test sets for the remaining problems - although further experiments indicated using different seeds for the test sets produced almost identical results.

The recorded distances were sorted into one of 500 bins, each corresponding to an interval between two consecutive entries in the look-up table. (Each interval corresponds to an increment in I_{dist} of 10nA.) The frequency of the samples in each bin was then calculated and the results overlaid onto the two-transistor V_{out} vs. I_{dist} curves. The resulting graphs are shown in Figures 7.7(a)-(h) and 7.8(a)-(h).

[9] When the bias term is included as an additional centre, these chips would have 16 and 32 centres respectively.

These graphs show that in each case, the majority of the normalised distance measures are fairly small, usually lying in the first 100 to 150 bins. From the superposition of the frequency distribution graphs onto the transistor non-linearity curves, it is clear that these distance measures fall in the region of the steepest gradients for the narrowest curves: exactly where a small change in the calculated current could lead to a large change in the calculated RBF output and a potential increase in the number of misclassifications. This was concluded to be the reason for the dips in the classification surfaces for the narrower non-linearities.

Further, an explanation can also be hypothesised as to why the classification for even narrower curves increases. It is believed that the gradients are fairly shallow whenever good interpolation results are obtained. However, these interpolations could be occurring where the RBF outputs are consistently high (0.8 to 1.0) or consistently low (0.0 to 0.2). The SVD algorithm does not care, it simply finds a set of numbers that solves the problem in the least squares sense. Thus, as long as the distance measures are interpolated **consistently** onto the same area of the graph for the training set and test set, then the test set classification rate should be similar to that of the training set.

7.3.3. Gaussian Width Investigation

The results from the classification experiments indicated that the classification performance of some of the transistor non-linearities were better than those from the networks using Gaussians. It was decided that this situation required further investigation, considering that the widths of the transistor non-linearity networks were fixed before any learning took place, while the widths of the Gaussians are determined after the centre locations have been adapted.

Often RBF networks with Gaussian non-linearities are developed on the premise that as many centres as required can be chosen to solve the problem. Heuristics such as distance to nearest neighbour etc. are then used to calculate the spread of the non-linearities to ensure that input space is adequately covered.

This is not the case for the Gaussian networks generated by the simulator program. Here the number of allowable centres is restricted and it was hypothesised that in this situation, the width heuristic used is not allowing the Gaussians to adequately cover input space. Meanwhile, the fixed width transistor non-linearities afford better coverage and hence produce a superior classification rate.

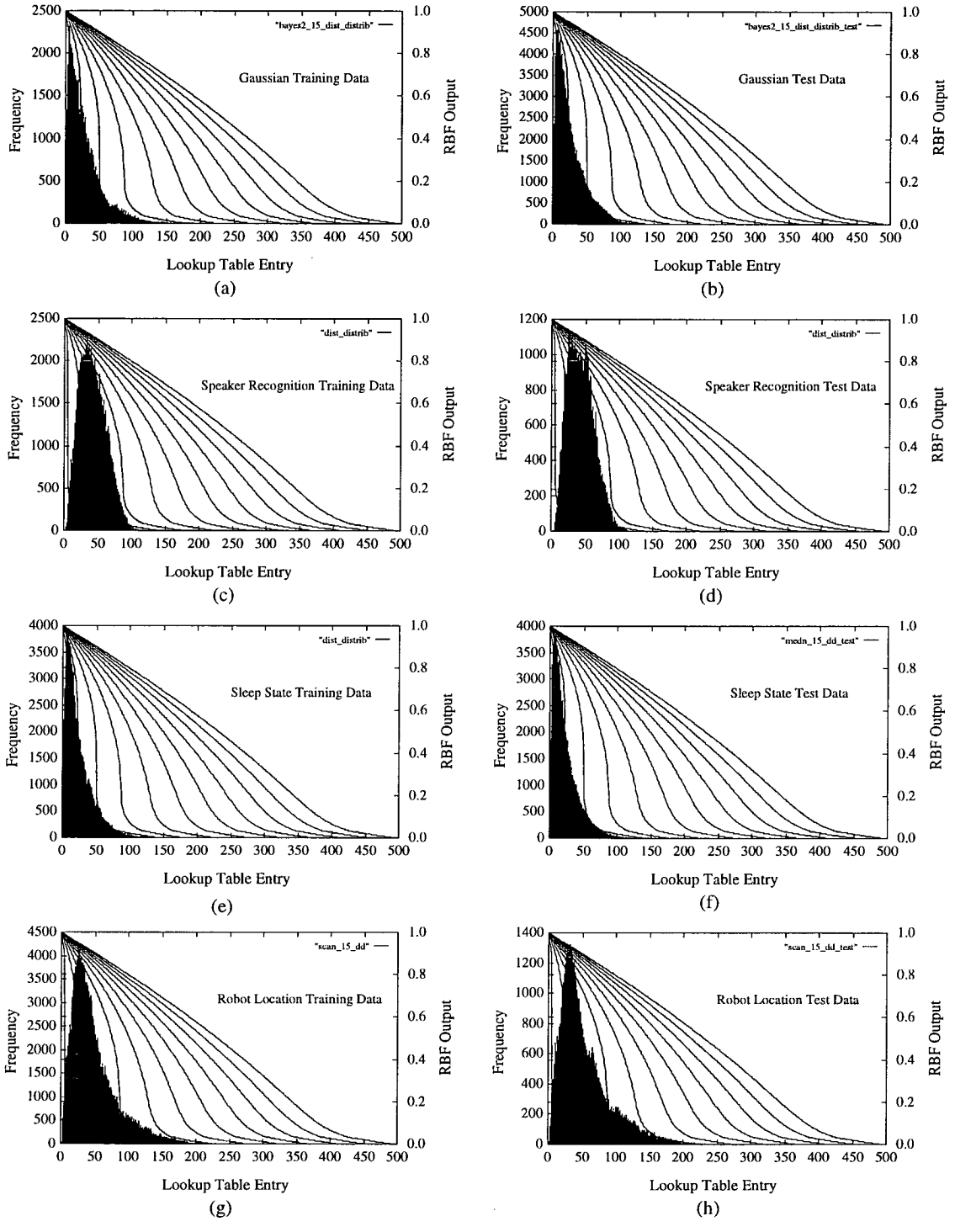


Figure 7.7 - Distance Interpolations for a 15 Centre RBF Network

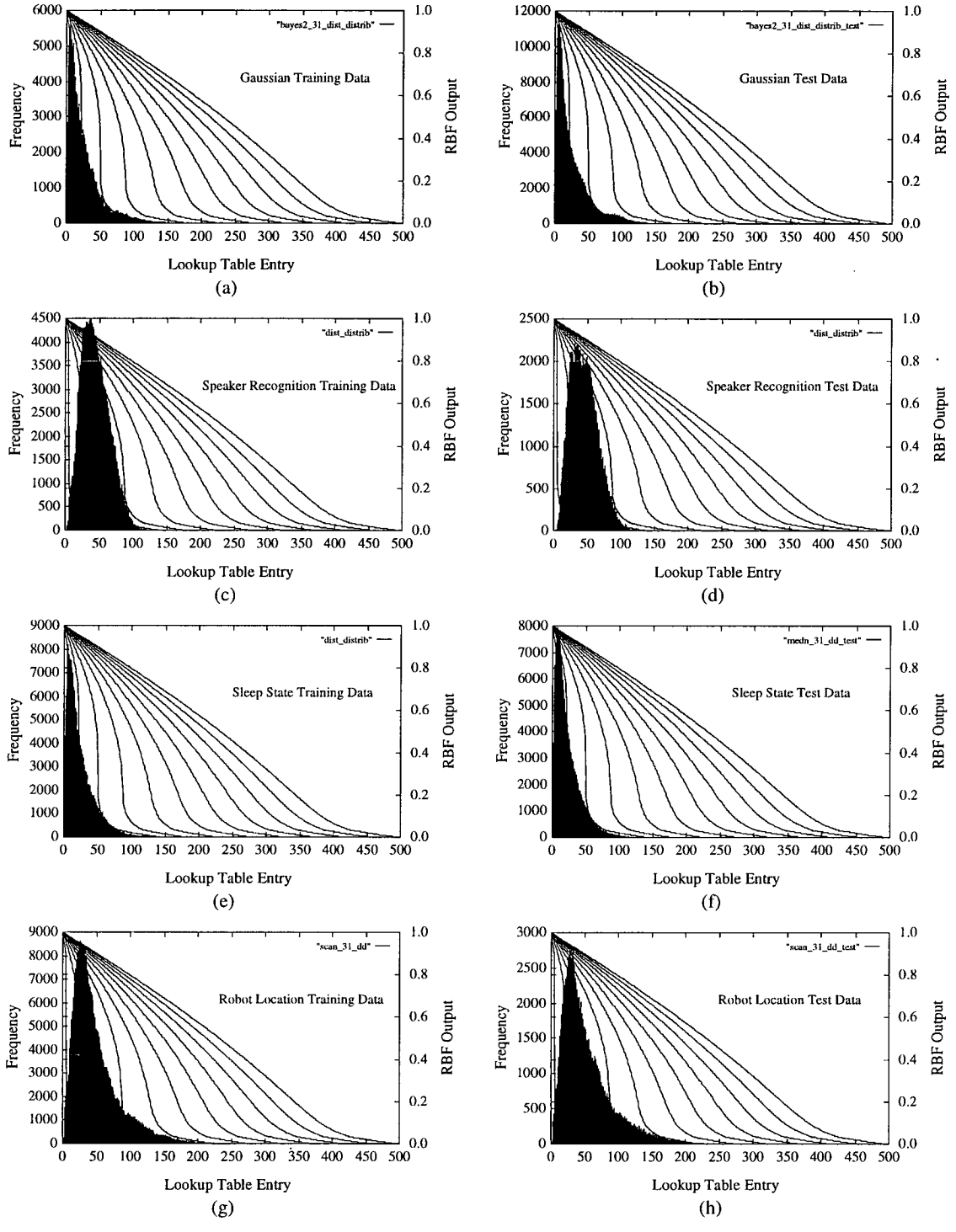


Figure 7.8 - Distance Interpolations for a 31 Centre RBF Network

To explore this hypothesis, the simulator was used to investigate the performance of the two Gaussian networks again, this time for progressively larger widths. Again 25 seeds were used to train RBFs with between 10 and 32 centres and the classification rates for the test sets were recorded. This time, though, the heuristically calculated widths were varied up to 25 times the original value and the classification rates recorded in each case. From these measures, the mean classification rates were found and graphs of mean test set classification performance vs. width factor for networks using both types of Gaussian curves are shown in Figures 7.9(a)-(h).

Clearly, with the exception of the 2 class Gaussian problem, the classification rate of all the networks increases with increasing Gaussian width. However, for most networks, after an initial steep rise in performance when the width factor lies between 1 and 5, it levels off or increases at a much slower rate for further width increases.

For the 2 class Gaussian problem, the best classification rates occur for small values of the width factor (up to approx. 3) and the classification rate actually decreases as the coverage of input space increases. This suggests this problem can be solved using only a small number of carefully chosen centres and that the width heuristic used is adequate for this problem. This should not be too surprising considering the problem is concerned with classifying vectors drawn from two distributions with the same functional form as the RBF non-linearity !

The best performance from the Gaussian networks with increased widths were also compared with the best transistor curves from Tables 7.3 and 7.4. These results are summarised in Table 7.5. The first number in the brackets in the *Curve 11* and *Curve 12* columns represents the number of centres and the second number represents the width multiplication factor in the network producing the maximum classification result for each problem.

Width Investigation - Comparison of Best Transistor and Gaussian Curves			
Data Set	Best Transistor (%)	Curve 11 (%)	Curve 12 (%)
Gaussian Test	87.45 ± 0.29	87.78 ± 0.10 (14,1)	87.81 ± 0.09 (13,5)
Speech Test	76.67 ± 2.06	76.51 ± 2.08 (31,8)	76.03 ± 1.48 (13,25)
Sleep Test	75.10 ± 1.54	85.31 ± 0.24 (32,25)	83.52 ± 0.60 (32,25)
Robot Test	80.18 ± 0.78	81.29 ± 1.08 (32,24)	79.72 ± 1.14 (32,6)

Table 7.5 - Comparison of the Best Classification Performances of the Gaussian and Transistor Non-linearities.

From the results of the width comparison experiments, it is clear that:

- In general, when the widths of the Gaussians are increased the performance of the networks tends to increase also, confirming the original hypothesis regarding inadequate coverage of input space by the original network.
- When the widths of the Gaussians are increased, the performances of the best Gaussian and best Transistor non-linearities are similar, giving further confirmation that the developed centre circuits can be used as basis functions in small RBF networks.

7.4. Quantisation Experiments

Having shown the potential of the two-transistor non-linearity within RBF networks, the performance of the software model of the hardware was investigated when the neural parameters were quantised after training.

Quantisation of the inputs and the weights is an issue relevant to many analogue, as well as digital, neural implementations. In the PAR chip, for example, quantisation is introduced through the use of chip-in-the-loop learning to train the hardware (the outputs are sampled in time by a RAM chip before being decoded), the use of a global dynamic refresh scheme where the neural weights are stored in off-chip RAM, and the use of DACs to generate the neural parameters and chip inputs.

Thus it was important to have an idea as to how input and weight quantisation might affect the hardware. To achieve this, quantisation experiments were performed on all four classification problems.

It is worth noting that in these experiments, only the performance of the the transistor non-linearities were investigated. Looking at the performance of the Gaussians would raise issues over the need for, and best way to implement, quantisation of the calculated widths. This was unnecessary for the thesis.

To perform these experiments, the software model of the hardware was again used to train RBF networks, to 64-bit floating point precision, using the adaptive k-means and SVD algorithms. The generated weight sets were quantised to the desired levels after network training and before the test vectors were classified. For these experiments, the hidden layer and output layer neural parameters were quantised to 16-bit & 16-bit, 12-bit & 12-bit, 8-bit & 12-bit, 12-bit & 8-bit and 8-bit & 8-bit respectively. Again 25 seeds were used for each quantisation experiment to obtain meaningful results.

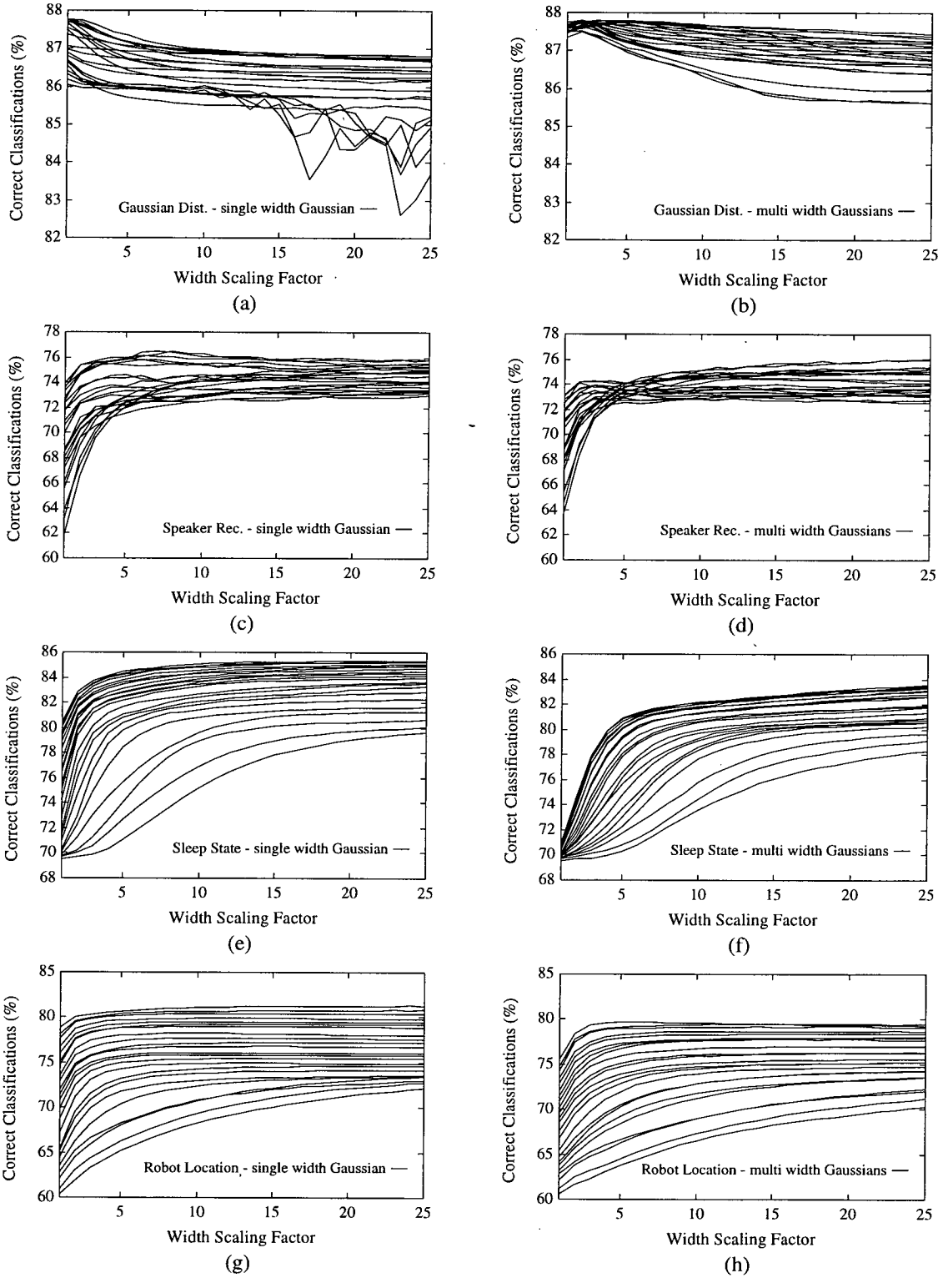


Figure 7.9 - Classification Performance vs. Width Scaling Factor Graphs

7.4.1. Parameter Quantisation

The following simple routine was used to round the floating point weights to the nearest equivalent fixed point "bit" in the specified weight range.

for all centre components{

/ quantise value to nearest integer in "range" */*

$$scaled\_value=(int)\left[\left(\frac{normal\_value * quant\_level}{weight\_range}\right)+0.5\right];$$

/ rescale rounded value back into original range */*

$$quantised\_value=\frac{(double)scaled\_value * range}{quant\_level}$$

}

Note that the recipe shown here is used for quantising the unipolar centre positions and the value of *weight_range* for this task is 3.0. Since the output weights and thresholds are bipolar, it was necessary to determine whether each of these parameters was positive or negative **before** quantising them. The principle for quantising bipolar weights is the same as shown in the recipe above, except that negative weights have 0.5 subtracted from them before they are rounded to the nearest integer. The *weight_range* for the output weights and thresholds was found by simply doubling the magnitude of the largest positive or negative weight.

7.4.2. Classification Performance

Rather than obtain results for many RBFs with different numbers of centres, only networks with 15, 31 and 63 centres were studied. This supposes that PAR chips with these numbers of centres, plus a bias term centre, could be fabricated. The mean and standard deviation of the classification results obtained from the experiments on the test sets of the problems are tabulated in Appendix C.

The results from the quantisation experiments show some interesting features.

- As with the original classification experiments detailed in Chapter 7, the results are problem dependent.
- Classification performance tends to decrease as the quantisation becomes coarser.
- The performance of the networks is greatly decreased, for all but the narrowest curves, when the quantisation of the output layer parameters is decreased from

12-bit precision to 8-bit precision. This degradation is more severe for the wider curves and networks with larger hidden layers.

- The performance of the narrower transistor non-linearities (especially the curves tagged 8, 9 and 10) remains fairly constant for all the problems, despite coarser quantisation.

The decrease in performance with coarser quantisation for networks with wide centres and large hidden layers is due to the system of equations describing the operation of the RBF network being *ill-conditioned*[10] [152]. This phenomenon can be explained with reference to Figures 7.7 and 7.8 - the Distance Interpolation graphs. These diagrams show all the distances calculated in these problems are small, hence all the RBF outputs in the hidden layer will be almost identical for all but the narrowest of curves. Also, adding more and more wide centres tends to "flatten" input space, reducing the severity of the terrain and increasing the number of similar outputs.

If all the hidden layer outputs in the RBF are approximately equal, then large, *precise* weights will be required to allow the discriminating functions in the output layer to differentiate between the classes - indeed, examination of the maximum weights produced for the different non-linearities indicated that the weight range increases as the non-linearities widen.

Taking these effects into account, it can be hypothesised that, when a solution set of output weights and thresholds is obtained to 64-bit floating point precision using the SVD algorithm, it produces very precise, accurate weights for solving the problem. However, because the system of equations represented by equation 7.3 is inherently ill-conditioned for wider widths and larger networks, the performance of these networks depends critically on the values of these weights. Hence, as the quantisation in the output layer becomes coarser, more noise is added to the multiplication process, greatly increasing the likelihood of misclassifying a vector in the ill-conditioned networks.

In contrast, for the narrow curves, there is sufficient variability in the range of the RBF outputs in the hidden layer to allow the system to remain *well-conditioned* despite the increase in quantisation and the number of centres. Hence the classification performance for these networks remains approximately constant. A further effect

[10] An ill-conditioned system is one in which small errors in the coefficients can have a large effect on the solution.

of this increased variability is that all the weights in the output layer of these networks remain small.

Therefore, in summary, it can be concluded from these results that, when using the developed centre circuit as the basis function in a small RBF network, the widths should be kept small since this allows network performance to remain adequate and well-conditioned, despite the limitations introduced by quantising the neural parameters. However, as shown earlier in this chapter, the narrow curves tend to produce the lowest classification performances due to their steep sides.

Furthermore, these experiments have shown that the precision of the output weights in an RBF network has a greater effect on network performance than the precision of the centre locations.

7.5. Software Experiments - Discussion

In general, two trends were noted from the software experiments.

- The performance of RBF networks using the transistor non-linearities with narrow widths are unaffected by reducing the precision of the hidden layer and output layer parameters. However, the narrow widths produced the poorest classification performance in the full precision classification experiments due to their rapid fall-off as I_{dist} increases.
- The performance of RBF networks using wider widths was generally good in the full precision classification experiments. However, these networks were inherently ill-conditioned and the performance degraded as the quantisation of the output layer became coarser.

Thus, it is clear from these experiments that, whilst the developed non-linearity could be successfully used in small, software RBF networks, the level of parameter quantisation on the PAR chip - 12-bit precision in the hidden layer, 8-bit precision in the output layer - will degrade the classification performance of the network. The degradation will be due to either the coarse quantisation in the output layer for wider non-linearities or the rapid fall-off of the narrower non-linearities. The acceptable level of degradation, and hence the chosen value for V_{width}, will be problem specific.

The results also show, however, that if 12-bit precision is used in the output layer, the performance of the reduced precision networks compares very well with that of the full precision network. Thus it can be concluded that the performance of RBFs using the developed non-linearity can be maintained if the full precision output weights are

quantised to 12-bits.

7.6. Summary

This chapter has discussed the software experiments undertaken with a specially developed model of the hardware. The purpose of these experiments was to investigate whether the designed centre circuitry would be capable of solving a variety of classification problems and to discover what effect parameter quantisation would have on the results. In addition to describing the operation of the software simulator and summarising the classification results, each problem was discussed and a couple of interesting observations from the full precision classification experiments explored.

From the results, it was concluded that the transistor non-linearity could be used in implementations of small RBF networks for solving classification tasks when the neural parameters were calculated and stored using 64-bit floating point precision. However, when the neural parameters were subsequently quantised after training, it was discovered that the classification performance of most networks suffered. Networks using narrow hidden layer non-linearities were unaffected by coarser parameter quantisation, however these networks produced poorer classification performances in the full precision trials.

Thus, a hardware RBF implementation with the fixed precision parameter storage of the PAR chip is likely to produce degraded classification performances, on these problems, compared to a full precision software solution. This clearly has implications for the development of application specific chips using the pulsed analogue hardware developed in this work.

Chapter 8

Hardware Results

In the last chapter, the suitability of the two-transistor non-linearity for solving a selection of classification problems using both full precision and quantised weights was investigated. This chapter describes the results and observations from hardware classification experiments using the PAR chip.

Several 2 input, 2 class Gaussian distribution problems, of differing complexity, were used for these experiments. A description of how the problems were cast into hardware is given, along with a discussion of the constraints imposed on the hardware experiments. The Gaussian distribution problems were chosen for the hardware demonstrations not only because they require the minimum external hardware for implementation, but because it is also possible to visualise the distribution of the training vectors and test vectors in input space.

Classification results from the chip, taken both before and after learning, are presented and observations from the learning experiments are discussed. A summary of the conclusions from the hardware experiments is given, and some consideration is given to the issues that arose whilst trying to solve problems on a pulsed analogue RBF chip whose internal circuits had been proven to function correctly.

8.1. Classification with the PAR Chip

The remainder of the work in this thesis attempts to assess the performance of the PAR chip on several problems, demonstrate that it can be trained *in-the-loop* and investigate if any further constraints are imposed on the RBF architecture through implementing it in VLSI.

The object of the hardware experiments was to discover what the classification performances of the chips were for problems of differing complexity using **unadapted** software generated weights, and then investigate if, and by how much, this performance could be adapted using *chip-in-the-loop* learning. The importance of the work is not so much the actual classification rates, but the performance of the hardware network, both before and after learning, compared to the software network.

The results of the hardware experiments will be presented in the next section; the remainder of this section describes how the hardware and the training procedure was configured for them.

8.1.1. Training Data and Test Data

For the hardware experiments, three different Gaussian distribution problems were created. These were labelled *Easy*, *Intermediate* and *Hard* to indicate their complexity.

The *Easy* problem consisted of two separate, distinct, linearly separable distributions. The *Intermediate* problem consisted of two distributions that overlapped slightly, whilst the *Hard* problem consisted of two over-lapping, non-linearly separable distributions, with one distribution embedded in the other. The *Hard* problem is therefore similar to the one used in the software experiments in Chapter 7.

Figures 8.1 to 8.3 illustrate the input space distributions of the training set and test set for the problems. All problems consisted of a 200 vector training set and a 200 vector test set. The circles in the diagrams indicate the range of the distributions: each circle is located on the centre of the distribution and has a radius equal to twice the standard deviation of the distribution.

All the data files were quantised to 12-bit precision before being used to generate the software solution sets. Thus the weights to be downloaded to the PAR chip had already been tailored to the quantised vectors to be presented to the hardware.

8.1.2. Generation of Weight Sets in Software

The software simulator described in Chapter 7 was used to generate the software solutions. As usual, the weight sets were found to 64-bit floating point precision using a combination of the adaptive k-means and the SVD algorithms. This time, however, the quantised training vectors were used in the training process.

Once the simulator was trained, the **unquantised** centre, output weight and threshold values were saved to files, ready to be downloaded to the hardware. These centre, output weight and threshold files were then used by the simulator to classify the training set and test set data. The classification rate and *mean square error* (MSE) value were obtained from the software simulator using both the full precision neural parameters and versions of the centre, output weight and threshold files quantised to 12-bit, 8-bit and 8-bit precision respectively.

By using the software model of the hardware, it was possible to generate several solution sets to the problems (through the use of random number generator seeds) and find their classification performance in software both before and after quantisation. All that remained was to download these solutions sets to hardware and investigate their classification performance there.

8.1.3. Network Set-Up

The PAR chip has 15 centres, so a 2 input, 15 centre, 2 output RBF was the only network considered for solving the problems.

Since the chip has 8 inputs, the PAR Development Board was hardwired such that inputs 0, 2, 4 and 6 all received one component of each 2-dimensional vector, with inputs 1, 3, 5 and 7 all receiving the other. By interleaving the inputs in this way, all the hidden layer circuits were utilised and any lateral across-chip variations should have averaged out.

8.1.4. Chip-in-the-Loop Learning

The hardware was trained using *chip-in-the-loop* learning - a process that allows a PC host to "train" the chip.

The first stage of this learning process involves training an ideal software version of the RBF under consideration to find a solution set of software weights for the problem. This set of centre positions, widths and output weights and thresholds is then downloaded to the hardware. However, using software generated weight sets in a hardware network generally leads to very poor performance: the software network has not been trained to account for hardware idiosyncrasies during learning.

By performing additional training iterations with the chip performing the feedforward calculations and the PC host calculating the weight updates based on a model of the network, the software generated weights can be adapted to the hardware environment, Figure 8.4. This is the second stage of the chip-in-the-loop learning process and usually improves the performance of the hardware.

8.1.5. Training Set-Up

Owing to the vast number of variations that could be made to the training schedules for the hardware, and because of the limited time left for completing the project, a number of constraints had to be placed on the hardware learning experiments.

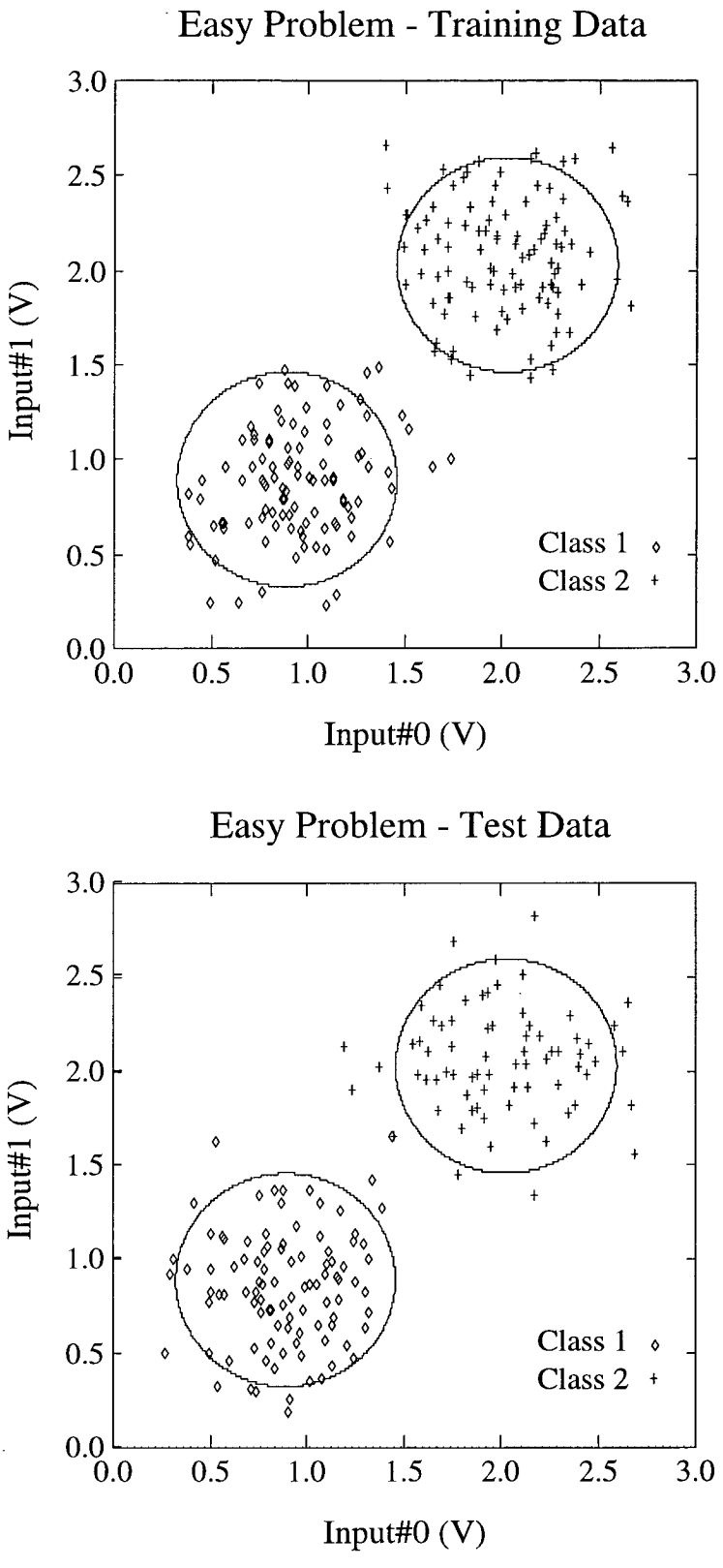


Figure 8.1 - Input Space Distribution of Training Data and Test Data for the *Easy* Problem

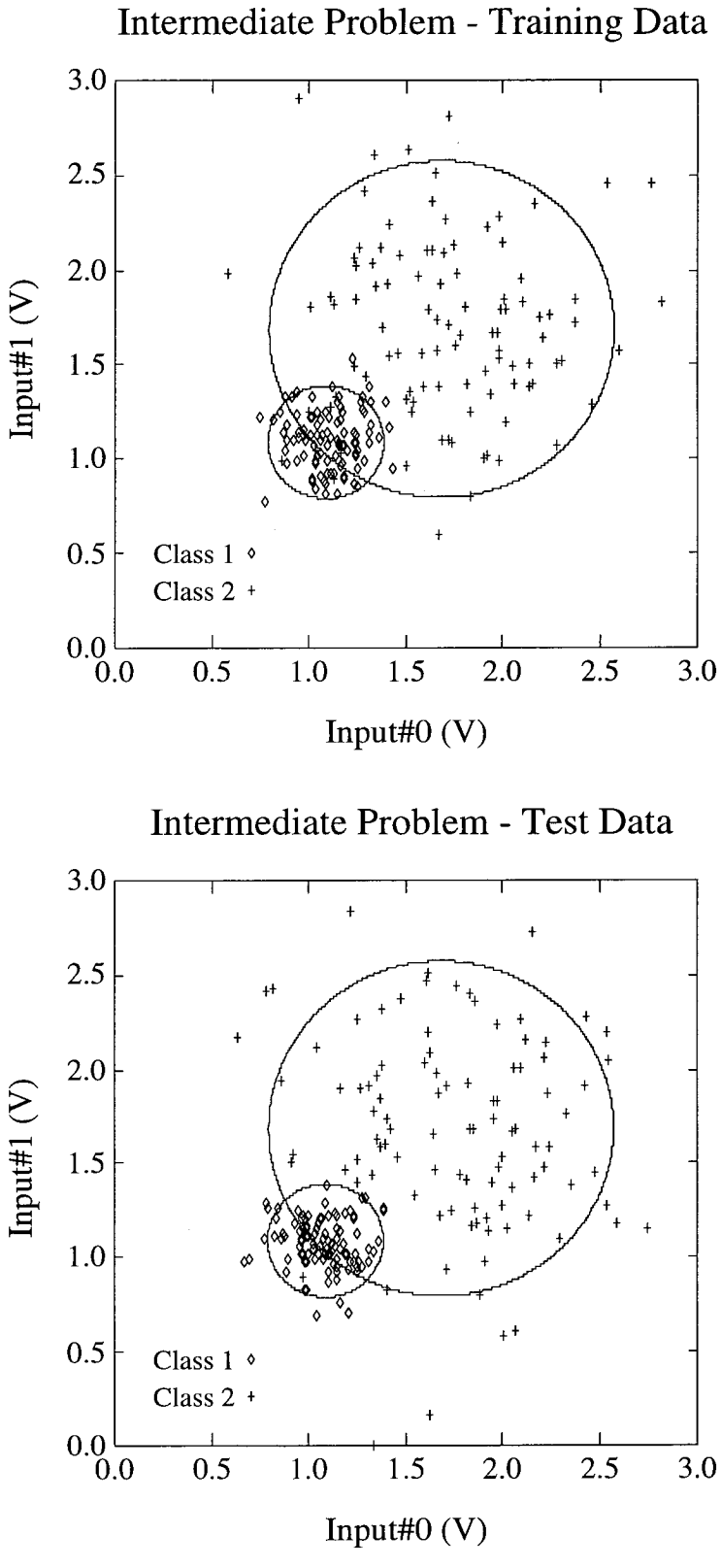
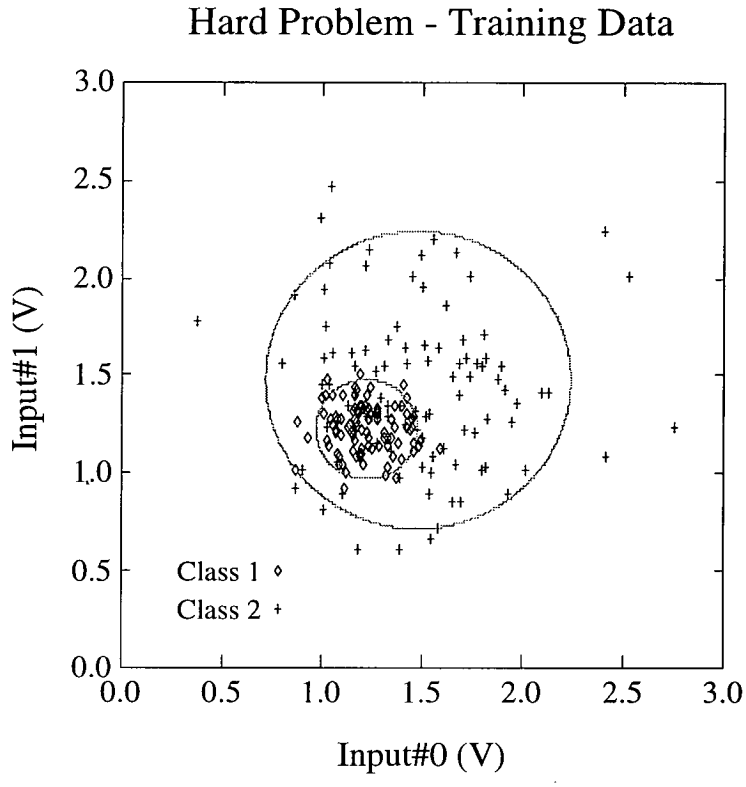


Figure 8.2 - Input Space Distribution of Training Data and Test Data for the *Intermediate* Problem



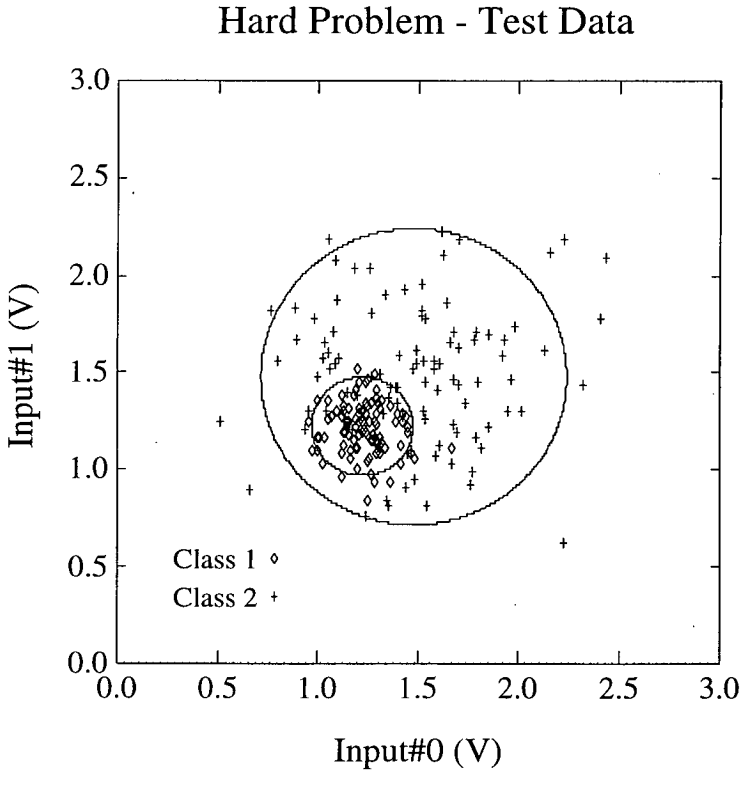


Figure 8.3 - Input Space Distribution of Training Data and Test Data for the *Hard* Problem

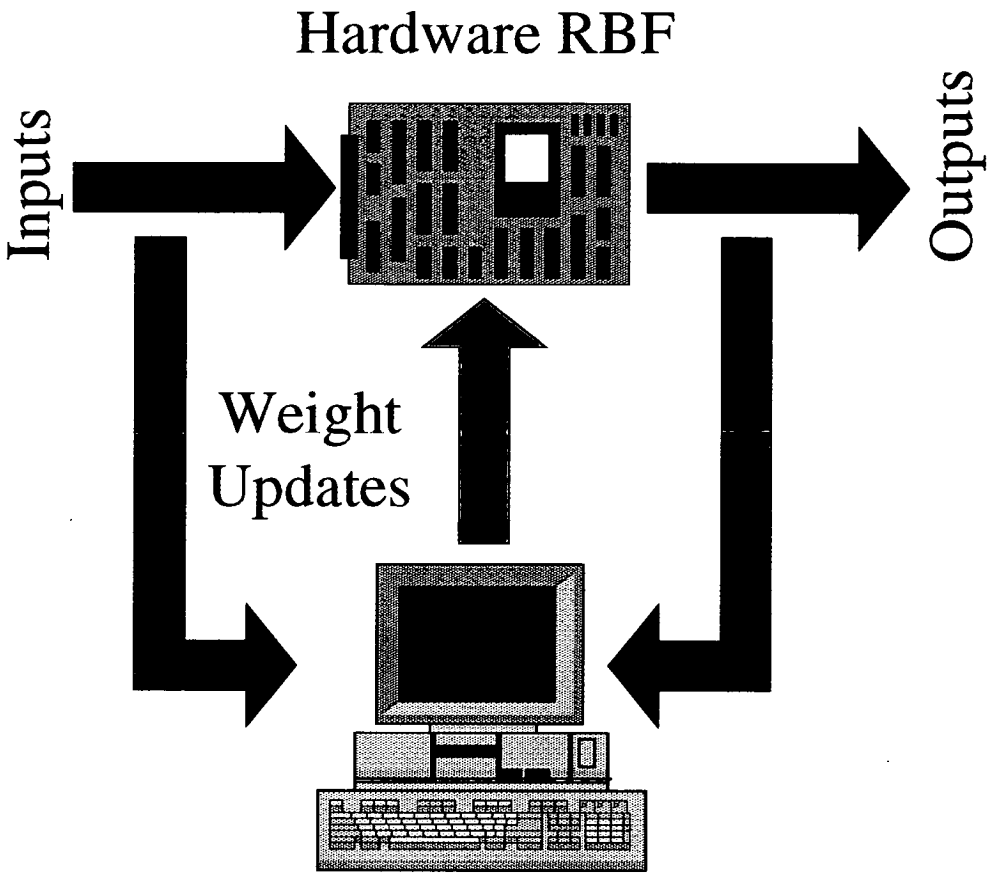


Figure 8.4 - Schematic Diagram of the *chip-in-the-loop* learning scheme

- Only two widths of transistor non-linearity were considered for the hardware experiments, one narrow width and a wider one. These *RBF Output* vs. I_{dist} non-linearities were used to find the original software solutions and subsequently by the PC program executing chip-in-the-loop learning. The non-linearities were presented to the software as look-up tables, having been generated using an HSPICE model of the circuit in Figure 6.9(a) - V_{limit} was set to 3.8V, whilst V_{width} was set to 2.0V for generating the narrow curve and 1.0V for the wider one. Figure 8.5(a) shows the *RBF Output* vs. I_{dist} representation of the non-linearities used by the software, whilst Figure 8.5(b) shows the equivalent *RBF Output* vs. $|V_{in} - V_{centre}|$ non-linearities for the same V_{limit} and V_{width} values[11].

[11] These latter curves have been generated by an HSPICE model of the complete centre circuit in Figure 6.7, assuming there is zero bulk-source voltage.

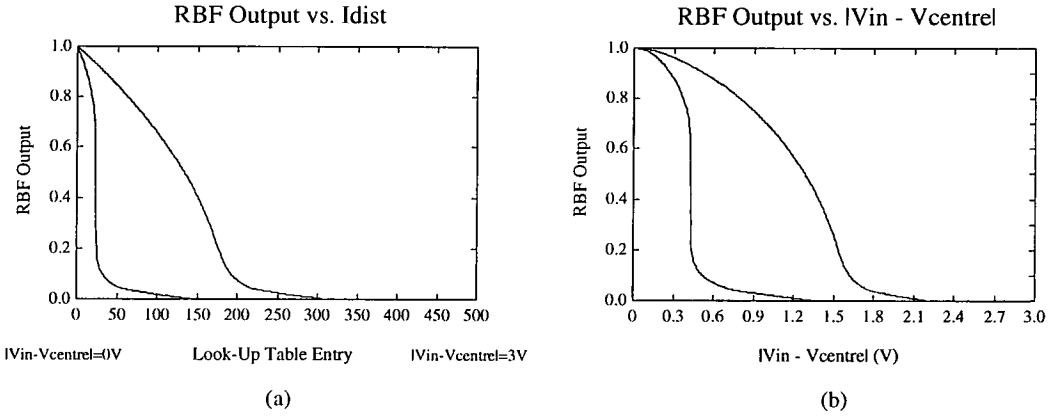


Figure 8.5 - (a) The *RBF Output* vs. I_{dist} graphs used for the hardware experiments and (b) the equivalent *RBF Output* vs. $|V_{in} - V_{centre}|$ characteristic

- Only the output layer weights were altered in the training experiments. This was done for three reasons. Firstly, since the output layer is linear, its error surface is a quadratic function of the weights and an optimal output weight set exists. Gain and offset errors in the hardware were assumed to have moved the minimum of the error surface, thus the software generated weight set corresponds to a non-minimal error for the chip. Further training of the output layer would help each RBF find a better solution, closer to the ideal minimum. Secondly, altering all the neural parameters in both layers using gradient descent would make the training process non-linear, with the same pathologies as similarly trained MLPs. This was not desirable. Finally, the limited time available meant that any training had to be simple if any conclusions were to be reached.
- Batch-mode LMS updating was used to alter the output weights and thresholds. Although this is not an optimal training technique, it provided two distinct advantages over other techniques such as stochastic mode gradient descent or weight perturbation [153, 154]. Firstly, since the weight updates are accumulated over the entire training set, the order of presentation need not be changed between epochs. Secondly, the weight updates are only made at the end of each epoch, significantly reducing the number of write instructions to the output weight RAM and consequently reducing the training times.
- The centres, widths and output weight and threshold values were stored in the PC as 64-bit floating point numbers. Consequently they were only quantised when being downloaded to the development board RAMs. Storing the weights in the PC to this precision helps avoid any update precision problems that may

be encountered during learning [148, 155] without the need to develop new training algorithms [156, 19].

- Finally, to avoid overflow problems in the DACs during learning, all the output weights and thresholds were normalised and clipped to the original weight range before being downloaded. Thus weights could continue to grow in the software model, but were assumed to have saturated by the hardware if they exceeded ±0.5 the original range. (The original range for each software weight set was calculated by doubling the magnitude of the largest positive or negative weight.)

8.2. Hardware Results

In order to assess the performance of the hardware, the neural parameter solution sets generated in software were downloaded to the classification system and the training set and test set for each problem were fed through the network 10 times. The results from these feedforward presentations were accumulated and averaged and are presented as the *Initial* hardware results in the following tables. The output layer of each RBF was then trained and, once this was complete, the same data sets were reprocessed a further 10 times using the new neural parameters. The classification results were again accumulated and averaged and form the *Final* results in the same tables. The hardware results presented in Tables 8.2 to 8.5 are the average results from each set of 10 ten passes, whilst the errors represent ±1 standard deviation of the results[12].

The hardware system used to generate the hardware results is discussed in Appendix D, along with a description of the software used to control the board and process the results. A performance anomaly was found with the software control of the board and this is also discussed in Appendix D.

Experimentation with the classification system confirmed that the LMS training procedure was dependent on the learning rate, η. For these experiments, the learning rates were chosen to allow the network to be trained fairly rapidly and the learning rates used for the different problems are summarised in Table 8.1. Momentum was also added ($\alpha = 0.9$) to ensure that weight changes were influenced by the local terrain of the error surface.

[12] Standard deviations are not presented for the MSE results because the variability in them was less than 0.35% for all the forward passes.

	1V Curve		2V Curve	
Problem	η	Epochs	η	Epochs
Easy	0.01	3	0.001	4
Intermediate	0.05	100	0.01	20
Hard	0.2	100	0.005	20

Table 8.1 - Summary of the Learning Rate and Training Epochs for the Different Problems

The learning rate and number of training epochs were not optimised for these experiments. To ensure rapid training, as high a learning rate as possible was used to show the chip could be trained. Training for the simplest problem was stopped as soon as the problem was solved (100% classification for the training set). Training was terminated on the other networks after a fixed number of epochs. The mean squared error (MSE) and the classification performance of all the networks were monitored during training.

8.2.1. Narrow Non-linearity Results

The classification performance and MSE results for a single PAR chip, using weight sets generated from a single seed, and a V_{width} value of 2V, are presented in Tables 8.2 and 8.3 respectively. Several observations were made from these results.

- There is very little difference in either classification performance or in the value of the MSE of the trained network when full precision and quantised weights are used in the software model of the hardware.
- In the hardware results, the initial classification performance decreases and the initial MSE increases as the problem becomes more difficult.
- When chip-in-the-loop learning is implemented, the MSE for the training set decreases for all three problems. Moreover, the performance on the test set, both in terms of the classification performance and MSE, is similar to the performance on the training set both before and after learning.
- Although the MSE decreases with LMS learning, the total classification performance does not necessarily increase significantly.
- The final MSE produced by the hardware is always much greater than that obtained from the software and the classification results from the trained hardware does not necessarily reflect those obtained in software using the original

Classification Performance - 2V Curve					
Chip 5, Seed 100		Software		Hardware	
Problem	Data Set	Unquantised	Quantised	Initial	Final
Easy	Training	100.0	100.0	95.80 ± 0.48	99.65 ± 0.34
	Test	98.0	98.0	95.05 ± 0.50	99.40 ± 0.21
Intermediate	Training	95.0	95.0	89.05 ± 1.26	91.80 ± 0.59
	Test	92.5	92.5	92.50 ± 0.71	94.95 ± 0.37
Hard	Training	84.5	85.0	72.90 ± 1.10	73.75 ± 0.35
	Test	88.8	88.0	74.35 ± 1.20	74.05 ± 0.37

Table 8.2 - Classification Performance Result Summary for the 3 problems using a single chip, a single seed and a V_{width} voltage of 2V

Mean Squared Error - 2V Curve					
Chip5, Seed 100		Software		Hardware	
Problem	Data Set	Unquantised	Quantised	Initial	Final
Easy	Training	$1.453\text{x}10^{-2}$	$1.456\text{x}10^{-2}$	$2.206\text{x}10^{-1}$	$2.038\text{x}10^{-1}$
	Test	$1.890\text{x}10^{-2}$	$1.885\text{x}10^{-2}$	$2.211\text{x}10^{-1}$	$2.038\text{x}10^{-1}$
Intermediate	Training	$4.187\text{x}10^{-2}$	$4.188\text{x}10^{-2}$	$2.418\text{x}10^{-1}$	$1.641\text{x}10^{-1}$
	Test	$3.478\text{x}10^{-2}$	$3.496\text{x}10^{-2}$	$2.407\text{x}10^{-1}$	$1.568\text{x}10^{-1}$
Hard	Training	$1.175\text{x}10^{-1}$	$1.175\text{x}10^{-1}$	$2.440\text{x}10^{-1}$	$2.166\text{x}10^{-1}$
	Test	$9.796\text{x}10^{-2}$	$9.776\text{x}10^{-2}$	$2.434\text{x}10^{-1}$	$2.112\text{x}10^{-1}$

Table 8.3 - MSE Result Summary for the 3 Problems using a single chip, a single seed and a V_{width} voltage of 2V

software weight sets.

- The hardware performance for the most difficult problem after training is far poorer than the software performance.
- When using the hardware to process the data sets, performance variations in both the classification performance and the MSE were noted when using both the original software generated weight sets and the weights sets obtained after training.

The significance of these results will be discussed in greater depth later.

8.2.2. Wider Non-linearity Results

The results obtained for a single chip using weight sets generated from a single seed for all three problems and a V_{width} value of 1V are presented in Tables 8.4 and 8.5.

The following observations were made from these results.

- Unlike the results obtained when V_{width} = 2V, there are greater discrepancies between the unquantised and quantised software results, for both the classification performance and the network MSE. Furthermore, the discrepancy increases as the problem becomes more difficult.
- For the training conditions imposed on the hardware network, there is a significant increase in classification performance, but only a relatively small decrease in MSE after chip-in-the-loop learning.
- The initial hardware classification performance is far poorer for all three problems when the wider non-linearity is used.
- Classification performance and MSE performance variations were again noted in all the forward pass experiments.

8.2.3. Chip and Seed Variations

To investigate the performance variations, under the same training conditions, for different chips or when different random number generator seeds were used, further hardware experiments were carried out using different seeds and different chips to solve all three problems. For these experiments, V_{width} was fixed at 2V.

The results from these experiments are tabulated in Appendix E and the observations are summarised below.

- The final classification performance, final MSE and the variation of the MSE during learning all depended on the chip or seed used in the experiment.
- Although every experiment produced a decrease in the MSE for the network, again the classification performance need not improve for the *Intermediate* or *Hard* problems.
- Quantising the weights in software made very little change to either the classification performance or the MSE performance. These results were in agreement with those obtained for the initial experiments listed in Tables 8.2 and 8.3.

Classification Performance - 1V Curve					
Chip 5, Seed 100		Software		Hardware	
Problem	Data Set	Unquantised	Quantised	Initial	Final
Linear	Training	100.0	99.0	73.80 ± 2.00	99.50 ± 0.53
	Test	99.0	99.0	76.30 ± 1.55	99.10 ± 0.61
Intermediate	Training	95.5	94.0	49.15 ± 1.23	84.00 ± 0.56
	Test	96.0	97.0	48.50 ± 1.31	84.85 ± 0.63
Hard	Training	85.5	72.5	34.90 ± 1.73	69.45 ± 0.64
	Test	79.5	73.0	34.80 ± 2.99	72.15 ± 0.75

Table 8.4 - Classification Performance Result Summary for the 3 Problems using a single chip, a single seed and a V_{width} voltage of 1V

Mean Squared Error - 1V Curve					
Chip5, Seed 100		Software		Hardware	
Problem	Data Set	Unquantised	Quantised	Initial	Final
Easy	Training	1.414×10^{-2}	5.051×10^{-2}	2.383×10^{-1}	2.343×10^{-1}
	Test	1.436×10^{-2}	4.908×10^{-2}	2.380×10^{-1}	2.343×10^{-1}
Intermediate	Training	5.906×10^{-2}	7.800×10^{-2}	2.478×10^{-1}	2.232×10^{-1}
	Test	4.272×10^{-2}	6.104×10^{-2}	2.476×10^{-1}	2.215×10^{-1}
Hard	Training	1.167×10^{-1}	2.501×10^{-1}	2.498×10^{-1}	2.431×10^{-1}
	Test	1.025×10^{-1}	2.198×10^{-1}	2.498×10^{-1}	2.418×10^{-1}

Table 8.5 - MSE Result Summary for the 3 Problems using a single chip, a single seed and a V_{width} voltage of 1V

- The classification results and MSE results again varied between each forward pass when using both the untrained and trained weight sets.
- Software and hardware performance again decreased as the problem became more difficult.
- Again the performance, after training, for the most difficult problem is significantly poorer than that obtained from the software.

8.2.4. Additional Observations

In addition to the observations already listed for the hardware experiments, several others were noted. These are summarised below.

- The performance improvements noted in these trials were all obtained in a short number of epochs using a high learning rate.
- When the same chip was trained under exactly the same conditions, it was observed that, except for small variations due to noise, the MSE decreased in the same way each time, Figure 8.6. For the experiments shown in Figure 8.6, the learning rate was set to $\eta = 0.005$, a momentum term of $\alpha = 0.9$ was used and the network was trained for 50 epochs.

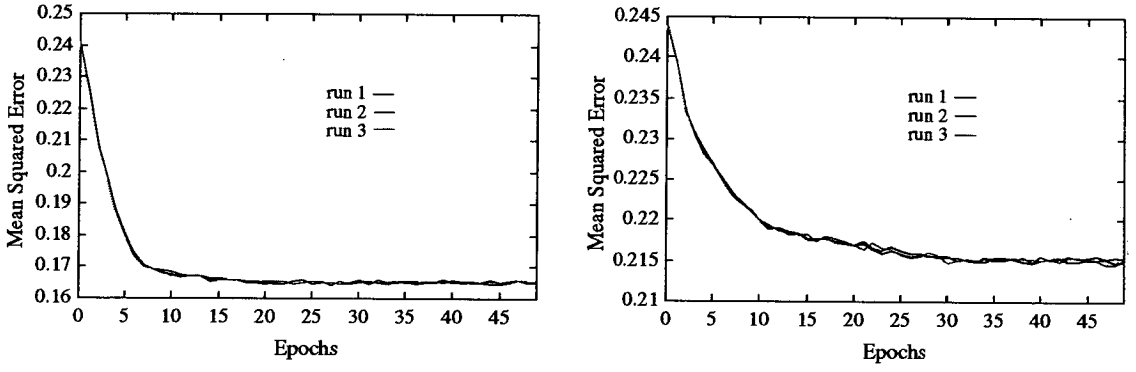
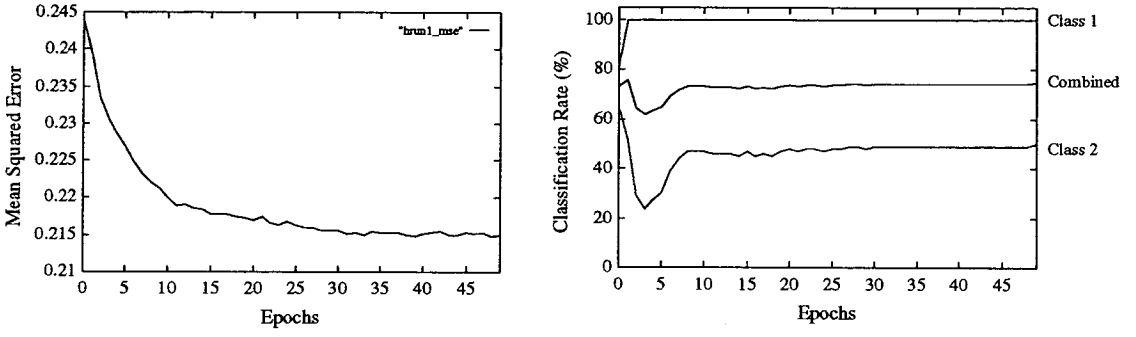


Figure 8.6 - (a) The variation of the MSE for the *Intermediate* and (b) *Hard* problems for several learning runs under the same conditions

- Although the total classification performance for some networks may not appear to have altered significantly as a result of LMS learning, the proportion of each class correctly identified did change radically. Class 1 was always quickly preferred to Class 2 and the proportion of Class 2 identified recovered after an initial drop, Figure 8.7. The *Hard* problem was used for the simulation shown in this figure, with a learning rate of 0.005 and momentum term of 0.9.

8.3. Discussion of the Hardware Experiments

Before drawing conclusions from the observations of the hardware learning experiments, the limitation of the experiments will be discussed along with some of the features of the learning environment.



(a) (b)

Figure 8.7 - (a) The variation of the MSE and (b) the classification rate during hardware training

8.3.1. Use of LMS Learning

The main disadvantage with LMS learning is that the best value for the learning rate, and the number of training epochs, must be determined experimentally. If the learning rate is too high, the training will be quite fast, but will also be coarse since large weight changes will be made at the end of every training epoch. The results of the hardware experiments indicate that the learning rate used was too high.

If a small learning rate was used, the training times in this study would have been prohibitively long, although a superior performance is likely to have been obtained. However, if the learning rate is too small, then the analogue noise present in the hardware could dominate, rendering any weight changes insignificant. The best solution, therefore, would use an adjustable learning rate that starts off large and is reduced as learning proceeds.

So, although these experiments confirm that RBF networks can be cast into pulsed analogue VLSI and can have their performance adapted using LMS learning, the results obtained are unlikely to be optimal. Many further experiments will need to be carried out before definitive results regarding the ability of the hardware to emulate software performance can be obtained.

However, the results using LMS learning are encouraging nonetheless.

8.3.2. Training Times

The training times for the PAR chip were fairly long, with the vast majority of time spent interrogating the Output RAM chip. The long training times were principally due to the slow clock speed of the IBM PS2 286 PC used to control the training

process. By using a faster PC, it will be possible to increase the training speed, allowing more training epochs to be completed in a given time interval. Faster training times will also allow smaller values of learning rate to be tried, without incurring prohibitively long training times.

8.3.3. Biased Classification

It was noted with all three problems that the classification system was heavily biased towards one of the classes - Class 1, the denser cluster in the *Intermediate* and *Hard* problems.

As observed in the experiments, even although the total classification performance did not seem to vary for some of the experiments, the proportion of each class correctly identified did vary. As training proceeded, the classifier quickly learned to identify all of Class 1 correctly, usually at the expense of Class 2. Thus, although the total classification rate for some of the problems did not seem to alter by much, the proportion of each class correctly classified varied significantly.

The biased classification in hardware, Figure 8.7, was also observed when a software version of the LMS algorithm was applied to these problems, Figure 8.8. For the results in this figure, the *Hard* problem was used, the output weights were initialised to small random numbers and a learning rate of $\eta = 0.000005$ was applied without momentum.

Thus, since there are clear similarities between the hardware and software learning, the biased classification was concluded to be a feature of the chosen problems and the LMS learning rule, not of the chip or development board.

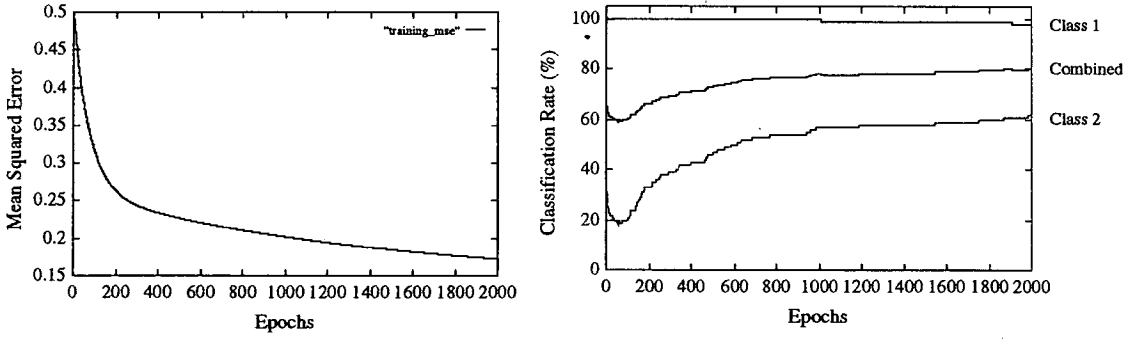


Figure 8.8 - (a) The variation in MSE and (b) the classification rate for LMS learning in software

For these biased classification results, the following hypothesis is proposed by way of an explanation. For the Gaussian distribution problems, misclassifying vectors in the denser distribution (Class 1) leads to a higher cost, for the MSE cost function, than misclassifying vectors in the less dense distribution. Thus, with LMS learning, the greatest reduction in the cost function comes from identifying all of the denser cluster initially, with further reductions occurring as a result of beginning to correctly classify more vectors of Class 2, whilst still correctly classifying all the vectors of Class 1.

8.3.4. Size of the Final MSE

It was noted that the size of the final MSE for all the networks was far larger than the MSEs obtained in software. For the results from the *Easy* problem, this was due to stopping the training once 100% recognition of the training set occurred. However, as shown in Figures 8.6(a) and (b), after an initial decrease, the MSEs for the *Intermediate* and *Hard* problems level off.

The reason for the difference between the hardware and software MSE is due to casting the RBF algorithm into VLSI [157], and using dedicated circuits to approximate the arithmetic functions usually carried out by the CPU in a computer. For the classification system incorporating the PAR chip, the difference in MSE performance is due to a combination of the *strength* of the synapse and the accepted widths for the maximum and minimum pulse widths from the classifier.

8.3.4.1. Synapse Strength

The DYMPLE synapse was designed to be cascadable. Each synapse is connected to the local 5pf capacitance it has been designed to charge or discharge by 1V in 10μs. (A 10μs input pulse corresponds to a hidden layer output neural state of 1.0).

A full positive synaptic weight can charge the capacitor by 1V, whilst a full negative weight discharges it by this amount in 10μs. However, when N synapses are cascaded in a synaptic column, the capacitance to be charged or discharged becomes $5N$pf. This capacitance will only be charged or discharged by 1V if **all** the synaptic weights are fully positive or fully negative respectively and maximum input pulses are applied to all the synapses.

For the problems used in these experiments, the weights in the column were distributed between the maximum and minimum limits. Also, due to the nature of the

RBF paradigm, each input pulse to the multiplier will not have the maximum width. Thus it is impossible to fully charge or discharge the distributed capacitance by 1V during normal operation, ie the synapse is not strong enough, and it is impossible to produce the acceptable maximum and minimum output pulse widths.

8.3.4.2. Acceptable Pulse Widths

For this classifier, the input vector is assigned to the class whose output unit produces the largest width of output pulse. Ideally, the output unit for the correct class would output a maximum width pulse (220 RAM locations) whilst the unit for the other class would produce a minimum width pulse (30 RAM locations). These represent the ideal acceptable maximum and minimum pulse widths the classifier is trained to produce.

However, as discussed above, the synapse was not strong enough to charge or discharge the total output capacitance by an amount that would allow these pulse widths to be generated. Thus for each vector, both output pulses lay between the ideal maximum and minimum widths.

Within the classifier, the controlling software simply assigned the input vector to the class associated with the output producing the widest pulse, and calculated the MSE from the difference between the ideal and actual output pulse widths for the output units. This means the defined limits for the pulse widths have a large effect on the value of the MSE, although the classification performance can be good even if the network MSE is large.

The residual MSE can be reduced if the accepted minimum pulse width is increased and the accepted maximum pulse width is reduced in the controlling software, Figure 8.9.

8.3.4.3. The Effect of Altering the Acceptable Pulse Widths

To investigate the effect the defined pulse width limits had on the residual MSE, the acceptable pulse widths were altered and learning re-started. Figures 8.10(a) and (b) show the variations in network training, using different acceptable pulse widths, for both the MSE and the classification rate.

These results were obtained from the *Hard* problem using a defined maximum pulsed width of 220 RAM Locations and a defined minimum pulse of 30 RAM Locations, a maximum pulse width of 177 RAM Locations and a minimum pulse width of

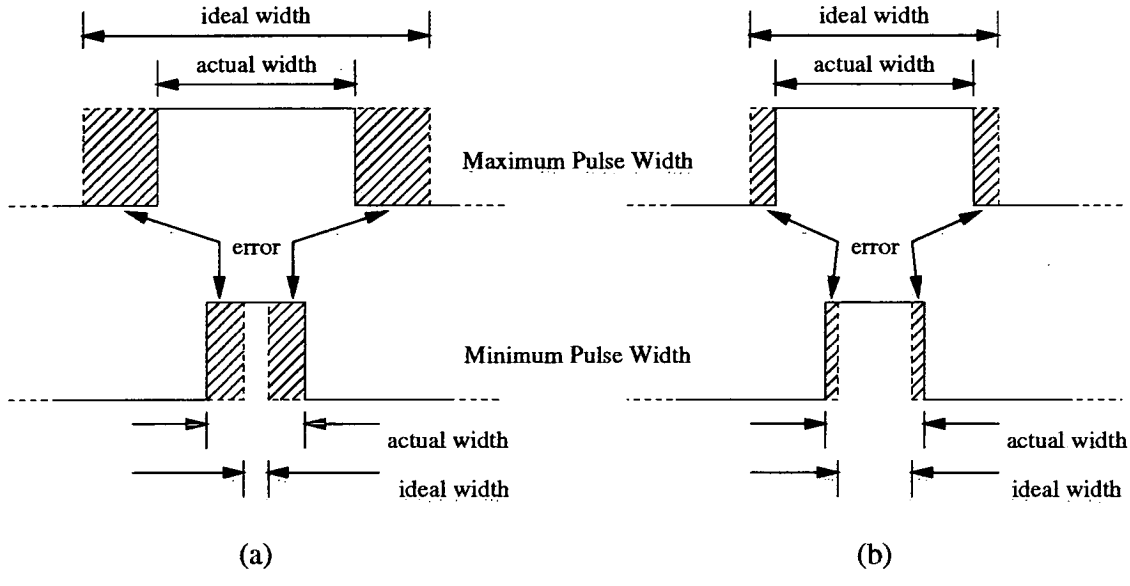


Figure 8.9 - (a) The MSE is calculated by summing the squared differences between the actual and acceptable pulse widths. (b) If the acceptable pulse widths are altered, the MSE can be reduced.

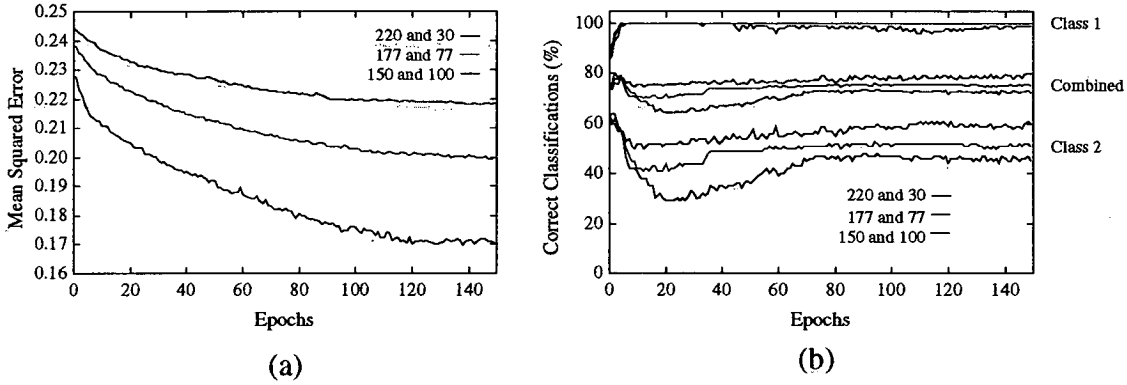


Figure 8.10 - Altering the acceptable pulse widths for correct classification (a) lowers the final MSE and (b) increases the performance for the network

77 RAM Locations, and finally a maximum pulse width of 150 RAM Locations and a minimum pulse width of 100 RAM Locations. Each training run consisted of 150 learning epochs using a learning rate of $\eta = 0.0005$ and a momentum term of $\alpha = 0.9$. Clearly from these results, the values for the MSE decrease and the classification rates increase, whilst following the same trend, as the difference between the maximum and minimum pulse widths is reduced. Indeed, the overall classification rate for

the 150 and 100 RAM Location limits is almost 80%, higher than the performance obtained in the original hardware experiments (Table 8.2) and 5% short of the performance obtained in software for the training set of the *Hard* problem.

Therefore, these results indicate that improved performance can be obtained from the hardware if both the learning rate and the defined acceptable pulse widths are altered. Thus the maximum and minimum acceptable pulse widths will also need to be determined empirically for all problems, further increasing the complexity of applying the chips to specific applications.

The improved performance obtained using the 150 RAM Location and 100 RAM Location limits may not be too surprising, though, as the following explanation suggests.

When the difference between the defined maximum and minimum pulse widths is large, the weight magnitudes will grow as the synapse tries to charge or discharge C_{out} to re-produce the desired output pulse widths and reduce the value of the MSE. However, because the synapse is not strong enough, the actual output pulses will never be longer than 220 RAM Locations nor narrower than 30 RAM Locations. Therefore, only unipolar weight changes will be made, the synaptic weights will never decrease in magnitude and will eventually saturate at the positive or negative extremes of the original hardware weight range. When this occurs, the network performance cannot improve further and the MSE levels off.

Reducing the difference between the defined limits of the pulse widths effectively strengthens the synapse. The voltage changes required to reproduce the defined pulses are smaller, the defined extremes can be exceeded by the actual pulse widths and bi-directional weight changes can occur. Now the synaptic weights do not saturate and increase or decrease in magnitude, as required, to reduce the network MSE and produce the defined output widths. Since the weights do not saturate, the network can continue to improve its performance until the optimal classification rate is reached (ideally) or the VLSI constraints prevent further improvement.

8.3.5. Differences Between Software and Hardware Results

Other reasons for the differences between the hardware and software results could be due to differences between the hardware RBF implementation and the software model used for *chip-in-the-loop* training.

The software uses models (formulae, look-up tables, ideal multiplication and addition) to represent the operation of hardware circuits. The hardware itself uses the characteristics of MOS transistors and voltage ramps to reproduce the RBF operations. It has been assumed during this work that the software model approximates the operation of the hardware. How well this has been achieved has not been investigated. However, if there are built-in differences between the software model and the hardware, it may not be possible to train these out. Fundamental differences between the two include the following.

Precision of the Weight Storage and the Feedforward Calculations - All the weights stored in the hardware have limited precision, as do the feedforward calculations. Operations in the software model are carried out to 64-bit floating point precision using using full precision or quantised parameters. Thus the precision in the hardware may not be high enough to allow it to emulate the software solution for more complex problems.

Shape of the Hidden Layer Non-linearity - The actual hidden layer non-linearity need not be exactly the same shape or height as the non-linearity modelled in HSPICE. Improved learning may be possible by determining the actual shape of the on-chip non-linearity and using this in the software model for finding the initial weights sets and performing subsequent *in-the-loop* training.

Location of the centres - The location of the centres in the hardware is affected by charge injection, whilst those in the software model are not. The centres and widths could also be trained using *in-the-loop* training, although this has not been attempted in this work. Training the centres and widths in this way may help to improve the network performance.

Noise - The hardware is affected by noise, the software model isn't. Whilst noise does not prevent learning, it may stop the hardware from reaching as good a solution as the software.

8.4. Hardware Experiments - Conclusions

Considering the limitations imposed on the hardware experiments, and the features of the classification system and Gaussian problems discussed in the last section, the following conclusions were drawn from the experiments.

- Since a decrease in the MSE occurs for all the training experiments, and the performance of the test set mirrors that of the training set both before and after learning, it was concluded that LMS learning had been successfully implemented, and the hardware could be trained *in-the-loop* to account for hardware non-idealities. This is similar to the results obtained in other studies of implementing neural networks in hardware [12, 63, 158, 71].
- Inherent analogue noise does not prevent learning, in agreement with several other studies [13, 158, 71], but it does affect the classification and MSE performance during separate passes of the data through the network.
- The higher classification performance and lower MSE obtained with V_{width} set to 2V, compared to when it was set to 1V, indicates that the 2V curve is more suited to solving these classification problems than the 1V curve, for 8-bit output layer precision. This is in agreement with the weight quantisation experiments conducted in Chapter 7.
- The results of the quantisation experiments in Chapter 7 indicated that a decrease in performance was likely when the PAR chip was used as a classifier. However, the final hardware results for the *Easy* problem (for both the 1V and 2V curves) and the *Intermediate* problem (for the 2V curve) indicate that the capability of the PAR chip to produce excellent classification results, comparable to full-precision software results, depends to some extent on the complexity of the problem.
- For these experiments, as the problem becomes harder, the difference between the full precision software results and the final hardware results becomes unacceptably large. This inability to replicate the software performance for the more difficult problems was concluded to be due to either using inappropriate parameter limits in the learning system, or to differences between the software model and the actual hardware, principally the reduced precision in the hardware.
- Under the same training conditions, for a given chip and a given initial weight set, network training in terms of the change in the MSE proceeds in the same manner during different chip training runs. However, although the shape of the

MSE reduction is the same, each training run is affected by random noise, so the precise value of the MSE varies between the runs, Figure 8.6.

- The experiments indicated the problems could be solved using different seeds and different chips. However, given that different chips will have different offset and gain variations to be trained out, it is recommended that each chip is trained individually, as also stated in [158].
- Whilst it has been proven that RBFs can be implemented in pulsed analogue VLSI, from the discussion in this chapter it is apparent that questions remain regarding the effective training of pulsed hardware RBF chips *in-the-loop* and their applicability to real world applications. Further research should now be directed towards studying the training of hardware implementations of Radial Basis Function neural networks. It is recommended this research concentrates on investigating the learning algorithms used for in-the-loop training, and studying the validity of the software models used, and the assumptions made, when simulating the operation of the hardware for training purposes. Fruitful research in this area will allow a fuller understanding of the constraints pulsed analogue VLSI places on the RBF architecture, and this will subsequently allow the true potential of hardware RBFs, for real applications, to be assessed.

8.5. Summary

This chapter has presented the results obtained from the classification experiments carried out with the PAR chip.

The results show the PAR chip is capable of being trained to solve a range of classification problems, but that its performance is variable and appears to depend both on the complexity of the problem and the parameters used in the hardware learning algorithm.

Probable reasons for the discrepancies between the hardware and software results were presented, and some inherent features of the classification system and the Gaussian distribution problems were discussed.

Further, it was concluded that, whilst the capability of using the developed pulsed RBF chip as a classifier had been demonstrated by this work, issues remained regarding the optimal training of the developed hardware *in-the-loop*. Only by addressing these issues can the true potential of pulsed RBFs for real applications be assessed.

Chapter 9

Summary and Conclusions

This chapter draws together all the work detailed in the preceding chapters, summarising the work completed at various stages of the project and presenting the conclusions reached at each stage.

9.1. Overall Project Summary

This project has investigated the circuit, system and operational issues affecting the implementation of the Radial Basis Function neural network architecture in pulsed analogue VLSI. In order to realise this aim, the goal of the work was the production of a functioning RBF demonstrator chip that was able to solve classification problems. The production of the demonstrator and its application as a classifier served as a vehicle for the investigation.

To achieve the project goal, several hybrid pulsed circuits, operating using PWM, were designed and developed. Furthermore, to investigate the functionality of the circuits, assess their operation and highlight any potential problems, the DYMPLES and RHO test chips were fabricated and tested.

After drawing appropriate conclusions from experiments conducted on the two test chips, some circuit modifications were made and the final demonstrator chip - the PAR chip - was fabricated. Subsequent hardware measurements were made using the PAR chip to assess the functionality of the modified circuits.

Software experiments were performed to investigate the performance of the developed two-transistor non-linearity circuit and see how it compared with Gaussian non-linearities when both were used as the hidden layer non-linearity in small RBF networks solving a selection of classification problems.

Subsequent software experiments were also performed to investigate how the classification performance of the hardware network would alter if the hidden layer and output layer neural parameters were quantised. From a consideration of the results from the quantisation trials, it was possible to determine which of the non-linearities should give the best on-chip performance.

Neural parameters for a small number of networks were then generated in software for a number of two-class Gaussian distribution problems. These parameters were downloaded to the PAR chip and initial classification results from the hardware were recorded. Subsequent classification measurements were also made after the output layer of PAR chip had been given some additional training using a *chip-in-the-loop* implementation of the LMS algorithm.

From the hardware experiments, it was possible to draw conclusions about the performance of the hardware, compare it to the performance of the software model and hypothesise as to the causes of the observed differences.

The development of the PAR chip and its application to 2-class classification problems realised the goal of this project. As a direct result, it was also possible to fulfil the aim expressed in Section 1.4.

9.2. Detailed Summary and Conclusions

This section summarises in more detail the issues addressed and the work undertaken at each stage of the project. The conclusions reached as a result of each piece of work are then presented.

9.2.1. Introduction and Background

Summary

The concept of discriminant functions in pattern classifiers was reviewed as a prelude to introducing the MLP and RBF neural architectures. The operation of both neural paradigms was discussed, along with methods for training them.

Having introduced the theoretical background, the operating modes of the CMOS transistor was discussed. The motivations for producing digital and analogue hardware neural networks were reviewed and examples from the literature of complete CMOS RBF networks, or their constituent parts, were presented.

The hybrid *pulse stream* neural technique was then reviewed, and circuits from the EPSILON Cell Library were used to illustrate the elegance of both PWM and PFM.

Finally, the motivations for this project were discussed and the scope of the investigation was defined.

Conclusions

Clearly RBF networks can be implemented in both analogue and digital VLSI. However, since a pulsed RBF had not yet been attempted, this project was of academic interest from the outset. Further, the developed circuitry would allow the use of pulsed RBFs in classification problems to be assessed and this could have implications for other analogue implementations.

Another benefit of this work would be the further development of the pulse stream neural technique.

9.2.2. Circuit Issues

Since this heading covers a large proportion of the work covered in this thesis, it has been sub-divided to allow each chip to be presented individually.

9.2.2.1. The DYMPLES Chip

Summary

The DYMPLES chip was designed and fabricated to allow the functionality of the current-mode synapse, developed to implement the output layer of a pulsed RBF, to be tested and assessed.

The DYMPLE synapse is a current-mode, PWM, two-quadrant multiplier. It uses dynamic current mirrors (DCMs) to store local copies of currents produced by a global, on-chip, current DAC. These currents are then used to selectively charge or discharge an output capacitance and a comparator is used to produce the output pulse. The synapse was designed to be simple, easily set-up and operated and easily transferred between different fabrication processes.

Conclusions

Measured results from the DYMPLES chip indicated that the synapse, DCMs and on-chip DAC all functioned as required. Additionally, the simulated and measured results from the circuit indicated that the use of DCMs allowed process variations to be accounted for implicitly, as well as increasing the on-chip integration of the pulse

stream technique. It was concluded that the synapse is a valid design of two-quadrant multiplier and could therefore be used to implement the output layer of a pulsed RBF chip.

9.2.2.2. The RHO Chip

Summary

The RHO chip was designed and fabricated to test and assess the operation of circuits developed to implement the basis function layer of an RBF chip. This chip consisted of two centre circuit arrays, both based on a distance circuit previously produced from this research group and published in the literature [143]. One array consisted of circuits using a capacitor-based technique for generating the hidden layer non-linearity, whilst the other consisted of a circuit that exploited the natural physics of MOS transistors.

Both centre circuits were designed to have an easy interface to the outside (analogue) world, be easy to set-up and operate, use PWM and interface easily to the DYM-PLES circuit.

Conclusions

Experiments on the RHO chip indicated that all the developed circuits functioned as required. However, discrepancies were noted in them all.

Distance Circuit - It was concluded that the discrepancies in the distance circuit were due to the use of an inappropriate layout technique and the use of global, as opposed to local, compensation circuits. Furthermore, the observations from the capacitor and transistor non-linearities could also be attributed to these effects.

Capacitor Circuit - Results from the capacitor array indicated that the ability of each chip to produce a Gaussian non-linearity varied and often depended on the time allocated to discharging the output capacitor. This latter effect was due to offset currents produced by the distance circuit and these currents would have more serious consequences if the capacitor size were scaled down.

Transistor Circuit - Offset currents had a negligible effect on the transistor-based array and the main discrepancy with this circuit was the variation of the shape of the

non-linearity both across-chip and between chips. However, it was believed that this variation could be accounted for in subsequent designs.

From a consideration of the requirements of the capacitor circuit and transistor circuit arrays, in addition to the experiences and observations acquired from setting up and experimenting with the RHO chip, it was concluded that the transistor-based non-linearity offered the greater potential for the final RBF chip.

9.2.2.3. The PAR Chip

Summary

The PAR chip was the final demonstrator chip fabricated for this project and consisted of the hidden layer and output layer circuits developed for the two test chips. The PAR chip was designed as a small, non-trivial RBF network that could be applied to classification problems.

Some improvements were made to the design and layout of the centre circuit and the correct operation of the modified version was demonstrated via HSPICE simulation. The DYMPLES circuit also required to be re-designed for the MIETEC 2.4μm fabrication process.

Since the PAR chip was to implement a complete RBF, consideration was given to system issues such as network size, on-chip DAC precision, chip refresh scheme, and storage of the V_{width} voltage for each centre.

The PAR chip was finally configured as an 8 input, 16 centre, 4 output RBF network. One centre on each chip was configured as the bias unit for the output layer and always produces a maximum width pulse.

Conclusions

Measured results indicated that all the circuits on the PAR chip functioned as required, and correct performance of the chip as an RBF network was therefore expected.

DACs and DCMs - The variation between the 8-bit DACs on the different chips was very low, however the output current was non-monotonic. This was due to the simplicity of the design: it only comprises two 4-bit DACs connected by a 16:1

attenuating current mirror. Unfortunately this 16:1 ratio is not being implemented accurately enough. The DAC non-monotonicity permeated through to the NMOS DCMs, which, along with the PMOS DCMs, otherwise functioned as expected.

Transistor Non-Linearity - The two-transistor circuit was observed to exhibit a good static response, which was repeated and well-matched over all the centres on 80% of the chips. However, the shape of the non-linearity varied between chips. Further, a constant displacement of the centre of the non-linearity from the ideal value of V_{centre} was observed on all the chips. The size of the displacement varied between chips and was due to charge injection, from the access transistors, onto the capacitor storing V_{centre}.

DYMPLE Synapse - The DYMPLE synapse again exhibited good performance and was linear with variations in both the applied input pulse width and the loaded synaptic weight. The ease with which the synapse was re-designed for the new process, and its proven performance in the new technology, highlighted that the design was indeed simple to transfer between different fabrication processes. Furthermore, although a discrepancy still existed between the positive and negative weights, none of the synaptic columns on any of the PAR chips produced a weight dependent offset error as witnessed on the original DYMPLES chip.

9.2.2.4. Overall Circuit Conclusions

This work has highlighted that it is possible to implement the constituent operations required for RBF neural networks in pulsed analogue VLSI, provided adequate care is taken when designing the circuits and laying out the chips.

Furthermore, this work has also allowed the *pulse stream* technique to be extended through the development of:

- a current-mode synapse, using an on-chip current DAC, which implicitly accounts for process variations and allows greater on-chip integration of pulsed circuitry
- a centre circuit which produces a static, instead of a dynamic, voltage for the pulse generating PWM neuron
- a neural network implementation that requires only linear ramps, which increases the potential for even greater on-chip integration since linear ramps can be easily generated on-chip, eg [159].

9.2.3. Basis Function Issues

Summary

Having developed a novel radial basis function non-linearity, it was necessary to have an indication of its likely performance when it was used in RBFs.

To assess the performance of the non-linearity, it was used in small RBF networks applied to classification problems. The performance of these networks was then compared to identically trained RBFs which used Gaussians. A specially developed software simulator of the hardware was used to perform these experiments.

For these trials, the transistor-based non-linearity was modelled as a look-up table extracted from an HSPICE simulation of the circuit. Also, to ensure that the results were consistent, the performance of both types of non-linearity on several classification problems, both real and artificial, was studied.

Conclusions

The overall conclusion from the classification experiments was that a small RBF using the non-linearity derived from the transistor-based circuit was capable of performing at least as well as a Gaussian-based RBF, when both were implemented in software to 64-bit floating point precision. These results are in agreement with those from other studies that claim the precise shape of the non-linearity is not important [107, 108]. A necessary proviso to the results from this work however, is that the non-linear function used was still a "bump" function: localised, monotonic and differentiable.

However, it was also concluded from these simulations that, when the distance measures were continuously interpolated onto shallow regions of the transistor non-linearities, the network performance was better than if the distance measures were continuously interpolated onto steep regions. This suggests that, whilst the non-linearity need not be a well-defined mathematical function, the gradient of the "bump" function used is important.

Further, it was found that increasing the width of the Gaussians led to an increase in the initial classification performance, which levelled off as the widths increased further. This suggests that the simple "distance to nearest neighbour" heuristic, often

used for determining the widths in Gaussian RBFs, may not be suitable in RBFs possessing a small number of centres.

9.2.4. Parameter Quantisation Issues

Summary

To study the effect of parameter quantisation on the software model of the hardware, additional software experiments were carried out. Parameter solutions were obtained for all four classification problems using three different sizes of RBF network. By subsequently quantising the hidden layer and output layer parameters of these neural solutions, it was possible to investigate how parameter quantisation, to different levels, affected the performance of trained RBF networks.

Conclusions

These experiments indicated that quantising the output layer to 8-bits had the biggest effect on the classification performance of the classifier, severely degrading the performance of those networks using wide non-linearities and with large hidden layers. Quantisation of the output layer parameters to 16 bits or 12 bits produced some slight performance degradation, as did quantising the hidden layer parameters to 16 bits, 12 bits or 8 bits.

From a consideration of the results, it was concluded that, due to the distances calculated in these problems, use of wide non-linearities led to numerical ill-conditioning. This ill-conditioning was exacerbated by increasing the number of centres in the hidden layer, or increasing the width of the non-linearities.

Meanwhile, the networks using narrower centres were well-conditioned and were unaffected by coarser quantisation of the neural parameters. However, a degraded classification performance was obtained from the networks using narrow non-linearities in the full precision classification experiments.

These results indicated that an inter-dependence existed between the shape of the basis function non-linearity and the precision of the weight storage in the output layer. Specifically, if narrow widths were used in the hidden layer, 8-bit precision could be used in the output layer, without performance degradation. However, if the

wider non-linearities were used, 12-bit precision was required in the output layer.

Therefore, it was concluded from this work that, if the PAR chip was used as a classifier for any of the four problems studied, its performance would be poorer than that obtained from a full precision software simulation. This performance degradation would be due to either the shape of the non-linearity (if narrow widths were used) or the ill-conditioning of the network (if wider widths were used). The tolerable degradation will be problem dependent, however pulsed RBFs using the developed non-linearity and 8-bit precision in the output layer may not be suitable for practical solutions to real problems.

9.2.5. Hardware Performance Issues

Summary

In order to investigate the functionality and potential of the developed RBF hardware, and explore some of the operational issues of pulsed RBF hardware, the PAR chip was used in a hardware classification system. Three Gaussian distribution problems, of different complexity, were applied to this system.

Conclusions

The hardware experiments confirmed that the PAR chip was able to act as an RBF classifier. It could solve classification problems of different complexity and have its performance adapted by LMS learning on its output layer parameters.

Although it was observed that a performance comparable to that from full precision software could be obtained from the simpler problems, the performance of the network, compared to the software results, fell as the problem became more complex. Whilst the reason for the performance drop is not fully understood, several explanations were hypothesised.

Although LMS learning was observed to cause a reduction in the MSE in every experiment, this did not necessarily cause a significant improvement in the overall classification performance of the network. LMS learning did cause a radical change in the proportion of each class correctly identified, though, and this phenomenon was shown to be attributable to the problem and learning rule rather than the hardware

system. However, the use of LMS learning in future developments of this work is not recommended.

Further experiments also showed:

- learning was repeatable on a given chip, under the same conditions
- different chips should be individually trained
- different seeds produce different solutions to the problem.

Finally, although the hardware experiments determined the potential of the developed system, questions remain with regard to optimally training the developed hardware for real applications and assessing the limits of the constraints that pulsed analogue VLSI places on the RBF paradigm.

9.3. Comparison with Other Implementations

A comparison of the PAR chip with other RBF implementations cited in the literature is given in Table 9.1.

Implementation	Inputs	Centres	Outputs	λ	Size (mm)	Power
PAR Chip (Pulsed)	8	16	4	3.0	6.5x4.8	35mW
WRBF Chip (Pulsed)	8	16	8	1.0	5.2x4.9	75mW
Kirk *et al* (Analog)	8	159	4	2.0	2.2x9.6	2mW
Collins *et al* (Analog)	32	150	16	2.0	10x10	0.5W
Ni1000 (Digital)	256	1024	64	0.8	16x14	5W

Table 9.1 - A comparison of different RBF implementations[13]

As can be seen from this table, analogue RBF implementations currently have an order of magnitude more centres per chip, and digital implementations two orders of magnitude more centres, than the pulsed RBF implementations. Using the cell sizes implemented on the PAR chip - 248μm by 168μm for the centre circuit and 404μm by 168μm for the DYMPLES circuit - it is estimated that a 16 input, 64 centre, 8 output PAR chip would require a core area of 10.752mm by 6.720mm, whilst dissipating less than 130mW. The DAC, ZCM and PWM neurons have not been included in this

[13] In this table, the estimated sizes are presented for the Collins chip [102]. The parameter, λ, is the minimum dimension size for the chosen fabrication process.

calculation, but, assuming a square chip is required, can be implemented in the remaining 10.752mm by 4.032mm of silicon. By moving to a smaller geometry process and re-designing the circuits, more centres could be implemented on a chip of this size.

However, although the desire for most implementations has been to implement as many centres as possible on an RBF chip, little consideration has been given in the open literature as to what precision these circuits require, what range of widths the basis function circuits can produce compared to the range of input space, or what performance can be obtained from software models of the system.

The work in this thesis has shown that all these issues are critical to the design of hardware RBF solutions. Therefore, until a need for a 64 centre pulsed RBF chip is identified, it is not recommended that one is designed.

9.4. Implementation Issues for Pulsed Analogue RBFs

So what issues need to be considered before RBFs are designed in pulsed analogue VLSI for real problems ?

Obviously, it must be established that:

- a hardware solution is the best option for the problem
- pulsed analogue VLSI is the best implementation technique for the solution
- the specifications with respect to performance, power dissipation and chip area can be met.

These issues are generic to any potential neural hardware solution and do not form a part of this thesis.

However, assuming that a pulsed RBF is required, the following issues must be addressed.

- Can suitable circuitry be designed to reproduce the operation of the centres and output units in an RBF, including a basis function of suitable shape and width ?
- Can the problem be solved in simulation, within performance specifications, using models of these circuits ?
- What precision is required in the hidden and output layers of an RBF using the developed basis function non-linearity and can this precision be realised in pulsed analogue VLSI ?

- Can the resulting network be trained to account for process variations and how long will the training take ?

This investigation has shown that it is possible to realise the constituent operations of RBF networks using pulsed analogue VLSI, provided that adequate care is taken whilst designing the circuits and laying out the chip. However, as shown by the subsequent software and hardware classification experiments, circuit functionality is no guarantee of system performance.

It has been shown that the precision required in the output layer of an RBF depends on the shape and width of the basis function used in the hidden layer. Specifically, whilst 12-bit precision was shown to be acceptable for the output layer in this work, an output layer precision of 8-bits was unacceptable for solving most of the problems. However, it is believed that the relationship between the basis function shape and output layer precision will be problem dependent. Thus, whilst it is impossible to make firm conclusions here, it is reckoned that individual, as opposed to generic, solutions will be required for real problems - requiring a thorough, and potentially time-consuming, design process in each case.

Further, this work has indicated that, whilst LMS learning was adequate to highlight the potential of the developed system, it should not be used for further developments of this work, and it is recommended that the use of other hardware training algorithms is investigated.

9.5. Further Work

Having considered the pertinent issues for implementing pulsed analogue RBFs, the following areas have been identified as suitable areas for further work.

9.5.1. Circuit Level

Work yet to be tackled includes attempting to reduce the area of the developed circuits, using floating gates to store the neural parameters within the hidden layer and output layer cells and investigating the effect of temperature on the circuit performance.

Since it has been shown that the non-linear characteristics of MOS transistors can produce suitable basis function non-linearities, the use of other shapes of transistor non-linearity can be investigated.

In addition, the operational characteristics and performance of combinations of circuit ideas from the literature can also be studied. For example, combining the two-transistor circuit from this work with the Euclidean distance cell of Collins *et al* [84], yields a pulsed RBF centre circuit that can be realised using only 4 transistors, Figure 9.1.

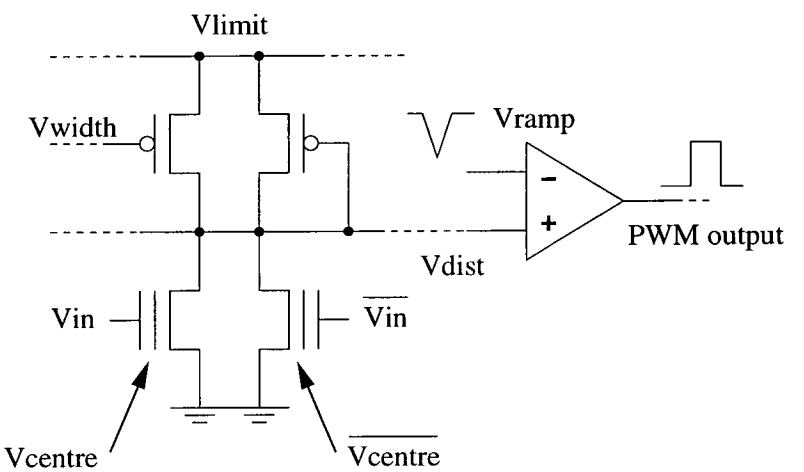


Figure 9.1 - Schematic Diagram illustrating how the Two-Transistor Circuit could be combined with the Euclidean Distance Cell of Collins *et al*

9.5.2. System Level

From this work, it is clear that the inter-dependence between the shape of the basis function and the precision of the output layer in an RBF implementation is critical to the successful operation of the system. Therefore it will be necessary to investigate what precision is required for the output layer in a range of RBF applications, for several transistor non-linearities, and determine if this precision can be realised in pulsed or conventional analogue VLSI. Such an investigation could begin with a determination of the minimum output layer precision required for each of the four problems used in this work.

Also, since the experiments in this work quantised the parameters after training, the use of limited precision software LMS learning should be investigated to determine whether it could compensate for a low precision output layer.

9.5.3. Operational Level

Further work on the operational issues of the developed system should be aimed at investigating the actual precision achievable in the hidden and output layers of the chip (12-bit and 8-bit precision has been predicted) and then determining the hardware performance that can be obtained with the PAR chip, using this precision, for a range of problems. This should allow an assessment of whether the software performance, for the reduced precision parameters, can be realised in the developed hardware.

Also, the extent to which differences between the hardware and its software model affects the training process can be investigated by developing software that models the hardware more realistically and using it for both off-line and *in-the-loop* training.

For undertaking these additional trials, however, it is recommended that a fast processor is used and that the performance anomaly on the board is corrected.

Assuming that a faster processor, with more memory, is available, then it should be possible to estimate the output weights for the PAR chip using SVD on the **measured** hidden layer outputs produced by the hardware. The hidden layer positions and widths of a trained network can be downloaded to the PAR chip, the training vectors can then be presented to the trained hidden layer and the **actual** outputs read off the chip and into RAM via the switching arrangement shown in Figure 6.24. The hidden layer pulses can then be read back from the off-chip RAM and appropriately scaled to produce the hidden layer response matrix (Φ in equation 7.3).

Using SVD to invert the matrix of actual hidden layer responses should lead to an output weight set that already accounts for any non-idealities in the trained hidden layer. The estimated output weights can then be scaled in software and downloaded to the chip and, if required, further chip-in-the-loop training can be carried out to try and achieve further performance improvements.

Adapting the hidden layer parameters, to investigate if learning could be improved using fully supervised techniques, can also be attempted.

Other learning algorithms, suitable for hardware RBFs, should also be investigated to prevent consuming too much time optimising the LMS learning rate for each problem. These new learning algorithms may also lead to some performance improvements.

9.6. Final Conclusion

This thesis has investigated the circuit, system and operational issues raised when the RBF neural architecture is implemented in pulsed hardware and applied to real problems.

It has been shown that, provided adequate care is taken whilst designing the circuits and laying out the chips, the constituent RBF operations can be realised in pulsed analogue VLSI.

However, simply reproducing these operations on silicon does not guarantee system performance and it was shown that the inter-dependence between the shape of the basis function and the precision in the output layer is critical to the operation of the hardware.

Further, the development of a robust learning environment and an understanding of the constraints imposed by the implementation technique are also essential to the final development of a working RBF system.

Appendix A

Chip Development Boards and Experiments

The development boards designed and built for the DYMPLES, RHO and PAR chips were used to:

- correctly bias the chips
- allow the chips to be tested automatically
- provide test points to allow the hardware to be calibrated and checked manually.

System Overview

Although separate development boards with different functionality were constructed for the three chips, they all operated as part of the set-up shown in Figure A.1.

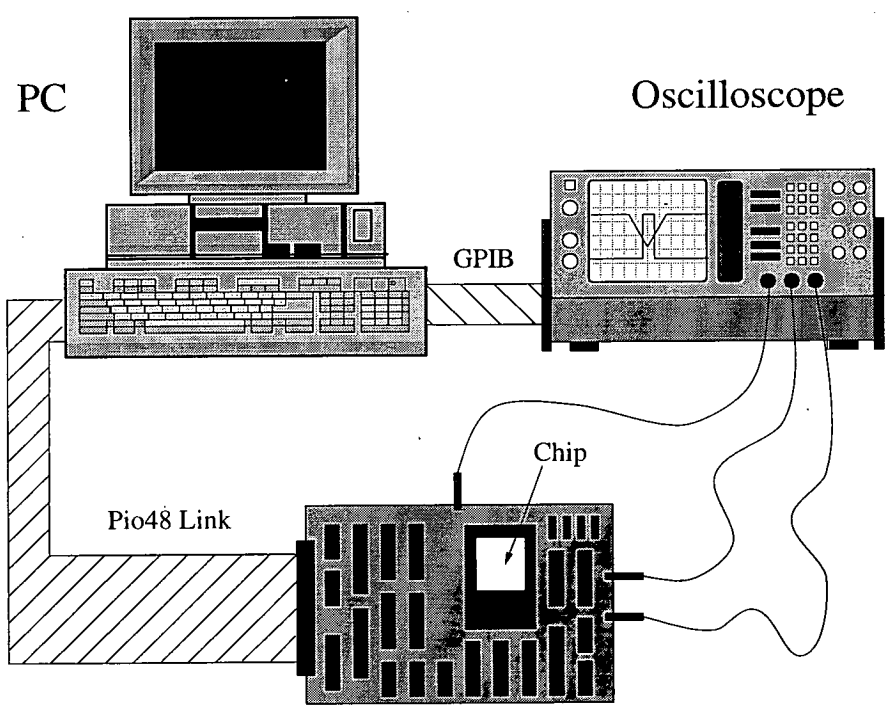


Figure A.1 - Chip Measurement System.

An IBM PS/2 286 PC was used to set-up and control each development board via a suite of 'C' programs and a Blue Chip Technology PIO-48 interface card. Provision was provided within each 'C' program to:

- set up the board for generating the desired outputs
- set up the board and/or oscilloscope for recording the results
- trigger the board
- interpret and process the raw measurements
- transfer the processed results to appropriate files.

Development Board Overview

At the heart of each development board is a **development board clock**, a counter and 8-bit or 12-bit digital buses which are split and terminated by various buffers, RAMs and DACs. The PAR chip uses 12-bit digital buses for all the neural parameters except the output weights and thresholds, whilst the DYMPLES and RHO chips use 8-bit buses.

The flow of information on the bus is controlled by a mixture of combinatorial logic and control bits. The control bits are generated by:

- the PC whilst the board is being set-up or interrogated
- the board counter once the board has been triggered and the results are being generated.

By considering which results would best characterise the performance of each chip and designing the development boards to easily record the relevant measurements, it was possible to produce simple boards with the desired functionality.

Development Board Features

The following features appeared on the three development boards.

- Test Pins -

By providing the boards with test points, it was possible to check signal flow, board functionality, voltage levels and waveform generation. This was deemed to be a necessary part of the development board design process and helped to confirm it was operating correctly, as well as help in the process of locating and diagnosing development board faults.

- Off-Chip DACs -

The RHO and PAR chips required the generation of analogue voltages for quantities such as V_{in}, V_{centre} and V_{width} etc. These were produced using 8-bit or 12-bit off-chip voltage DACs, loaded and latched by the PC. The DACs were biased and amplified to produce voltages within a defined range and could be calibrated before running the experiments.

- Op-Amp Measurement Circuits -

Measurement transistors were included on all three chips. By connecting them to an Op-Amp circuit similar to that in Figure A.2, copies of on-chip currents were converted into buffered off-chip voltages and, since there is a linear relationship between I_{sink} and V_{output}, equation A.1, this enabled on-chip current variations to be tracked as off-chip voltage variations.

$$V_{output} = V_{ref} + I_{sink} R_{feedback} \tag{A.1}$$

When multiple circuits required to be characterised off-chip, analogue multiplexors were used, under PC control, to steer current from a particular measurement transistor through the Op-Amp circuit. This ensured that the experimental set-up was consistent for all the results and so any observed differences between the measurements from the outputs would be due to on-chip variations, as opposed to off-chip ones.

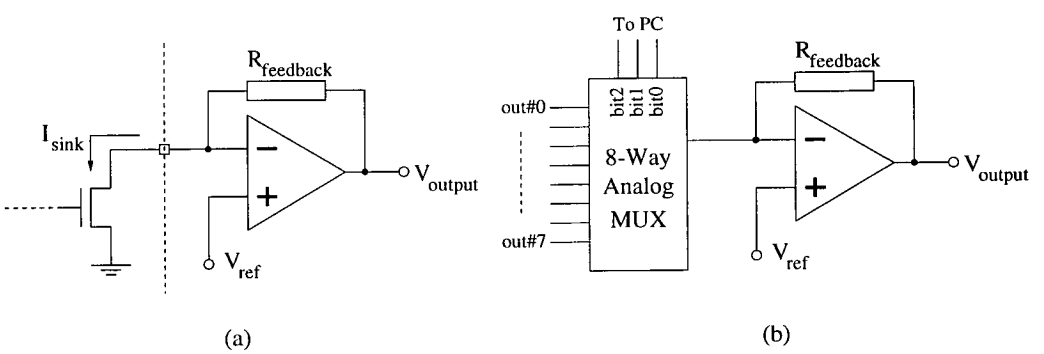


Figure A.2 - The Op-Amp circuit for (a) a single measurement transistor and (b) eight transistors connected to an 8-way analogue multiplexor

- Board Counter -

To fully test the three chips, measure the non-linear characteristics of the hidden layers and the multiplication characteristics of the output layer, the chips were operated in real time. For this, pulses and ramps were produced, applied to the chip and the

output pulse widths were recorded and measured.

The timing control of the input pulse and ramp generation and the output pulse recording was facilitated by the development board counter. This was incremented using either a 1MHz, 12MHz or 24MHz development board clock - a 24MHz clock was used for the DYMPLES and PAR chip development boards, whilst the RHO chip development board counter was incremented using a 1MHz, 12MHz or 24MHz counter.

Whilst setting up the board for each experiment, the PC disabled the board counter and loaded the RAMs with the necessary bit patterns for the input pulses and off-chip ramps. Once the RAMs were loaded, the PC triggered the board, ceding control to the counter, and entered a timed "idle-state" loop. Having been triggered, the board counter was incremented by the clock, addressing the RAMs as it did so and causing bitstreams to be read out of memory. These bitstreams produced the input pulse widths and time varying ramps for the chip, eg Figure A.3(a).

As the output ramp was generated, the output pulses from the chips were simultaneously read into another RAM by setting it into **write** mode and using the development board counter to address it. This circuit basically sampled the chips PWM outputs, Figure A.3(b).

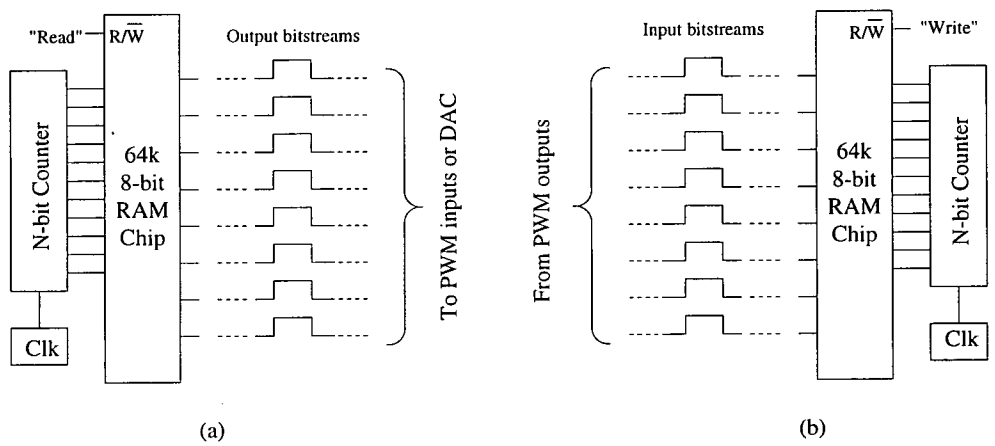


Figure A.3 - The Development Board hardware for (a) generating the PWM input pulses and time-varying analogue ramps and (b) recording the PWM outputs from the chip

After the counter reached its maximum value, it was automatically disabled and the system remained idle until the PC loop timed-out and the PC retook control of the board. The RAM containing the sampled data from the chips PWM outputs was then interrogated by the PC and the output pulse widths calculated.

In this way it was possible to automatically test the chips and simultaneously gather corresponding results from all the outputs on a given chip.

The DYMPLES Chip Experiments

This section details how the experimental results from the DYMPLES chip were generated and measured.

- DAC Characteristics -

i) All sixteen 4-bit binary words were applied to the DAC in turn.

ii) The output voltages from an Op-Amp measurement circuit (Figure A.2) were recorded and the corresponding on-chip currents calculated.

- DCM Characteristics -

i) All sixteen 4-bit binary words were applied to the DAC in turn.

ii) All the synapses on the chip were loaded 10 times with the DAC current and the 2.5μA "zero" current. This multiple loading ensured the synapses had the correct gate voltages for a given current.

iii) The current to be measured was steered to the Op-Amp circuit through an 8-way analogue multiplexor and the output voltage recorded. Again the corresponding on-chip current was calculated.

- Multiplication Characteristics -

i) A 4-bit binary word was applied to the DAC.

ii) All the synapses on the chip were loaded 10 times with the DAC current and the 2.5μA "zero " current.

iii) A pulse was fired into the PWM inputs of the chip and a linear time varying ramp was then generated at the appropriate input, Figure A.4.

iv) The output pulse widths were measured - the outputs from all 8 neurons being logged simultaneously.

v) Steps i) to iv) were repeated for all 16 4-bit words and 250 input pulse widths between 0μs and 10μs.

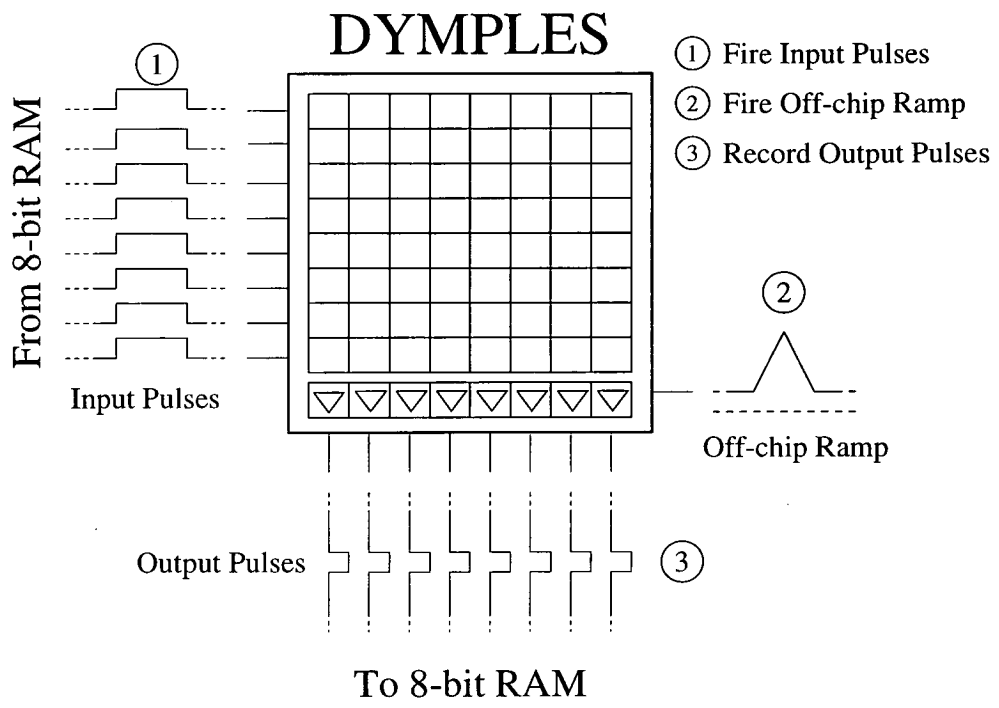


Figure A.4 - Block diagram indicating the inputs and outputs to the DYMPLES chip during the multiplication characteristic experiments

The RHO Chip Experiments

This section details how the experimental results from the RHO chip were generated and measured.

- Distance Circuit Characteristics -

i) The V_{centre} DAC was loaded and latched with the 8-bit digital word required to produce the required output voltage.

ii) The V_{in} DAC was loaded likewise.

iii) The V_{in} and V_{centre} voltages combined to produce an output current, I_{dist}. A scaled version of I_{dist} was taken off-chip where measurable output voltages were produced by an Op-Amp and 8-way switching circuit. These voltages were automatically measured using a Philips digital storage oscilloscope controlled by the PC. (All the cells were continuously refreshed as the measurements were taken.)

iv) Steps ii) and iii) were repeated for up to 256 values of V_{in}.

v) Steps i) to iv) were repeated for several values of V_{centre} (if required).

- CAP Array and TRAN Array Circuit Characteristics -

Note: Two separate arrays of circuits were produced on the RHO chip and, since these operated slightly differently, two separate programs (one for each array) were used to set up and control the board. Both programs were almost identical (only the ramps and some control sequences were different) and operated using the recipe below.

i) The V_{centre} DAC was loaded and latched with an 8-bit value.

ii) The V_{width} DAC was loaded and latched with an 8-bit value.

iii) The V_{in} DAC was loaded and latched with an 8-bit value.

iv) All the cells on the chip were loaded with V_{centre} and this was continuously refreshed as the experiment proceeded.

v) The input ramps were fired onto the chip and the output pulse widths were read into the off-chip RAM. Again the pulses from all 8 outputs were logged simultaneously.

vi) Steps iii) to v) were repeated for all values of V_{in} (if required).

vii) V_{width} was altered and steps i) and iii) to vi) were repeated (if required).

viii) V_{centre} was altered and steps ii) to vi) were repeated (if required).

For the other experiments mentioned in Chapter 5, such as the common mode trials performed on the *CAP* Array and the Transistor Centre "Block Tests", the two programs were simply reconfigured in such a way so as to allow the quantities to be assessed.

The PAR Chip Experiments

This section details how the experimental results from the PAR chip were generated and measured.

- Current DAC Characteristic -

i) All 256 8-bit words were applied to the on-chip current DAC.

ii) The output voltages from an Op Amp measurement circuit similar to that in Figure A.2(a) were recorded.

iii) The DAC test currents (scaled versions of the actual DAC currents) were calculated using the following formula:

$$I_{dac\_i} = \frac{V_{zero} - V_{word\_i}}{R_{feedback}} \tag{B.2}$$

where I_{dac_i} is the calculated DAC test current, V_{zero} is the Op Amp output voltage for a current DAC word of 0_{10}, V_{word_i} is the recorded voltage for DAC word i_{10} ($0 \leq i \leq 255$) and $R_{feedback}$ is the feedback resistor in the Op Amp circuit. Processing the results using this formula removes the zero input dc offset from the characteristic.

- DCM Characteristics -

i) All 256 8-bit DAC words were used.

ii) The chip was refreshed continuously ensuring that correct weight and zero currents were read into the DCMs.

iii) The currents to be measured were steered to the Op Amp circuit through an 8-way analogue multiplexor, Figure A.2(b), and the Op Amp's output voltage was recorded.

iv) The DCM test currents (again scaled versions of the actual currents flowing in the DCMs) were calculated using a similar formula to that used for finding the DAC test currents. When finding the DCM test currents, however, the zero input dc offset was not removed.

- Distance Circuit Characteristics -

i) The V_{centre} off-chip voltage DAC was loaded and latched with the required 12-bit word.

ii) The chip was refreshed continuously.

iii) The V_{in} DAC was loaded and latched with 256 12-bit words in the range 0_{10} to 4095_{10}.

iv) For each combination of V_{centre} and V_{in}, all 4 characterisable distance circuits were connected, in turn, to a measurement Op Amp via a 4-way analogue multiplexor. The Op Amp output voltage was recorded in each case.

v) Steps iii) and iv) could be repeated for several values of V_{centre} (if required).

- Hidden Layer Non-linearity Characteristics -

i) The V_{width}, V_{centre} and V_{in} off-chip voltage DACs were loaded and latched with the required 12-bit words. Again the PAR chip was continuously refreshed.

ii) The board counter was stepped through under PC control. This applied the hidden ramp to the hidden layer PWM neurons and the current state (**on** or **off**) of all the neurons was recorded for each counter step. The output pulse width was then calculated by summing the number of logic HIGH states for each neuron.

iii) The values latched into the V_{in}, V_{centre} and V_{width} DACs were then altered as required and step ii) repeated for each combination.

- Output Layer Characteristics: Output Pulse vs. Input Pulse -

i) 18 8-bit words between 0_{10} and 255_{10} were applied to the on-chip current DAC.

ii) 250 different input pulse widths were applied to the output layer circuits.

iii) After each input pulse width application, the output ramp was fired onto the chip and the output pulse widths from all 4 outputs were recorded.

- Output Layer Characteristics: Output Pulse vs. Synaptic Weight -

i) All 256 8-bit words were applied to the on-chip current DAC.

ii) Pulse widths of 0, 50, 100, 150, 200 and 250 RAM locations were applied to the output layer.

iii) Again, after each pulse width application, the output ramp was fired onto the chip and the output pulse widths were recorded.

Appendix B

Software Simulator Classification Results

This appendix contains the mean classification rates and the standard deviations for the software simulator experiments described in Section 7.3. The results are presented in tabular format. Each entry in the mean classification tables is the percentage of that particular data set that was **correctly** classified. Graphical depictions of these results are shown in Chapter 7. The results in the standard deviation tables are again given as percentages.

Gaussian Distributions Training Set - Mean Classifications

Gaussian Distributions - Training Set - Mean Classifications													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	85.64	85.80	85.74	85.80	85.78	85.56	85.60	82.58	80.44	83.88	84.38	85.66	86.78
11	85.90	85.90	85.88	86.04	85.70	85.70	85.60	82.80	80.18	84.44	84.28	85.94	86.28
12	86.04	86.02	85.96	85.94	85.86	85.76	85.84	82.80	80.24	84.58	84.64	86.02	86.20
13	86.00	85.78	85.76	85.80	85.68	85.74	85.56	83.06	80.74	84.90	85.28	86.16	86.06
14	85.92	85.74	85.74	85.94	85.98	85.98	85.78	83.72	80.64	85.18	85.56	86.12	85.74
15	85.92	85.68	85.94	85.78	86.16	86.06	85.68	84.12	81.16	85.20	85.44	86.06	85.66
16	85.98	85.62	85.82	85.78	86.20	86.16	86.04	84.96	81.84	85.78	85.52	86.02	85.64
17	86.26	86.18	86.24	86.24	86.48	86.40	86.22	85.26	81.88	85.84	85.40	85.66	85.84
18	86.52	86.36	86.34	86.52	86.60	86.50	86.36	84.98	82.22	85.56	85.46	85.82	85.78
19	86.60	86.48	86.66	86.62	86.66	86.66	86.64	85.44	82.46	85.64	85.56	85.76	86.00
20	86.60	86.68	86.72	86.72	86.70	86.76	86.76	85.80	82.48	85.58	85.44	85.64	85.76
21	86.80	86.74	86.52	87.02	86.98	86.92	86.76	85.78	82.46	85.46	85.76	85.56	85.80
22	86.72	86.68	86.64	86.72	86.88	86.72	86.92	85.74	83.06	85.54	85.58	85.58	85.84
23	86.92	86.64	86.78	86.74	86.80	86.62	86.82	85.96	82.94	85.70	85.68	85.68	85.94
24	87.28	86.64	86.90	86.52	87.16	86.68	86.48	85.94	82.84	85.56	85.94	85.74	85.96
25	86.92	86.58	86.74	86.70	86.84	86.64	86.66	85.86	83.14	85.70	85.88	85.68	85.84
26	87.12	86.70	86.64	86.82	87.06	86.74	86.44	85.96	83.48	85.80	85.56	85.86	86.06
27	87.16	86.82	86.70	86.82	86.86	86.76	86.34	86.28	83.64	85.82	85.98	85.94	86.06
28	87.04	86.84	86.92	86.86	86.86	86.80	86.32	86.06	84.04	86.04	86.22	86.12	86.20
29	87.14	86.86	86.84	86.74	86.92	86.62	86.64	86.22	84.50	85.88	86.38	86.12	86.14
30	87.10	86.84	86.86	86.82	86.98	86.76	86.76	86.50	84.62	86.18	86.30	86.14	86.22
31	87.42	87.16	87.22	87.02	87.18	87.04	87.12	86.70	84.82	86.10	86.20	86.16	86.38
32	87.44	87.14	87.26	87.28	87.32	86.96	87.20	86.66	85.28	85.98	86.60	86.06	86.46

Table B.1 - Percentage Classification Rate of trained RBF networks for the training set of the two class Gaussian Distributions problem.

Gaussian Distributions Training Set - Standard Deviations

Gaussian Distributions - Training Set - Standard Deviations													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	0.88	0.82	0.92	0.66	0.94	1.07	0.83	1.62	2.21	1.50	1.30	0.65	0.53
11	0.68	0.72	0.89	0.82	0.85	0.72	1.00	1.98	2.19	1.32	1.42	0.56	0.66
12	0.85	0.77	0.93	0.84	0.83	0.75	0.87	1.88	2.27	1.18	0.95	0.52	0.79
13	0.94	0.90	1.05	0.97	0.98	1.21	0.91	1.34	1.80	1.13	1.14	0.44	0.47
14	1.04	0.89	0.89	0.94	0.77	0.81	0.91	1.15	2.35	1.11	0.80	0.47	0.53
15	0.90	0.88	0.91	0.85	1.01	0.68	0.76	1.18	1.94	0.74	0.91	0.50	0.64
16	0.88	0.80	0.83	0.77	0.85	0.70	0.75	0.88	1.54	0.64	0.87	0.52	0.54
17	0.90	0.82	0.82	0.68	0.95	0.71	0.92	0.75	1.68	0.87	1.05	0.53	0.58
18	0.71	0.85	0.94	0.71	0.77	0.68	0.91	0.68	1.58	0.66	0.64	0.55	0.57
19	0.79	0.76	0.79	0.66	0.76	0.67	0.79	0.90	1.61	0.63	0.84	0.47	0.53
20	0.77	0.89	0.85	0.72	0.74	0.92	0.82	0.53	1.66	0.83	0.92	0.40	0.59
21	0.72	0.87	0.68	0.83	0.90	0.64	0.83	0.77	1.62	0.85	0.87	0.40	0.39
22	0.69	0.84	0.70	0.69	0.81	0.74	0.76	0.75	1.52	0.76	0.78	0.41	0.50
23	0.90	0.76	0.88	0.82	0.74	0.47	0.78	0.62	1.89	0.81	0.96	0.41	0.68
24	0.83	0.68	0.66	0.70	0.82	0.66	0.70	0.85	1.68	0.63	0.73	0.51	0.69
25	0.69	0.79	0.77	0.69	0.80	0.71	0.76	0.58	1.33	0.64	0.82	0.53	0.86
26	0.91	0.82	0.62	0.57	0.68	0.66	0.74	0.61	1.74	0.79	1.12	0.40	0.81
27	0.67	1.03	0.39	0.66	0.54	0.82	0.76	0.55	1.41	0.75	0.83	0.40	0.89
28	0.81	1.05	0.70	0.73	0.80	0.92	0.67	0.54	1.44	0.67	0.87	0.37	0.87
29	0.56	0.85	0.65	0.94	0.80	0.71	0.76	0.79	1.38	0.78	0.97	0.34	0.81
30	0.90	0.80	0.91	0.79	0.71	0.78	0.77	0.72	1.03	0.75	0.83	0.40	0.78
31	0.95	0.84	0.93	0.74	0.73	0.71	0.76	0.83	1.24	0.49	0.79	0.41	0.66
32	0.94	0.97	0.72	0.82	0.91	0.88	0.78	0.62	1.10	0.67	0.83	0.37	0.78

Table B.2 - Standard Deviations of Mean Classification Percentages in Table B.1

Gaussian Distributions Test Set - Mean Classifications

Gaussian Distributions - Test Set - Mean Classifications													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	87.06	87.04	86.96	86.89	86.71	86.59	86.74	84.25	82.72	85.76	86.38	87.48	87.74
11	87.02	86.99	86.85	86.81	86.70	86.63	86.81	84.35	82.27	86.25	86.11	87.66	87.58
12	86.88	86.93	86.84	86.81	86.68	86.69	86.94	84.07	82.52	86.23	85.84	87.69	87.48
13	86.74	86.73	86.64	86.69	86.55	86.62	86.93	84.71	82.80	86.48	86.17	87.76	87.45
14	86.72	86.70	86.59	86.65	86.59	86.66	86.95	84.71	82.87	86.52	86.50	87.78	87.47
15	86.71	86.70	86.69	86.67	86.69	86.76	87.04	84.55	82.97	86.74	86.79	87.78	87.50
16	86.62	86.65	86.59	86.66	86.62	86.77	87.00	84.93	83.45	86.85	86.92	87.78	87.51
17	86.65	86.73	86.68	86.73	86.60	86.77	87.02	84.83	83.54	86.91	87.02	87.77	87.64
18	86.67	86.72	86.63	86.71	86.62	86.73	87.06	84.89	83.74	86.92	87.09	87.71	87.61
19	86.65	86.73	86.66	86.76	86.63	86.84	87.15	85.00	83.66	87.00	87.11	87.55	87.67
20	86.64	86.64	86.60	86.72	86.60	86.80	87.11	85.02	83.74	87.07	87.18	87.39	87.69
21	86.56	86.67	86.67	86.67	86.55	86.85	87.08	84.93	83.79	87.09	87.28	87.08	87.74
22	86.66	86.64	86.64	86.70	86.61	86.79	86.98	85.16	84.42	87.13	87.25	86.99	87.70
23	86.53	86.62	86.50	86.65	86.52	86.72	86.84	85.15	84.28	87.14	87.37	86.90	87.69
24	86.49	86.54	86.54	86.52	86.53	86.71	86.78	85.12	84.14	87.12	87.41	86.82	87.67
25	86.45	86.56	86.50	86.51	86.51	86.69	86.76	85.20	84.50	87.12	87.36	86.69	87.70
26	86.44	86.54	86.50	86.59	86.43	86.82	86.81	85.25	84.53	87.09	87.34	86.63	87.62
27	86.41	86.47	86.44	86.47	86.38	86.69	86.73	85.32	84.69	87.16	87.45	86.56	87.63
28	86.41	86.47	86.35	86.42	86.34	86.63	86.60	85.32	84.73	87.19	87.29	86.41	87.67
29	86.34	86.41	86.27	86.25	86.21	86.63	86.53	85.35	84.62	87.21	87.28	86.33	87.57
30	86.33	86.38	86.20	86.23	86.21	86.47	86.40	85.26	84.84	87.14	87.23	86.34	87.53
31	86.31	86.18	86.13	86.14	86.17	86.33	86.31	85.24	84.64	87.15	87.13	86.20	87.50
32	86.12	86.05	86.02	85.95	86.07	86.23	86.08	85.42	84.70	86.96	86.97	86.04	87.35

Table B.3 - Percentage Classification Rate of trained RBF networks for the test set of the two class Gaussian Distributions problem.

Gaussian Distributions Test Set - Standard Deviations

Gaussian Distributions - Test Set - Standard Deviations													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	0.40	0.40	0.39	0.39	0.44	0.51	0.41	1.47	1.49	0.78	0.70	0.43	0.18
11	0.33	0.33	0.36	0.35	0.38	0.31	0.41	1.43	1.33	0.67	0.76	0.30	0.23
12	0.30	0.28	0.31	0.29	0.36	0.32	0.36	1.47	0.97	0.71	0.82	0.18	0.20
13	0.41	0.39	0.40	0.39	0.39	0.33	0.39	1.00	1.18	0.65	0.68	0.12	0.18
14	0.32	0.27	0.36	0.29	0.31	0.25	0.37	0.85	1.23	0.54	0.58	0.10	0.22
15	0.27	0.30	0.24	0.27	0.25	0.30	0.27	0.91	1.26	0.59	0.54	0.12	0.17
16	0.31	0.26	0.26	0.29	0.27	0.27	0.42	0.72	0.85	0.51	0.42	0.11	0.18
17	0.22	0.16	0.24	0.21	0.27	0.28	0.40	0.61	0.91	0.59	0.34	0.18	0.21
18	0.22	0.20	0.23	0.23	0.24	0.30	0.29	0.82	0.75	0.66	0.32	0.20	0.28
19	0.24	0.20	0.25	0.24	0.29	0.30	0.24	0.81	0.98	0.47	0.41	0.29	0.20
20	0.27	0.27	0.23	0.26	0.26	0.27	0.28	0.71	1.31	0.55	0.37	0.30	0.20
21	0.29	0.29	0.22	0.26	0.33	0.22	0.30	0.78	1.30	0.40	0.36	0.27	0.19
22	0.23	0.28	0.21	0.22	0.23	0.21	0.24	0.81	0.94	0.34	0.34	0.21	0.19
23	0.21	0.22	0.32	0.25	0.20	0.32	0.33	0.70	1.01	0.50	0.23	0.20	0.26
24	0.39	0.39	0.29	0.31	0.41	0.33	0.41	0.59	1.03	0.47	0.26	0.28	0.32
25	0.36	0.33	0.30	0.32	0.36	0.34	0.42	0.58	1.02	0.44	0.26	0.35	0.31
26	0.35	0.33	0.24	0.31	0.38	0.37	0.35	0.59	1.10	0.37	0.34	0.40	0.36
27	0.29	0.39	0.29	0.30	0.34	0.47	0.47	0.63	0.56	0.35	0.29	0.37	0.33
28	0.38	0.39	0.34	0.28	0.41	0.50	0.54	0.76	0.86	0.44	0.39	0.41	0.31
29	0.44	0.37	0.38	0.52	0.53	0.56	0.63	0.52	0.88	0.44	0.37	0.37	0.38
30	0.39	0.30	0.45	0.46	0.56	0.56	0.75	0.65	0.88	0.42	0.41	0.31	0.41
31	0.38	0.41	0.40	0.32	0.44	0.50	0.63	0.64	0.80	0.44	0.49	0.32	0.35
32	0.47	0.41	0.51	0.40	0.52	0.61	0.67	0.67	1.02	0.64	0.54	0.38	0.40

Table B.4 - Standard Deviations of Mean Classification Percentages in Table B.3

Speaker Recognition Training Set - Mean Classifications

Speaker Recognition - Training Set - Mean Classifications													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	77.59	77.37	77.44	77.59	77.23	76.93	76.36	72.87	69.49	69.36	67.52	75.80	77.33
11	79.88	79.87	79.83	79.56	79.19	78.96	78.28	74.00	70.99	70.65	68.97	77.15	78.61
12	79.80	79.71	79.79	79.67	79.39	79.05	78.73	74.68	72.17	72.40	70.25	78.59	80.15
13	80.33	80.33	80.37	80.19	79.97	79.55	79.08	74.76	72.69	73.24	71.88	79.52	80.53
14	81.73	81.83	81.76	81.45	81.43	81.03	79.85	75.64	73.60	74.67	73.03	80.67	81.73
15	82.49	82.45	82.11	82.37	82.03	81.13	79.60	76.36	74.81	75.55	75.29	81.37	82.19
16	82.81	82.53	82.28	82.47	82.09	81.75	80.36	76.83	75.17	76.07	75.89	81.81	82.56
17	82.77	82.51	82.27	82.55	82.28	81.56	80.52	77.59	75.76	76.32	76.19	82.07	82.64
18	82.99	83.12	82.87	82.32	82.25	81.68	80.47	77.73	76.93	76.56	77.27	81.96	82.80
19	83.12	83.21	83.05	82.77	82.56	82.11	81.01	78.16	78.04	77.20	77.63	83.11	83.27
20	83.43	83.28	83.53	83.20	82.89	82.25	81.00	78.47	78.33	77.52	78.48	83.61	84.13
21	83.29	83.39	83.19	82.99	82.92	82.16	81.15	79.12	78.97	78.32	79.07	83.85	84.49
22	83.40	83.53	83.09	83.36	83.08	82.32	81.68	79.36	79.56	78.73	79.43	84.28	84.53
23	83.52	83.47	83.51	83.29	83.09	82.56	81.87	80.11	79.89	79.56	79.63	84.81	84.87
24	83.55	83.76	83.75	83.44	83.35	82.72	82.11	80.60	80.45	79.85	80.17	85.05	85.21
25	83.40	83.43	83.47	83.51	83.16	82.85	82.39	80.72	80.57	80.39	80.49	85.67	86.05
26	83.64	83.67	83.48	83.45	83.11	82.80	82.55	81.05	80.41	80.79	80.83	86.17	86.24
27	83.61	83.47	83.51	83.53	83.39	83.39	82.65	81.27	80.72	81.23	81.44	86.49	86.77
28	83.71	83.55	83.53	83.55	83.56	83.24	82.76	81.51	81.17	80.81	81.20	86.60	86.64
29	83.84	84.01	83.92	83.72	83.49	83.59	83.11	81.87	80.89	81.47	81.56	86.99	87.35
30	84.31	84.11	84.09	83.89	83.72	83.56	83.36	82.53	81.51	81.88	82.04	87.03	87.49
31	84.08	84.04	83.87	83.93	83.64	83.51	83.65	82.71	81.83	82.37	82.00	87.24	87.52
32	84.41	84.36	84.11	83.87	83.85	84.05	84.00	83.29	82.16	82.57	81.81	87.80	87.80

Table B.5 - Percentage Classification Rate of trained RBF networks for the training set of the three class Speaker Recognition problem.

Speaker Recognition Training Set - Standard Deviations

Speaker Recognition - Test Set - Standard Deviations													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	2.45	2.22	2.25	2.17	2.35	2.21	1.79	1.60	1.91	4.14	3.04	3.32	3.08
11	1.60	1.47	1.46	1.55	1.45	1.50	1.43	1.82	2.56	3.36	3.47	3.05	2.62
12	2.71	2.92	2.88	2.75	2.42	2.14	1.69	1.37	1.90	2.47	4.08	2.51	2.50
13	2.01	1.87	2.16	1.85	2.11	2.04	1.55	1.55	1.67	2.44	3.14	2.15	1.96
14	1.67	1.61	1.87	1.50	1.48	1.51	1.11	1.55	2.10	2.56	2.88	1.72	2.00
15	1.49	1.43	1.67	1.62	1.45	1.25	1.18	1.78	1.79	1.93	3.21	1.59	1.60
16	1.19	1.36	1.29	1.40	1.01	1.22	1.28	1.58	2.31	1.78	2.76	1.29	1.41
17	1.29	1.22	1.38	1.11	1.58	1.51	1.16	1.45	2.07	1.51	2.49	1.13	1.46
18	1.11	0.88	1.24	1.04	0.99	1.11	1.01	1.38	2.20	1.80	2.59	1.50	1.37
19	1.00	1.14	1.25	1.14	1.14	1.22	1.33	1.53	1.58	1.84	2.77	1.25	1.25
20	0.98	1.03	1.12	1.33	1.11	1.20	1.08	1.60	1.95	1.93	2.60	1.53	1.52
21	0.87	0.97	1.29	1.30	1.18	1.14	1.28	1.29	1.94	1.86	2.38	1.41	1.43
22	1.04	1.09	1.25	1.39	1.24	1.40	1.20	1.52	1.40	1.94	2.23	1.26	1.27
23	1.40	1.55	1.40	1.43	1.36	1.09	1.09	1.59	1.36	1.83	2.06	1.50	1.38
24	1.22	0.96	0.98	1.10	1.22	0.84	1.18	1.58	1.24	1.89	1.86	1.45	1.45
25	1.05	1.15	1.14	1.05	1.05	0.91	1.10	1.43	1.77	1.94	1.75	1.52	1.24
26	0.99	1.06	1.15	1.03	1.24	1.22	0.95	1.13	1.56	1.66	1.46	1.61	1.41
27	0.94	0.97	0.86	0.81	0.91	1.13	1.06	1.52	1.57	1.70	1.72	1.34	1.47
28	1.28	1.08	1.25	1.12	0.97	0.98	1.24	1.65	1.49	1.48	1.79	1.86	1.91
29	0.77	0.94	0.98	0.96	1.13	1.19	1.41	1.52	1.59	1.66	1.87	1.96	1.68
30	1.05	0.97	1.10	1.13	1.12	1.40	1.50	1.75	1.73	1.54	1.73	1.84	1.80
31	1.19	0.96	0.82	1.08	1.10	0.93	0.88	1.56	1.77	1.65	1.51	1.91	1.81
32	1.16	1.02	1.07	1.04	1.00	1.01	0.83	1.31	2.07	1.37	1.59	2.24	1.82

Table B.6 - Standard Deviations of Mean Classification Percentages in Table B.5

Speaker Recognition Test Set - Mean Classifications

Speaker Recognition - Test Set - Mean Classifications													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	73.07	73.28	73.09	73.25	73.25	73.55	73.81	72.69	66.64	54.13	59.07	63.07	64.51
11	73.41	73.36	73.39	73.07	73.44	73.81	74.24	73.52	68.43	54.96	58.51	61.81	63.76
12	73.07	73.28	73.31	73.39	73.52	73.57	74.16	73.68	69.09	57.44	60.03	63.81	65.44
13	73.84	73.63	73.87	74.13	74.11	73.97	74.48	74.45	70.35	58.99	61.15	65.63	67.12
14	73.44	73.41	73.79	73.84	73.81	73.76	74.59	74.19	71.01	61.47	61.68	66.05	67.60
15	73.25	72.91	73.31	73.47	73.28	73.57	74.69	74.96	70.59	61.60	63.92	67.57	68.24
16	72.96	73.15	73.41	73.44	73.20	73.73	74.77	75.92	71.49	62.13	63.52	67.23	68.05
17	72.85	72.72	72.77	73.01	73.12	73.49	74.75	76.43	72.19	62.61	64.16	68.03	68.13
18	72.83	72.85	72.64	73.01	72.96	73.31	74.37	76.48	72.56	62.61	65.44	68.45	69.04
19	72.77	72.37	72.85	72.75	72.93	73.65	74.93	76.35	72.48	63.89	66.03	68.51	69.01
20	72.32	72.19	72.32	72.37	72.56	73.04	74.91	76.19	73.49	63.09	65.15	68.75	68.83
21	71.92	71.87	71.84	71.92	72.37	73.20	74.61	76.48	73.36	63.23	65.71	70.03	70.56
22	72.21	72.11	72.56	72.43	72.35	72.96	74.61	76.48	73.17	63.79	66.48	70.61	70.93
23	72.64	72.88	72.67	72.69	72.88	73.55	74.77	76.61	73.25	64.19	66.29	70.93	70.93
24	72.24	72.24	72.05	72.21	72.56	73.44	74.67	76.29	73.31	66.03	67.01	71.81	71.20
25	72.53	72.64	72.69	72.88	73.28	73.76	75.57	76.32	72.77	66.03	67.36	71.81	71.87
26	72.21	72.32	72.35	72.37	72.83	73.79	75.12	76.19	73.28	65.92	67.49	72.37	71.97
27	71.87	71.84	71.41	71.92	72.00	72.77	74.13	75.73	73.23	66.72	67.89	72.80	72.51
28	71.44	71.57	71.65	71.73	72.24	72.61	74.19	76.37	73.49	67.12	68.48	72.59	71.84
29	72.05	71.60	72.11	72.11	72.48	73.23	74.96	76.67	73.15	67.89	67.71	73.65	73.07
30	71.97	71.81	71.63	71.87	71.95	72.83	74.37	76.03	72.85	67.31	68.03	73.33	72.59
31	72.13	72.11	71.87	72.59	72.53	73.25	74.64	76.32	72.83	67.87	68.29	73.63	73.20
32	72.27	71.79	72.00	72.13	72.64	73.23	74.91	76.64	73.49	67.89	68.67	73.95	72.85

Table B.7 - Percentage Classification Rate of trained RBF networks for the test set of the three class Speaker Recognition problem.

Speaker Recognition Test Set - Standard Deviations

Speaker Recognition - Test Set - Standard Deviations													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	2.48	2.46	2.50	2.45	2.45	2.59	2.53	2.11	3.27	5.48	5.16	4.47	3.81
11	2.94	2.71	2.91	2.48	2.63	2.20	1.95	2.22	3.23	5.19	5.46	3.34	2.44
12	2.43	2.48	2.39	2.31	2.02	2.05	2.35	2.21	3.10	3.48	4.89	2.98	3.53
13	2.70	2.70	2.72	2.87	2.77	2.65	2.99	2.46	3.01	3.57	4.72	2.63	3.38
14	2.74	2.77	2.57	2.49	2.47	2.49	2.54	2.56	2.21	3.55	4.38	2.63	3.36
15	2.68	2.31	2.61	2.65	2.65	2.54	2.37	1.98	2.13	4.55	4.17	2.38	2.59
16	2.16	2.13	2.50	2.01	2.10	2.40	2.90	2.20	2.70	3.55	3.90	2.43	2.36
17	2.11	1.79	1.83	2.03	2.19	2.04	2.03	2.14	2.98	3.54	3.41	2.71	2.83
18	2.13	1.74	1.93	1.86	2.05	2.19	2.21	2.57	2.47	3.37	3.47	2.63	3.22
19	1.89	2.17	1.94	2.12	2.12	2.14	2.18	1.91	2.28	3.23	3.44	2.64	2.44
20	1.44	2.01	1.70	1.79	1.47	1.86	1.79	2.38	2.63	2.85	2.88	2.61	2.55
21	1.64	1.52	1.73	1.75	2.13	2.25	1.85	2.26	2.08	3.52	3.35	2.26	1.89
22	1.83	2.13	1.76	2.05	2.08	2.27	2.11	2.11	2.08	3.14	3.17	2.72	2.69
23	1.93	1.79	1.69	1.99	2.35	2.14	2.56	1.94	2.17	3.10	3.24	2.45	2.10
24	2.05	2.03	1.77	1.81	2.12	2.19	2.47	2.21	2.45	2.77	3.65	2.53	1.99
25	2.11	2.19	2.41	2.13	2.27	1.81	2.48	2.06	1.82	2.55	3.08	2.01	2.49
26	1.52	1.47	1.63	1.78	2.17	1.78	2.40	2.04	2.51	3.03	3.28	2.07	2.39
27	1.62	1.34	1.41	1.72	1.42	1.81	2.14	2.27	2.54	2.83	2.26	2.07	2.34
28	1.82	1.64	1.43	1.66	1.76	1.48	2.01	2.50	2.59	3.31	3.10	2.57	2.29
29	1.39	1.57	1.63	1.43	1.74	1.85	1.84	2.06	2.53	2.75	2.22	2.43	2.09
30	1.85	2.16	2.09	2.24	2.36	1.90	1.82	2.38	2.13	2.61	2.73	1.69	1.84
31	1.69	1.81	1.98	1.72	1.55	2.42	2.16	2.44	2.14	2.81	2.73	1.97	1.77
32	1.52	2.07	1.61	1.90	1.60	1.81	1.72	2.45	2.34	2.38	2.97	1.87	1.73

Table B.8 - Standard Deviations of Mean Classification Percentages in Table B.7

Sleep State Training Set - Mean Classifications

Sleep State - Training Set - Mean Classifications													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	76.20	76.21	76.64	76.51	76.25	76.49	76.28	76.25	72.21	74.83	74.65	75.64	75.87
11	76.37	76.43	76.41	76.45	76.16	76.40	76.47	76.36	72.39	74.37	74.43	75.29	75.73
12	76.16	76.16	76.53	76.43	76.20	76.47	76.32	76.49	72.88	75.12	74.45	75.52	75.84
13	76.44	76.25	76.55	76.60	76.45	76.57	76.63	76.29	72.95	75.04	74.21	75.96	75.92
14	76.61	76.53	76.72	76.75	76.69	76.73	76.65	76.43	73.37	75.21	74.51	75.96	75.64
15	76.48	76.48	76.59	76.73	76.56	76.65	76.84	76.44	73.40	75.40	74.40	76.29	75.61
16	76.83	76.76	77.15	76.85	77.03	76.96	77.19	76.84	73.81	75.80	74.77	76.73	75.60
17	76.79	76.93	77.28	76.92	77.12	77.11	77.29	76.61	73.93	75.88	74.88	77.07	75.87
18	77.24	77.25	77.44	77.33	77.64	77.32	77.73	77.12	74.12	76.23	75.49	77.53	75.80
19	77.57	77.27	77.85	77.32	77.81	77.72	78.01	77.28	74.37	76.15	75.44	78.37	75.95
20	77.69	77.87	78.09	77.84	77.89	77.77	78.27	77.47	74.92	76.61	75.13	79.65	75.96
21	77.80	77.88	78.39	77.91	78.49	78.01	78.67	77.65	74.96	76.49	75.12	80.52	76.09
22	78.13	78.20	78.56	78.09	78.47	78.12	78.93	77.79	74.91	76.91	75.36	80.69	76.25
23	78.31	78.35	78.80	78.35	78.79	78.40	79.05	78.04	74.99	77.23	75.15	81.59	76.01
24	78.61	78.45	79.11	78.43	79.03	78.67	79.11	78.15	74.83	77.09	75.28	82.36	76.19
25	78.75	78.91	79.43	78.80	78.95	78.99	79.20	78.59	75.07	77.39	75.64	83.17	76.59
26	79.32	79.20	79.75	79.35	79.56	79.27	79.87	78.93	75.49	77.45	75.80	84.08	76.96
27	79.47	79.59	79.87	79.37	79.88	79.64	80.15	79.32	75.81	77.76	75.92	84.75	77.09
28	79.96	79.97	80.45	80.04	80.41	80.29	80.44	79.64	75.99	78.28	75.80	85.48	77.16
29	80.35	80.40	80.67	80.21	80.72	80.77	80.67	80.16	76.16	78.29	76.13	85.65	77.55
30	80.56	80.80	81.00	80.84	81.09	81.13	81.07	80.48	76.24	78.49	76.39	86.13	77.81
31	80.60	80.89	81.16	80.75	81.08	81.08	81.20	80.60	76.93	78.49	76.61	86.47	78.00
32	80.89	81.03	81.45	81.11	81.19	81.29	81.51	80.71	76.99	78.55	77.08	86.65	78.03

Table B.9 - Percentage Classification Rate of trained RBF networks for the training set of the three class Sleep State problem.

Sleep State Training Set - Standard Deviations

Sleep State - Training Set - Standard Deviations													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	0.70	0.55	0.73	0.86	0.74	0.88	0.89	0.43	0.73	1.09	0.93	0.55	0.46
11	0.91	0.88	0.86	0.89	0.87	0.93	0.67	0.53	0.70	1.16	1.17	0.48	0.43
12	0.97	1.01	0.96	1.15	0.87	1.20	0.94	0.66	0.68	0.97	1.11	0.64	0.64
13	0.75	0.83	1.08	0.71	0.97	0.84	0.78	0.46	0.84	1.23	1.17	0.79	0.51
14	0.97	0.94	0.83	0.86	0.97	1.06	1.00	0.65	0.92	0.93	0.93	0.74	0.62
15	1.02	1.02	1.00	1.01	0.94	1.22	1.06	0.74	0.87	1.15	0.69	0.87	0.63
16	1.05	0.81	1.00	1.04	1.09	0.93	0.88	0.87	0.83	0.82	0.89	1.34	0.69
17	1.23	0.99	1.02	0.94	1.10	1.07	1.28	0.80	0.86	1.02	0.68	1.46	0.59
18	1.41	1.16	1.10	1.18	1.30	1.13	1.16	0.84	1.16	1.01	0.75	1.63	0.54
19	1.24	0.96	0.96	1.16	1.04	0.98	1.06	0.93	0.91	1.05	1.00	1.52	0.83
20	1.01	0.99	0.85	1.00	1.14	0.80	1.02	0.93	1.13	1.15	0.69	2.25	0.75
21	1.08	1.14	0.96	0.89	1.27	0.81	1.03	0.95	1.00	1.13	0.73	2.58	0.66
22	1.16	1.19	1.07	1.36	1.23	1.05	1.23	1.15	0.95	1.30	0.84	2.82	0.79
23	0.88	0.90	1.36	1.16	1.23	0.93	1.27	1.22	1.04	1.44	0.72	2.81	0.70
24	0.93	1.11	0.96	1.09	1.15	0.99	1.14	1.28	0.95	1.07	0.90	2.95	0.60
25	1.14	1.17	1.16	1.46	1.23	1.38	1.45	1.42	0.94	1.29	0.99	2.92	0.91
26	1.57	1.53	1.29	1.62	1.70	1.78	1.70	2.08	1.04	0.90	0.96	2.61	1.35
27	1.41	1.49	1.35	1.86	1.71	1.52	1.57	1.84	1.35	1.15	0.78	2.14	1.35
28	1.52	1.39	1.17	1.71	1.73	1.45	1.66	1.72	1.59	1.01	0.75	1.91	1.26
29	1.87	1.71	1.53	1.97	1.97	1.82	1.84	2.22	1.53	1.10	1.08	2.18	1.53
30	1.72	1.70	1.35	1.61	1.55	1.75	1.66	1.89	1.49	0.99	1.14	1.81	1.42
31	1.62	1.75	1.27	1.65	1.66	1.63	1.66	1.96	1.93	1.30	1.12	1.84	1.20
32	1.86	1.64	1.54	1.49	1.95	1.56	1.59	1.90	1.74	1.11	1.17	1.82	1.62

Table B.10 - Standard Deviations of Mean Classification Percentages in Table B.9

Sleep State Test Set - Mean Classifications

Sleep State - Test Set - Mean Classifications													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	70.44	70.52	70.56	70.56	70.51	70.57	70.54	70.43	68.69	68.59	68.93	69.52	69.57
11	70.51	70.56	70.62	70.63	70.58	70.65	70.62	70.49	68.84	68.49	68.85	69.70	69.68
12	70.81	70.81	70.86	70.84	70.81	70.88	70.86	70.58	69.01	68.97	69.00	69.76	69.76
13	70.92	70.93	70.97	70.98	70.99	71.04	71.00	70.76	69.02	68.96	69.11	69.89	69.81
14	71.11	71.11	71.19	71.12	71.19	71.20	71.21	70.97	68.91	69.21	69.40	70.14	69.81
15	71.16	71.21	71.27	71.26	71.28	71.32	71.34	71.07	68.93	69.08	69.43	70.29	69.79
16	71.39	71.43	71.51	71.45	71.46	71.56	71.60	71.24	69.07	69.27	69.64	71.00	69.89
17	71.34	71.37	71.46	71.46	71.44	71.53	71.62	71.24	69.04	69.12	69.58	71.41	69.94
18	71.42	71.41	71.55	71.44	71.55	71.65	71.68	71.33	69.02	69.16	69.55	71.82	69.93
19	71.61	71.63	71.84	71.74	71.80	71.85	71.97	71.48	69.03	69.15	69.58	72.91	69.99
20	71.80	71.76	72.06	71.80	72.02	72.03	72.21	71.67	69.11	69.14	69.50	74.25	69.98
21	71.98	71.90	72.22	71.95	72.26	72.26	72.40	71.85	69.10	69.07	69.42	75.08	70.04
22	72.37	72.25	72.48	72.25	72.69	72.62	72.75	72.13	69.21	68.96	69.26	75.54	70.09
23	72.32	72.27	72.58	72.26	72.74	72.71	72.84	72.26	69.13	68.80	69.29	76.45	70.13
24	72.34	72.21	72.65	72.28	72.80	72.77	72.95	72.26	69.08	68.99	69.27	76.71	70.10
25	72.54	72.42	72.82	72.49	73.07	73.09	73.15	72.55	69.08	68.84	69.06	77.39	70.06
26	73.13	73.00	73.32	73.11	73.68	73.58	73.70	73.04	69.35	68.81	69.05	78.36	70.42
27	73.35	73.20	73.55	73.20	73.92	73.84	73.90	73.24	69.39	68.79	69.08	78.98	70.52
28	73.70	73.46	73.86	73.51	74.22	74.11	74.27	73.51	69.39	68.66	69.07	79.41	70.59
29	73.86	73.65	74.04	73.82	74.54	74.30	74.41	73.69	69.34	68.49	68.92	79.62	70.75
30	74.18	74.07	74.38	74.17	74.91	74.72	74.79	74.16	69.33	68.44	68.89	80.06	70.94
31	74.27	74.11	74.57	74.29	74.91	74.94	75.04	74.31	69.59	68.30	69.00	80.29	70.95
32	74.30	74.26	74.66	74.34	74.95	74.94	75.10	74.34	69.56	68.22	68.84	80.52	70.83

Table B.11 - Percentage Classification Rate of trained RBF networks for the test set of the three class Sleep State problem.

Sleep State Test Set - Standard Deviations

Sleep State - Test Set - Standard Deviations													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	0.35	0.40	0.45	0.38	0.37	0.44	0.46	0.31	0.82	0.41	0.73	0.27	0.21
11	0.48	0.57	0.51	0.51	0.53	0.60	0.58	0.29	0.68	0.47	0.80	0.28	0.20
12	0.59	0.68	0.69	0.59	0.62	0.72	0.66	0.35	0.57	0.79	0.91	0.31	0.21
13	0.58	0.65	0.65	0.60	0.60	0.69	0.66	0.38	0.56	0.54	0.79	0.33	0.27
14	0.58	0.68	0.71	0.71	0.63	0.70	0.70	0.57	0.51	0.66	0.65	0.53	0.20
15	0.56	0.62	0.65	0.58	0.53	0.64	0.64	0.60	0.54	0.57	0.49	0.90	0.20
16	0.68	0.70	0.75	0.72	0.71	0.74	0.71	0.63	0.49	0.62	0.35	1.40	0.19
17	0.65	0.71	0.75	0.75	0.70	0.76	0.71	0.51	0.61	0.65	0.29	1.74	0.19
18	0.77	0.87	0.82	0.82	0.83	0.84	0.81	0.60	0.61	0.63	0.35	1.43	0.19
19	0.77	0.79	0.83	0.86	0.78	0.82	0.87	0.57	0.58	0.63	0.36	1.99	0.24
20	0.62	0.63	0.66	0.65	0.59	0.63	0.64	0.58	0.68	0.60	0.28	2.28	0.21
21	0.61	0.55	0.65	0.67	0.63	0.58	0.67	0.65	0.63	0.58	0.32	2.76	0.28
22	0.88	0.86	0.95	0.96	0.98	0.95	1.03	0.94	0.59	0.51	0.39	2.88	0.37
23	0.83	0.76	0.83	0.78	0.89	0.92	0.92	0.93	0.83	0.67	0.28	2.77	0.44
24	0.87	0.73	0.78	0.72	0.92	0.87	0.91	0.94	0.74	0.61	0.26	2.95	0.49
25	0.92	0.90	0.95	0.95	1.00	1.15	1.09	1.07	0.64	0.49	0.40	2.93	0.48
26	1.39	1.50	1.38	1.43	1.56	1.58	1.52	1.57	0.79	0.57	0.44	2.39	0.85
27	1.47	1.48	1.41	1.43	1.59	1.62	1.55	1.68	0.89	0.70	0.45	2.03	1.11
28	1.51	1.47	1.43	1.46	1.63	1.57	1.50	1.59	0.94	0.59	0.48	1.61	1.21
29	1.62	1.77	1.61	1.67	1.79	1.82	1.86	1.93	1.19	0.58	0.52	1.98	1.35
30	1.53	1.59	1.51	1.54	1.65	1.65	1.64	1.82	1.15	0.49	0.56	1.86	1.34
31	1.49	1.59	1.56	1.61	1.66	1.65	1.66	1.82	1.41	0.46	0.55	1.75	1.31
32	1.42	1.48	1.46	1.55	1.48	1.54	1.54	1.60	1.37	0.58	0.59	1.76	1.37

Table B.12 - Standard Deviations of Mean Classification Percentages in Table B.11

Robot Location Training Set - Mean Classifications

Robot Location - Training Set - Mean Classifications													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	68.43	68.39	68.22	68.26	68.10	67.85	64.19	62.67	60.46	59.57	58.59	63.15	63.37
11	69.12	69.10	68.93	69.19	68.92	68.59	65.45	63.18	61.07	59.84	58.97	63.37	63.60
12	70.03	69.97	69.84	69.89	69.82	69.77	66.53	63.78	61.47	60.95	59.85	64.24	64.35
13	70.89	71.10	70.71	70.85	70.64	70.61	67.35	64.56	62.05	62.23	60.45	64.66	64.89
14	71.99	72.32	72.05	72.31	71.95	71.99	68.47	65.05	62.35	62.29	60.93	65.75	65.43
15	73.39	73.49	73.61	73.67	73.45	72.98	70.30	66.09	62.57	63.69	61.85	66.89	66.37
16	74.71	74.69	74.92	74.94	74.77	74.07	71.47	66.21	63.13	64.55	62.28	67.83	67.25
17	75.77	75.75	75.87	76.06	75.99	75.31	72.26	67.14	63.10	64.49	63.01	68.57	68.11
18	76.97	77.06	77.13	77.44	77.35	76.46	74.31	67.73	63.65	65.15	62.75	70.55	69.03
19	77.82	77.88	78.25	78.13	78.15	77.17	75.07	68.40	64.41	65.66	63.67	72.29	70.43
20	78.73	78.90	79.08	79.15	79.15	78.12	76.09	68.85	64.43	66.91	64.42	73.50	71.61
21	79.42	79.56	79.89	79.95	79.84	78.83	77.00	70.00	64.78	67.42	64.60	74.65	72.54
22	80.64	80.96	81.06	81.01	81.05	79.74	77.46	70.31	64.92	67.73	65.31	75.85	73.81
23	81.61	81.67	81.87	81.84	81.71	80.87	78.21	70.53	64.95	67.75	65.85	76.49	74.81
24	82.10	82.31	82.49	82.49	82.40	81.51	79.02	71.17	65.39	68.51	66.45	77.63	75.51
25	82.89	83.00	83.11	83.14	83.11	81.99	79.73	71.71	65.89	68.94	67.11	78.44	76.13
26	83.59	83.87	84.05	83.91	83.88	82.73	80.15	72.62	66.86	69.51	67.64	79.45	76.79
27	84.12	84.59	84.69	84.29	84.34	83.23	80.73	73.07	67.11	70.06	68.43	80.19	77.71
28	84.59	84.99	85.09	84.67	84.83	83.77	81.00	73.17	67.27	69.94	68.65	81.08	78.35
29	85.02	85.31	85.29	85.16	85.07	83.97	81.75	73.63	67.97	70.35	68.88	81.83	78.89
30	85.31	85.64	85.83	85.41	85.49	84.29	82.01	74.21	68.35	70.91	69.61	82.51	79.25
31	85.77	85.79	86.15	85.73	85.74	84.97	82.47	74.33	68.26	71.37	70.09	82.89	79.93
32	86.05	86.26	86.53	86.19	86.06	85.09	82.67	75.04	68.77	71.87	70.48	84.10	81.10

Table B.13 - Percentage Classification Rate of trained RBF networks for the training set of the six class Robot Location problem.

Robot Location Training Set - Standard Deviations

Robot Location - Training Set - Standard Deviations													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	1.16	1.13	1.20	1.15	1.12	1.15	1.63	1.20	2.33	1.97	1.59	1.06	0.94
11	1.51	1.50	1.58	1.71	1.44	1.24	1.11	1.40	1.74	2.39	1.36	1.14	1.01
12	1.72	1.73	1.81	1.94	1.61	1.50	1.23	1.50	2.13	1.73	1.44	1.40	1.27
13	2.06	2.01	2.01	1.97	1.89	2.02	1.49	1.30	1.87	1.94	1.71	1.59	1.49
14	1.89	1.85	1.83	1.85	1.74	1.62	1.68	1.47	2.17	1.54	1.57	1.79	1.52
15	1.86	1.77	1.81	1.78	2.08	1.67	2.60	1.59	2.33	2.03	2.04	1.84	1.68
16	1.98	1.89	1.83	1.77	2.00	1.89	2.86	1.75	2.45	1.77	2.31	1.66	1.71
17	1.51	1.68	1.47	1.73	1.92	2.14	2.44	1.48	2.06	2.12	2.27	1.60	1.73
18	1.26	1.14	1.29	1.44	1.57	1.59	2.05	1.46	2.57	1.90	1.61	1.87	1.60
19	1.95	2.07	1.82	1.85	1.76	1.72	2.09	1.62	2.46	2.23	1.35	2.10	1.70
20	1.72	1.69	1.57	1.62	1.72	1.69	1.50	1.87	2.06	2.21	1.53	1.92	1.41
21	1.73	1.46	1.35	1.33	1.55	1.88	1.46	2.20	1.79	1.90	1.67	1.89	1.91
22	1.42	1.52	1.18	1.52	1.37	1.67	1.21	1.93	2.02	1.52	1.36	1.97	1.75
23	1.50	1.39	1.23	1.43	1.42	1.49	1.53	1.93	2.05	1.68	1.39	2.30	1.62
24	1.29	1.19	1.23	1.11	1.09	1.26	1.36	2.20	1.87	1.28	1.68	2.14	2.25
25	1.26	1.42	1.49	1.37	1.28	1.31	1.68	2.27	1.95	1.33	1.88	2.19	2.05
26	1.40	1.53	1.15	1.29	1.39	1.30	1.31	1.71	1.59	1.14	1.13	1.93	1.93
27	1.05	0.82	0.98	1.18	1.12	0.96	1.12	1.94	1.68	1.42	1.25	1.75	1.66
28	0.92	1.19	1.20	1.06	0.92	0.88	1.18	1.95	2.00	1.33	1.45	1.86	1.98
29	1.30	1.34	1.08	1.34	1.25	1.30	1.41	2.03	2.13	1.31	1.28	1.96	1.73
30	1.32	1.10	1.00	1.18	0.99	0.96	1.27	2.12	1.89	1.68	1.35	1.89	1.97
31	1.17	1.14	1.11	1.14	1.04	1.14	1.25	2.11	2.36	1.32	1.83	2.11	2.06
32	0.95	0.72	0.81	0.80	0.79	1.12	1.20	2.18	1.90	1.42	1.53	1.87	2.05

Table B.14 - Standard Deviations of Mean Classification Percentages in Table B.13

Robot Location Test Set - Mean Classifications

Robot Location - Test Set - Mean Classifications													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	67.83	67.78	67.64	67.49	66.92	66.22	61.96	58.04	56.49	56.79	58.50	60.33	60.63
11	68.42	68.32	68.28	68.13	67.69	67.23	63.17	58.37	56.78	57.30	58.64	60.94	61.17
12	69.13	69.07	69.06	68.94	68.59	68.32	64.26	59.06	57.17	57.80	58.64	62.08	62.15
13	69.70	69.70	69.66	69.58	69.37	69.05	65.23	59.86	57.32	58.56	58.89	62.81	62.80
14	70.08	70.11	70.04	70.05	69.92	69.88	66.22	60.32	57.33	58.14	59.24	63.49	63.22
15	70.99	71.08	71.02	71.04	71.08	70.94	67.64	61.30	57.75	58.99	59.34	64.52	63.82
16	72.11	72.24	72.22	72.28	72.31	71.78	68.46	61.58	58.30	59.76	59.80	65.22	64.18
17	72.44	72.61	72.67	72.66	72.86	72.31	69.30	61.81	58.78	59.60	59.97	65.41	64.72
18	73.46	73.63	73.63	73.73	73.79	73.20	70.83	62.80	59.46	59.89	59.68	67.14	65.41
19	74.05	74.20	74.26	74.31	74.37	73.48	71.22	63.64	59.99	60.27	60.50	68.78	66.77
20	74.96	75.07	75.14	75.18	75.30	74.50	72.06	64.20	60.23	60.90	60.93	69.87	67.55
21	75.51	75.64	75.76	75.78	75.93	74.74	72.54	65.45	60.56	61.22	61.41	70.89	68.45
22	76.23	76.36	76.53	76.48	76.77	75.78	73.28	66.08	60.66	61.65	61.84	71.48	69.01
23	76.85	76.92	77.07	77.02	77.19	76.34	74.10	66.26	60.76	61.88	62.41	72.02	69.60
24	77.07	77.17	77.36	77.34	77.61	76.76	74.55	66.81	61.49	62.57	62.97	72.99	70.20
25	77.72	77.92	78.00	77.90	78.02	77.20	74.93	67.25	61.61	62.68	63.46	73.62	70.90
26	78.05	78.21	78.28	78.13	78.23	77.49	75.45	68.12	62.22	63.20	64.17	74.78	71.43
27	78.40	78.58	78.70	78.46	78.64	77.77	75.91	68.47	62.67	63.80	65.03	75.14	72.02
28	78.92	79.12	79.15	79.02	79.09	78.33	76.36	69.05	62.96	63.74	65.16	76.03	72.82
29	79.24	79.50	79.48	79.33	79.35	78.69	76.66	69.33	63.50	64.02	65.43	76.74	73.44
30	79.58	79.84	79.89	79.69	79.74	79.10	77.12	69.99	63.76	64.40	66.19	77.70	74.22
31	79.81	79.98	80.06	79.75	79.77	79.18	77.23	70.05	64.12	64.83	66.78	78.16	74.70
32	79.99	80.17	80.18	79.88	79.94	79.36	77.67	70.85	64.66	65.10	67.16	78.86	75.61

Table B.15 - Percentage Classification Rate of trained RBF networks for the test set of the six class Robot Location problem.

Robot Location Test Set - Standard Deviations

Robot Location - Test Set - Standard Deviations													
	Transistor Curve											Gaussian	
Cts	0	1	2	3	4	5	6	7	8	9	10	11	12
10	1.14	1.21	1.21	1.24	1.19	1.11	1.44	0.87	1.93	1.38	1.19	1.30	1.11
11	1.40	1.41	1.40	1.42	1.21	0.88	1.12	0.80	1.35	1.76	1.13	1.08	1.05
12	1.69	1.73	1.79	1.81	1.88	2.01	1.19	1.14	1.51	1.41	1.44	1.82	0.96
13	1.91	1.79	1.87	1.85	1.89	1.83	1.36	0.99	1.69	1.42	2.04	1.66	1.12
14	1.93	1.87	1.96	1.98	1.92	1.87	1.81	0.87	1.79	1.47	2.12	1.59	1.29
15	1.65	1.62	1.55	1.56	1.60	1.74	2.16	1.20	1.71	1.70	2.29	1.85	1.12
16	1.71	1.70	1.66	1.65	1.59	1.67	2.23	1.18	1.87	1.85	2.29	1.54	1.05
17	1.25	1.23	1.30	1.24	1.43	1.52	1.60	1.54	2.03	1.64	2.13	1.33	1.22
18	1.16	1.24	1.19	1.30	1.14	1.18	1.37	1.43	2.25	1.76	1.88	1.83	1.24
19	1.52	1.49	1.47	1.49	1.60	1.34	1.37	1.51	2.29	1.78	1.51	1.96	1.39
20	1.32	1.33	1.30	1.23	1.30	1.01	1.35	1.73	1.72	1.57	1.28	1.99	1.33
21	1.35	1.30	1.41	1.41	1.55	1.52	1.14	1.55	1.77	1.65	1.30	1.62	1.79
22	1.16	1.20	1.25	1.25	1.37	1.06	1.01	1.50	1.74	1.54	1.33	1.57	1.37
23	1.11	1.11	1.22	1.11	1.29	1.41	1.25	2.13	2.27	1.48	1.33	1.85	1.40
24	0.91	0.94	0.90	0.94	0.95	1.18	1.28	2.27	2.19	1.02	1.31	1.65	1.63
25	1.21	1.15	1.16	1.17	1.17	1.42	1.41	1.99	2.09	1.30	1.68	1.61	1.57
26	1.25	1.20	1.19	1.09	1.23	1.44	1.39	1.96	1.86	1.33	1.16	1.83	1.69
27	1.02	1.09	0.96	0.98	1.16	1.01	1.15	1.82	1.68	1.39	1.44	1.67	1.57
28	0.80	0.79	0.73	0.78	0.91	1.10	1.25	2.07	2.21	1.26	1.89	1.60	1.70
29	1.11	1.11	1.03	1.03	1.12	1.16	1.06	2.15	2.13	1.33	1.80	1.45	1.49
30	0.94	0.97	0.88	0.98	1.07	1.03	1.22	1.98	1.85	1.21	1.91	1.40	1.88
31	1.00	0.99	1.00	1.02	0.96	0.84	1.02	1.76	2.19	1.24	1.84	1.66	1.81
32	0.80	0.84	0.78	0.69	0.70	0.80	1.08	1.57	1.76	1.21	1.54	1.59	1.99

Table B.16 - Standard Deviations of Mean Classification Percentages in Table B.15

Appendix C

Quantisation Experiments - Classification Results

This appendix contains the results for the quantisation experiments discussed in Chapter 7. Results for different levels of parameter quantisation on the test set for all four problems are given. As explained in Chapter 7, the weight values were found using full 64-bit floating point arithmetic and the quantisation was only introduced for the forward passes of the test data through the trained network. The classification performances are given as percentages along with ± 1 standard deviation of each result.

The results for each problem have been tabulated as follows. The first table in each section contains the average classification performances for the problems for RBFs with 15, 31 and 63 centres and no parameter quantisation. The remaining three tables then summarise the results for the 15, 31 and 63 centre networks with different levels of quantisation. The levels of quantisation chosen are defined as hidden layer quantisation and output layer quantisation. This corresponds to quantising the centre positions in the hidden layer and the weight and threshold values in the output layer to the precision indicated, eg *16-bit : 16-bit* means that both the centre positions and the output weights and thresholds were quantised to 16-bit precision.

Gaussian Distributions

Gaussian Distributions - No Quantisation			
Curve	15 Centres	31 Centres	63 Centres
0	86.71 ± 0.27	86.31 ± 0.38	83.85 ± 1.02
1	86.70 ± 0.30	86.18 ± 0.41	83.09 ± 1.13
2	86.69 ± 0.24	86.13 ± 0.40	82.97 ± 1.14
3	86.67 ± 0.27	86.14 ± 0.32	81.98 ± 1.19
4	86.69 ± 0.25	86.17 ± 0.44	83.48 ± 1.04
5	86.76 ± 0.30	86.33 ± 0.50	83.21 ± 0.99
6	87.04 ± 0.27	86.31 ± 0.63	81.55 ± 1.18
7	84.55 ± 0.91	85.24 ± 0.64	82.07 ± 1.36
8	82.97 ± 1.26	84.64 ± 0.80	81.62 ± 1.18
9	86.74 ± 0.59	87.15 ± 0.44	83.16 ± 1.76
10	86.79 ± 0.54	87.13 ± 0.49	84.75 ± 1.13

Table C.1 - Classification performance for all 3 networks with no parameter quantisation

Gaussian Distributions - 15 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	86.71 ± 0.27	86.58 ± 0.36	86.31 ± 0.60	72.39 ± 13.14	72.84 ± 13.54
1	86.70 ± 0.30	86.63 ± 0.28	86.25 ± 0.66	75.24 ± 12.53	75.33 ± 12.00
2	86.69 ± 0.24	86.62 ± 0.28	86.33 ± 0.58	74.99 ± 13.79	74.72 ± 13.75
3	86.67 ± 0.27	86.66 ± 0.24	86.47 ± 0.42	77.92 ± 9.66	77.80 ± 9.25
4	86.69 ± 0.25	86.63 ± 0.33	86.47 ± 0.42	75.59 ± 12.05	76.07 ± 12.29
5	86.76 ± 0.30	86.74 ± 0.28	86.64 ± 0.27	80.92 ± 7.24	80.42 ± 8.21
6	87.03 ± 0.28	87.01 ± 0.35	86.93 ± 0.36	84.62 ± 3.51	84.37 ± 3.85
7	84.55 ± 0.91	84.54 ± 0.92	84.52 ± 0.96	82.64 ± 3.68	82.39 ± 5.10
8	82.97 ± 1.26	82.98 ± 1.25	83.01 ± 1.32	82.87 ± 1.37	82.90 ± 1.43
9	86.74 ± 0.59	86.73 ± 0.59	86.76 ± 0.57	86.73 ± 0.64	86.75 ± 0.60
10	86.79 ± 0.54	86.78 ± 0.54	86.75 ± 0.57	86.75 ± 0.53	86.72 ± 0.55

Table C.2 - Quantisation classification performances for the 15 centre network

Gaussian Distributions - 31 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	86.31 ± 0.37	84.88 ± 3.13	84.32 ± 3.85	58.22 ± 13.45	58.91 ± 14.37
1	86.15 ± 0.42	85.46 ± 1.18	84.77 ± 1.47	58.24 ± 13.50	58.53 ± 13.85
2	86.14 ± 0.38	85.41 ± 1.35	84.74 ± 2.00	61.94 ± 15.33	61.82 ± 15.22
3	86.13 ± 0.32	85.63 ± 0.86	85.07 ± 1.74	57.35 ± 13.68	57.73 ± 13.44
4	86.17 ± 0.44	84.85 ± 2.19	84.53 ± 1.90	61.28 ± 12.21	62.63 ± 13.07
5	86.33 ± 0.50	85.57 ± 2.08	84.82 ± 2.85	57.98 ± 10.60	59.28 ± 10.93
6	86.30 ± 0.63	85.74 ± 1.45	84.95 ± 2.75	62.11 ± 11.20	63.20 ± 11.65
7	85.24 ± 0.64	85.14 ± 0.66	84.74 ± 0.80	69.30 ± 14.75	69.31 ± 14.31
8	84.64 ± 0.81	84.60 ± 0.84	84.39 ± 1.05	79.86 ± 8.32	79.42 ± 8.73
9	87.15 ± 0.44	87.14 ± 0.45	87.12 ± 0.51	87.13 ± 0.48	87.12 ± 0.51
10	87.13 ± 0.49	87.13 ± 0.49	87.08 ± 0.49	87.13 ± 0.50	87.08 ± 0.48

Table C.3 - Quantisation classification performances for the 31 centre network

Gaussian Distributions - 63 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	83.78 ± 1.03	79.56 ± 6.96	76.96 ± 6.57	54.54 ± 10.97	53.98 ± 10.26
1	83.01 ± 1.17	77.83 ± 9.49	70.62 ± 10.90	52.98 ± 7.01	52.49 ± 6.28
2	82.95 ± 1.19	76.94 ± 7.79	72.49 ± 11.14	52.03 ± 6.09	52.75 ± 6.43
3	81.94 ± 1.23	77.26 ± 7.30	71.96 ± 9.58	50.88 ± 2.83	51.40 ± 4.38
4	83.46 ± 1.09	78.63 ± 5.02	69.42 ± 10.36	52.01 ± 6.13	52.60 ± 7.94
5	83.11 ± 0.97	77.41 ± 8.24	72.58 ± 7.98	50.83 ± 3.10	51.98 ± 6.01
6	81.57 ± 1.21	77.81 ± 5.82	74.60 ± 6.32	53.71 ± 7.44	53.43 ± 5.67
7	82.08 ± 1.30	78.75 ± 6.18	69.57 ± 9.81	58.86 ± 12.05	56.73 ± 9.63
8	81.64 ± 1.20	80.47 ± 2.76	76.62 ± 5.04	60.78 ± 11.51	59.53 ± 11.43
9	83.16 ± 1.75	83.14 ± 1.79	82.94 ± 1.84	80.54 ± 7.69	80.41 ± 7.43
10	84.75 ± 1.13	84.76 ± 1.12	84.54 ± 1.07	84.71 ± 1.11	84.50 ± 1.08

Table C.4 - Quantisation classification performances for the 63 centre network

Speaker Recognition

Speaker Recognition - No Quantisation			
Curve	15 Centres	31 Centres	63 Centres
0	73.25 ± 2.68	72.13 ± 1.69	72.05 ± 1.90
1	72.91 ± 2.31	72.11 ± 1.81	71.33 ± 2.22
2	73.31 ± 2.61	71.87 ± 1.98	71.25 ± 1.76
3	73.47 ± 2.65	72.59 ± 1.72	71.97 ± 2.29
4	73.28 ± 2.65	72.53 ± 1.55	72.35 ± 2.91
5	73.57 ± 2.54	73.25 ± 2.42	72.53 ± 2.71
6	74.69 ± 2.37	74.64 ± 2.16	74.24 ± 2.34
7	74.96 ± 1.98	76.32 ± 2.44	74.64 ± 2.16
8	70.59 ± 2.13	72.83 ± 2.14	70.21 ± 3.50
9	61.60 ± 4.55	67.87 ± 2.81	70.83 ± 3.22
10	63.92 ± 4.17	68.29 ± 2.73	72.19 ± 2.96

Table C.5 - Classification performance for all 3 networks with no parameter quantisation

Speaker Recognition - 15 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	73.23 ± 2.64	73.17 ± 2.65	73.39 ± 2.85	67.76 ± 8.95	67.71 ± 9.38
1	72.91 ± 2.22	72.99 ± 2.33	72.96 ± 2.52	60.61 ± 9.42	60.85 ± 9.49
2	73.23 ± 2.61	73.17 ± 2.71	73.01 ± 2.54	67.52 ± 7.51	67.09 ± 7.64
3	73.41 ± 2.64	73.28 ± 2.69	73.09 ± 2.71	68.67 ± 5.16	68.67 ± 5.50
4	73.31 ± 2.68	73.36 ± 2.49	73.25 ± 2.63	70.59 ± 3.76	70.32 ± 4.08
5	73.55 ± 2.54	73.55 ± 2.58	73.73 ± 2.72	70.67 ± 5.03	70.61 ± 5.25
6	74.75 ± 2.38	74.85 ± 2.51	74.77 ± 2.63	73.01 ± 3.06	72.99 ± 2.95
7	74.96 ± 1.98	74.99 ± 1.98	75.36 ± 2.35	74.37 ± 2.53	74.56 ± 2.68
8	70.61 ± 2.11	70.59 ± 2.13	70.40 ± 2.06	70.29 ± 2.35	70.19 ± 2.29
9	61.60 ± 4.55	61.60 ± 4.55	61.63 ± 4.54	61.57 ± 4.60	61.55 ± 4.58
10	63.92 ± 4.17	63.79 ± 4.21	63.81 ± 4.01	63.68 ± 4.44	63.84 ± 4.46

Table C.6 - Quantisation classification performances for the 15 centre network

Speaker Recognition - 31 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	72.16 ± 1.56	72.13 ± 1.81	71.92 ± 1.89	56.43 ± 13.16	56.40 ± 12.98
1	72.11 ± 1.84	71.84 ± 1.72	72.03 ± 1.95	56.72 ± 13.24	56.56 ± 13.05
2	71.87 ± 1.83	72.03 ± 1.65	72.11 ± 1.80	57.95 ± 9.55	57.71 ± 9.74
3	72.59 ± 1.64	72.75 ± 2.03	72.27 ± 2.06	63.52 ± 8.21	63.87 ± 8.60
4	72.53 ± 1.75	72.51 ± 1.66	72.53 ± 1.85	64.08 ± 8.26	63.39 ± 8.31
5	73.23 ± 2.43	73.15 ± 2.53	73.25 ± 2.42	66.99 ± 6.59	66.80 ± 6.67
6	74.69 ± 2.16	74.67 ± 2.11	74.21 ± 2.18	69.33 ± 6.04	69.15 ± 6.15
7	76.35 ± 2.49	76.24 ± 2.61	76.27 ± 2.61	74.93 ± 2.75	75.09 ± 2.89
8	72.83 ± 2.14	72.77 ± 2.06	72.75 ± 2.33	72.77 ± 2.14	72.75 ± 2.34
9	67.87 ± 2.81	67.89 ± 2.79	67.87 ± 2.60	68.05 ± 2.63	67.89 ± 2.68
10	68.32 ± 2.73	68.32 ± 2.73	68.27 ± 2.71	68.24 ± 2.74	68.21 ± 2.76

Table C.7 - Quantisation classification performances for the 31 centre network

Speaker Recognition - 63 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	72.00 ± 1.93	71.20 ± 2.32	70.93 ± 2.28	44.00 ± 11.53	44.03 ± 11.71
1	71.12 ± 2.46	70.40 ± 2.07	70.24 ± 3.00	43.47 ± 10.02	43.73 ± 10.81
2	71.41 ± 1.97	70.83 ± 2.18	70.67 ± 2.27	40.56 ± 10.26	40.61 ± 10.55
3	72.05 ± 2.34	71.76 ± 2.22	71.09 ± 3.18	45.28 ± 10.97	45.09 ± 11.02
4	72.37 ± 2.85	72.00 ± 2.89	71.49 ± 2.98	44.21 ± 11.64	44.13 ± 11.31
5	72.43 ± 2.93	71.92 ± 2.81	72.11 ± 3.24	48.21 ± 11.14	47.52 ± 10.87
6	74.24 ± 2.29	73.81 ± 2.32	73.55 ± 2.38	52.48 ± 12.45	51.79 ± 12.05
7	74.69 ± 2.17	74.77 ± 2.33	74.61 ± 2.19	72.24 ± 4.22	72.27 ± 4.15
8	70.24 ± 3.50	70.19 ± 3.56	69.97 ± 3.09	70.13 ± 3.29	70.19 ± 2.98
9	70.83 ± 3.22	70.80 ± 3.25	70.83 ± 3.34	71.01 ± 3.06	70.91 ± 3.10
10	72.16 ± 2.98	72.13 ± 2.82	72.08 ± 2.90	72.21 ± 2.85	72.29 ± 2.78

Table C.8 - Quantisation classification performances for the 63 centre network

Sleep State

Sleep State - No Quantisation			
Curve	15 Centres	31 Centres	63 Centres
0	71.16 ± 0.56	74.27 ± 1.49	78.10 ± 1.14
1	71.21 ± 0.62	74.11 ± 1.59	78.46 ± 0.83
2	71.27 ± 0.65	74.58 ± 1.56	78.67 ± 0.89
3	71.26 ± 0.58	74.29 ± 1.61	78.44 ± 1.00
4	71.28 ± 0.53	74.91 ± 1.66	79.03 ± 0.84
5	71.32 ± 0.64	74.94 ± 1.65	79.18 ± 0.76
6	71.34 ± 0.64	75.04 ± 1.66	79.28 ± 0.61
7	71.07 ± 0.60	74.31 ± 1.82	78.84 ± 0.83
8	68.93 ± 0.54	69.59 ± 1.41	71.43 ± 2.13
9	69.08 ± 0.57	68.30 ± 0.46	66.99 ± 0.83
10	69.43 ± 0.49	69.00 ± 0.55	68.29 ± 0.72

Table C.9 - Classification performance for all 3 networks with no parameter quantisation

Sleep State - 15 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	71.17 ± 0.56	71.12 ± 0.52	71.17 ± 0.53	58.30 ± 11.41	58.26 ± 11.38
1	71.21 ± 0.63	71.18 ± 0.65	71.19 ± 0.69	60.05 ± 12.80	59.96 ± 12.75
2	71.26 ± 0.65	71.26 ± 0.63	71.28 ± 0.64	56.80 ± 11.25	56.69 ± 11.24
3	71.26 ± 0.57	71.25 ± 0.61	71.29 ± 0.62	64.04 ± 10.83	63.85 ± 10.97
4	71.28 ± 0.53	71.26 ± 0.55	71.28 ± 0.57	64.61 ± 9.25	64.63 ± 8.96
5	71.32 ± 0.64	71.34 ± 0.69	71.34 ± 0.68	63.75 ± 9.51	63.57 ± 9.66
6	71.34 ± 0.64	71.35 ± 0.66	71.36 ± 0.69	67.71 ± 5.13	67.61 ± 5.13
7	71.08 ± 0.61	71.08 ± 0.61	71.07 ± 0.63	69.14 ± 3.18	69.01 ± 3.33
8	68.93 ± 0.54	68.93 ± 0.55	68.93 ± 0.54	68.77 ± 0.83	68.77 ± 0.83
9	69.08 ± 0.57	69.08 ± 0.56	69.08 ± 0.61	69.11 ± 0.56	69.08 ± 0.60
10	69.43 ± 0.50	69.43 ± 0.49	69.41 ± 0.49	69.42 ± 0.48	69.41 ± 0.49

Table C.10 - Quantisation classification performances for the 15 centre network

Sleep State - 31 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	74.27 ± 1.47	73.30 ± 1.75	73.15 ± 1.85	40.19 ± 10.27	40.16 ± 10.45
1	74.11 ± 1.59	73.12 ± 1.64	73.06 ± 1.84	37.41 ± 9.36	37.60 ± 9.32
2	74.54 ± 1.56	73.96 ± 2.13	73.79 ± 2.39	44.10 ± 11.35	44.28 ± 11.19
3	74.30 ± 1.61	73.67 ± 1.51	73.56 ± 1.63	44.66 ± 13.60	44.72 ± 13.58
4	74.90 ± 1.64	74.06 ± 1.91	74.02 ± 1.89	46.91 ± 15.06	46.94 ± 15.10
5	74.96 ± 1.65	74.46 ± 1.58	74.29 ± 1.67	42.54 ± 13.30	43.07 ± 13.05
6	75.03 ± 1.64	74.42 ± 1.66	74.33 ± 1.74	43.67 ± 10.27	43.25 ± 9.83
7	74.33 ± 1.83	74.13 ± 1.70	74.00 ± 1.94	52.94 ± 14.02	52.95 ± 13.86
8	69.59 ± 1.40	69.51 ± 1.42	69.47 ± 1.52	62.88 ± 10.11	62.97 ± 9.90
9	68.30 ± 0.46	68.29 ± 0.45	68.30 ± 0.49	68.32 ± 0.43	68.31 ± 0.48
10	69.00 ± 0.55	69.00 ± 0.56	68.97 ± 0.57	68.98 ± 0.56	68.97 ± 0.58

Table C.11 - Quantisation classification performances for the 31 centre network

Sleep State - 63 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	78.08 ± 1.15	74.52 ± 2.47	74.13 ± 2.59	35.21 ± 5.02	35.10 ± 4.38
1	78.42 ± 0.85	75.33 ± 3.59	74.79 ± 3.29	36.27 ± 7.14	36.75 ± 7.92
2	78.61 ± 0.92	75.42 ± 2.34	74.53 ± 3.06	38.64 ± 9.32	38.71 ± 9.49
3	78.41 ± 1.01	75.62 ± 3.08	75.55 ± 3.03	39.39 ± 9.53	39.36 ± 9.75
4	78.98 ± 0.84	74.64 ± 4.80	73.06 ± 6.75	37.48 ± 8.77	37.71 ± 8.89
5	79.16 ± 0.80	76.39 ± 3.02	75.70 ± 2.94	37.81 ± 9.33	38.54 ± 9.85
6	79.28 ± 0.60	76.83 ± 3.29	76.62 ± 2.88	42.92 ± 15.00	43.03 ± 14.13
7	78.83 ± 0.82	77.55 ± 1.68	76.49 ± 2.33	37.03 ± 6.96	36.86 ± 6.67
8	71.42 ± 2.13	70.96 ± 2.32	70.95 ± 2.11	47.53 ± 11.36	48.25 ± 11.47
9	66.99 ± 0.83	66.98 ± 0.81	66.93 ± 0.84	66.92 ± 0.84	66.89 ± 0.89
10	68.29 ± 0.72	68.28 ± 0.71	68.25 ± 0.73	68.26 ± 0.75	68.26 ± 0.74

Table C.12 - Quantisation classification performances for the 63 centre network

Robot Location

Robot Location - No Quantisation			
Curve	15 Centres	31 Centres	63 Centres
0	70.99 ± 1.65	79.81 ± 1.00	84.28 ± 0.63
1	71.08 ± 1.62	79.98 ± 0.99	84.44 ± 0.65
2	71.02 ± 1.55	80.06 ± 1.00	84.28 ± 0.67
3	71.04 ± 1.56	79.75 ± 1.02	84.03 ± 0.74
4	71.08 ± 1.60	79.77 ± 0.96	84.22 ± 0.61
5	70.94 ± 1.74	79.18 ± 0.84	83.99 ± 0.72
6	67.64 ± 2.16	77.23 ± 1.02	81.93 ± 0.77
7	61.30 ± 1.20	70.05 ± 1.76	76.05 ± 1.03
8	57.75 ± 1.71	64.12 ± 2.19	72.46 ± 1.50
9	58.99 ± 1.70	64.83 ± 1.24	72.73 ± 1.29
10	59.34 ± 2.29	66.78 ± 1.84	77.69 ± 2.11

Table C.13 - Classification performance for all 3 networks with no parameter quantisation

Robot Location - 15 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	70.99 ± 1.64	70.73 ± 1.82	70.70 ± 1.77	49.10 ± 13.47	48.98 ± 13.63
1	71.06 ± 1.61	70.76 ± 1.88	70.73 ± 2.01	50.95 ± 12.38	51.06 ± 12.34
2	71.03 ± 1.56	70.71 ± 1.73	70.73 ± 1.83	50.49 ± 13.58	50.50 ± 13.59
3	71.04 ± 1.57	70.88 ± 1.92	70.83 ± 2.05	50.98 ± 13.42	50.81 ± 13.31
4	71.10 ± 1.58	70.79 ± 1.72	70.70 ± 1.81	54.60 ± 9.83	54.60 ± 9.98
5	70.93 ± 1.76	70.74 ± 1.76	70.59 ± 1.82	62.70 ± 8.19	62.64 ± 8.28
6	67.63 ± 2.15	67.64 ± 2.17	67.62 ± 2.15	64.44 ± 3.57	64.45 ± 3.56
7	61.30 ± 1.20	61.32 ± 1.20	61.30 ± 1.19	60.98 ± 1.39	60.96 ± 1.41
8	57.75 ± 1.71	57.75 ± 1.73	57.74 ± 1.66	57.74 ± 1.71	57.71 ± 1.68
9	58.99 ± 1.70	59.00 ± 1.70	59.02 ± 1.68	58.97 ± 1.78	58.99 ± 1.78
10	59.34 ± 2.29	59.34 ± 2.29	59.33 ± 2.30	59.54 ± 2.52	59.56 ± 2.51

Table C.14 - Quantisation classification performances for the 15 centre network

Robot Location - 31 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	79.78 ± 1.02	78.96 ± 1.82	78.73 ± 2.06	39.03 ± 16.73	39.14 ± 16.69
1	79.98 ± 1.01	79.22 ± 1.69	78.79 ± 1.76	35.88 ± 13.38	36.00 ± 13.42
2	80.05 ± 1.01	79.50 ± 1.74	79.14 ± 1.87	43.44 ± 19.72	43.36 ± 19.87
3	79.75 ± 0.99	79.31 ± 1.31	79.08 ± 1.53	46.86 ± 14.61	47.30 ± 14.23
4	79.78 ± 0.93	79.65 ± 1.34	79.43 ± 1.36	48.55 ± 14.65	48.52 ± 14.39
5	79.20 ± 0.82	79.06 ± 1.10	78.90 ± 1.20	51.82 ± 16.09	51.97 ± 15.88
6	77.24 ± 1.00	77.22 ± 1.23	77.00 ± 1.25	62.65 ± 7.93	62.32 ± 8.03
7	70.05 ± 1.76	70.11 ± 1.77	70.12 ± 1.78	68.10 ± 2.58	68.08 ± 2.58
8	64.12 ± 2.19	64.12 ± 2.18	64.10 ± 2.15	64.16 ± 2.16	64.16 ± 2.17
9	64.83 ± 1.24	64.84 ± 1.24	64.84 ± 1.21	64.83 ± 1.28	64.85 ± 1.24
10	66.78 ± 1.85	66.79 ± 1.86	66.78 ± 1.86	66.63 ± 2.14	66.64 ± 2.14

Table C.15 - Quantisation classification performances for the 31 centre network

Robot Location - 63 Centres					
Curve	16-bit : 16-bit	12-bit : 12-bit	8-bit : 12-bit	12-bit : 8-bit	8-bit : 8-bit
0	84.32 ± 0.66	82.14 ± 2.15	82.12 ± 2.40	33.96 ± 15.62	34.80 ± 15.37
1	84.45 ± 0.67	82.66 ± 2.15	82.47 ± 2.43	32.52 ± 18.43	32.74 ± 18.58
2	84.31 ± 0.69	82.57 ± 1.98	82.40 ± 1.93	24.74 ± 13.94	24.42 ± 13.37
3	84.05 ± 0.78	82.95 ± 1.89	82.76 ± 2.11	34.48 ± 15.51	34.72 ± 15.58
4	84.25 ± 0.59	83.21 ± 1.63	82.80 ± 1.93	29.57 ± 18.21	29.62 ± 17.63
5	83.98 ± 0.71	83.33 ± 1.24	82.59 ± 1.91	32.98 ± 12.58	32.30 ± 12.32
6	81.93 ± 0.77	81.50 ± 1.16	81.39 ± 1.41	49.17 ± 11.44	50.18 ± 10.45
7	76.05 ± 1.04	76.03 ± 1.12	75.86 ± 1.15	66.57 ± 7.86	66.24 ± 8.30
8	72.46 ± 1.50	72.46 ± 1.48	72.41 ± 1.46	72.48 ± 1.60	72.43 ± 1.58
9	72.73 ± 1.29	72.73 ± 1.28	72.70 ± 1.30	72.70 ± 1.25	72.69 ± 1.28
10	77.70 ± 2.11	77.69 ± 2.12	77.72 ± 2.10	77.56 ± 2.02	77.60 ± 1.97

Table C.16 - Quantisation classification performances for the 63 centre network

Appendix D

Chip Development Boards and Experiments

This appendix illustrates how the PAR chip development board was configured for taking the results presented in Chapter 8 and describes, via a flow diagram, how the controlling software allowed the results to be collected.

Board Set-Up

This section describes how the signals necessary to operate the PAR chip as a classifier were generated using hardware on the PAR chip development board. The external hardware requirement is shown in Figure D.1.

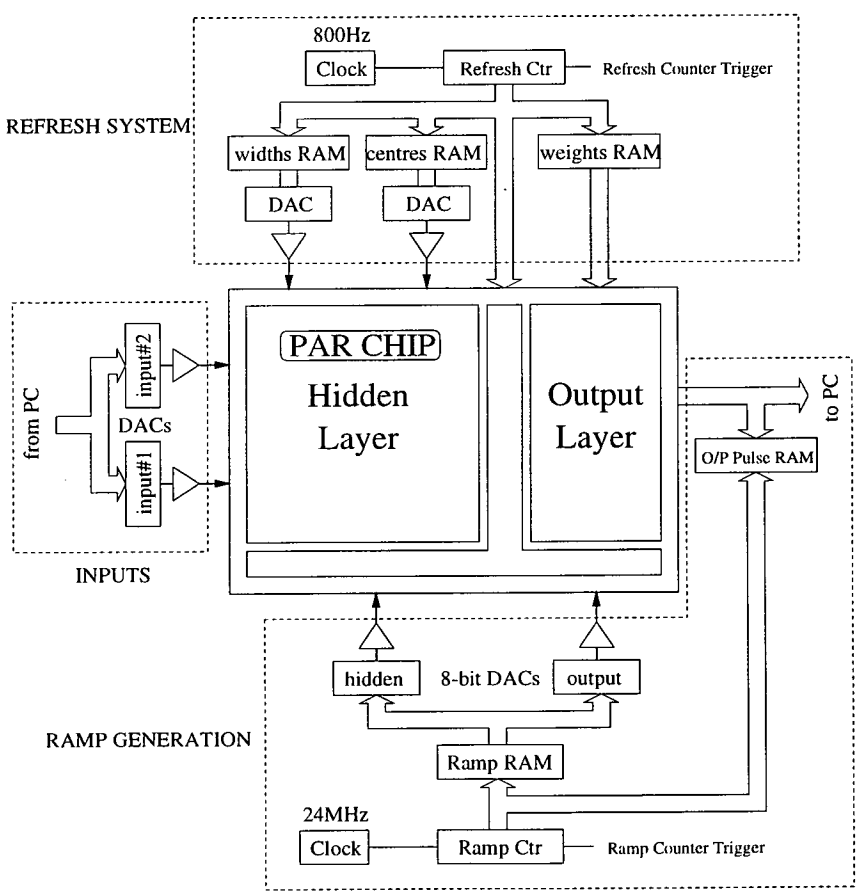


Figure D.1 - Schematic Diagram of the PAR chip Classification System

REFRESH SYSTEM

The different neural parameters were downloaded to the *widths*, *centres* and *weights* RAMs before vector classification began. The *widths* and *centres* RAMs were each 12 bits wide and their digital words were read to off-chip 12-bit voltage DACs. The *weights* RAM was 8 bits wide and output its digital words to the on-chip current DAC.

The top 7 bits of a 9-bit refresh counter were used to address these RAMs during normal operation. By Ex-ORing some of the LSBs, it was possible to generate loading signals with guard bands as described in Chapter 6.

The refresh operation was designed to run continuously and to be transparent to the vector throughput. It could be reset using the *Refresh Counter Trigger* control bit.

RAMP GENERATION

A single RAM and two 8-bit off-chip voltage DACs were used to generate the linear, double-sided, hidden layer and output layer ramps. These ramps were applied to the hidden layer and output layer PWM neurons respectively, and were used to convert the analogue voltages to pulse widths. The Ramp RAM was addressed using a 10-bit counter that also addressed the Output Pulse RAM. Thus the output pulses from the chip were recorded as the ramps were generated.

As with the Refresh RAM, the Ramp RAM was loaded before vector processing began. Unlike the Refresh counter, however, the Ramp counter was designed to terminate, until reset, after it had cycled through four 256-location RAM pages. The RAM pages consisted of ramp pages interleaved with blank pages as shown in Table D.1. By configuring the Ramps in this way, it was possible to prevent overlap of the pulses from the hidden layer and output layer and allow output pulses that "spilled-over" the Output Pulse page (page 3 in the Output Pulse RAM) to be recorded.

Ramp RAM Contents	
Page	Contents
1	Hidden Layer Ramp
2	Blank Page
3	Output Layer Ramp
4	Blank Page

Table D.1 - Contents of the Ramp RAM

INPUTS

The two components of each input vector were presented to the chip as analogue voltages between 0V and 3V. To generate the voltages, the analogue values of the input vector components (stored in PC memory) were converted to 12-bit digital words and downloaded and latched into the 12-bit DACs. The input DACs were calibrated to produce output voltages in the desired range.

Classification Software Operation

The operation of the software used to control the PAR chip classification system is described by the flow diagram in Figure D.2.

The software worked as follows.

- **Initialisation** - After reading all the input vectors, neural parameters and look-up tables into computer memory, the hardware board was accessed and all the neural parameters and the ramps were downloaded to the relevant RAM chips.
- **Vector Presentation** - Each input vector component was loaded and latched into the 12-bit input DACs and the Ramp RAM was subsequently triggered. This fired the ramps onto the board. The generated output pulses were automatically read into the Output Pulse RAM.
- **Pulse Width Calculation** - The Output Pulse RAM was interrogated by reading its contents back into PC memory. The widths of the stored pulses were then calculated.
- **Classification Decision** - The classification of the input vector was made by assigning it to the class represented by the output that had produced the longest pulse. The errors between the actual and target output vectors were then calculated.
- **Result Presentation** - Once all the vectors had been processed by the network, the performance results (classification rate and MSE) were recorded.
- **Optional Learning** - If the system was required to implement chip-in-the-loop learning, then the output weights and thresholds were updated using the LMS Rule. The new weights and thresholds were over-written into the *weights* RAM and the chip was refreshed for 1 second before the training vectors were re-processed. If learning was not implemented, no weight updates were made and the data set being classified was simply fed through the hardware again.

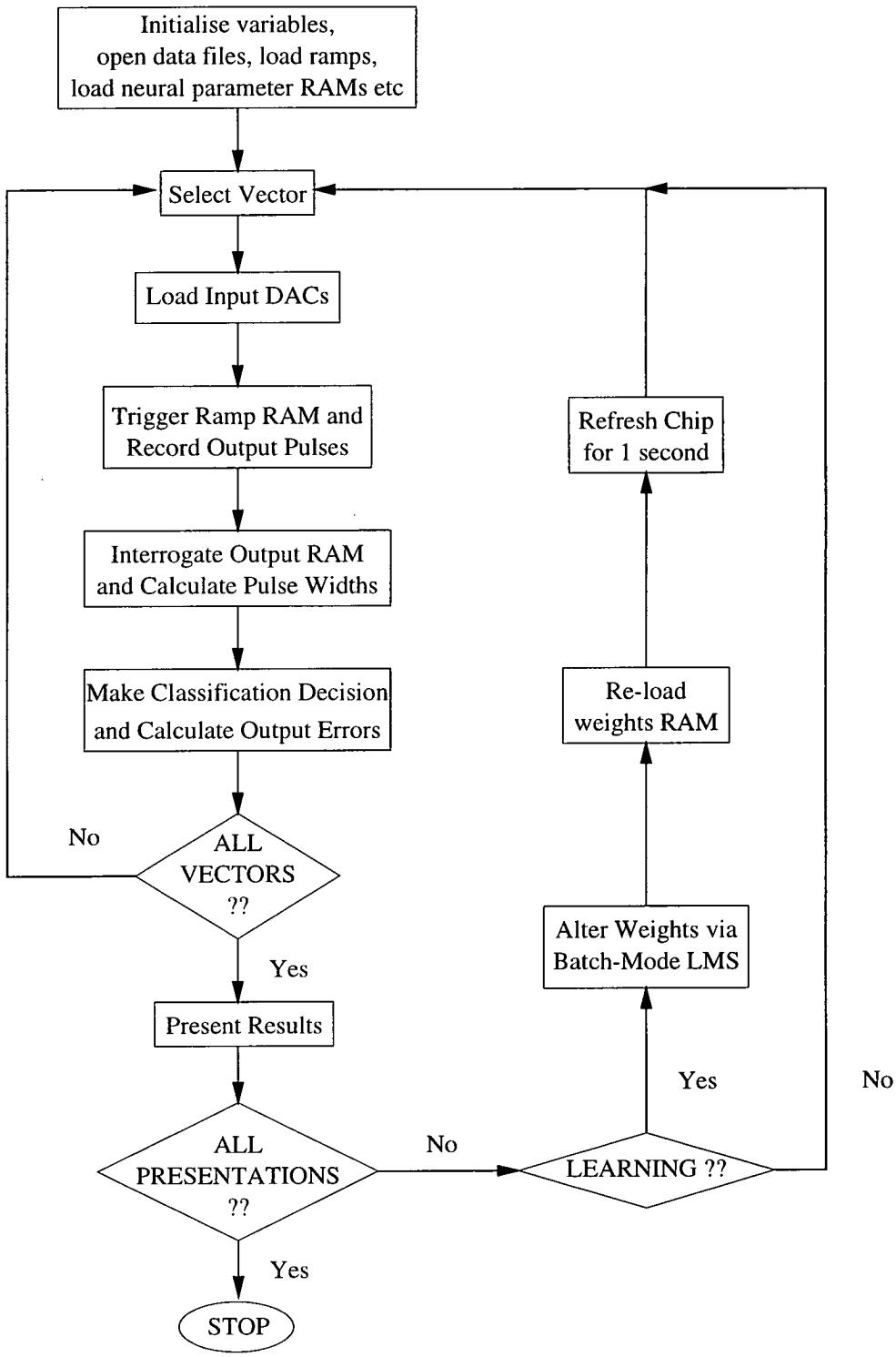


Figure D.2 - Flow diagram describing the operation of the software used to control the hardware classification system

Performance Anomaly

During the development of the classification and learning system, a problem was observed with the software control of the board.

When a delay was added between loading the input DACs and triggering the Ramp RAM, the classification performance of the PAR chip varied periodically with the length of the delay. Figure D.3 shows the classification variation with the delay in ms for four different PAR chips attempting to solve the *Hard* classification problem. Using a different PC and compiler to compile the code and control the development board resulted in the same variations. Although the results were not identical for the new PC, the same variations were noted if the time delays were altered or the code changed. It may be significant that the clock speed of the second PC was much faster than the first: this would account for the results not being identical, whilst using the same code and board could account for the same variations in performance being observed.

The performance of the board RAMs, clocks and ramps were investigated and were found to function as required. Further, all the required signals were connected to the relevant chip pins and were generated as required. Also, Chapter 6 had shown that the static performance of the chip was consistent and as expected. Thus, having ruled out a chip or a board problem, attention was focused on the software control of the board and on the chip refresh system.

For the experiments that indicated the performance variations, the chip refresh had always been **on** only during the time delay. Thus for a given, fixed, delay, the refresh would be addressing the same location in the refresh RAMs. The refresh system was therefore altered and allowed to run continuously. The periodic variation remained, though.

However, further investigations indicated that the variation appeared to depend on the time between reseting the global refresh clock and triggering the Ramp RAM. Due to the way that the board was set up and controlled, when the refresh clock was running continuously, it was reset when the Output Pulse RAM was being interrogated. Since the delay was added between loading the input DACs and triggering the Ramp RAM, the refresh counter of the continuously running, and supposedly asynchronous, refresh system would again be addressing approximately the same location in RAM for each vector presentation.

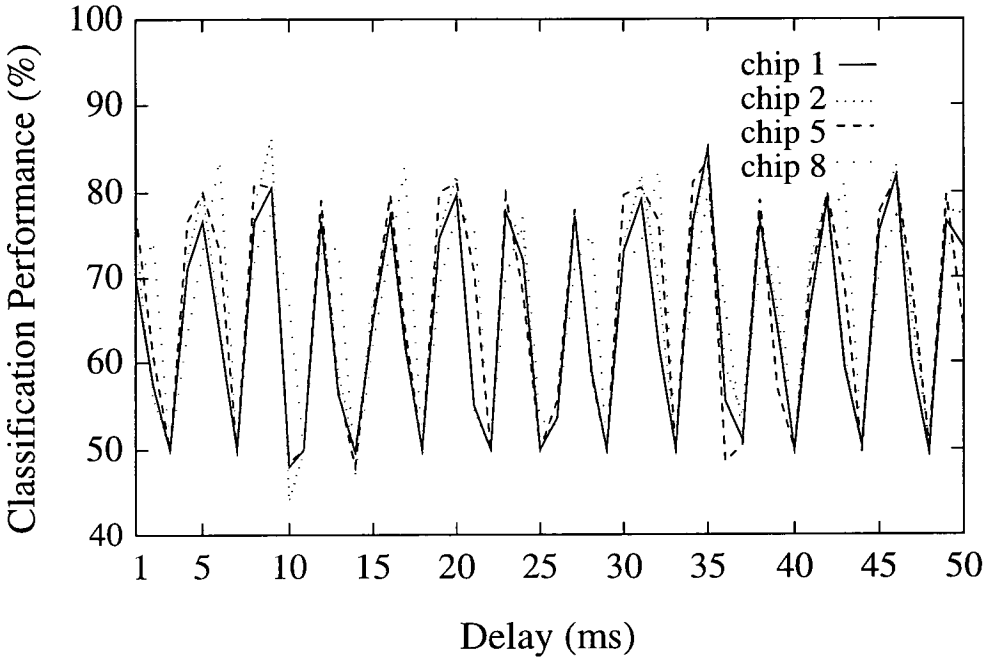


Figure D.3 - Periodic Variation in Classification Performance with added Delay

In essence, the refresh was pseudo-synchronous to the data through-put.

By not re-setting the refresh counter during normal operation, it was possible to remove the periodicity in the Classification Performance, Figure D.4. Figure D.4(a) shows the observed variation for a continuous refresh clock reset when the Output Pulse RAM was interrogated, whilst Figure D.4(b) shows the observed variation when the refresh clock is again continuous but not reset when the RAM is interrogated, ie the refresh is completely asynchronous to the vector presentation.

Clearly, there is a periodic variation when the clock is reset that is not present when refresh is completely asynchronous to the vector presentation. Furthermore, the classification performance for a system with an asynchronous clock varies from run to run, Figure D.4(b). However, it was noted that the performance of the chip when the refresh is asynchronous and continuous is generally lower than that obtained for the performance peaks when the variations are periodic. This is believed to be due to random variations introduced into the results by digital switching noise.

From these results, it was concluded that the variation in performance was indeed due to the pseudo-synchronism of the global refresh system with vector throughput. Although removing the source of this variation would be necessary in the development of a final version of this system, in the context of this thesis it was not

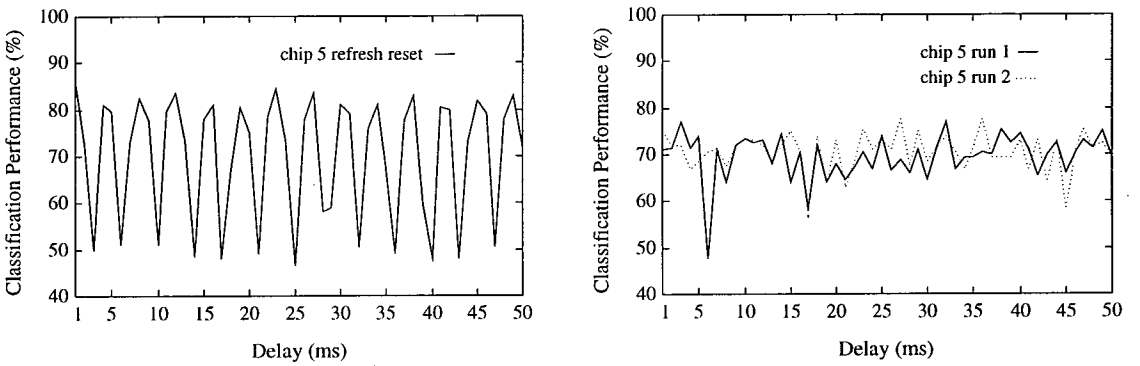


Figure D.4 - Graphs showing how the Classification Performance varies with the added Delay for a continuously running refresh clock that is (a) reset before each vector presentation and (b) never reset

justifiable to spend an unpredictable amount of time correcting it. Thus, a pragmatic decision was made to regard the performance "peaks" in Figure D.3 as a good measure of the expected performance from a final system with a corrected refresh system. The recorded results in this thesis were therefore measured when the board was configured to be at one of these peaks.

To fix the refresh system problem, and avoid a drop in performance due to digital noise, it is recommended that the refresh counter is redesigned to refresh the chip a set number of times before switching itself off. Thus, it is envisaged that the refresh system will only be used **between** vector presentations (in addition to loading the board initially and during learning). As already mentioned, at present the counter is designed to run continuously until it is terminated by a global master reset signal (MR_bar). If a more significant, and presently unused, counter bit is fed back to an AND gate as shown in Figure D.5, then the counter will terminate once bit *N-1* becomes set, ie if *N* is 12 in Figure D.5, the chip will be refreshed 16 times before the counter needs to be re-set. To restart the refresh counter, it must be globally reset and re-triggered using the *Refresh Counter Trigger* input.

Altering the board in this way will:

- allow the board to be completely refreshed between separate vector presentations

- allow the refresh to be off during vector presentations thus reducing the level of noise on the chip.

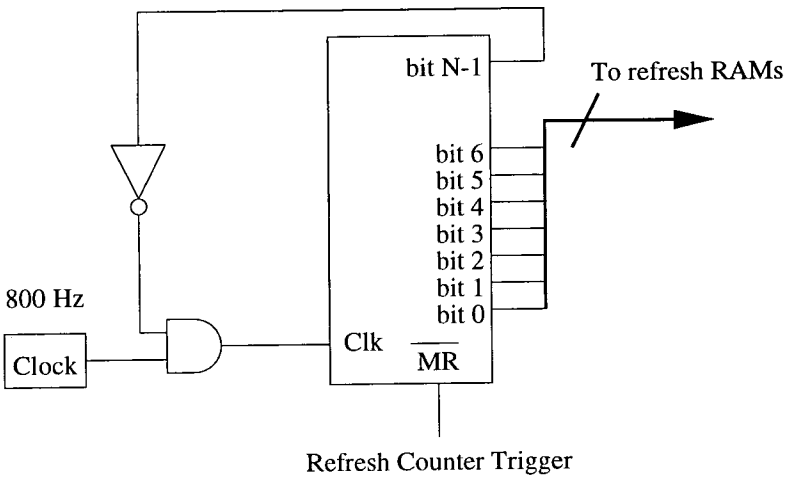


Figure D.5 - Suggested re-design for the Refresh Counter

Appendix E

Hardware Experimental Results

This appendix contains the multi-chip and multi-seed classification performance and mean squared error (MSE) measurements from the hardware experiments detailed in Chapter 8. Software results are also given for comparison purposes. For these results, the transistor non-linearity corresponding to V_{width} = 2V was used. The learning rates and training epochs for each of the experiments are tabulated below.

Learning Rate	Seed				Chip		
Problem	100	101	102	103	2	5	8
Easy	0.001	-	0.001	0.0005	0.001	0.001	0.001
Intermediate	0.01	0.01	0.01	-	0.01	0.01	0.01
Hard	0.005	0.005	0.005	-	0.005	0.005	0.005

Table E.1 - Learning Rates

Epochs	Seed				Chip		
Problem	100	101	102	103	2	5	8
Easy	4	-	4	7	6	4	5
Intermediate	20	20	20	-	20	20	20
Hard	20	20	20	-	20	20	20

Table E.2 - Epochs

The software results were obtained from a single execution of the software model of the PAR chip using the listed random number generator seed to control network training. The *Unquantised* results were obtained by using the weights generated by adaptive k-means and SVD training in software to process the training set and test set **without** any quantisation. The *Quantised* results were obtained by quantising the

64-bit floating point centre locations to 12-bit precision and the output weights and thresholds to 8-bit precision after training and before processing the vectors.

The hardware results were obtained by calculating the average classification performance from a single chip after passing the data set through 10 times. The mean classification performance over the ten runs is presented, along with an error term representing ± 1 standard deviation of the results. The average MSE over the 10 runs is also presented, however no standard deviations are presented for the MSEs as they were all less than 0.35%.

The initial results were obtained by downloading the unaltered software generated weights to the chip and using them to process the vectors. The final average results were obtained in the same manner except that the original software generated output weights and thresholds had been adapted for the chip using *chip-in-the-loop* learning.

Easy Problem - Single Chip : Multiple Seed

Classification Performance - Easy Problem					
Chip 5 Used		Software		Hardware	
Data Set	Seed	Unquantised	Quantised	Initial	Final
Training	100	100.0	100.0	95.80 ± 0.48	99.65 ± 0.34
Test	100	98.0	98.0	95.05 ± 0.50	99.40 ± 0.21
Training	102	100.0	100.0	96.85 ± 0.82	100.0 ± 0.0
Test	102	98.0	98.0	97.45 ± 0.80	99.35 ± 0.24
Training	103	100.0	100.0	99.00 ± 0.24	99.45 ± 0.28
Test	103	99.0	99.0	98.65 ± 0.34	99.15 ± 0.34

Table E.3 - Classification Performance Result Summary for the Easy Problem
Single Chip : Multiple Seed

Mean Squared Error - Easy Problem					
Chip 5 Used		Software		Hardware	
Data Set	Seed	Unquantised	Quantised	Initial	Final
Training	100	1.453×10^{-2}	1.456×10^{-2}	2.206×10^{-1}	2.038×10^{-1}
Test	100	1.890×10^{-2}	1.885×10^{-2}	2.211×10^{-1}	2.038×10^{-1}
Training	102	1.397×10^{-2}	1.398×10^{-2}	2.298×10^{-1}	2.075×10^{-1}
Test	102	2.157×10^{-2}	2.155×10^{-2}	2.298×10^{-1}	2.080×10^{-1}
Training	103	1.560×10^{-2}	1.562×10^{-2}	2.299×10^{-1}	2.092×10^{-1}
Test	103	1.908×10^{-2}	1.907×10^{-2}	2.295×10^{-1}	2.080×10^{-1}

Table E.4 - MSE Performance Summary for the Easy Problem
Single Chip : Multiple Seed

Easy Problem - Multiple Chip : Single Seed

Classification Performance - Easy Problem					
Seed was 100		Software		Hardware	
Data Set	Chip	Unquantised	Quantised	Initial	Final
Training	2	100.0	100.0	95.45 ± 0.76	99.90 ± 0.21
Test		98.0	98.0	94.90 ± 0.57	99.40 ± 0.21
Training	5	100.0	100.0	95.80 ± 0.48	99.65 ± 0.34
Test		98.0	98.0	95.05 ± 0.50	99.40 ± 0.21
Training	8	100.0	100.0	98.50 ± 0.41	100.0 ± 0.0
Test		98.0	98.0	97.80 ± 0.26	99.40 ± 0.21

Table E.5 - Classification Performance Result Summary for the Easy Problem Multiple Chip : Single Seed

Mean Squared Error - Easy Problem					
Seed was 100		Software		Hardware	
Data Set	Chip	Unquantised	Quantised	Initial	Final
Training	2	1.453×10^{-2}	1.456×10^{-2}	2.199×10^{-1}	1.899×10^{-1}
Test		1.890×10^{-2}	1.885×10^{-2}	2.204×10^{-1}	1.897×10^{-1}
Training	5	1.453×10^{-2}	1.456×10^{-2}	2.206×10^{-1}	2.038×10^{-1}
Test		1.890×10^{-2}	1.885×10^{-2}	2.211×10^{-1}	2.038×10^{-1}
Training	8	1.453×10^{-2}	1.456×10^{-2}	2.192×10^{-1}	1.954×10^{-1}
Test		1.890×10^{-2}	1.885×10^{-2}	2.197×10^{-1}	1.957×10^{-1}

Table E.6 - MSE Performance Summary for the Easy Problem Multiple Chip : Single Seed

Intermediate Problem - Single Chip : Multiple Seed

Classification Performance - Intermediate Problem					
Chip 5 used		Software		Hardware	
Data Set	Seed	Unquantised	Quantised	Initial	Final
Training	100	95.0	95.0	89.05 ± 1.26	91.80 ± 0.59
Test	100	97.5	97.5	92.50 ± 0.71	94.95 ± 0.37
Training	101	95.5	95.5	94.95 ± 0.16	94.55 ± 0.55
Test	101	96.5	96.5	95.95 ± 0.16	96.90 ± 0.21
Training	102	95.0	95.0	94.30 ± 0.26	92.50 ± 0.62
Test	102	97.5	97.5	96.80 ± 0.26	95.45 ± 0.28

Table E.7 - Classification Performance Result Summary for the Intermediate Problem Single Chip : Multiple Seed

Mean Squared Error - Intermediate Problem					
Chip 5 used		Software		Hardware	
Data Set	Seed	Unquantised	Quantised	Initial	Final
Training	100	4.187×10^{-2}	4.188×10^{-2}	2.418×10^{-1}	1.641×10^{-1}
Test	100	3.478×10^{-2}	3.496×10^{-2}	2.407×10^{-1}	1.568×10^{-1}
Training	101	4.212×10^{-2}	4.209×10^{-2}	2.384×10^{-1}	1.808×10^{-1}
Test	101	2.426×10^{-2}	2.438×10^{-2}	2.377×10^{-1}	1.763×10^{-1}
Training	102	4.353×10^{-2}	4.355×10^{-2}	2.388×10^{-1}	1.830×10^{-1}
Test	102	3.662×10^{-2}	3.662×10^{-2}	2.382×10^{-1}	1.787×10^{-1}

Table E.8 - MSE Performance Summary for the Intermediate Problem Single Chip : Multiple Seed

Intermediate Problem - Multiple Chip : Single Seed

Classification Performance - Intermediate Problem					
Seed was 100		Software		Hardware	
Data Set	Chip	Unquantised	Quantised	Initial	Final
Training	2	95.0	95.0	93.10 ± 0.66	91.00 ± 0.47
Test		97.5	97.5	95.50 ± 0.53	94.25 ± 0.26
Training	5	95.0	95.0	89.05 ± 1.26	91.80 ± 0.59
Test		97.5	97.5	92.50 ± 0.71	94.95 ± 0.37
Training	8	95.0	95.0	91.10 ± 0.70	88.00 ± 0.53
Test		97.5	97.5	93.35 ± 0.53	91.20 ± 0.63

Table E.9 - Classification Performance Result Summary for the Intermediate Problem Multiple Chip : Single Seed

Mean Squared Error - Intermediate Problem					
Seed was 100		Software		Hardware	
Data Set	Chip	Unquantised	Quantised	Initial	Final
Training	2	4.187×10^{-2}	4.188×10^{-2}	2.408×10^{-1}	1.637×10^{-1}
Test		3.478×10^{-2}	3.496×10^{-2}	2.398×10^{-1}	1.559×10^{-1}
Training	5	4.187×10^{-2}	4.188×10^{-2}	2.418×10^{-1}	1.641×10^{-1}
Test		3.478×10^{-2}	3.496×10^{-2}	2.407×10^{-1}	1.568×10^{-1}
Training	8	4.187×10^{-2}	4.188×10^{-2}	2.400×10^{-1}	1.655×10^{-1}
Test		3.478×10^{-2}	3.496×10^{-2}	2.394×10^{-1}	1.585×10^{-1}

Table E.10 - MSE Performance Summary for the Intermediate Problem Multiple Chip : Single Seed

Hard Problem - Single Chip : Multiple Seed

Classification Performance - Hard Problem					
Chip 5 used		Software		Hardware	
Data Set	Seed	Unquantised	Quantised	Initial	Final
Training	100	84.5	85.0	72.90 ± 1.10	73.75 ± 0.35
Test	100	88.5	88.0	74.35 ± 1.20	74.05 ± 0.37
Training	101	85.5	85.5	57.65 ± 1.78	74.10 ± 0.61
Test	101	88.5	88.5	61.00 ± 2.05	77.00 ± 0.78
Training	102	85.5	86.0	68.05 ± 2.13	68.60 ± 0.39
Test	102	87.5	87.0	72.80 ± 2.25	84.70 ± 0.95

Table E.11 - Classification Performance Result Summary for the Hard Problem
Single Chip : Multiple Seed

Mean Squared Error - Hard Problem					
Chip 5 used		Software		Hardware	
Data Set	Seed	Unquantised	Quantised	Initial	Final
Training	100	1.175×10^{-1}	1.175×10^{-1}	2.440×10^{-1}	2.166×10^{-1}
Test	100	9.796×10^{-2}	9.776×10^{-2}	2.434×10^{-1}	2.112×10^{-1}
Training	101	1.156×10^{-1}	1.155×10^{-1}	2.452×10^{-1}	2.235×10^{-1}
Test	101	1.021×10^{-1}	1.023×10^{-1}	2.445×10^{-1}	2.208×10^{-1}
Training	102	1.166×10^{-1}	1.167×10^{-1}	2.441×10^{-1}	2.217×10^{-1}
Test	102	9.796×10^{-2}	9.775×10^{-2}	2.434×10^{-1}	2.230×10^{-1}

Table E.12 - MSE Performance Summary for the Hard Problem
Single Chip : Multiple Seed

Hard Problem - Multiple Chip : Single Seed

Classification Performance - Hard Problem					
Seed was 100		Software		Hardware	
Data Set	Chip	Unquantised	Quantised	Initial	Final
Training	2	84.5	85.0	75.00 ± 0.75	71.65 ± 0.34
Test		88.5	88.0	78.45 ± 1.57	72.75 ± 0.35
Training	5	84.5	85.0	72.90 ± 1.10	73.75 ± 0.35
Test		88.5	88.0	74.35 ± 1.20	74.05 ± 0.37
Training	8	84.5	85.0	76.00 ± 1.39	68.60 ± 0.57
Test		88.5	88.0	74.15 ± 0.63	69.00 ± 0.41

Table E.13 - Classification Performance Result Summary for the Hard Problem Multiple Chip : Single Seed

Mean Squared Error - Hard Problem					
Seed was 100		Software		Hardware	
Data Set	Chip	Unquantised	Quantised	Initial	Final
Training	2	1.175×10^{-1}	1.175×10^{-1}	2.432×10^{-1}	2.172×10^{-1}
Test		9.796×10^{-2}	9.776×10^{-2}	2.427×10^{-1}	2.121×10^{-1}
Training	5	1.175×10^{-1}	1.175×10^{-1}	2.440×10^{-1}	2.166×10^{-1}
Test		9.796×10^{-2}	9.776×10^{-2}	2.434×10^{-1}	2.112×10^{-1}
Training	8	1.175×10^{-1}	1.175×10^{-1}	2.435×10^{-1}	2.191×10^{-1}
Test		9.796×10^{-2}	9.776×10^{-2}	2.430×10^{-1}	2.146×10^{-1}

Table E.14 - MSE Performance Summary for the Hard Problem Multiple Chip : Single Seed

References

References

1. W.S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-33, 1943.
2. F. Rosenblatt, "The Perceptron: A probabilistic Method for Information Storage and Organisation in the Brain", *Psychological Review*, vol. 65, pp. 386-408, 1958.
3. M.L. Minsky and S.A. Papert, *Perceptrons,* MIT Press, 1969.
4. P.J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences", PhD Thesis, Harvard University, Boston, USA, 1974.
5. D.B. Parker, "Learning Logic: Casting the Cortex of the Human Brain on Silicon", in *Technical Report TR-47*, Centre for Computational Research in Economics and management Science, MIT, Cambridge, USA, 1985.
6. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Backpropagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, ed. D.E. Rumelhart and J.L. McClelland, vol. 1, MIT Press, 1986.
7. D.S. Broomhead and D. Lowe, "Multivariate Functional Interpolation and Adaptive Networks", *Complex Systems 2*, pp. 321-55, 1988.
8. J. Moody and C.J. Darken, "Fast Learning in Networks of Locally Tuned Processing Units", *Neural Computation*, vol. V 1, pp. 281-94, 1989.
9. A.F. Murray and A.V.W Smith, "Asynchronous Arithmetic for VLSI Neural Systems", *Electronics Letters*, vol. 23, no. 12, pp. 642-3, June 1987.
10. D.J. Baxter, "Process-Tolerant VLSI Neural Networks for Applications in Optimisation", PhD Thesis, University of Edinburgh, 1993.

11. S. Churcher, "VLSI Neural Networks for Computer Vision", PhD Thesis, University of Edinburgh, 1993.

12. A Hamilton, "Pulse Stream Circuits and Techniques for the Implementation of Neural Networks", PhD Thesis, University of Edinburgh, 1993.

13. P.J. Edwards, "Analogue Imprecision in MLPs - Implications and Learning Improvements", PhD Thesis, University of Edinburgh, 1994.

14. G.B. Jackson, "Hardware Neural Systems for Applications: A Pulsed Analog Approach", PhD Thesis, University of Edinburgh, 1996.

15. C.M. Bishop, *Neural Networks for Pattern Recognition,* Clarendon Press, Oxford, 1995.

16. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis,* Wiley, 1973.

17. R. Beale and T. Jackson, *Neural Computing: An Introduction,* Adam Hilger, Bristol, 1990.

18. R.P. Lippmann, "Pattern Classification Using Neural Networks", *IEEE Comms. Magazine*, pp. 47-64, November 1989.

19. M. Hoehfeld and S.E. Fahlman, "Learning with Limited Numerical Precision Using the Cascade-Correlation Algorithm", *IEEE Trans. Neural networks*, vol. 3, no. 4, pp. 602-11, July 1992.

20. J. Platt, "A Resource-Allocating Network for Function Interpolation", *Neural Computation*, vol. 3, pp. 213-25, 1991.

21. N.J. Nilsson, *Learning Machines: Foundations of Trainable Pattern Classifying Systems,* McGraw-Hill, 1965.

22. D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* MIT Press, 1986.

23. K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, vol. 2, pp. 359-66, 1989.

24. B.J. Wythoff, "Backpropagation Neural Networks - A Tutorial", *Chemometrics and Intelligent Laboratory Systems*, vol. 18, no. 2, pp. 115-55, 1993.

25. S. Haykin, *Neural Networks: A Comprehensive Foundation,* Macmillan College Publishing, 1994.

26. D.R. Hush, B. Horne, and J.M. Salas, "Error Surfaces for Multilayer Perceptrons", *IEEE Trans. Systems, Man and Cybernetics*, vol. 22, no. 5, pp. 1152-61, September/October 1992.

27. P.D. Wasserman, *Advanced Methods in Neural Computing,* Van Nostrand Reinhold, 1993.

28. T. Tollenaere, "SuperSAB: Fast Adaptive Backpropagation with Good Scaling Properties", *Neural Networks*, vol. 3, pp. 561-73.

29. C.M. Bishop, "Neural Networks and Their Applications", *Rev. Sci. Instrum.*, vol. 65, no. 6, pp. 1803-32, June 1994.

30. J. Park and I.W. Sandberg, "Universal Approximation Using Radial Basis Function Networks", *Neural Computation*, vol. 3, pp. 246-57, 1991.

31. T.M. Cover, "Geometrical and Statistical Properties of Linear Inequalities with Applications in Pattern Recognition", *IEEE Transactions on Electronic Computers EC-14*, pp. 326-34, 1965.

32. S. Lee and R.M. Kil, "A Gaussian Potential Function Network with Hierarchically Self-Organising Learning", *Neural Networks*, vol. 4, no. 2, pp. 207-24, 1991.

33. M.J.D. Powell, "Radial Basis Functions for Multivariate Interpolation: A Review", *Proc. IMA Conf. on Algorithms for the Approximation of Data*, pp. 143-167, Shrivenham, 1985.

34. M.J.D. Powell, "Radial Basis Function Approximations to Polynomials", *Proc. Numerical Analysis 1987*, pp. 223-41, Dundee, 1987.

35. T. Poggio and F. Girosi, "Networks for Approximation and Learning", *Proc. IEEE*, vol. 78, no. 9, pp. 1481-97, September 1990.

36. B. Widrow and M.A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation", *Proc. IEEE*, vol. 78, pp. 1415-42, 1990.

37. L. Tarassenko and S. Roberts, "Supervised and Unsupervised Learning in Radial Basis Function Classifiers", *IEE Proc.-Vis. Image Signal Process*, vol. 141, no. 4, pp. 210-6, August 1994.

38. S. Chen, C.F.N Cowan, and P.M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Trans. on Neural Networks*, vol. 2, no. 2, pp. 302-9, March 1991.

39. S. Chen, P.M. Grant, and C.F.N. Cowan, "Orthogonal Least Squares Algorithm for Training Multi-Output Radial Basis Function Networks", *IEE Proc.-F*, vol. 139, no. 6, pp. 378-84, December 1992.

40. "Special Issue on Neural Network Hardware", *IEEE Trans. Neural Networks*, vol. 3, no. 3, May 1992.

41. "Neural Network Hardware", *Int. Journal Neural Systems*, vol. 4, no. 4, December 1993.

42. "Analog VLSI and Neural Networks", *IEEE Micro*, vol. 14, no. 1, June 1994.

43. P.E. Allen and D.R. Holberg, *CMOS Analog Circuit Design,* Saunders College Publishing, 1987.

44. E. Vittoz and J. Fellrath, "CMOS Analog Integrated Circuits Based on Weak Inversion Operation", *IEEE J. Solid-State Circuits*, vol. SC-12, no. 3, pp. 224-31, June 1977.

45. E.A. Vittoz, "Micropower Techniques", in *Design of MOS VLSI Circuits for Telecommunications*, ed. Y. Tsividis and P. Antognetti, pp. 104-44, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

46. A.G. Andreou, K.A. Boahen, P.O. Pouliquen, A. Pavasovic, R.E. Jenkins, and K. Strohnbehn, "Current-Mode Subthreshold MOS Circuits for Analog VLSI Neural Systems", *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 205-13, March 1991.

47. P.H.W. Leong and M.A. Jabri, "A Low-Power VLSI Arrhythmia Classifier", *IEEE Trans. Neural Networks*, vol. 6, no. 6, pp. 1435-45, November 1995.

48. S.S. Watkins and P.M. Chau, "A Radial Basis Function Neurocomputer Implemented with Analog VLSI Circuits", *Proc. of Int. Joint Conf. on Neural Networks (Baltimore, USA)*, vol. 2, pp. 607-12, June, 1992.

49. C. Mead, *Analog VLSI and Neural Systems,* Addison Wesley, 1989.

50. C. Mead, "Neuromorphic Electronic Systems", *Proc. IEEE*, vol. 78, no. 10, pp. 1629-36, October 1990.

51. L.E. Atlas and Y. Suzuki, *IEEE Circuits and Devices Magazine*, pp. 20-4, November 1989.

52. H.P.Graf, E. Sackinger, and L.D. Jackel, "Recent Developments of Electronic Neural Nets in North America", *Journal of VLSI Signal Processing*, vol. 5, pp. 19-31, Kluwer Academic Publishers, 1993.

53. R.W. Means, "High Speed Parallel Hardware Performance Issues for Neural Network Applications", *Proc. IEEE Conf. Neural Networks*, vol. 1, pp. 10-6, 27th June-2nd July 1994.

54. D. Hammerstrom and S. Rehfuss, "Neurocomputing Hardware: Present and Future", *Artificial Intelligence Review*, vol. 7, pp. 285-300, 1993.

55. J. Cloutier, E. Cosatto, S. Pigeon, F.R. Boyer, and P.Y. Simard, "VIP: An FPGA-Based Processor for Image Processing and Neural Networks", *Proc. MicroNeuro 96*, pp. 330-6, February 12th-14th 1996.

56. M. Duranton, "L-Neuro 2.3: A VLSI for Image Processing by Neural Networks", *Proc. MicroNeuro 96*, pp. 157-60, February 12th-14th 1996.

57. S. S. Watkins and P. M. Chau, "Reduced-Complexity Circuit for Neural Networks", *Electronics Letters*, vol. 31, no. 19, pp. 1644-6, 14th September 1995.

58. C. Park, K. Buckmann, J. Diamond, U. Santoni, S-C. The, M. Holler, M. Giler, C.L. Scofield, and L. Nunez, "A Radial Basis Function Neural Network with On-Chip Learning", *Proc. Int. Joint Conf. on Neural Networks*, pp. 3035-8, 1993.

59. E. Parzen, "On estimation of a probability density function and mode", *Ann. Math. Stat.*, vol. 33, pp. 1065-76, 1962.

60. D.F. Specht, "Probabilistic Neural Networks", *Neural Networks*, vol. 3, no. 1, pp. 109-18, 1990.

61. P. Maffezzoni and P. Gubian, "VLSI Design of Radial Basis Functions Hardware Generator for Neural Computations", *Proc. 4th Int. Conf. Microelectronics for Neural Networks and Fuzzy Systems (Turin, Italy)*, pp. 252-59, 1994.

62. A.H. Kramer, "Array-Based Analog Computation: Principles, Advantages and Limitations", *Proc. MicroNeuro 96*, pp. 68-79, 12th-14th February 1996.

63. S.M. Tam, B. Gupta, H.A. Castro, and M. Holler, "Learning on an Analog VLSI Neural Network Chip", *Proc. IEEE Int. Conf. Systems, Man and Cybernetics*, pp. 701-3, November 1990.

64. R. Tawel, "Learning in Analog Neural Network Hardware", *Computers and Electrical Engineering*, vol. 19, no. 6, pp. 453-67, 1993.

65. D.J. Mayes, "Multilayer Perceptrons: A Comparison of Analogue Hardware and Digital Software Implementations", *BEng.(Hons.) Project Report (HSP 928)*, University of Edinburgh, Scotland, UK, May 1993.

66. P.W. Hollis and J.J. Paulos, "Artificial Neural Networks Using MOS Analog Multipliers", *IEEE J. Solid-State Circuits*, vol. 25, no. 3, pp. 849-55, June 1990.

67. M. Jabri, S. Pickard, P. Leong, and Y.Xie, "Algorithmic and Implementation Issues in Analog Low Power Learning Neural Network Chips", *Journal of VLSI Signal Processing*, vol. 6, pp. 67-76, 1993.

68. P. Heim and M.A. Jabri, "Long-term CMOS Static Storage Cell Performing AD/DA Conversion for Analogue Neural Network Implementations", *Electronics Letters*, vol. 30, no. 25, pp. 2124-25, 8th December 1994.

69. A.F. Murray and A.V.W. Smith, "Asynchronous VLSI Neural Networks Using Pulse-Stream Arithmetic", *IEEE Trans. Solid-State Circuits*, vol. 23, no. 3, pp. 688-97, 1988.

70. E. Vittoz, H. Oguey, M.A. Maher, O. Nys, E. Dijkstra, and M. Chevroulet, "Analog Storage of Adjustable Synaptic Weights", in *VLSI Design of Neural Networks*, ed. U. Ramacher and U. Ruckert, pp. 47-63, Kluwer Academic Publishers, 1991.

71. S.P. Eberhardt, R. Tawel, T.X. Brown, T. Daud, and A.P. Thakoor, "Analog VLSI Neural Networks: Implementation Issues and Examples in Optimization and Supervised Learning", *IEEE Trans. Industrial Electronics*, vol. 39, no. 6, pp. 552-64, December 1992.

72. R. Castello, D.D. Caviglia, M. Franciotta, and F. Montecchi, "Selfrefreshing Analogue Memory Cell for Variable Synaptic Weights", *Electronics Letters*, vol. 27, no. 20, pp. 1871-3, 26th September 1991.

73. B. Hochet, "Multivalued MOS Memory for Variable-Synapse Neural Networks", *Electronics Letters*, vol. 25, no. 10, pp. 669-70, 11th May 1989.

74. G. Cauwenberghs and A. Yariv, "Fault-Tolerant Dynamic Multilevel Storage in Analog VLSI", *IEEE Trans. on Circuits and Systems - II: Analog and Digital Processing*, vol. 41, no. 12, pp. 827-9, December 1994.

75. J.J. Chang, "Nonvolatile Semiconductor Memory Devices", *Proc. of the IEEE*, vol. 64, no. 7, pp. 1039-59, July 1976.

76. R. Hecht-Neilson, Neurocomputing, pp. 292-6, Addison-Wesley, 1990.

77. C. Diorio, S. Mahajan, P. Hasler, B. Minch, and C. Mead, "A High-Resolution Non-Volatile Analog Memory Cell", *Proc. ISCAS 95*, 1995.

78. T. Shibata, H. Kosaka, H. Ishii, and T. Ohmi, "A Neuron-MOS Network Using Self-Learning-Compatible Synapse Circuits", *IEEE Journal of Solid-State Circuits*, vol. 30, no. 8, pp. 913-22, August 1995.

79. Y. Berg, R.L. Sigvartsen, T.S. Lande, and A. Abusland, "An Analog Feed-Forward Neural Network with On-Chip Learning", *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 1, pp. 65-75, January 1996.

80. B.W. Lee, B.J. Sheu, and H. Yang, "Analog Floating-Gate Synapses for General-Purpose VLSI Neural Computation", *IEEE Trans. Cicuits and Systems*, vol. 38, no. 6, pp. 654-8, June 1991.

81. D.A. Durfee and F.S. Shoucair, "Comparison of Floating Gate Neural Network Memory Cells in Standard VLSI CMOS Technology", *IEEE Trans. Neural Networks*, vol. 3, no. 3, pp. 347-53, May 1992.

82. O. Fujita, Y. Amemiya, and A. Iwata, "Characteristics of Floating Gate Device as Analogue Memory for Neural Networks", *Electronics Letters*, vol. 27, no. 11, pp. 924-6, 19th March 1991.

83. H.A. Castro, S.M. Tam, and M.A. Holler, "Implementation and Performance of an Analog Nonvolatile Neural Network", *Analog Integrated Circuits and Signal Processing*, vol. 4, pp. 97-113, 1993.

84. S. Collins, D.R. Brown, and G.F. Marshall, "An Analogue Vector Matching Architecture", *Analog integrated Circuits and Signal Processing*, vol. 8, pp. 247-57, Kluwer Academic Publishers, 1995.

85. A.J. Holmes, "The Use of Non-Volatile a-Si:H Memory Devices for Synaptic Weight Storage in Artificial Neural Networks", PhD Thesis, University of Edinburgh, 1996.

86. L.D. Jackel, R.E. Howard, H.P. Graf, B. Straughn, and J.S. Denker, "Artificial Neural Networks for Computing", *Journal Vac. Science and Technology*, vol. B61, p. 61, 1986.

87. A.J. Holmes, A.F. Murray, S. Churcher, and J. Hajto, "Pulsestream Synapses with Non-Volatile Analogue Amorphous-Silicon Memories", *Advances in Neural Information Processing Systems (NIPS) 7*, pp. 763-9, Morgan Kaufman, 1995.

88. A.A. Reeder, I.P. Thomas, C. Smith, J. Wittgreffe, J.D. Godfrey, J. Hajto, A.E. Owen, A.J. Snell, A.F. Murray, M.J. Rose, I.S. Osbourne, and P.G. LeComber, "Application of Analogue a-Si Memory Devices to Resistive Synapses for Neural Networks", *Proc. Materials Research Society Symposium*, vol. 258, pp. 1081-6, 1992.

89. Y. Tsividis and S. Satyanarayana, "Analogue Circuits for Electronic Neural Networks", *Electronics Letters*, vol. 23, no. 24, pp. 1313-4, 19th November 1987.

90. P.B. Denyer and J. Mavor, "MOST Transconductance Multipliers for Array Applications", *IEE Proc., Part I*, vol. 128, no. 3, pp. 81-6, June 1981.

91. F.J. Kub, K.K. Moon, I.A. Mack, and F.M. Long, "Programmable Analog Vector-Matrix Multipliers", *IEEE J. Solid-State Circuits*, vol. 25, no. 1, pp. 207-14, February 1990.

92. F. J. Pelayo, B. Pino, J. Ortega, and F. J. Fernandez, "CMOS Implementation of Synapse Matrices with Programmable Analog Weights", *Lecture Notes in Computer Science*, vol. 540, pp. 307-14, 1991.

93. S-C Qin and R.L. Geiger, "A ±5-V CMOS Analog Multiplier", *IEEE Journal Solid-State Circuits*, vol. SC-22, no. 6, pp. 1143-6, December 1987.

94. J. Babanezhad and G. Temes, "A Versatile Building Block: the CMOS Differential Differencing Amplifier", *IEEE Journal Solid-State Circuits*, vol. SC-20, pp. 1158-68, December 1985.

95. S.S. Watkins and P.M. Chau, "Different Approaches to Implementing A Radial Basis Function NeuroComputer", *Proc. RNNS/IEEE Symp. on Neuroinformatics and Neurocomputers (Roston-on-Don, Russia)*, pp. 1149-55, 1992.

96. A.J. Montalvo, J.J. Paulos, and R.S. Gyurcsik, "An Analog VLSI Neural Network Architecture with On-Chip Learning", *Proc. IEEE Int. Conf. Neural Networks*, vol. 3, pp. 1364-8, 1994.

97. T. H. Borgstrom, M. Ismail, and S. B. Bibyk, "Programmable Current-Mode Neural Network for Implementation in Analogue MOS VLSI", *IEE Proceedings*, vol. 137 (Part G), no. 2, pp. 175-84, April 1990.

98. O. Landolt, E. Vittoz, and P. Heim, "CMOS Selfbiased Euclidean Distance Computing Circuit with High Dynamic Range", *Electronics Letters*, vol. 28, no. 4, pp. 352-3, 1992.

99. G.T. Tuttle, S. Fallahi, and A.A. Abidi, "A Low-Power Analog CMOS Vector Quantizer", *Proc. Data Compression Conf.*, pp. 410-19, 30th March-2nd April 1993.

100. K. Bult and H. Wallinga, "A Class of Analog CMOS Circuits Based on the Square-Law Characteristic of an MOS Transistor in Saturation", *IEEE Journal Solid-State Circuits*, vol. SC-22, no. 3, pp. 357-65, June 1987.

101. G. Marshall and S. Collins, "A Compact Analogue Radial Basis Function Circuit", *Proc. ANN 95*, pp. 471-6, 26-28 June 1995.

102. G. Marshall and S. Collins, "An Analogue Radial Basis Function Circuit incorporating Floating-gate Devices", *Analog Integrated Circuits and Signal Processing*, vol. 11, pp. 21-34, Kluwer Academic Publishers, 1996.

103. J. Anderson, J.C. Platt, and D.B. Kirk, "An Analog VLSI Chip for Radial Basis Functions", in *Advances in Neural Information Processing Systems*, ed. C.L. Giles, S.J. Hanson, and J.D Cowan, pp. 765-71, Morgan Kaufmann, 1993.

104. R. Shorten and R. Murray-Smith, "Side Effects of Normalising Radial Basis Function Networks", *Int. J. Neural Systems*, vol. 7, no. 2, pp. 167-79, May 1996.

105. E. Seevinck and R.F. Wassenaar, "A Versatile CMOS Linear Transconductor/Square-Law Function Circuit", *IEEE J. Solid-State Circuits*, vol. SC-22, no. 3, pp. 366-77, June 1987.

106. M. Verleysen, P. Thissen, and J. Madrenas, "Analog VLSI Implementation of Kernel-Based Classifiers", *Proc. 4th Int. Conf. Microelectronics for Neural Networks and Fuzzy Systems (Turin)*, pp. 138-44, 1994.

107. M. Verleysen, P. Thissen, J-L. Voz, and J. Madrenas, "An Analog Processor Architecture for a Neural Network Classifier", *IEEE Micro*, vol. 14, no. 3, pp. 16-28, June 1994.

108. R. Dogaru, A.T. Murgan, S. Ortmann, and M. Glesner, "A Modified RBF Neural Network for Efficient Current-Mode VLSI Implementation", *Proc. MicroNeuro 96*, pp. 265-70, 12-14th February 1996.

109. G. Cauwenberghs and V. Pedroni, "A Charge-Based CMOS Parallel Analog Vector Quantiser", *Advances in Neural Information Processing Systems (NIPS) 7*, pp. 779-86, Morgan Kaufman, 1995.

110. J.G. Harris, "Implementing Radial Basis Functions Using Bump-Resistor Networks", *Proc. IEEE Int. Conf. on Neural Networks*, vol. 3, pp. 1894-8, 27th June - 2nd July, 1994.

111. T. Delbruck, "'Bump' Circuts for Computing Similarity and Dissimilarity of Analog Voltages", *Int. Joint. Conf. Neural Networks (IJCNN 91)*, vol. 1, pp. I475-9, 1991.

112. J.G. Harris. Private Communication

113. B.E. Boser, E. Sackinger, J. Bromley, Y. Le Cun, and L.D. Jackel, "An Analog Neural Network Processor with Programmable Topology", *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 2017-25, December 1991.

114. J-C. Lee, B. J. Sheu, and R. Chellappa, "A VLSI Neuroprocessor for Image Restoration Using Analog Computing-Based Systolic Architecture", *Journal of VLSI Signal Processing*, vol. 5, pp. 185-99, 1993.

115. A. Hamilton, A.F. Murray, D.J. Baxter, H.M. Reekie, and L. Tarassenko, "Integrated Pulse Stream Neural Networks: Results, Issues and Pointers", *IEEE Trans. on Neural Networks*, vol. 3, no. 3, pp. 385-93, May 1992.

116. M. J. Brownlow, L. Tarassenko, and A. F. Murray, "Analogue Computation Using VLSI Neural Network Devices", *Electronics Letters*, vol. 26, no. 16, pp. 1297-99, 2nd August 1990.

117. L.W. Massengill, "A Dynamic CMOS Multiplier for Analog Neural Network Cells", *Proc. 1990 Custom Integrated Circuits Conf.*, pp. 26.4.1-4, 1990.

118. O. Schwartsglass, J. Shappir, and A. J. Agranat, "VLSI implementation of Pulse Activated Synapses", *Electronic Letters*, vol. 29, no. 16, pp. 1433-5, 5th August 1993.

119. G. Moon, M.E. Zaghloul, and R.W. Newcomb, "VLSI Implementation of Synaptic Weighting and Summing in Pulse Coded Neural-Type Cells", *IEEE Trans. Neural Networks*, vol. 3, no. 3, pp. 394-403, May 1992.

120. M.E. Zaghloul, J.L. Meador, and R.W. Newcomb, *Silicon Implementations of Pulse Coded Neural Networks,* Kluwer Academic Publishers, 1994.

121. R.P. Lippmann, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, vol. 4, pp. 4-22, 1987.

122. J.N. Tombs, "Multi-layer Neural Networks and their Implementation in Analogue VLSI", PhD Thesis, University of Oxford, 1992.

123. L.M. Reyneri, M. Chiaberge, and D. Del Corso, "Using Coherent Pulse Width and Edge Modulations in Artificial Neural Systems", *Int. Journal of Neural Systems*, vol. 4, no. 4, pp. 407-18, December 1993.

124. L.M. Reyneri, "Weighted Radial Basis Functions for Improved Pattern Recognition and Signal Processing", *Neural Processing Letters*, vol. 2, no. 3, pp. 2-6, 1995.

125. M. Chiaberge, E. Miranda Sologuren, and L.M. Reyneri, "A Low-Power Neuro-Fuzzy Pulse Stream System", *Proc. MicroNeuro 96*, pp. 191-9, 12-14th Februray 1996.

126. L.M. Reyneri, M. Chiaberge, and L. Zocca, "CINTIA: A Neuro-Fuzzy Real Time Controller for Low Power Embedded Systems", *Proc. 4th Int. Conf. Microelectronics for Neural Networks and Fuzzy Systems (Turin, Italy)*, pp. 392-403, 1994.

127. C. Toumazou, F.J. Lidgey, and D.G. Haigh (eds), *Analogue IC Design: The Current-Mode Approach,* Peter Peregrinus Ltd., London, 1990.

128. Z. Wang, "Current-Mode CMOS Integrated Circuits for Analog Computation and Signal Processing: A Tutorial", *Analog Int. Circuits and Sig. Processing 1*, pp. 287-95, Kluwer Academic Publishers, 1991.

129. M. Ismail and T. Fiez, *Analog VLSI Signal and Information Processing,* McGraw-Hill, 1994.

130. S.J. Daubert, D. Vallancourt, and Y.P. Tsividis, "Current Copier Cells", *Electronics Letters*, vol. 24, no. 25, pp. 1560-2, 8th December 1988.

131. E.A. Vittoz and G. Wegmann, "Dynamic Current Mirrors", in *Analogue IC Design: The Current-Mode Approach*, ed. C. Toumazou, F.J. Lidgey, and D.G. Haigh, pp. 297-326, Peter Peregrinus Ltd., London, 1990.

132. T. Serrano and B. Linares-Barranco, "The Active-Input Regulated-Cascode Current Mirror", *IEEE Trans. on Circuits and Systems - I: Fundamental Theory and Applications*, vol. 41, no. 6, pp. 464-7, June 1994.

133. H.C. Yang and D.J. Allstot, "An Active-Feedback Cascode Current Source", *IEEE Trans. Circuits and Systems*, vol. 37, no. 5, pp. 644-6, May 1990.

134. E. Sackinger and W. Guggenbuhl, "A High-Swing, High-Impedance MOS Cascode Circuit", *IEEE Journal Solid-State Circuits*, vol. 25, no. 1, pp. 289-97, February 1990.

135. C. Toumazou, J.B. Hughes, and D.M. Pattullo, "Regulated Cascode Switched-Current Memory Cell", *Electronics Letters*, vol. 26, no. 5, pp. 303-5, March 1990.

136. G. Wegmann, E.A. Vittoz, and F. Rahali, "Charge Injection in Analog MOS Switches", *IEEE Journal of Solid-State circuits*, vol. SC-22, no. 6, pp. 1091-7, December 1987.

137. E. A. Vittoz, "Dynamic Analog Techniques", in *Design of MOS VLSI Circuits for Telecommunications*, ed. Y. Tsividis and P. Antogretti, pp. 145-70, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

138. J-H. Shieh, M. Patil, and B.J. Sheu, "Measurement and Analysis of Charge Injection in MOS Analog Switches", *IEEE J. Solid-State Circuits*, vol. SC-22, no. 2, pp. 277-81, April 1987.

139. D. Macq and P. Jespers, "Charge Injection in Current Copier Cells", *Electronics Letters*, vol. 29, no. 9, pp. 780-1, 29th April 1993.

140. E. A. Vittoz, "Analog VLSI Signal Processing: Why, Where and How ?", *Analog Integrated Circuits and Signal Processing*, pp. 27-44, July 1994.

141. H-U. Post and K. Waldschmidt, "A High-Speed NMOS A/D Convertor with a Current Source Array", *IEEE Journal Solid-State Circuits*, vol. SC-15, no. 3, pp. 295-300, June 1980.

142. C.A.A. Bastiaansen, D.W.J. Groeneveld, H.J. Schouwenaars, and H.A.H. Termeer, "A 10-bit 40MHz 0.8μm CMOS Current-Output D/A Converter", *IEEE Journal Solid-State Circuits*, vol. 26, no. 7, pp. 917-21, July 1991.

143. S. Churcher, A.F. Murray, and H.M. Reekie, "Programmable Analogue VLSI for Radial Basis Function Neural Networks", *Electronics Letters*, vol. 29, no. 18, pp. 1603-5, September 1993.

144. A. Nedungadi and T.R. Viswanathan, "Design of Linear CMOS Transconductance Elements", *IEEE Trans. Circuits and Systems*, vol. CAS-31, pp. 891-4, October 1984.

145. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design,* Addison Wesley, 1988.

146. E.S. Yang, *Microelectronic Devices,* McGraw-Hill, 1988.

147. E. A. Vittoz, "The Design of High-Performance Analog Circuits on Digital CMOS Chips", *IEEE Journal of Solid-State Circuits*, vol. SC-20, no. 3, pp. 657-65, June 1985.

148. G.A. Cairns, "Learning with Analogue VLSI Multilayer Perceptrons", PhD Thesis, University of Oxford, 1995.

149. S.Furui, "Cepstral Analysis Technique for Automatic Speaker Verification", *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-29, no. 2, pp. 254-72, 1981.

150. L. Tarassenko, J. Tombs, and G. Cairns, "On-chip Learning with Analogue VLSI Neural Networks", *Int. Journal Neural Systems*, vol. 4, no. 4, pp. 419-26, December 1993.

151. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C,* 2nd Edition, Cambridge University Press, 1992.

152. E. Kreyszig, *Advanced Engineering Mathematics,* 6th Edition, Wiley, 1988.

153. M. Jabri and B. Flower, "Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks", *IEEE Trans. Neural Networks*, vol. 3, no. 1, pp. 154-7, January 1992.

154. A.J. Bostel, "Training RBF Networks with Perturbation Methods", *Proc. ISCAS 95 (Seattle, USA)*, vol. 3, pp. 1699-702, 1995.

155. P.W. Hollis, J.S. Harper, and J.J. Paulos, "The Effects of Precision Constraints in a Backpropagation Learning Network", *Neural Computation*, vol. 2, pp. 363-73, 1990.

156. S. Sakaue, T. Kohda, H. Yamamoto, S. Maruno, and Y. Shimeki, "Reduction of Required Precision Bits for Back-Propagation Applied to Pattern Recognition", *IEEE Trans. Neural Networks*, vol. 4, no. 2, pp. 270-5, March 1993.

157. L.M. Reyneri and E. Filippi, "An Analysis on the Performance of Silicon Implementations of Backpropagation Algorithms for Artificial Neural Networks", *IEEE Trans. Computers*, vol. 40, no. 12, pp. 1380-8, December 1991.

158. R.C. Frye, E.A. Rietman, and C.C. Wong, "Back-Propagation Learning and Nonidealities in Analog Neural Network Hardware", *IEEE Trans. Neural Networks*, vol. 2, no. 1, pp. 110-7, January 1991.

159. K. Papathanasiou and A. Hamilton, "Pulse Based Signal Processing : VLSI Implementation of a Palmo Filter", *International Symposium on Circuits and Systems, Atlanta*, vol. 1, pp. 270-273, May 1996.

List of Publications

References

1. D.J. Mayes, J.L. Louvet, and A. Hamilton, "A VLSI Current Mode Synapse Chip", *Proc. of Int. Workshop on Artificial Neural Networks (Malaga, Spain)*, pp. 815-21, 1995.
2. D.J. Mayes, J.L. Louvet, and A. Hamilton, "DYMPLES - An Analog Current Mode Pulsed Synapse", *Proc. of 4th Int. Conf. on Artificial Neural Networks (Cambridge, UK)*, pp. 477-82, 1995.
3. D.J. Mayes, A.F. Murray, and H.M. Reekie, "Pulsed VLSI for RBF Neural Networks", *Proc. MicroNeuro 96 (Lausanne, Switzerland)*, pp. 177-84, 12th-14th February, 1996.
4. D.J. Mayes, A. Hamilton, A.F. Murray, and H.M. Reekie, "A Pulsed VLSI Radial Basis Function Chip", *Proc. ISCAS 96 (Atlanta, USA)*, pp. 297-300, 12th-15th May, 1996.