Boolean Weightless Neural Network Architectures

BY

James Robert Armstrong

A thesis submitted in partial fulfilment for the requirements of the degree of

DOCTOR OF PHILOSOPHY

April 2011

The research work presented in this thesis was carried out in the
Applied Digital Signal and Image Processing (ADSIP) Centre,
School of Computing, Engineering and Physical Sciences,
University of Central Lancashire
in collaboration with
BAE SYSTEMS

Student Declaration

Concurrent registration for two or more academic awards

Either *****I declare that while registered as a candidate for the research degree, I have not been a registered candidate or enrolled student for another award of the University or other academic or professional institution

or ~~*I declare that while registered for the research degree, I was with the University's specific permission, a *registered candidate/*enrolled student for the following award:~~

____

Material submitted for another award

Either *****I declare that no material contained in the thesis has been used in any other submission for an academic award and is solely my own work.

or ~~*I declare that the following material contained in the thesis formed part of a submission for the award of~~

____
(state award and awarding body and list the material below):

Collaboration

Where a candidate's research programme is part of a collaborative project, the thesis must indicate in addition clearly the candidate's individual contribution and the extent of the collaboration. Please state below

Signature of Candidate ______________________________

Type of Award ______________________________

School ______________________________

Dedicated to my wife and family,
for their support and trust

Acknowledgements

I owe my deepest gratitude to my Director of Studies, Dr Simon Platt, and my Second Supervisors, Dr Doug King and Ian MacDiarmid for their support, valuable advice and encouragement. Special thanks to Doug King for the technical discussions which often sparked new ideas.

I thank BAE SYSTEMS for supporting this collaborative venture with the University of Central Lancashire. Special thanks are due to:-

- Mark Ford and Andrew Cassie for supporting this research.
- Nick Jaques, for laying out the printed circuit boards.
- William Barlow for his assistance producing some of the diagrams incorporated in this work and proof reading.
- Martin MacLean for managing the patent process and liaising with the University of Central Lancashire with respect to the Intellectual Property Rights.
- BAE SYSTEMS Technical Information Centre; especially Claire, Maureen and Julie.

I thank William Newell, patent agent, for his contribution to the patent process.

I also thank my academic advisor, Professor Jim Austin, University of York, for discussions, visits to his laboratory and constructive criticism.

Abstract

A collection of hardware weightless Boolean elements has been developed. These form fundamental building blocks which have particular pertinence to the field of weightless neural networks. They have also been shown to have merit in their own right for the design of robust architectures.

A major element of this is a collection of weightless Boolean sum and threshold techniques. These are fundamental building blocks which can be used in weightless architectures particularly within the field of weightless neural networks. Included in these is the implementation of L-max also known as N point thresholding. These elements have been applied to design a Boolean weightless hardware version of Austin's ADAM neural network. ADAM is further enhanced by the addition of a new learning paradigm, that of non-Hebbian Learning. This new method concentrates on the association of 'dis-similarity', believing this is as important as areas of similarity.

Image processing using hardware weightless neural networks is investigated through simulation of digital filters using a Type 1 Neuroram neuro-filter. Simulations have been performed using MATLAB to compare the results to a conventional median filter. Type 1 Neuroram has been tested on an extended collection of noise types. The importance of the threshold has been examined and the effect of cascading both types of filters was examined.

This research has led to the development of several novel weightless hardware elements that can be applied to image processing. These patented elements include a weightless thermocoder and two weightless median filters. These novel robust high speed weightless filters have been compared with conventional median filters.

The robustness of these architectures has been investigated when subjected to accelerated ground based generated neutron radiation simulating the atmospheric radiation spectrum experienced at commercial avionic altitudes. A trial investigating the resilience of weightless hardware Boolean elements in comparison to standard weighted arithmetic logic is detailed, examining the effects on the operation of the function when implemented on hardware experiencing high energy neutron bombardment induced single event effects.

Further weightless Boolean elements are detailed which contribute to the development of a weightless implementation of the traditionally weighted self ordered map.

Contents

List of Figures

List of Tables

Abbreviations

ADAM	Advanced Distributed Associative Memory
ADSIP	Applied Digital Signal and Image Processing
ANITA	Atmospheric like Neutrons from Thick Target
AURA	Advanced Uncertain Reasoning Architecture
BNC	Bayonet Neill-Concelman Connector
CBV	Compact Binary Vector
CCD	Charge Coupled Device
CEPS	School of Computing, Engineering and Physical Sciences
CMM	Correlation Memory Matrix
CNNAP	Cellular Neural Network Associative Processor
DAME	Distributed Aircraft Maintenance Environment
DBIS	Department of Business, Skills and Innovation
DC	Direct Current
DTI	Department of Trade and Industry
DVD	Digital Versatile Disc
EDIF	Electronic Data Interchange Format
EEPROM	Electrically Erasable Programmable Read Only Memory
EX-OR	Exclusive OR
EX-NOR	Exclusive NOR
FAE	Fused Adaptive Element
FDC	Xilinx Symbol – a single ‘D’ type flip-flop
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GB	Gigabyte
IEE	Institution of Electrical Engineers
IET	Institution of Engineering and Technology
JTAG	Joint Test Action Group
MATLAB	Matrix Laboratory software tool
MHz	Mega Hertz
MoD	Ministry of Defense

MSE	Mean Square Error
NIF	Neutron Irradiation Facility
PC	Personal Computer
PCB	Printed Circuit Board
PIC	Programmable Interface Controller
PRESENCE	Parallel Structured Neural Computing Engine
PSNR	Peak SNR
PSU	Power Supply Unit
RAM	Random Access Memory
RTC	Real Time Clock
SAT	Sum and Threshold
SDRAM	Synchronous Dynamic Random Access Memory
SEE	Single Event Effect
SEU	Signal Event Upset
SNR	Signal-to-Noise Ratio
SOM	Self Ordered Map
SPAESRANE	Solutions for the Preservation of Aerospace Electronic Systems Reliability in the Atmospheric Neutron Environment
TRIUMF	Tri University Meson Facility
TSL	Theodor Svedberg Laboratory
WISARD	Wilkie, Stonham and Aleksander's Recognition Device
WNNW	Weightless Neural Networks Workshop
WSOM	Weightless Self Ordered Map

Chapter 1

1 Introduction

1.1 Weighted Boolean Logic

Boolean logic was first described by Charles Boole in 1854, and since has been used in everyday electronics and forms the basis of modern day computing [Boole 1854]. In 1936 Turing produced his paper on computable numbers; where he proposed the fundamental operation of modern computing [Turing 1936]. The work of John Von Neumann introduced the architecture of the modern day stored programme computer [von Neumann 1958]. These architectures are based on weighted clocked logic operating with arithmetic units and counters, processing data in sequence. Weighted logic has advantages in that it is compact, allowing large numbers to be easily implemented in binary systems. Unfortunately it suffers from the fact that minor errors can cause massive data disruption. Therefore error detection and correction codes such as Hamming correction have been developed to increase the robustness of the data and resultant systems [Hamming 1950]. This approach does not remove the risk of corruption but reduces the occurrence of corruption to an acceptable level.

1.2 Weightless Logic

Weightless logic forms the heart of the operation of weightless neural networks. Weightless neural networks are sometimes referred to as Boolean neural networks due to their properties and mode of operation [Picton 1994]. Weightless Boolean neural networks were first described in the McCulloch and Pitts paper written in 1943 [McCulloch et al 1943]. McCulloch and Pitts proposed a simplified model for an artificial neuron based on the limited understanding of a neuron at the time. The original neuron model is a weightless neuron; however in the early implementations using computers engineers were unable to get the model to work [Rochester et al 1956]. The model was then altered to add the addition of weights which could be varied at each of the synapses feeding the neuron. This concept has now formed the main stream of neural networks and consists of multiple neural network systems [Rumelhart et al 1986, Hopfield 1982]. It was not until Bledsoe and Brownings' work on pattern recognition that weightless neural networks were rediscovered [Bledsoe et al 1959]. Weightless neural networks lend themselves very well to implementation on modern electronics due to only having two states, ‘on’ or ‘off’, or in neural terms ‘firing’ or not ‘firing’. This is analogous to Boolean logic which also has two states of operation, ‘zero’ or ‘one’. Weightless neural networks although only having two states of operation, are often implemented using weighted arithmetic counting units and weighted logic to perform the required sum and thresholding functions [Austin 1986].

1.3 Avionic Systems

Avionic systems are subject to corruption caused by atmospheric radiation. Atmospheric radiation is not a new phenomenon but has been known about since its discovery in 1913 by Hess [Hess 1913]. Atmospheric radiation has not changed in intensity since its discovery, however our understanding has become much greater and we now understand some of the effects that influence the intensity [Dyer et al 2001]. The effects of atmospheric radiation are causing more detrimental effects to avionic systems, due to the changes in electronic components used within them [Dyer et al 2000]. The avionics industry is moving away from bespoke parts to commercial off-the-shelf parts due to their lower cost and lack of military suppliers. Therefore the industry has to investigate ways of using commercial off-the-shelf parts, without compromising the safety or reliability of the overall avionics systems.

Military avionics systems are also developing new technology to implement evermore advanced systems, in order to give air superiority, these include things like voice recognition and visual targeting. These systems require more processing power and electronics than traditional hand controlled systems.

1.4 Atmospheric Radiation

Cosmic radiation originates from deep in space although its exact origins are not fully understood. It is believed the particles are generated by supernova activity [Dyer et al 2000]. Atmospheric radiation is generated when primary cosmic rays consisting of highly energetic particles which penetrate the earth’s magnetic field, which acts as a shield against the majority of these particles [Dyer et al 2001]. These primary charged particles mainly consist of protons (hydrogen nuclei) along with alpha particles, helium nuclei and other nuclear fragments. They interact with air molecules to generate a cascade of secondaries including neutrons as shown in Figure 1.1 from Chugg [Chugg 2003c]. It is the neutrons which have the most detrimental effect on electronic systems.

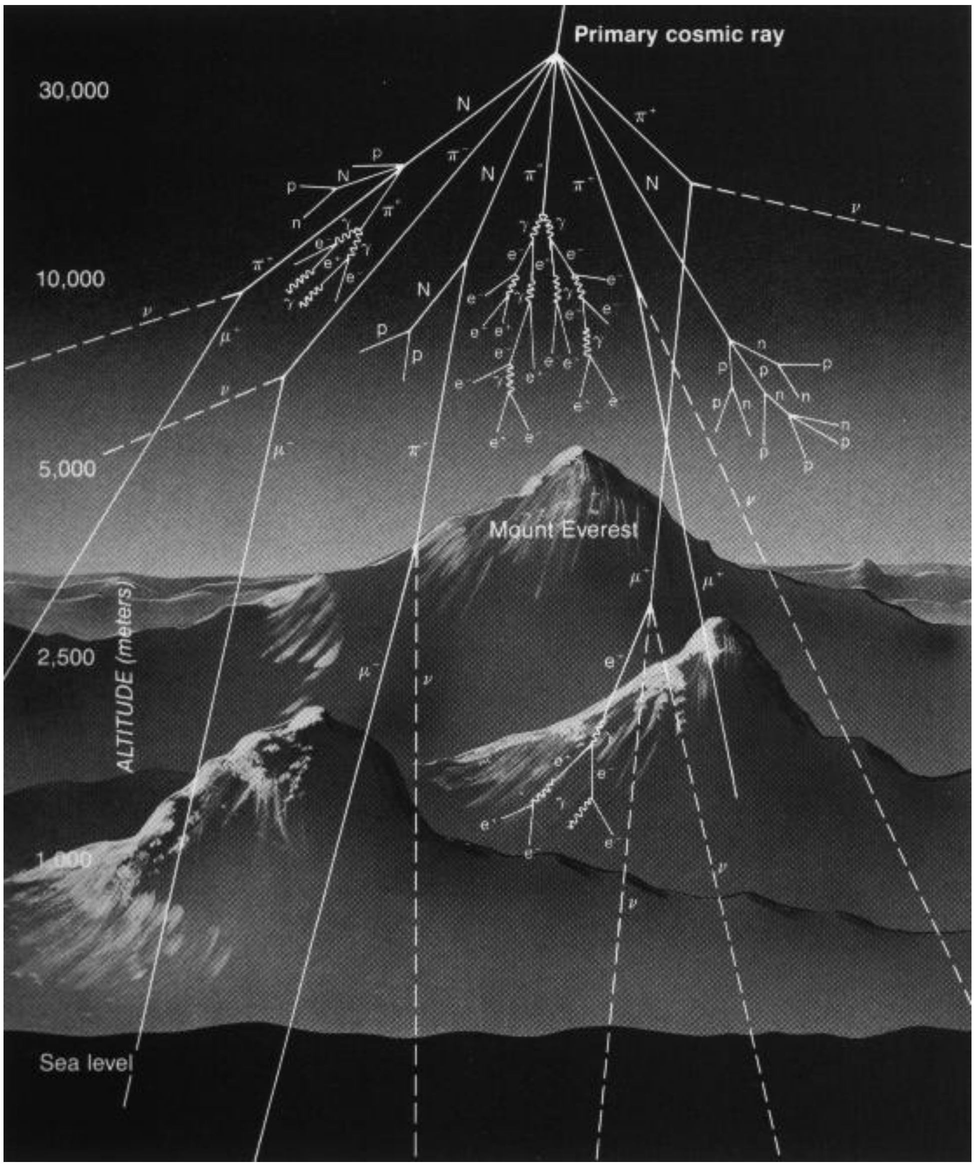


Figure 1.1 A Diagram of the Primary Cosmic Rays Interacting with Air Molecules

These reach a maximum flux at 60,000 feet, and reduce at lower altitudes. Figure 1.2 shows that the flux is 300 times less at sea level than at 60,000 feet. The flux of these secondary particles also alters with latitude with a maximum at the earth's poles, due to the increased effect of the earth's magnetic field. There are also anomalies such as the 'South Atlantic Anomaly' described by Dyer et al which makes the prediction of the levels over the earth's surface hard to determine [Dyer et al 2001]. With the lack of an exact origin and the fact that some of the primary particles were generated thousands of years ago, cosmic radiation level prediction is in its infancy. It is known that certain factors such as the solar cycle have an effect on the levels but knowledge of other factors is limited. The greatest effects on levels have been caused by events that were not and could not be predicted based on our present knowledge. These include a factor of 216 rise in cosmic fluxes back in 1956 and a factor of 31 rise in 1989 at approximately 32,000 feet altitude [Dyer et al 2001].

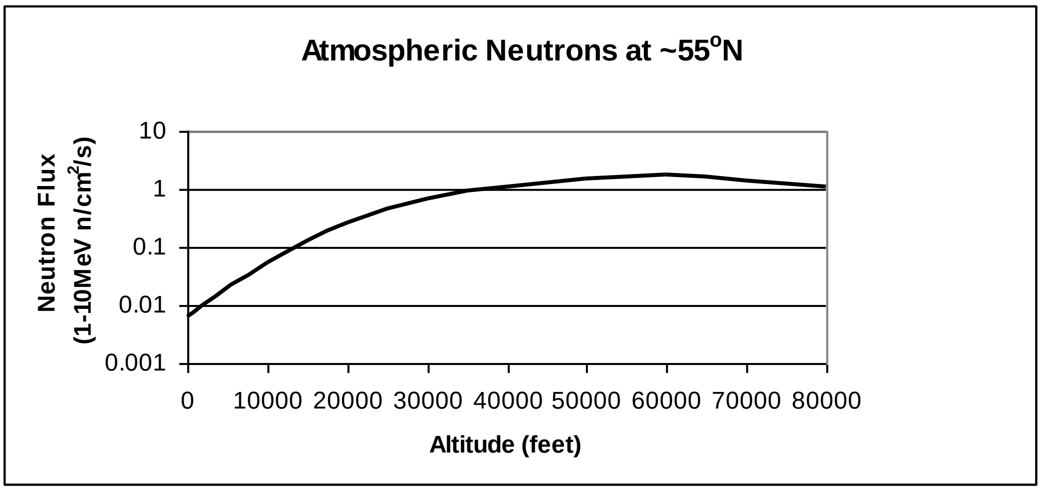


Figure 1.2 A Graph of Neutron Flux with Relationship to Altitude

With the current lack of understanding and inability to predict cosmic fluxes accurately, the tendency has been to rely on real time monitoring such as fitted to Concorde [Dyer et al 2001]. Commercial aircraft operate within this environment as they typically cruise at between 30,000 to 45,000 feet. From Figure 1.2 we see this gives an average neutron flux of approximately 2 n/cm^2/s. Until recently the main concern with operating in this environment has been for humans who are exposed to this radiation for long periods of time. This has resulted in legislation which now requires them to be classified as radiation workers and hence their radiation dose needs to be monitored [Schneur 1999]. This has resulted in a requirement for an accurate cosmic radiation monitor, particularly for neutrons which are the main component of atmospheric radiation. Until recently these detectors have been large and bulky and not suitable as discrete personal monitors.

1.5 Cosmic Radiation Effects on Avionic Components

More recently there has been concern for the electronic systems operating in this environment: the effects of single event upset, burnout and damage have been understood for a long time in space applications [Dyer et al 2000].

The problem of corruption or damage to semiconductor devices occurs when one of these highly energetic neutrons interacts with atoms within the device. The illustration in Figure 1.3 shows a typical interaction between an atmospheric neutron and a silicon atom [Chugg 2006].

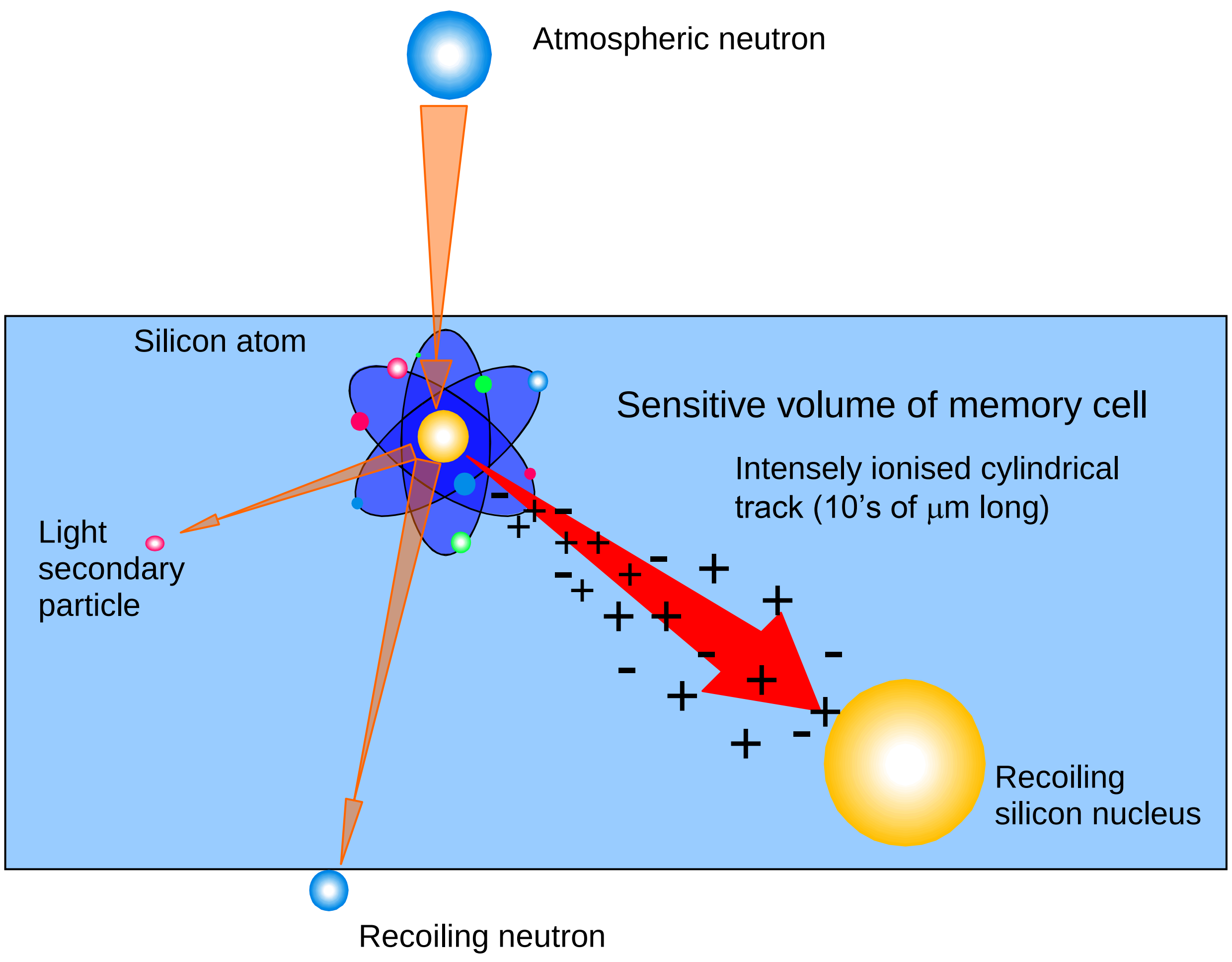

Atmospheric neutron
Silicon atom
Sensitive volume of memory cell
Intensely ionised cylindrical track (10's of μm long)
Light secondary particle
Recoiling silicon nucleus
Recoiling neutron

Figure 1.3 An Illustration of an Atmospheric Neutron Interacting with Silicon

It can be seen that the neutron interacts with the silicon atom leaving an intensely ionised cylindrical track. It is this highly charged track that leaves charge within the cells of the device that can cause the change of state for that cell. Due to the length of the tracks produced it is quite common to upset several cells with one interaction, meaning that single bit error detection and correction logic is insufficient.

It is important to note that these effects are not new but the severity of these effects is increasing due to changing technology. The main causes are the reduction of supply voltage levels, meaning the amount of charge required to flip data bits is less. The geometry of parts is also reducing; meaning the number of cells affected by the track is increasing. To compound matters the amount of memory used in systems is also increasing.

These atmospheric radiation effects have been widely observed at avionics altitudes due to the significantly lower shielding effects of the earth's magnetic field and the atmosphere [Dyer et al 2000]. Technology trends in the semiconductor industry have led to devices becoming more susceptible. These trends are improved speed, lower power consumption and reduced production costs along with increased capabilities within devices. These goals have been achieved by reducing device operating voltage, die geometry size and increasing density of components. Lowering the operating voltage of the device reduces the transition energy required to move from one logic state to another. Reducing the die geometry reduces the power needed by the device and subsequent heat produced. Smaller geometries also allow a greater gate count as the area taken by each logic gate is less and the frequency of operation of the device can be increased. Unfortunately all these characteristics are leading to devices that are more susceptible to atmospheric radiation. In addition, the avionics industry is moving away from high specification military parts toward commercial parts in an effort to reduce cost. This is causing an increase in susceptibility of aircraft avionics to atmospheric radiation effects.

1.6 Research in this Field

In order to counteract this worrying trend, more detail of these effects and a greater knowledge of the environment, coupled with the ability to better predict flux levels, is required before the problem impacts on aircraft safety. Therefore the need to design robust, resilient, systems for operating in this environment is growing, in conjunction with better capabilities for monitoring the environment. This will lead to better understanding of the interaction process caused by the atmospheric radiation within electronic devices.

The avionics industry is now aware of the need to study the potential issues caused by atmospheric radiation on avionics systems and has funded research programmes in conjunction with the Department for Business Enterprise and Regulatory Reform (formally the Department of Trade and Industry) and the Ministry of Defence [Chugg 2003c].

A potential solution to this ever increasing problem is that of weightless logic and weightless neural networks due to their ability to generalize which allows an element of self correction.

1.7 Weightless Neural Networks

Weightless neural networks are a technology which come under the banner of 'Artificial Intelligence' due to their ability to learn and adapt to presented information. A neural network is a group of interconnected simple processing units which try to approximate the operation of part of the human brain. They usually consist of a large number of processors which operate in parallel, each having its own knowledge and local memory. The way the processing units connect is usually fluid and it is these connections which increase the networks ability to adapt to different stimuli. Although not programmed, they may follow rules which allow them to successfully operate and are not constrained to perform in an algorithmic manner. The description of their operation is their structure and interconnects at any particular time given. This gives them the power to solve complex non-linear problems which are mathematically challenging.

Boolean hardware weightless neural networks are the principal focus of this research. Weightless neural networks differ from conventional neural networks such as the error back-propagation network and the Hopfield networks in several ways [Rumelhart et al 1986, Hopfield 1982]. Weightless neural networks tend to be logic based using Boolean operators instead of the mathematical floating point equations used in weighted techniques. Unlike conventional weighted networks, weightless neural networks in general have the unique property of being able to learn in one cycle, unlike weighted systems that require multiple cycles in order to iteratively alter the weights in the system. Hardware weightless neural networks also exhibit several other properties that offer improvements over the conventional techniques particularly in the areas of robustness and speed [Bedford et al 1996]. Improvements in robustness can be found in weightless neural networks for several reasons. The reduction or removal of the system's dependence on a clock. The data propagates through the system just incurring a small latency caused by the propagation delay of the logic. Hardware weightless neural networks hence have a greater propensity to operate in a harsh environment such as in a high electromagnetic environment due to lack of their reliance on a clock pulse [King 2000]. Hardware weightless neural networks are parallel in their nature and hence their speed of operation in comparison to an algorithmic approach based on a conventional processor is vastly superior in complex tasks such as scene analysis [Austin 1986].

1.8 Areas for Development

1.8.1 ADAM

The properties of weightless Boolean networks are ideally suited to the development of robust architectures designed to overcome the effects of atmospheric radiation experienced by modern avionics. In order to gain the full benefit of this technology implementation of the networks it is vital to understand the robustness of the overall system. Several weightless neural networks, although weightless in nature, are implemented using standard weighted architectures. These include Austin's ADAM network which is implemented with the use of arithmetic counter units to sum the rows and columns of the matrices [Austin et al 1987]. King has demonstrated with his Neuroram that it is possible to develop hardware which is also capable of sum and thresholding necessary for the implementation of these weightless neural networks that does not rely on arithmetic counter units [King 2000].

1.8.2 Neuroram

A promising technology which was researched under SPAESRANE is the use of Charged Coupled Devices (CCD's) as a radiation detector as patented and presented by Chugg [Chugg et al 2002]. Studies of this technology at ground based neutron facilities have supported the hypothesis that these CCD element devices hold great promise as neutron detectors [Torok et al 2006]. The neutron radiation causes degradation of the CCD pixel elements and causes the generation of unwanted noise on the data. Long-term exposure of the sensors causes more severe damage to the CCD pixel elements. Pixels can become stuck in a particular state usually a maximum or minimum. The effect of this is to generate white and black spots on an image similar to Salt and Pepper noise. It is known that the properties of the median filter make it suitable for removing salt and pepper noise. Pixels can also be damaged by the radiation exposure causing them to randomly oscillate in state even when not exposed to radiation; this effect is known as random telegraph noise. [Chugg et al 2003b]. King's Neuroram filter is an ideal technology for the removal of the salt-and-pepper noise. The Salt and Pepper noise is similar in type to the effects which are caused by radiation damage to CCD sensors. His research in this area has been limited to the analysis of his filter with salt-and-pepper noise. It is proposed that the filter may be able to deal with other types of noise, but these have yet to be investigated, but may hold significance in the removal of further noise damage caused to the CCD by radiation effects. The author was part of the SPAESRANE project organising and co-coordinating some of the earlier trials particularly at the Theodor Svedberg Laboratory located at the University of Uppsala in Sweden. This included working with Chugg and King on the set up of these early CCD experiments, the results of these trials are available in the papers in Appendix A [Chugg et al 2003a, Chugg et al 2003b].

1.8.3 Conversion of Weighted Neural Networks to Weightless Architectures

Neural networks tend to be robust by their nature; including weighted neural networks as they all possess the property to generalise. This gives them an inherent ability to deal with corrupt or incomplete data. If this technology is to be used the architecture in which it is implemented needs to be carefully considered to reduce the effect of corruption. An area of interest is the development of weightless neural networks based on the principles of existing weighted neural networks using weightless Boolean hardware technology.

1.8.4 Robustness of Weightless Boolean Architectures

Weightless technology by its very nature looks to offer a more graceful degradation in its performance when subjected to corruption due to its verbose nature and limited effect caused by any single corruption. The aim of this research is to investigate this hypothesis and compare it with compatible weighted implementations. This will be performed using ground-based neutron facilities as the use of flight trials is impractical due to cost and timescales.

1.8.5 Implementation of Standard Architectures in Weightless Boolean Logic

In order for weightless Boolean hardware technology to be a viable alternative to weighted binary implementations it is necessary to show that standard implementations of conventional weighted binary functions can be implemented in a weightless manner.

1.9 Research Objectives

The research objectives for this thesis are all based around the development of weightless Boolean hardware architectures and their associated elements which derive from the field of weightless hardware neural networks. A detailed chronological history of hardware weightless neural networks can be found in the next chapter which presents the background for the thesis.

There are several aims of this research which are described below:

The main objective is to develop a group of weightless Boolean elements that could be used to further improve existing weightless neural networks.

- A primary aim of this thesis is to investigate whether the transition from weighted binary to weightless binary and Boolean architectures offers an improved resilience with reduced effects of corruption particularly on systems subjected to atmospheric radiation. King had previously postulated that earlier weightless elements exhibited increased robustness [King 2000].

- The development of a group of weightless Boolean elements which could be used to further improve existing weightless neural networks. A key driver is to remove clocks and counters, often associated with weighted implementations of these networks to improve their robustness for avionics applications.

- Key to this research was to demonstrate the flexibility of these proposed weightless elements and show their applicability to existing weighted systems. The robust nature of these elements was also important.

- Show how a traditional weighted neural network which relied on a weighted algorithmic approach could be redesigned and implemented using weightless Boolean hardware elements.

- Further investigate the properties of King's Neuroram as a neurofilter on an extended range of noise types, whilst understanding the importance of the threshold criteria [King 2000].

1.10 Overview of the Thesis

The aim of the thesis is to develop a collection of weightless Boolean elements which can be used to enhance existing weightless neural networks. The viability of developing these weightless Boolean elements to implement an alternative to standard binary implementations of traditionally based arithmetic and clock based logic is also investigated. Furthermore it is proposed that these weightless Boolean elements could be used to redesign standard weighted neural networks. A key reason for moving to weightless Boolean elements is the removal of the clock and arithmetic units in conjunction with the removal of the weighting associated with traditional binary. The driver for this is to improve the robustness to corruption and generate a more predictable, graceful degradation in performance when subjected to single event failures. The necessity for this is driven by a new threat facing avionics, that of atmospheric radiation [MacDiarmid et al 2005]. Trials are performed at several ground-based neutron accelerator facilities comparing traditional implementations of the swap block commonly found in the median filter. This is compared with the novel swap block developed as part of a weightless Boolean median filter.

1.10.1 Chapter 2 A Chronological History of Weightless Neural Networks

This chapter is a continuation of the introduction and puts the thesis into context giving a chronological history of weightless neural networks which form the backbone of the research. The origins of hardware neural networks and some of the physiological thoughts of early philosophers including James and Hebb which have led to some of the learning paradigms still used today [James 1890, Hebb 1949]. This thesis and the chronological history are bound to the field of hardware weightless neural networks and do not cover algorithmic implementations of weightless neural networks. An overview of Austin's ADAM network in conjunction with Willshaw and N point thresholding are discussed [Willshaw et al 1969, Austin et al 1987]. The principles of King’s Neuroram are also described [King 2000].

1.10.2 Chapter 3 A New Weightless Boolean ADAM and Non - Hebbian Learning

Chapter 3 proposes a collection of new weightless Boolean elements to implement threshold techniques. The primary driver for the development of these weightless Boolean elements is the implementation of a weightless hardware ADAM neural network. Austin’s implementation although FPGA based is still implemented using arithmetic weighted binary counters to sum the columns and rows of the matrices before applying a threshold [Weeks et al 2005]. Novel weightless Boolean elements are developed which implement the key thresholding criteria used in the ADAM network, these being Willshaw and N point thresholding [Austin 1986]. These elements were also trialled on the author's custom-built FPGA circuit board. A description of the hardware is presented along with the test criteria applied during this research. An additional weightless Boolean thermocoding technique which is designed to operate on serial weightless data is given. This technique has been patented by BAE SYSTEMS and a copy of the author's patents can be found in Appendix C.

The learning criteria for the correlation matrix memory are examined. Until now the correlation matrix memory has always relied on Hebbian learning to train the neurons located at the intersections. In weightless logic this is implemented with a traditional 'AND' gate. It is proposed that this 'AND' gate can be replaced by several other logic operators to allow the development of further learning criteria. These additional learning criteria are investigated to see if they offer a plausible alternative. The application of these criteria were simulated to determine the effect on matrix saturation in comparison to the traditional Hebbian learning criteria. The chapter concludes with a schematic for a Boolean hardware weightless ADAM neural network which consists of the sum and threshold techniques proposed earlier in the chapter.

1.10.3 Chapter 4 Hardware Weightless Boolean Median Filters

Chapter 5 presents two new methods of implementing weightless median filtering. The first of these is a technique for ordering the data in which the author has contributed a tagging method to the architecture. The second is the author's novel weightless Boolean median filter which is clockless and has been implemented in an FPGA. Both these techniques have been patented, full details can be found in Appendix B. The novel technique was then compared with a weighted implementation of an FPGA when subjected to single event upsets caused by neutron radiation. Trials have been performed at TRIUMF and TSL and the results from TRIUMF were submitted and accepted for publication by the IET [King et al 2008].

1.10.4 Chapter 5 Performance of Weightless Neural Network Image Filters

Further examination of King’s Neuroram configured as a Type I Neuroram filter is performed in Chapter 4 [King 2000]. The analysis is performed on standard two-dimensional greyscale images and examines the capability of King’s Neuroram to remove an extended range of noise types. This is a continuation from King's initial research which trialled the filtering properties of Neuroram on similar images corrupted with salt and pepper noise. The images are corrupted with several noise types and an evaluation of both a conventional median filter and the neural filter are performed. The additional noise types include: additive Gaussian, additive uniform, multiplicative uniform and multiplicative Gaussian. A trial with salt and pepper noise is undertaken to compare with King's original analysis. Evaluation of the two filters is further extended with analysis of the performance when cascaded. The ability to alter the threshold on Neuroram is examined with respect to the effect on the noise removal of the image. King's hypothesis that the ideal threshold for the filter when operating on the removal of salt and pepper is half the number of data samples is tested [King 2000].

1.10.5 Chapter 6 Boolean Weightless Self Ordered Map

Chapter 6 describes a weightless Boolean implementation of the self ordered map [Aleksander et al 1995]. The chapter begins with a chronological history covering the development of the self ordered map. This history is not covered in Chapter 2 as it is traditionally a weighted neural network. The self ordered map differs from the previously described neural networks in that it is capable of ‘unsupervised learning’ [Kohonen 1984]. A description of the self ordered map and its operation is presented, in conjunction with the algorithms necessary to implement a simulation of the network on a sequential computer.

In order to develop the weightless Boolean self ordered map a collection of weightless elements was designed. Each of these individual elements is described in the chapter. These elements include a selector, an expander, and a Hamming distance reducer. Variations of these elements are also described. A simulation of the weightless self ordered map is also described and the associated program is available in Appendix D. A block diagram describing how these elements can be constructed to form a weightless Boolean implementation concludes the chapter.

1.10.6 Chapter 7 Summary and Conclusions

Chapter 7 gives an overview of the research objectives and descriptions of all the chapters and their contribution to the research.

The summary of chapter 1 describes the author's contribution to the two published papers, copies of which are attached in Appendix A [Chugg et al 2003a, Chugg et al 2003b]. These papers describe some of the effects observed when using CCD's to capture single event effects at trials performed at TSL.

Chapter 2 gives an overview of the field of hardware weightless neural networks describing the prior art relating to this thesis.

Chapter 3 describes the development of a collection of sum and threshold elements, including a serial thermocoder which has been patented and is attached in Appendix C [Armstrong et al 2003, Armstrong 2003a, Armstrong 2003b].

Two new additional sum and threshold elements are described which complement King's original elements [King 2000] A collection of alternative learning techniques for the correlation matrix memory are described which builds on the traditionally used Hebbian learning technique. A weightless Boolean hardware implementation of L–Max and Willshaw thresholding techniques are discussed for use in a novel weightless Boolean hardware implementation of ADAM.

Chapter 4 discusses the properties of King's Neuroram acting as an image filter on an extended range of noise types [King 2000]. It is demonstrated that King's Neuroram as an image filter is capable of removing multiplicative uniform, salt-and-pepper and multiplicative Gaussian noise. However it does not perform well with additive Gaussian and additive uniform noise and demonstrates similar characteristics to that of a standard median filter. King's filter performs slightly worse at noise removal than the conventional median filter however it does not cause as much blurring. An analysis of the ability to adjust the threshold and hence alter the performance of the Neuroram filter is also discussed. King originally suggested a threshold value equal to 50% of the number of exemplars presented to his Neuroram was optimum when performing image filtering on two dimensional greyscale images [King 2000]. This research shows a threshold of 33% is the optimum.

An evaluation into the performance of both a conventional median filter and Neuroram when cascaded is described which shows that the performance of the two filters differs. The Neuroram filter showed improvement in cascades of up to five deep; however this was only three for the median filter. Overall the performance with regard to noise removal was better for the median filter even with less numbers of cascades.

Chapter 5 described a tagging technique and a weightless Boolean median filter both of which have been patented, copies of which are included in Appendix C. The results from comparing the performance of the swap block taken from the weightless median filter and a conventional median filter when subjected to neutron radiation are presented. The results show that when implemented on an FPGA susceptible to SEU when subjected to neutron radiation the robustness and failure modes differ. The weightless swap block suffers more upsets due to its architecture but the resultant failures were more predictable. In contrast the weighted swap block failed less but the consequences were more unpredictable due to the weighting of the data. These findings were submitted and both presentations were accepted for publication at the IET (formerly IEE). One of the presentations was made and copies of both can be found in Appendix B.

Chapter 6 discusses the weightless Boolean hardware which was developed in order to implement a Weightless Self-Ordered-Map. The weightless Boolean architectures developed include: a selector, expander and a Hamming distance reducer. A block diagram of a weightless self ordered map is presented.

1.10.7 Chapter 8 Further Work

Chapter 8 offers suggestions for further work in the field of weightless Boolean neural elements as well as discussing further improvements to ADAM to reduce the effect of saturation. A further hardware implementation of the weightless self ordered map in an FPGA architecture is suggested. A possible extension to Neuroram and its filtering properties is proposed using adaptive filtering; two methods are discussed.

1.10.8 Appendices

Appendix A has the two papers resulting from trials at TSL in using CCD's to capture single event effects.

- Analyses of CCD Images of Nucleon-Silicon Interaction Events.

- Single Particle Dark Current Spikes Induced in CCD’s by High Energy Neutrons.

Appendix B contains the two presentations submitted and accepted by the IET for publication.

- BAE SYSTEMS, Air Systems Approach to the Problem of Atmospheric Radiation.
- System Level Prevention: Managing SEE Using Error Correction Techniques.

Appendix C contains three international patents, derived from the author’s BAE SYSTEMS invention reports. These have been examined and published.

- Ordering by Hamming Value.
- Serial Weightless Data to Thermocode Coded Data Converter.
- Ordering Weightless Binary Tuples According to Hamming Value.

Appendix D consists of a DVD retained on the inside back cover of this thesis. It contains MATLAB emulations, C code simulations, PIC C code, circuit diagrams and FPGA projects. It represents an archive of electronic data generated during the course of this research.

Chapter 2

2 A Chronological History of Hardware Weightless Neural Networks

2.1 Overview

This chapter presents a brief chronological history of the development of artificial hardware weightless neural networks.

The development of the field of artificial neural networks is driven by two factors. The first of these is the quest to understand the human brain. This has led to many biologically plausible models that demonstrate our growing understanding of the operation of neurons and brain structures. The second driver is the ability to use this technology to perform ‘intelligent’ operations, within specific boundaries, in order to solve complex non-linear problems. Intelligent operations are those which require ‘thought’ or decisions based on prior knowledge or experience. Intelligence cannot easily be achieved with traditional sequential computers as these require a defined algorithm. Neural networks overcome this limitation by forming model free estimators.

2.2 Boolean Logic

Artificial hardware weightless neural networks are based on Boolean logic, which derives from George Boole’s algebraic systems of logic published in 1854 [Boole 1854]. Claude Shannon proved in his Master’s thesis in 1937 that Boolean algebra and binary arithmetic could be used to simplify the switching circuits based on electromechanical relays used at the time in telephone routing switches [Shannon 1937]. His thesis took this concept further by showing that structures of electromechanical relays could be used to solve Boolean algebraic problems. Although McCulloch and Pitts are widely acknowledged as producing the first neuron model recent evidence shows that Turing proposed several neuron elements and associated structures [Copeland et al 1996]. Turing referred to these elements as ‘unorganised machines’ and proposed three separate machines known as ‘A’ type, ‘B’ type and ‘P’ type. The most relevant is the ‘B’ type machine which is described with two inputs and one output which could be connected to each other; today we know this element as the two input

NAND gate. Turing goes on to describe how initially these elements are randomly connected, and by appropriate interference these connections would alter and mimic education. Turing stated that a ‘B’ type machine can be trained to ‘do any required job’ [Copeland et al 1996].

2.3 The McCulloch and Pitts Neuron Models

McCulloch and Pitts wrote the first of two papers which are now recognised as the first model for a physiological neuron [McCulloch et al 1943, Pitts et al 1947]. Most artificial neural networks are fundamentally based upon these early models. The McCulloch and Pitts models are built up of several key elements. The inputs in to the model are known as *synapses*. The synapses are the connections through which information is fed to the neuron. Within the neuron there are two processing elements, one for summation and another for thresholding. The output state of the neuron is determined by the information received from the synapses combined with the neuron’s threshold function. The output of the neuron is delivered by the *axon*. The axon of a neuron can be connected to one or more neurons via their synapses. An example of the basic McCulloch and Pitts neuron is given in Figure 2.1.

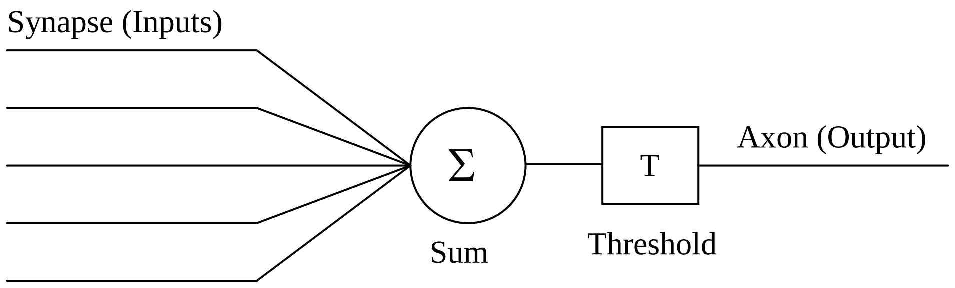


Figure 2.1 McCulloch and Pitts Neuron

The neuron ‘fires’ when the summed synaptic response overcomes the threshold, otherwise it remains in it quiescent state.

2.4 Inhibition of the Neuron

McCulloch and Pitts describe multiple types of operation for their basic model, in particular they examine inhibition of the neuron. Two models for inhibition are presented using inhibitory synapses, which prevent or make the neuron less likely to fire if they are triggered. These models are termed, absolute inhibition and relative inhibition respectively.

2.4.1 Absolute Inhibition

In the case of absolute inhibition the inhibitory synapse has total control of the neuron. If the inhibitory synapse fires then the neuron cannot ‘fire’ at all. If the inhibitory synapse is not active then the neuron will ‘fire’ if the sum of the excitatory inputs exceeds the threshold as shown in Figure 2.2.

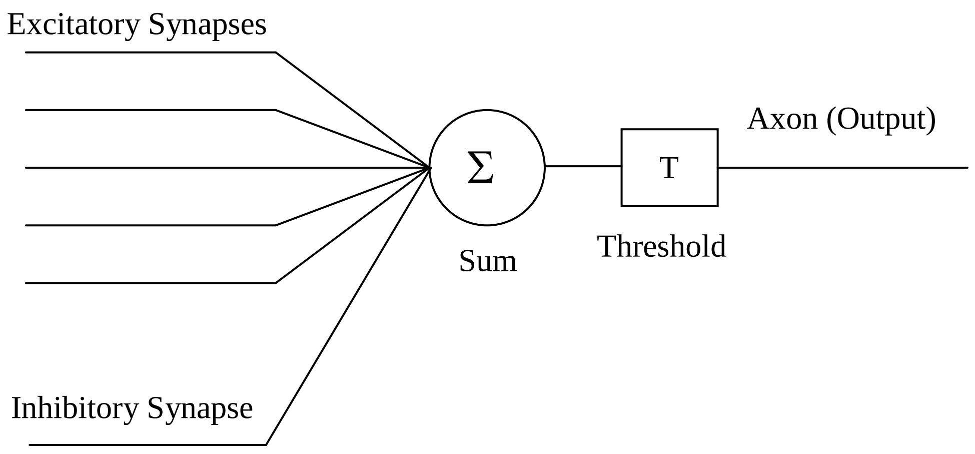


Figure 2.2 A McCulloch and Pitts Neuron with Absolute Inhibition

2.4.2 Relative Inhibition

In the case of relative inhibition, firing of the neuron is determined by the sum of the excitatory synapses minus the sum of the inhibitory synapses. If this result exceeds the threshold then the neuron will fire, otherwise it remains in its quiescent state as shown in Figure 2.3.

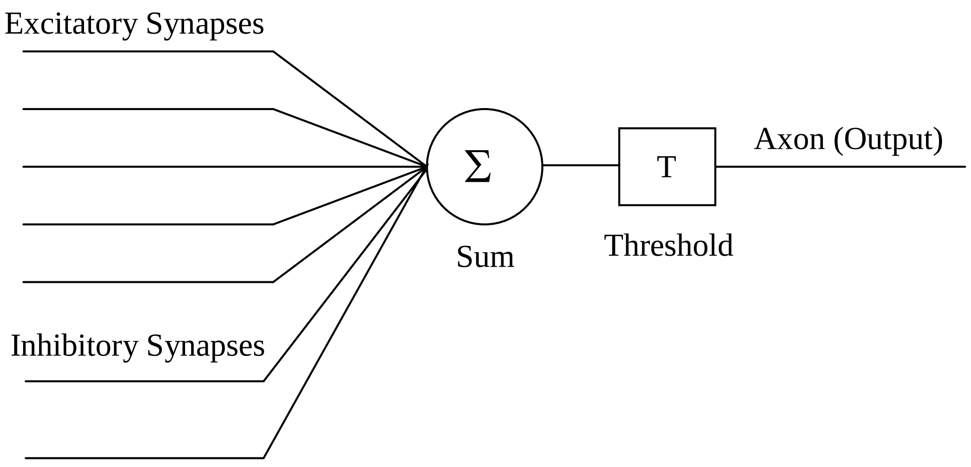


Figure 2.3 A McCulloch and Pitts Neuron with Relative Inhibition

2.4.3 The Common McCulloch and Pitts Model

Figure 2.4 shows the common McCulloch and Pitts model which forms the basis of most artificial neural networks. The McCulloch and Pitts Model can be described by equation 2.1 where w_i represents the weighting value for the input x_i with T being the threshold function. The weights *Wn* can only be integers. The model that McCulloch and Pitts present is weightless and hence the weights represent the number of weightless connections from a given source to the neuron.

$$Output = \sum_{i=1}^{n} w_i x_i > T \qquad \textbf{2.1}$$

2.4.4 A Mathematical Model for the McCulloch and Pitts Neuron.

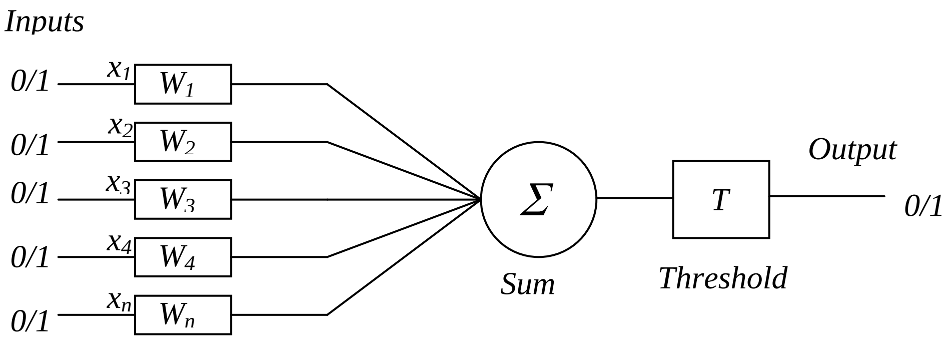


Figure 2.4 The Common McCulloch and Pitts Model

2.5 Hebbian Learning

In 1949 Donald Hebb wrote a paper, which built on McCulloch and Pitts earlier papers, and introduced several important advances in the field of neural networks [Hebb 1949]. Hebb's paper discusses the link between psychology and physiology: this understanding has not dated in the intervening years. The paper also introduces the term *connectionism,* which is used to describe the connections of single neurons to form neural structures containing multiple neurons.

One of Hebb's most important contributions to the field is his model for Hebbian learning: *'When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.'*

This is similar to the earlier thoughts of James. *'When two brain processes are active together or in immediate succession, one of them, on reoccurring tends to propagate its excitement into the other'* [James 1890]. If we substitute the 'brain processes' with 'neurons' we have a connectionism rule that is almost identical to that of a Hebbian neuron.

In 1956 Rochester et al at the IBM laboratories, in discussion with Hebb, implemented the first computer simulation of Hebbian learning [Rochester et al 1956]. This first simulation used neurons based on weightless binary devices and a threshold. This simulation failed because the strength of the simulated synapse of the neurons grew uncontrollably. In order to prevent this a normalisation rule was applied which meant that synapse values grew in strength at the expense of others. As well as the normalisation function the second simulation had several other differences. These are that the synapse strength could range from –1 to +1 in real values instead of 0 to +1, meaning the neurons were no longer weightless binary neurons. The output of these simulated neurons was dependent upon frequency of firing; this was graded from 0 to 15. The Hebbian rule was also modified so a synapse value increased if it fired whilst one that did not fire was decreased. The system that was simulated consisted of 512 neurons and showed signs of working. This resulted in the algorithmic weighted implementation of the common McCulloch and Pitts Model which forms the basis of the more prevalent weighted algorithm neural networks such as the Perceptron and Hopfield networks [Rumelhart et al 1986, Hopfield 1982].

2.6 The First Hardware Weightless Neural Network

The first practical hardware use of a weightless neural network was a pattern recognition system developed by Bledsoe and Browning [Bledsoe et al 1959a, Bledsoe et al 1959b]. They tackled the problem of character recognition. The hardware they implemented to perform this task was an array of photo-detectors in a 10 by 15 formation. These arrays were pseudo-randomly paired; they consisted of 150 photo-detectors creating 75 photo-detector pairs. Each photo-detector pair allowed the production of four possible states in memory. Figure 2.5 shows a smaller 8 by 8 matrix to illustrate the operation of the system.

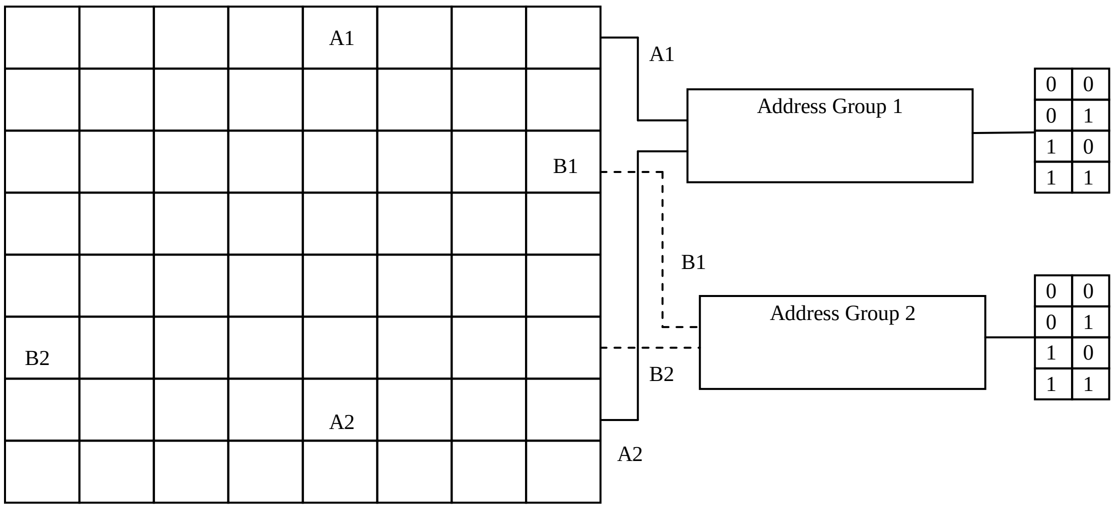


Figure 2.5 Example of the Bledsoe and Browning Technique

They introduced the term *tuple* to the forum. A tuple is a collection of weightless binary elements, which in this case are taken from a data space using pseudo-random mapping. Equation 2.2 is used to determine the number of storage sites in the memory matrix, *L*.

$$L = S^n \times \frac{N}{n} \times C \qquad \textbf{2.2}$$

Where *S* is the number of operational states of the photocell. (*S*=2 for states 0 and 1)
n is the parameter for *n*-tupling
N is the number of photocells
C is the number of categories of patterns learned and read (for Bledsoe and Browning's example, C=36)

Bledsoe and Browning also studied the effects of changing tuple size *n*. They noted that if the tuple size *n* =1, the system would behave as a standard template matcher. However they showed that for small tuple sizes, the system had the ability to generalise. Bledsoe and Browning also demonstrated that for small tuple sizes the memory could easily saturate if too many exemplars were shown. Multiple exemplars are required for noisy data or for data with variances such as positional inaccuracies. To compensate for over saturation the memory size can be increased by increasing *n*. This was also demonstrated by Ullmann [Ullman 1973]. In the early 1970's memory was an expensive commodity unlike today. Ullmann also performed a computer simulation assessment to determine the sensitivity of the system to the pseudo-random mapping process. This study concluded the memory was generally insensitive to this memory mapping process [Ullman 1969]. Bledsoe and Browning also investigated non-exclusive *n*-tuple mapping of the photocells in their system. They concluded that there was no real advantage to be gained with this method. The main reasons for this were the increased memory capacity required and the longer computation time [Bledsoe et al 1959b].

2.7 Correlation Matrix Memory

In 1961 Steinbuch developed a learning matrix, 'Die Lernmatrix' [Steinbuch 1961], as shown in Figure 2.6. The key property of this matrix is its ability to associate one pattern with another; the process of hetero-association. Steinbuch used this matrix as a self-correcting translator circuit because of its properties in dealing with incomplete or corrupt data [Steinbuch et al 1962, Steinbuch et al 1967].

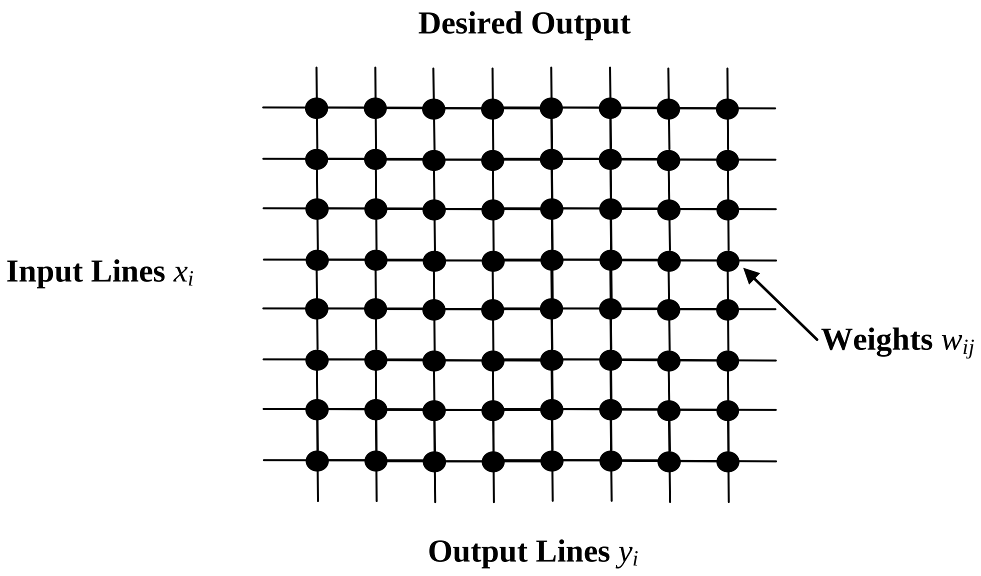


Figure 2.6 Die Lernmatrix

A mathematical description for this matrix is given by Picton [Picton 1994]. An example is now presented.

Matrices have been used to describe the inputs and outputs of the correlation memory.

Equation 2.3 shows the output matrix [*Y*] which is the product of the input matrix [*X*] and the correlation matrix [*W*].

$$[Y]=[X]\times[W] \qquad \textbf{2.3}$$

Equation 2.4 shows Equation 2.3 re-arranged where $[X]^t$ is the transpose of [*X*].

$$[W]=[X]^t\times[Y] \qquad \textbf{2.4}$$

Picton gives the following example showing a system trained on two patterns containing 4 inputs and 4 outputs so that *n*=4 and *P*=2.

Equation 2.5 shows that within the [*X*] matrix there are rows labelled X_1 and X_2. Vectors have been created to describe the inputs 1010 and 0110.

$$[X]=\begin{bmatrix}X_1\\X_2\end{bmatrix}=\begin{bmatrix}1010\\0110\end{bmatrix} \qquad \textbf{2.5}$$

Equation 2.6 shows that within the [*Y*] matrix there are rows labelled Y_1 and Y_2. Vectors have been created to describe the outputs 1000 and 0101.

$$[Y] = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} 1000 \\ 0101 \end{bmatrix} \tag{2.6}$$

Equation 2.7 shows the transposed [*X*] matrix for this example

$$[X]^t = \begin{bmatrix} 10 \\ 01 \\ 11 \\ 00 \end{bmatrix} \tag{2.7}$$

Equation 2.8 shows the trained correlation matrix for the given inputs and outputs.

$$[W] = \begin{bmatrix} 10 \\ 01 \\ 11 \\ 00 \end{bmatrix} \times \begin{bmatrix} 1000 \\ 0101 \end{bmatrix} = \begin{bmatrix} 1000 \\ 0101 \\ 1101 \\ 0000 \end{bmatrix} \tag{2.8}$$

Equation 2.9 shows the input vector multiplied by the correlation vector. The expected outputs of [1000] and [0101] are not seen until an appropriate method of thresholding is performed.

$$\begin{bmatrix} 1010 \\ 0110 \end{bmatrix} \times \begin{bmatrix} 1000 \\ 0101 \\ 1101 \\ 0000 \end{bmatrix} = \begin{bmatrix} 2101 \\ 1202 \end{bmatrix} \tag{2.9}$$

Equation 2.10 demonstrates the capability of the matrix to deal with corrupt data. In the example given, bit 1 is swapped giving [1011] instead of [0101]. If we examine the output with this given input we notice the matrix still gives the correct output. It should be noted that this example was carefully chosen to show this aspect of the matrix.

$$[1011] \times \begin{bmatrix} 1000 \\ 0101 \\ 1101 \\ 0000 \end{bmatrix} = [2101] \tag{2.10}$$

In 1969 Willshaw, Buneman and Longuet-Higgins published a paper describing an optical implementation of an associative memory [Willshaw et al 1969]. Their optical correlation matrix memory is called a *correlograph*. They took the correlograph one step further to develop a neural network model. In order to create this model they assumed the input lines in the horizontal direction act as axons and the vertical lines act as synapses. To this they applied Hebbian learning [Hebb 1949]. Because the system is a digital implementation this meant the synapses were logically 'ANDed' to the axons.

The training of this type of network is performed by example in a similar method to the learning matrix.

2.8 WISARD

WISARD is an acronym for Wilkie, Stonham and Aleksander's Recognition Device. The inspiration for WISARD came from the previous work of Bledsoe and Browning and Aleksander [Bledsoe et al 1959a, 1959b, Aleksander et al 1984]. Initially Aleksander demonstrated a fused adaptive element, which could be trained as shown in Figure 2.7.

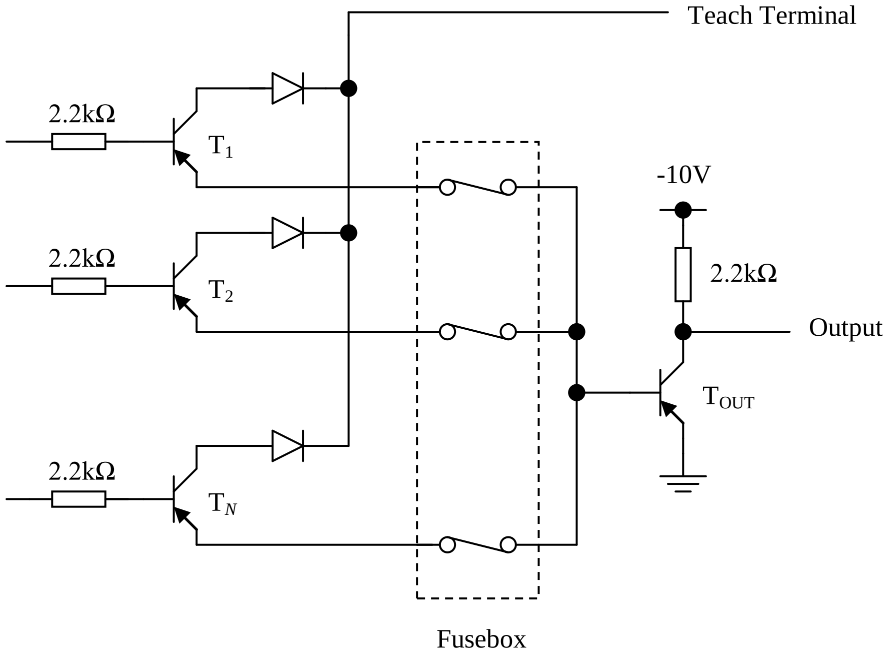


Figure 2.7 A Fused Adaptive Element

Aleksander proposed this circuit in 1965 [Aleksander 1965]. The fused adaptive element is taught by example. Showing the element, or collection of elements, all the required outputs for a given set of inputs performs the teaching of the network whilst the network is in teaching mode. Once the training has been performed and the network is set to recall mode the system will reproduce the trained association on the presentation of the same given input. Aleksander also discusses the potential to recognise patterns in the presence of noise and this can be clearly seen as a precursor to WISARD.

The development of WISARD did not come about until 1984 when Aleksander, Wilkie and Stonham, studied the recognition of patterns in a noisy environment [Aleksander et al 1984, Stonham 1985]. WISARD was inspired from the earlier work of Bledsoe and Browning. It is similar to the hardware character recogniser in the way the data space is randomly mapped into tuples stored in memory elements, in this case RAM blocks and the result of all the memory elements summed. Unlike the earlier hardware of Bledsoe and Browning, WISARD captures the image using a camera and digitising system instead of photo detectors to make the matrix space [Aleksander et al 1984]. Matrix elements are pseudo-randomly chosen to form tuples, the tuple size being determined by the designer and limitation on address size of the RAMs. Figure 2.8 shows an example of the mapping of the address space.

WISARD is trained by using a ‘teach’ methodology by showing exemplars to the data matrix. Before training can commence all the RAM locations need to be initialised to zero. Whilst set in the teach mode a pattern is input to the RAMs within a discriminator and the decoded address location from the *n*-tuple is stored with a ‘1’.

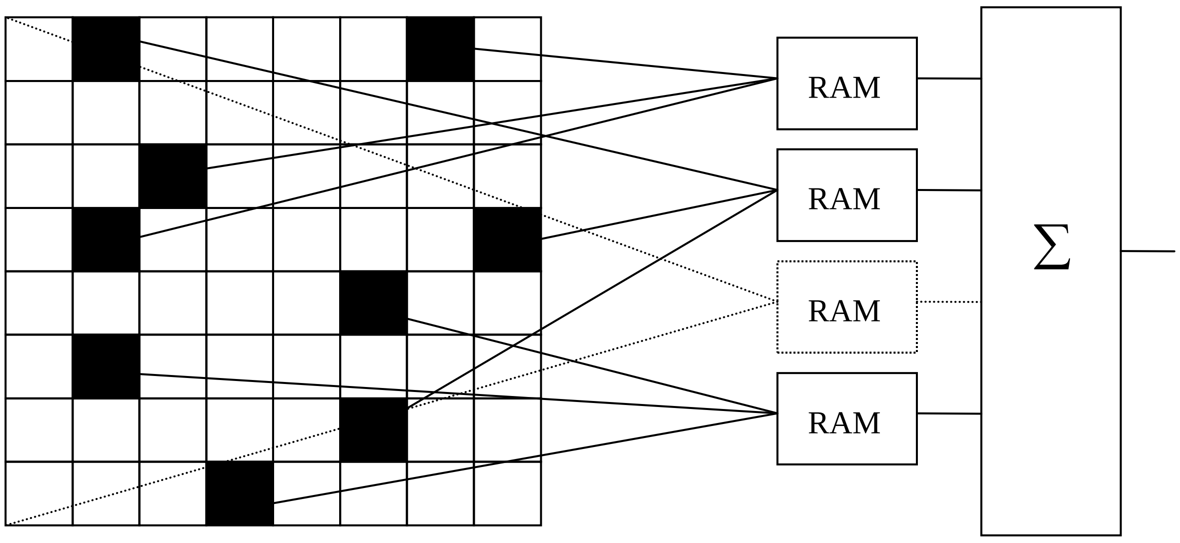


Figure 2.8 A RAM Discriminator

Once the system has been fully trained the network is switched to recall mode where the response for all the individual discriminators is monitored when a pattern is presented to the network as shown in Figure 2.9. The highest response to the pattern is the match or the closest match [Aleksander et al 1995].

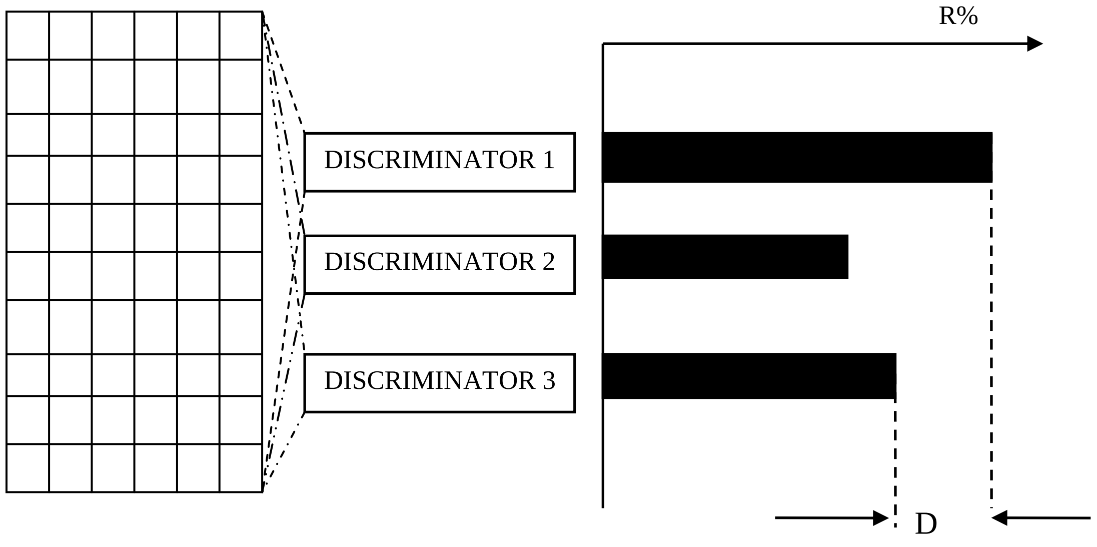


Figure 2.9 A Multi-Discriminator with Associated Response

Picton’s analysis of WISARD is presented here:

Equation 2.11 is derived from the training of the network, where *P* is the number of pixels, *n* is the tuple size and *K* is the number of RAMs per discriminator.

$$K = \frac{P}{n} \qquad \textbf{2.11}$$

Equation 2.12 shows the probability ρ of selecting pixels from the same particular RAM in the *j*th discriminator, where *A* is the area of overlap between the presented pattern and the previously taught pattern, *P* is the number of pixels and *n* is the tuple size.

$$\rho = \left(\frac{A}{P}\right)^n \qquad \textbf{2.12}$$

If *K* is large then it is probable that *p* of the RAMs in the discriminator will fire. Equation 2.13 shows the output of the *j*th discriminator r_j if all the outputs of the RAMs are summed.

$$r_j = K\left(\frac{A}{P}\right)^n \qquad \textbf{2.13}$$

It is worth noting that if the two patterns are identical $r_j = K$ and A=*P*. Equation 2.14 shows the normalisation of the output.

$$r_j = \left(\frac{A}{P}\right)^n \qquad \textbf{2.14}$$

These equations have been derived for a simple example of one trained pattern and one input, however the equation still applies for more complex situations [Picton 1994].

A further way of making a decision is to use a confidence measure C as shown in Equation 2.15. The confidence in a result is determined by taking the highest response from the strongest discriminator r_1 and comparing it to the response from the strongest incorrect discriminator r_2.

$$C = \frac{r_1 - r_2}{r_1} = 1 - \frac{r_2}{r_1} \qquad \textbf{2.15}$$

The highest confidence score for Equation 2.15 is 1 and this will only be gained if the response for $r_1 = 1$ and the response from $r_2 = 0$. This is highly unlikely and will only occur when a pattern totally matches a trained pattern and has no overlapping pixels with any other exemplar in the memory.

Substituting Equation 2.14 into 2.15 gives:

$$C = 1 - \left(\frac{A_2}{A_1}\right)^n \qquad \textbf{2.16}$$

Equation 2.16 indicates that as n increases the discriminatory power of the network increases, although there is a cost penalty in terms of increased memory requirement and reduction in the ability to generalise.

2.9 ADAM

ADAM is a bi-directional hetero-associative network, meaning the network can associate one set of data with another set of data in both directions [Austin et al 1987, Austin 1997, Austin et al 1994, Bolt et al 1992, Kennedy et al 1994]. ADAM is the acronym for Advanced Distributed Associative Memory and a block diagram of the architecture is shown in Figure 2.10.

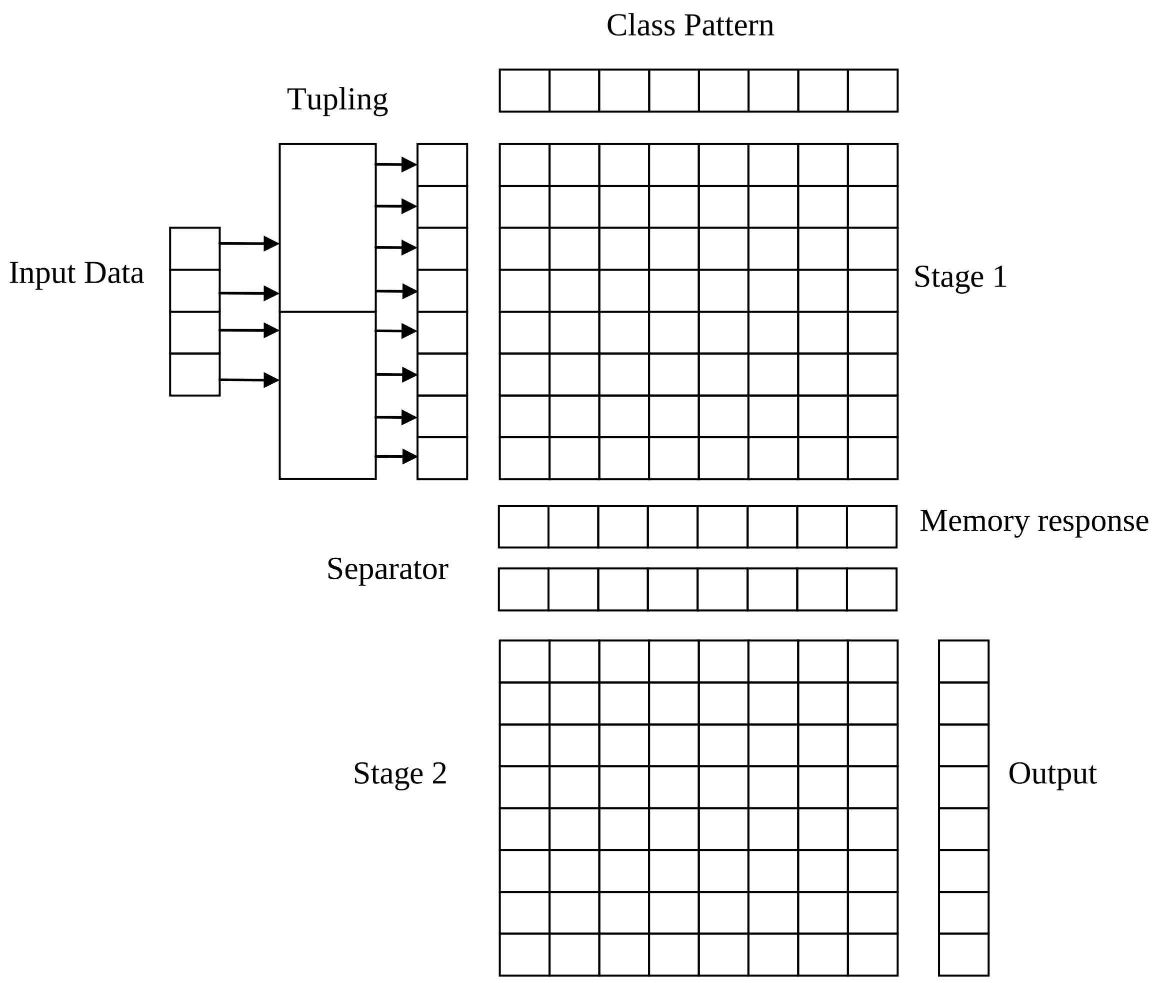


Figure 2.10 A Block Diagram of the ADAM Architecture

ADAM was created by Austin as part of his doctoral research in order to perform scene analysis [Austin 1986, Austin et al 1994]. Since then Austin has become a Professor at the University of York heading the Advanced Computer Architectures Group. This position has allowed Austin and his students to closely examine and develop ADAM for industrial applications [Austin 1993, Austin 1998]. A typical application of its use has been the development of a hardware board for use in a computer for the Post Office to allow data mining of postal addresses which contain inaccurate or missing information [Austin et al 1998, Austin et al 1994, Kennedy et al 1995]. The performance of ADAM has been closely studied. An assessment of ADAM's fault tolerance and reliability has been performed by Bolt. Bolt demonstrated that as the tuple size *n* increased so did the operational fault tolerance [Bolt 1991].

ADAM is based on fundamental building blocks from some of the earlier neural networks [Beale et al 1997]. The heart of ADAM is two weightless correlation matrix memories using a Hebbian learning rule [Willshaw et al 1969]. To reduce the problem of saturation ADAM uses pseudo-randomly mapped tupling to expand the input data. This random tuple mapping is similar to that used by Bledsoe and Browning in their optical character recogniser [Austin 1994].

Austin’s major contribution in the formation of the ADAM architecture was to add a class separator pattern between correlation matrix memories. The advantage of this is that instead of associating A to B, the network associates A to C to B, where C is the class pattern. It is this class separator that allows two different data types to be associated to each other. Figure 2.11 shows an example of the first correlation matrix with the class pattern 1 0 0 0 1 0 1 0 being presented and an input of 1 0 1 1 0 0 1.

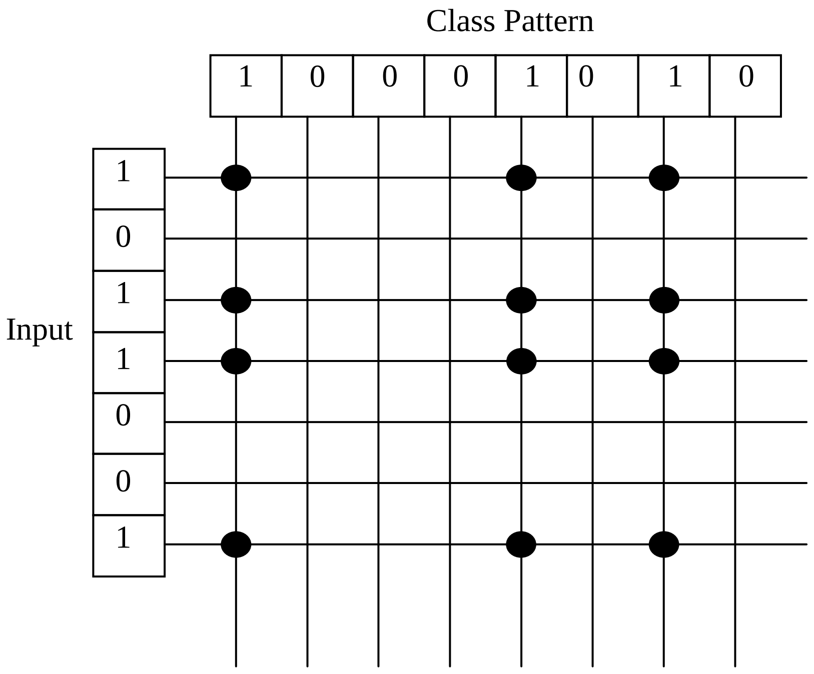


Figure 2.11 The First Correlation Matrix Memory with ADAM

Due to the nature of the class separator pattern being designer specified the distribution of the weightless code can be carefully controlled; this allows Austin to use his L-max (also known as N point) thresholding technique [Willshaw et al 1970].

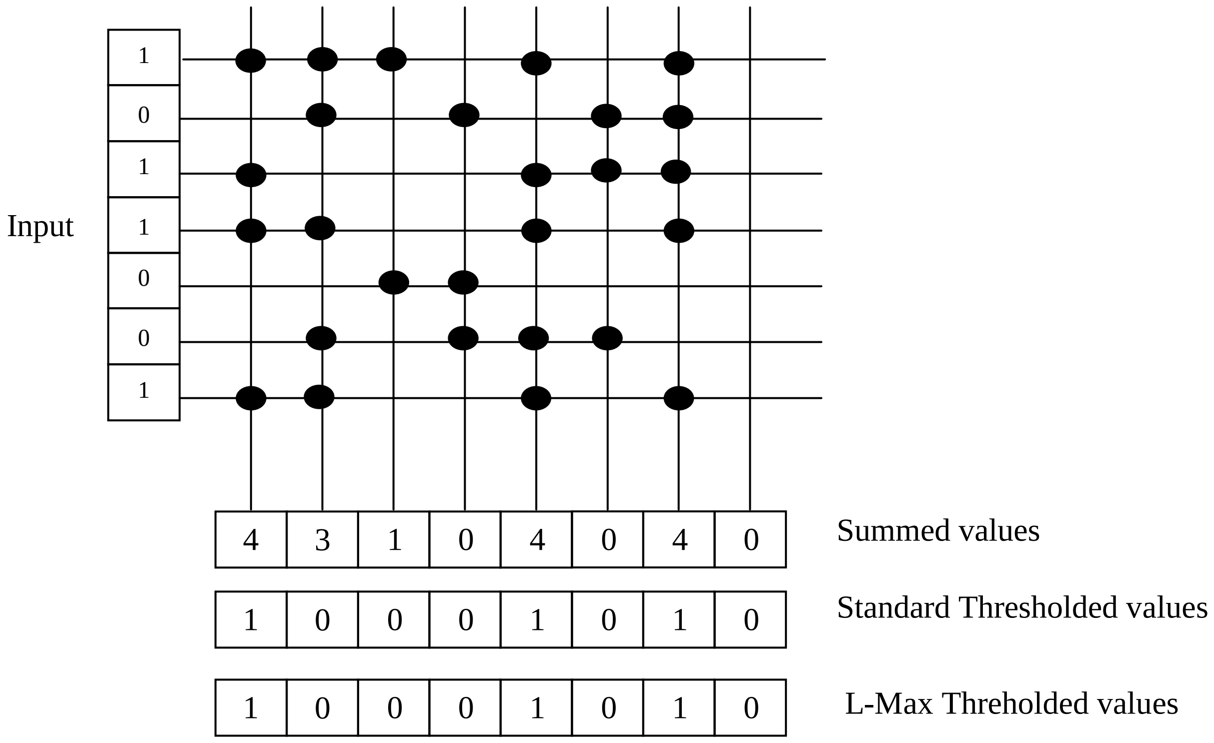


Figure 2.12 A Correlation Matrix Memory Containing Trained Data

Previously trained data input is shown to the correlation matrix memory containing all the trained data. Figure 2.12 shows the response of the summed outputs with a standard thresholding technique applied. The standard threshold value is determined by the number of bits set in the input pattern, in this case the threshold is 4 [Kennedy et al 1995]. The L-Max threshold value is generated by taking the highest summed value and making this the threshold, so in the example shown below the L-max threshold value is 4 [Willshaw et al 1970].

2.10 AURA

AURA is a hardware weightless neural network developed by Austin and his team at the University of York in 1997 [Austin et al 1998]. AURA is an acronym for Advanced Uncertain Reasoning Architecture. AURA was developed in conjunction with industry, one of the industrial sponsors was British Aerospace (now BAE SYSTEMS). British Aerospace were interested in AURA for the development of mission systems control for future aircraft. Other interested industrial parties included chemical companies and the Post Office. Austin and his team developed products for both these parties based on AURA. A system for the Post Office was developed to find the best matches for misspelt, incomplete or inaccurate addresses from its address database. AURA's strength is in searching large databases to match incomplete data a process known as 'data mining'.

AURA was developed for rule based learning systems. The heart of AURA is still based around the CMM as in ADAM. The main difference with AURA is its method of pre-processing which gives the network its reasoning capability. In order to process the data it first needs to be converted to a suitable form. To achieve this several stages of processing are required. The CMM can only deal with weightless binary strings of *k* bits, *k* being determined by the size of the CMM's being used. The first stage of the network converts lexical tokens into binary vector patterns of the appropriate *k* bits in length. This is followed by the binding of the variable names to the values then the superimposed coding of the sets of bound variables is performed. This pre-processed data is then routed to each of the appropriate CMM networks. Arity is defined as the number of antecedents in a rule. The outputs of the CMM's are resolved by taking the class separator pattern and identifying the matching rule. A diagram of this architecture is given in Figure 2.13.

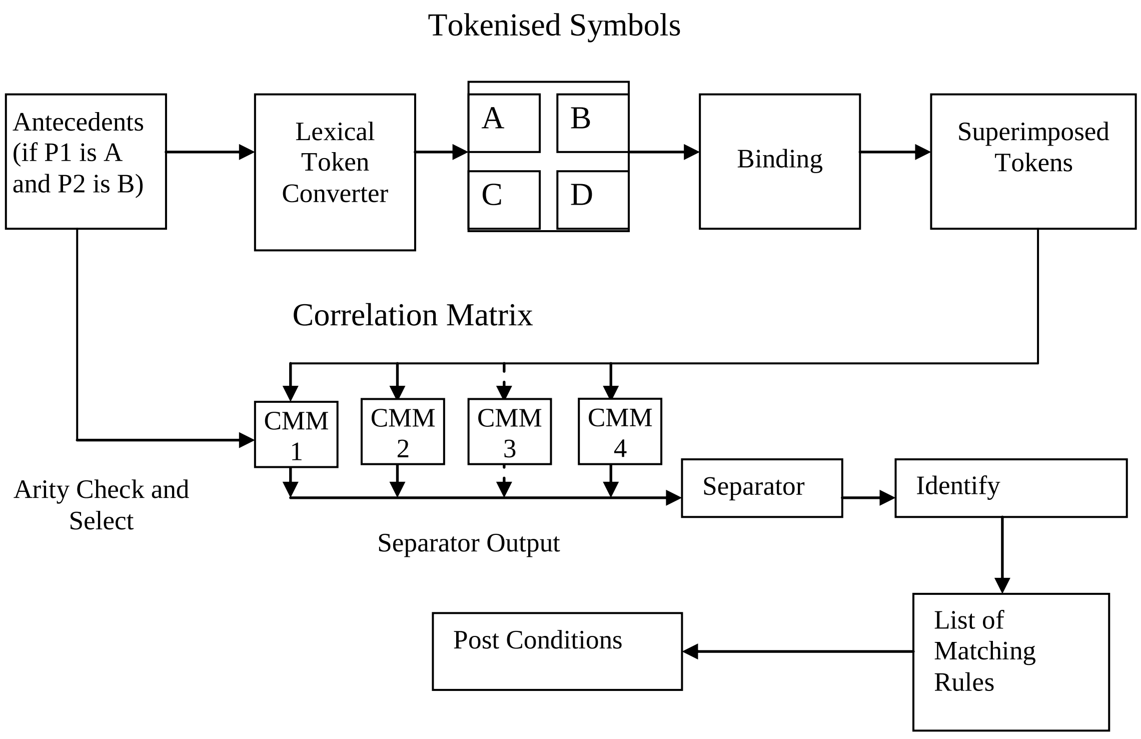


Figure 2.13 A Block Diagram of AURA

The key feature of AURA is its ability to match data which is incomplete given certain rules and gives it 'data mining' properties.

2.11 Cortex

Austin and his team at the University of York developed a suitable hardware and computing architecture for a neural computer called Cortex-1 [Weeks et al 2005]. Cortex-1 is an extension of AURA and is a scalable structure of multiple AURA's. The AURA systems are implemented on the PRESENCE cards developed earlier by the University. PRESENCE stands for Parallel Structured Neural Computing Engine [Kennedy et al 1995]. Cortex-1 consists of 7 Sun PC's which are networked and contain 4 PRESENCE cards [Moulds et al 1999]. Austin and his team are using this approach to perform large scale data mining to develop a system to detect benefit fraud using information from multiple databases. This project was sponsored by the Department of Trade and Industry (now Department of Business, Skills and Innovation). Other projects that Austin and his team are working on under his company Cybula include DAME which is an acronym for Distributed Aircraft Maintenance Environment. The company's involvement in DAME is to develop a distributed data mining and pattern matching engine based on AURA technology [Austin et al 1998]. This is a major multiple partner project including other universities and companies including Rolls Royce. Austin is now working in the field of bio-metrics again using AURA technology in the task of three dimensional facial recognition systems. Austin has updated his Cortex computing and hardware which has been re-named Cortex-2. Cortex-2 contains the new PRESENCE 2 hardware processing card, which has greater processing power, storage capacity and interconnection speeds [Weeks et al 2005]. PRESENCE 2 is based around a Xilinx Virtex 2 six million gate FPGA and 4 GB of SDRAM. Austin is investigating further improving the performance of his AURA system with a compression technique for the data retrieved from the CMMs, this technique is called Compact Binary Vector coding [Austin et al 1998].

2.12 Neuroram

Neuroram was designed by King in 1997, and constructed using hardware weightless Boolean elements [King 2000, King 1999a, b, c, King et al 1999d]. King was trying to create a vastly simplified version of some of the elements of the neo-cortical region of the human brain. The weightless binary elements that are described in his thesis, includes a Boolean method of performing a sum and threshold which is fundamental to the construction of artificial neurons [King 1999a]. King uses weightless asynchronous logic to perform this operation instead of using the standard clocked counters as found in other weightless neural systems such as ADAM. In his thesis King presents a philosophy of reducing clocked elements to improve the resilience and speed of his networks.

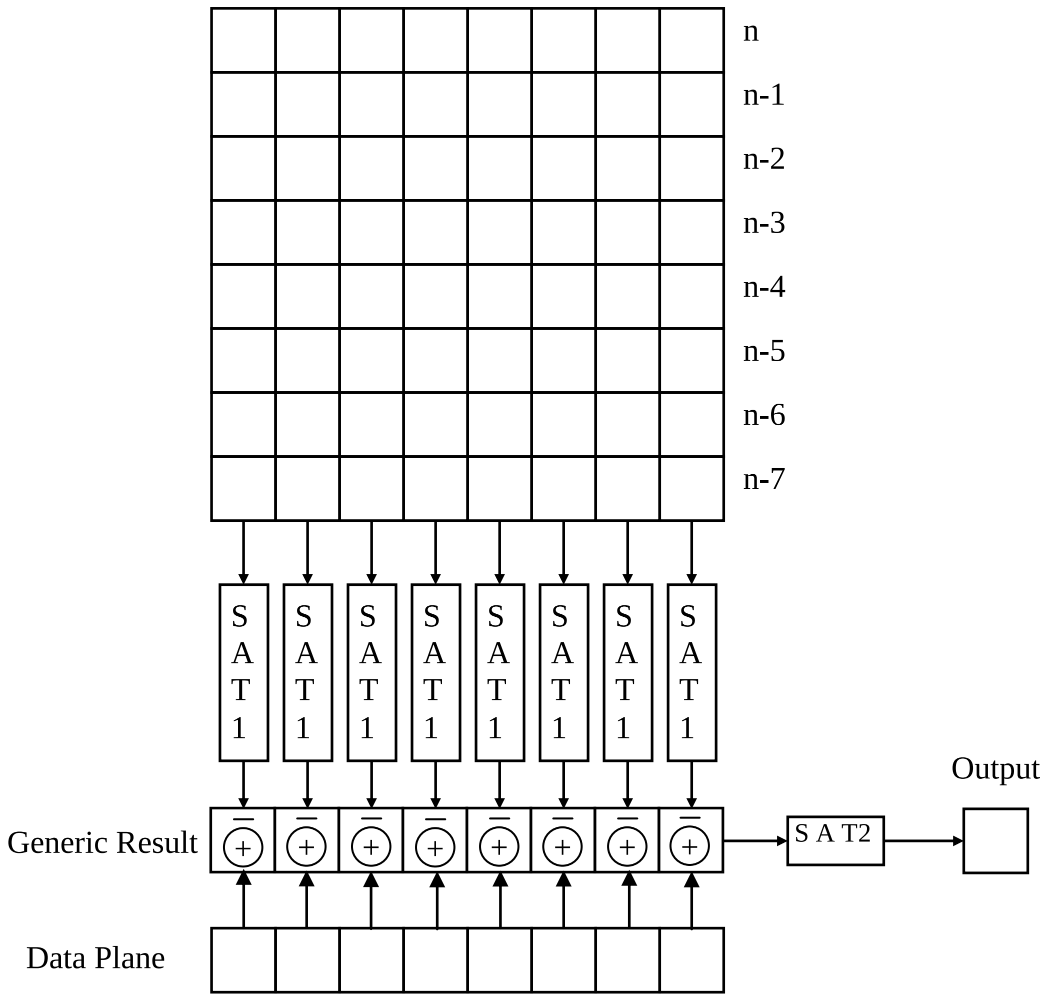


Figure 2.14 A Block Diagram of Neuroram

The block diagram of Neuroram shows the structure of the system. The first block contains a first-in first-out (FIFO) based memory where strings of data are loaded in time sequence and shifted down. The second stage is the sum and threshold element which sums and thresholds each of the columns to form the generic result, this is then further summed and thresholded to give the output.

It is worth noting that although King used his Boolean sum and threshold elements, conventional sum and threshold elements could be used without any change in functionality [King 2000]. King also studies coding methods that could be used to format the data into a suitable form for use within Neuroram [King 2000].

King then uses Neuroram as a simple digital filter to reduce the effects of white Gaussian noise on signals. The coding and pre-processing of the data is essential in the operation of Neuroram. King investigated the use of Gray code, K-code (which was his own code), thermocode and weighted binary coding for pre-processing using a simple test waveform with additive white Gaussian noise and correlating them to the original waveforms without the additive noise [King 2000]. His results show the importance of coding on Neuroram as demonstrated by the effects of K-coding which severely corrupted data. The pre-processing method which yielded the highest correlation score for this problem was a Gray code with a pseudo-random key. King also investigated the effects of thresholding for the (SAT1) threshold elements. This investigation was performed by training the network on 32 data sets of the same waveform in 8 bit Gray code with 256 samples and additive white Gaussian noise on each waveform. Each waveform had a signal to noise ratio of approximately 22 dB which was increased by 5 dB when filtered with a threshold between 12 and 17 [King et al 1998]. These waveforms were then fed into a Neuroram architecture and summed and thresholded by the sum and threshold elements (SAT1) to generate the generic template. The generic template was then correlated to the original reference waveform. The threshold was then varied from 0 to 32 to evaluate the signal to noise ratio. The results show a value around the mid threshold value greatly increases the signal to noise ratio demonstrating the suitability of Neuroram as a filter and ‘clean-up’ memory [King 2000].

King takes the filter investigation one stage further and adds the second sum and threshold element (SAT2) to the architecture [King 2000]. This element allows the generic result to be stored and used again or a new generic result to be used. This is determined on whether the new generic result is a significant improvement on the already stored generic result, if so, it is overwritten. The new architecture was then used to filter two dimensional images. The images were 8 bit Gray encoded with salt and pepper noise added. The aim of the network was to remove the noise. In order to do this a 3 by 3 window was slid over the image. The 8 bit Gray code for each element of the window was then fed into Neuroram and each of the elements summed and thresholded with a mid value to produce the generic template. This generic template was then correlated with the central pixel value; if the generic template is better than the central pixel value the value is changed. King demonstrates that this methodology provides an effective way of removing salt-and-pepper noise in images [King 2000]. King also repeats this process using thermocode instead of Gray code which yields a further small improvement. King also compares the results with a standard median filter, which performs slightly better than the weightless thermocode filter.

Chapter 3

3 A New Weightless Boolean ADAM and Non-Hebbian Learning

3.1 Introduction

Austin's ADAM network is traditionally implemented in a combination of hardware and software, often using hardware to accelerate the processing and maximise the parallel nature of the network [Weeks et al 2005]. Even though the principles of ADAM are weightless, Austin uses conventional weighted binary counters to calculate the sum and threshold values [Austin 1986]. This chapter proposes a fully Boolean hardware implementation of ADAM using the novel Boolean elements proposed in this thesis. The Correlation Matrix Memory CMM which forms the heart of ADAM is also examined and a new learning paradigm is presented: non-Hebbian learning [Hebb 1949]. Methods of combining CMMs to increase network capacity are also investigated. The chapter concludes with an architecture of a fully Boolean hardware implementation of the ADAM network. This chapter also describes the hardware and the test philosophy for all hardware weightless elements that have been developed through this research. A novel serial sum and threshold technique which has been patented is also described.

3.2 Design and Test Philosophy

This research is focused on the development of weightless Boolean hardware elements. Therefore in order to develop these weightless elements and test them a suitable platform and philosophy is required. There are two main approaches to this; the first of these is to simulate the networks on a personal computer using a high-level language. The disadvantage of this method is that a computer can only sequentially process single step operations due to the limitation of its processor. Conversely, neural networks by their very nature are parallel, updating multiple elements simultaneously. This means that the simulations tend to be slow in comparison; however this is mitigated due to the speed and multiple processor architecture of modern personal computers. Neural networks although vastly parallel operate at much lower refresh rates. Modern higher-level languages such as C and MATLAB are ideally suited to the simulation of weightless neural networks and their associated weightless elements. MATLAB is a matrix-based language which is compatible with the structure of weightless Boolean elements, because these can easily be described in matrix form. MATLAB supports the logical Boolean operators equivalent to the gate functions being implemented. The C programming language offers a more flexible approach and has been used in this research to simulate the weightless ADAM neural network. Therefore the software implementation of these networks has used Borland C++ version 4.52 operating in 'only C compiler mode' and MATLAB version 7.01 including the image processing toolbox version 5.01 in conjunction with the signal processing toolbox version 6.21. All the simulations and weightless toolboxes created within this research can be found in the enclosed DVD in Appendix D.

In conjunction with the software simulation of the weightless neural elements this research has also required suitable hardware in order to develop and verify the hardware elements developed during this research. The aim of this research was to create and develop a new weightless neural network followed by analysis of the network. A custom designed board known as 'Neuromorph' was designed and built to aid the evaluation of the hardware elements. The board was designed using the Innoveda 2 schematic capture tool (now Mentor Graphics) and the symbols manually created. The layout was performed at BAE SYSTEMS and the production of the printed circuit board was outsourced.

3.3 Neuromorph Board

An architecture which is much more suited to this task is that of a large field programmable gate array (FPGA). This is due to the fact that most weightless neural networks are constructed from multiple simple elements connected in parallel. FPGA's lend themselves to these architectures because they contain look up tables which can be configured as logical functions. Therefore simple elements which are connected in parallel are easy to implement on them.

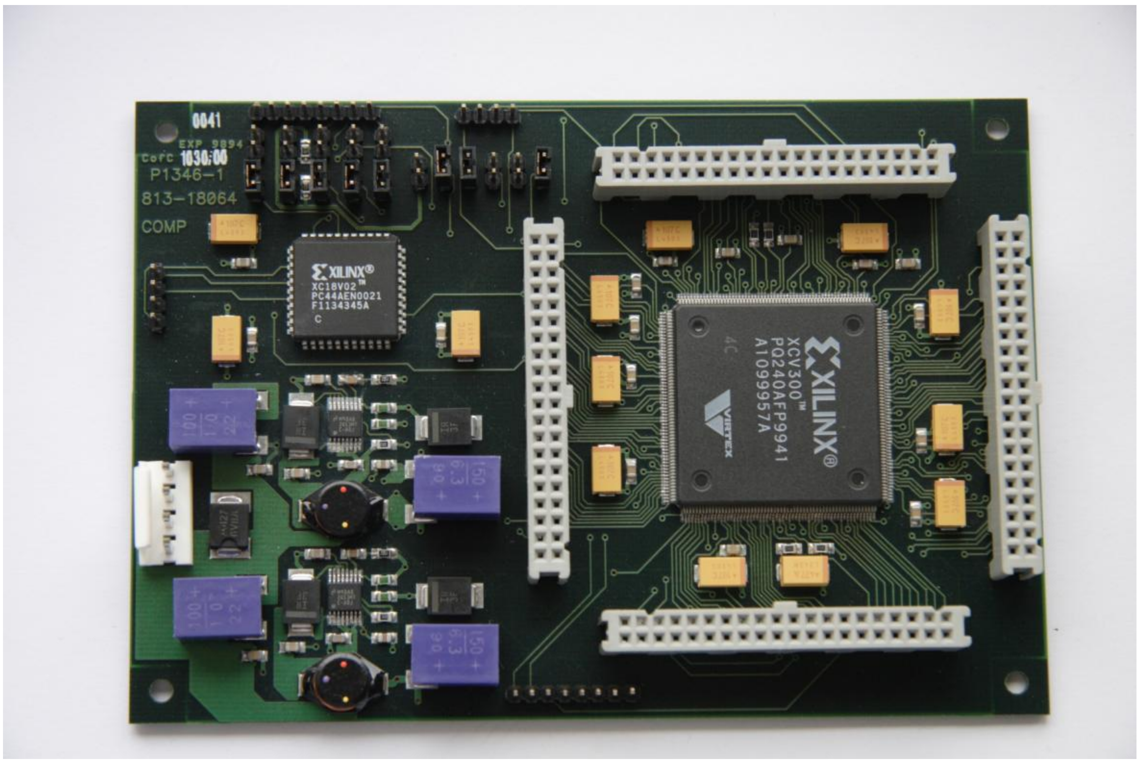


Figure 3.1 The Neuromorph Board Designed by the Author

The board is designed around a 300,000 gate Xilinx Virtex I FPGA which is the heart of the board [Xilinx 1998]. The board also contains two DC - DC power supplies which provide the FPGA with 2.5 volts and 3.3 volts. The 3.3 volts supply also supplies the on-board EEPROM which can store the FPGA programme. The FPGA can be configured so that at power on the EEPROM automatically configures the FPGA. This is required because the FPGA is static RAM based and needs programming as its configuration is lost on power down. The board has 160 input / output pins brought out to four connectors so that data can be easily interfaced to the FPGA. There are on-board jumpers which allow different configuration modes; including parallel, serial, JTAG and EEPROM. Initially implementations of weightless neural network designs were performed by entering the designs into the Innoveda 2 schematic capture tool using the Virtex symbol library. The design was then checked and compiled into an EDIF file for use by the Xilinx Alliance software, which checks, builds and compiles the code into a suitable format to program the FPGA. The resultant '.bit' file was then used to program the Neuromorph board using the Xilinx Multilinx system. Xilinx offer several ways of programming the device and the jumper settings on the Neuromorph board allow all of these. The most convenient way is using the JTAG port. The median filter discussed in Chapter 5 was developed using a new version of the Xilinx Alliance version 5.1 software which allows direct schematic entry and simulation. This negated the need of the Innoveda toolset and the associated exporting of the EDIF.

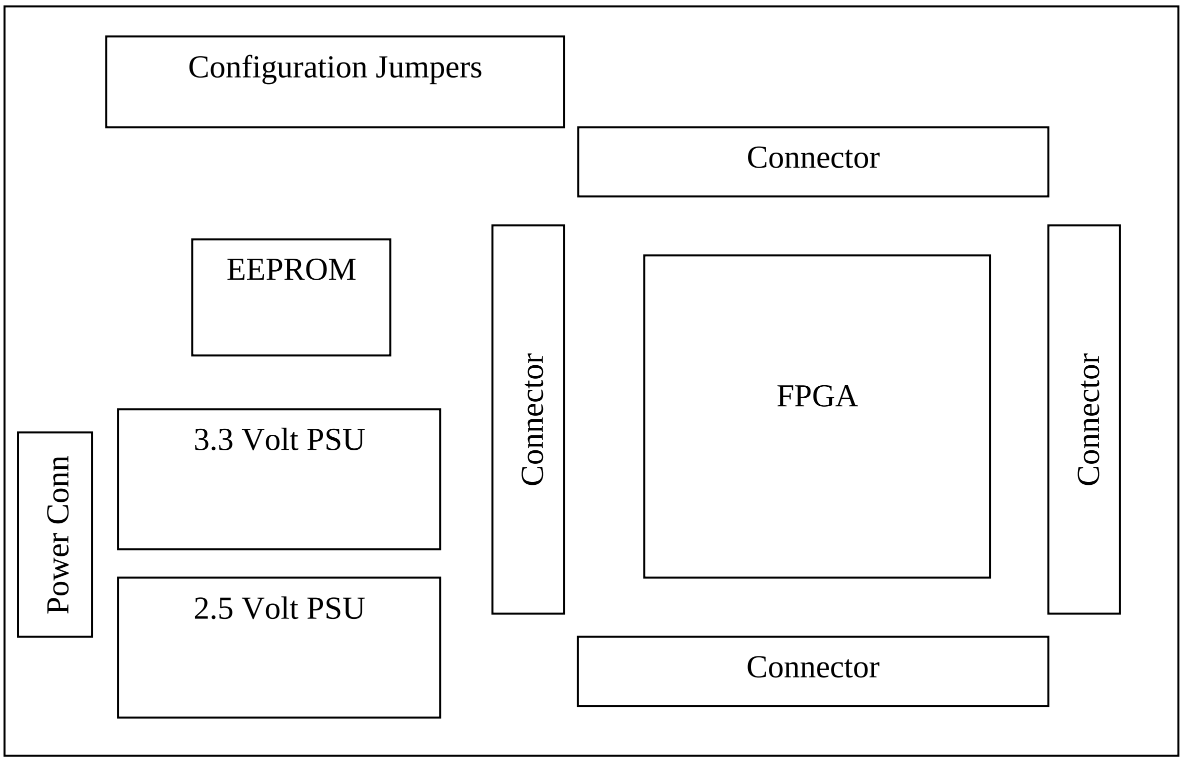


Figure 3.2 An Overview of the Neuromorph Board

A block diagram of the main components of the Neuromorph board are given in Figure 3.2.

3.4 Sum and Threshold

A fundamental element of any neural network is the sum and threshold function which forms the basic neuron; weightless neural networks are no exception. The difference in a weightless implementation is rather than being algorithm based, the output is determined by the number of active excitatory inputs minus the number of inhibitory inputs. King in his thesis likened this to a see-saw balance [King 2000]. All inputs and outputs in weightless neural networks only have two states 'firing' or 'not firing' and no weighting function. This means that these neurons lend themselves to simple implementation in conventional Boolean logic. A typical Boolean two input 'AND' gate could be also classed as a weightless neuron with a threshold of two and was appreciated by Turing [Copeland et al 1996]. Similarly if a two input 'OR' gate was used this could be described as a weightless neuron with a threshold of one as shown in Table 3.1.

Synaptic Input A	Synaptic Input B	‘AND’ Gate Threshold = 2	‘OR’ Gate Threshold = 1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Table 3.1 Boolean Logic Acting as a Neuron

In order to optimise the data for processing in a weightless neural network it is often advantageous to separate the ‘firing’ inputs from the ‘non-firing’ inputs. ‘Firing’ inputs are represented by ‘1’ and ‘non-firing’ inputs ‘0’ in conventional logic. An elegant method of performing this simple separation of logic within a tuple is to thermocode the data [King 2000]. This can be done in a number of ways; King has developed some Boolean structures which can perform this operation on small tuples or form hierarchies for larger tuples. The disadvantage of these structures is as the tuple data size grows so does the number of layers. The propagation delay of the structure increases with each additional layer.

3.5 King’s Weightless Thermocoder

Figure 3.3 shows a weightless thermocoder implemented to King’s design [King 2000].

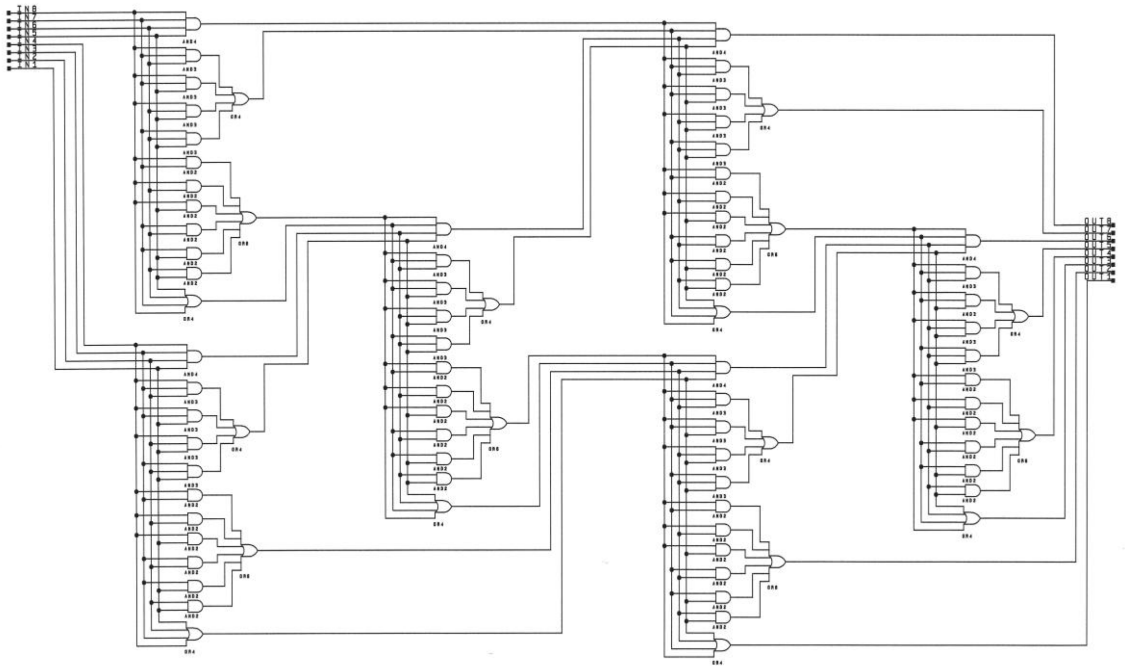


Figure 3.3 The Weightless Thermocoder

The design consists of six 4 input thermocoder blocks arranged into a follow through structure to ensure all bits are organised into thermocode. The structure allows 8 bits of weightless data to be converted into a weightless thermcode. The structure is clockless and parallel operating on Boolean logic, each layer adds < 1 ns propagation delay when implemented in modern high performance FPGA’s such as the Xilinx Virtex II [Xilinx 2001]. Due to the parallel nature of the structure the maximum propagation delay is < 8 ns equating to a clock speed of greater than 125 MHz. The weightless thermocode is used in weightless neural technology to perform the summation function, instead of standard weighted binary counter technology.

3.6 A Serial Weightless Thermocoder

A novel Boolean serial weightless thermocoder has been developed that generates an ‘n’ length thermocode from a weightless data stream [Armstrong 2003a]. This technique follows from the author’s earlier work on thermocoding [Armstrong 1999].

The technique converts a fast serial stream of weightless data into thermocode. The weightless stream of data is commonly found in weightless neural network systems, often in the sum and threshold areas of the network. This section of the weightless neural network needs to be robust and fast and this technique offers both these characteristics.

The operation of the thermocoder is based around the use of ‘D’ type flip-flops. As long as the initial state is a thermocode value the system is guaranteed to provide the correct thermocode after *n* clock cycles where *n* is the number of bits in the thermocoder. The thermocoder operates by performing a shift left operation or a shift right operation depending on whether the incoming data is ‘1’ or ‘0’. As the incoming data is introduced into the chain a bit of data is lost at the other end of the chain.

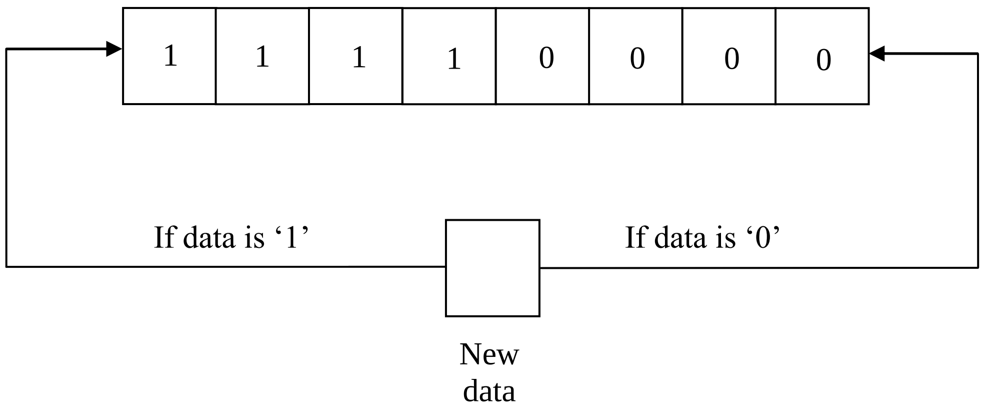


Figure 3.4 Initial Condition

Figure 3.4 shows the thermocode with an initial condition of half the bits set and the other half not set. The thermocoder can be initialised in any condition but it is recommend that the data is preset as thermocode otherwise it can take a number of cycles before the thermocode will give valid data.

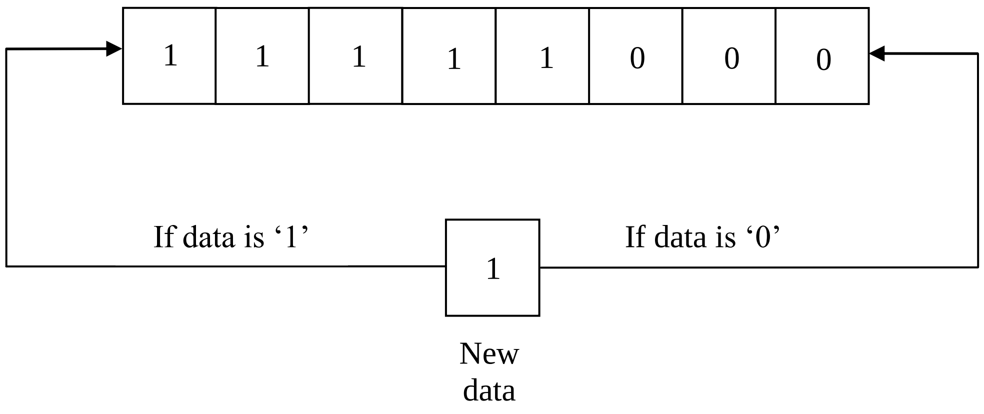


Figure 3.5 Input of Logic High ‘1’ Being Input to the Thermocoder

Figure 3.5 shows the thermocode as a serial bit of data of ‘1’ is presented and implemented in the thermocoder.

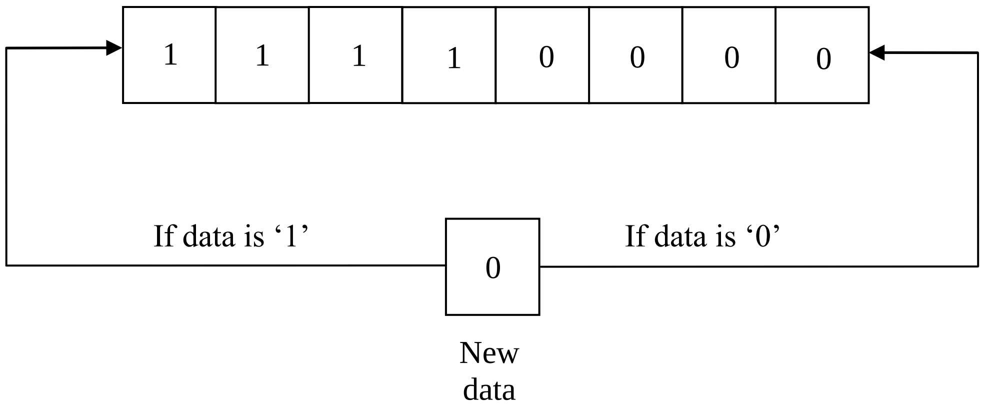


Figure 3.6 Input of Logic High‘1’ Implemented and Input of Logic Low ‘0’

Figure 3.6 shows the implementation of the next bit of data being presented to the thermocoder a logic low ‘0’.

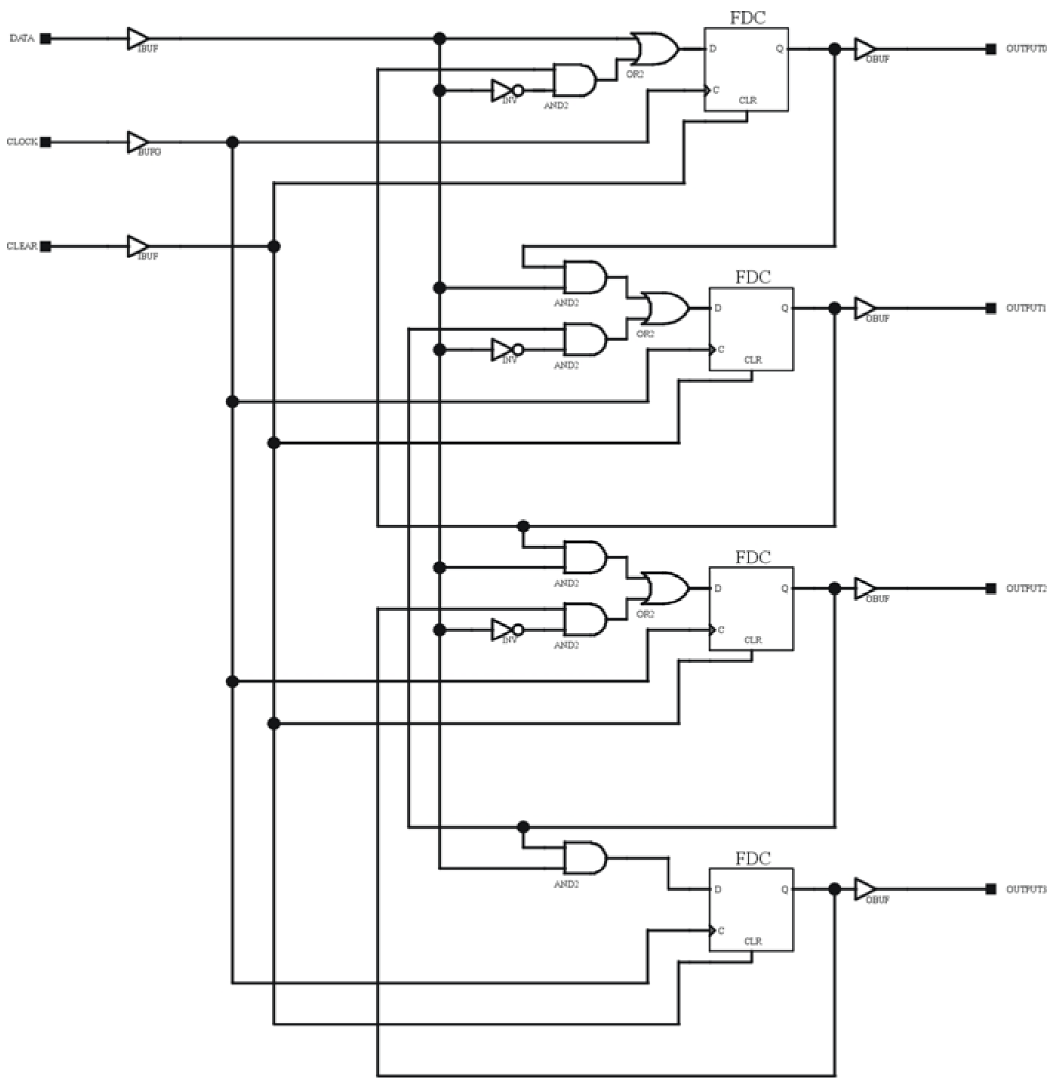


Figure 3.7 A Logical Implementation of Weightless Thermocoder

Figure 3.7 shows a four bit logical implementation of the weightless thermocoder. This implementation was performed using schematic entry into the Innoveda schematic entry tool using the Xilinx symbol library. The Xilinx symbol library often names the symbols for instance FDC represents a single D type flip-flop. The implementation was then checked and exported via an EDIF netlist to the Xilinx Alliance 4.2i toolbox where the design was checked and compiled to produce a suitable bit map for downloading into the Xilinx Virtex XCV300 FPGA on the Neuromorph card. The design was tested using a clock, light emitting diodes and switches. The technique is most suited to neural networks which generate the data in a stream such as ADAM [Austin 1986] and Neuroram [King 2000].

The most widely used technique for the summation and thresholding is that of using conventional arithmetic logic units and counters even in weightless neural networks.

Austin tends to use this approach in his ADAM and AURA architecture implementing them in fast FPGA's on bespoke acceleration hardware {Weeks et al 2005].

The thermocoder technique demonstrated can be used to make a sum and threshold module which can be used in neural networks to implement a basic McCulloch and Pitts neuron [McCulloch 1943]. This can be implemented by taking two of the thermocoder units, one for the 'sum' part and the other for the 'threshold' part and comparing the logic output of the two units used to determine the thermocode unit with the greatest number of '1's. If the 'sum' has the greatest Hamming value then the 'sum' is greater than the 'threshold' and the threshold has been exceeded. If the 'threshold' has the greatest Hamming value then the 'sum' is less than the 'threshold' and hence the threshold has not been exceeded. This comparison can be made using either the 'less than or equal to comparator' or alternatively the 'greater than or equal to comparator' shown in Figure 3.8 and Figure 3.9 respectively.

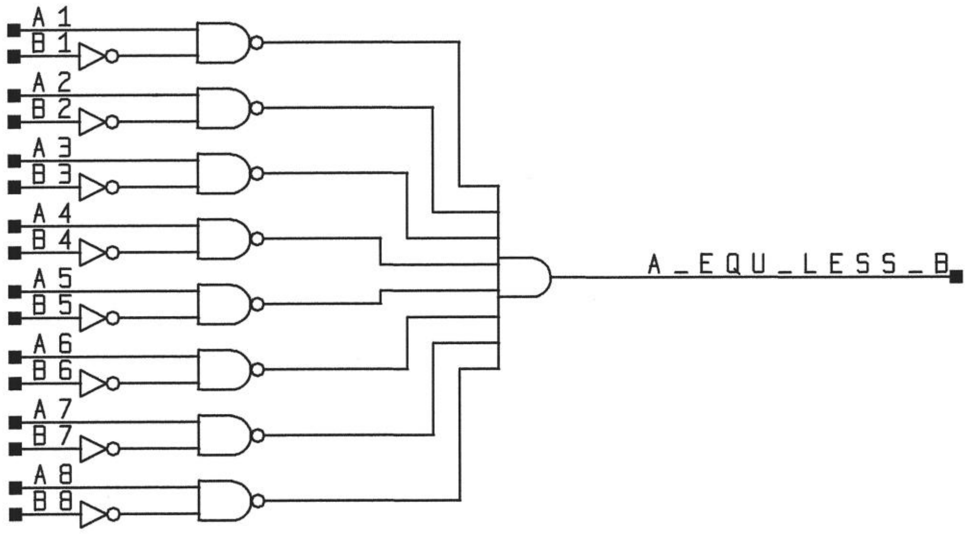


Figure 3.8 Less Than or Equal to Comparator

The greater than or equal functions are key elements to the implementation of a weightless neuron. The greater than or equal to sum and threshold element can be used to decide if the neuron should 'fire' or not. The greater than or equal to element takes a weightless thermocoded threshold and evaluates it against a weightless data stream. If the weightless data is not thermocoded then an additional weightless thermocoder will be required as in Figure 3.3.

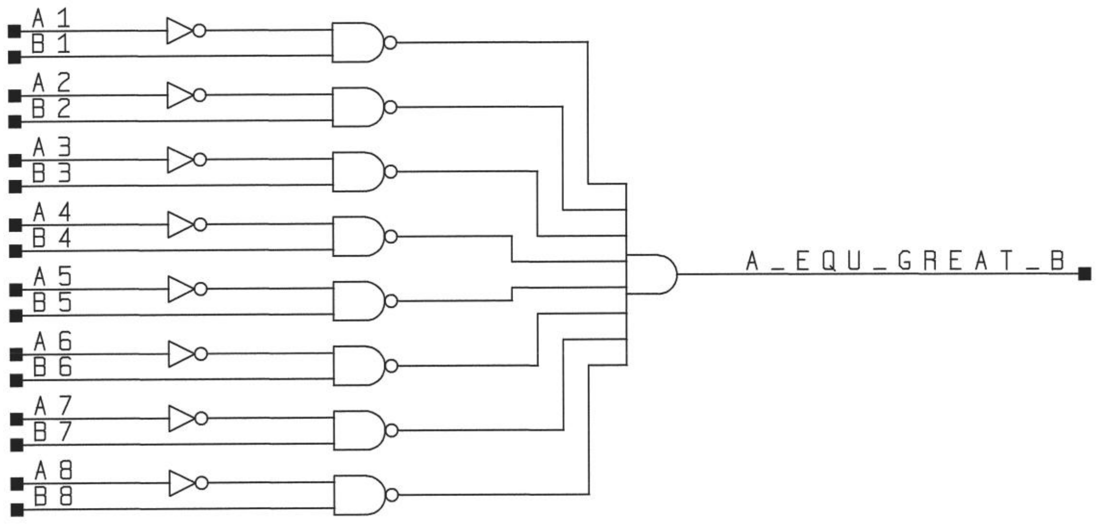


Figure 3.9 Greater than or Equal to Comparator

3.7 CMM Learning

The core of all of Austin's networks is the correlation matrix memory originally developed by Steinbuch to offer robustness and a form of error correction in telephone exchanges where components were prone to failure [Steinbuch et al 1962, Steinbuch et al 1967]. The 'learning' of the CMM in a weightless neural network is performed using Hebbian learning.

The learning concentrates on the reinforcement of neurons which 'fire'. Traditionally the CMM ignores neurons which do not fire and hence the associated junctions remain at a quiescent state. The matrix operates by performing an association between two inputs at each intersection where the neuron is located as shown in Figure 3.10.

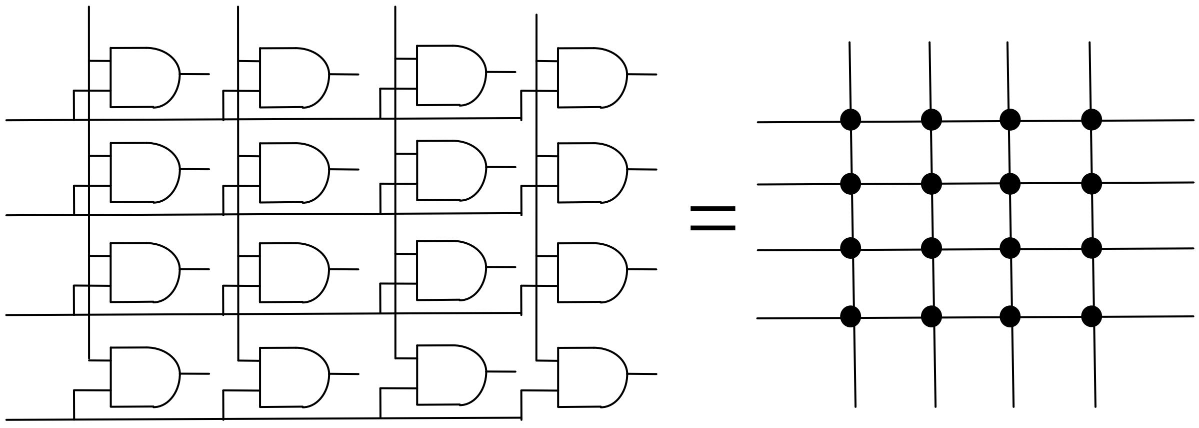


Figure 3.10 The Logic at the Intersections of the Correlation Matrix Memory

The neuron implements Hebbian learning using a two input 'AND' gate and a memory cell. The training of the neuron at the intersection means the inputs are strengthened by the storage of a ‘1’ at that intersection. This only occurs when both the inputs for a given intersection are ‘1’. This effectively generates a content addressable memory [Aleksander et al 1995]. Memory which is addressed by the data is equally known as content addressable memory or associative memory and differs from conventional memory as it does not require a prescribed address to where the information is located. The information is located based on its content.

3.8 The Performance of CMM as Neural Memory

The performance of the correlation matrix memory is dependent upon the level of saturation. Saturation is an issue when over 50% of the neurons in the matrix are trained with a '1'. Over-saturation of the network can severely affect the ability of the network to generalise. A more severe side effect of over-saturation is incorrect associations which can occur. The greater the saturation of the matrices, the greater the tendency of the network to produce incorrect associations [Bolt et al 1992]. Austin introduces several techniques to reduce the occurrence of saturation including tupling the incoming data. This ensures a pre-determined number of ‘1’s in a tuple for any given data possibility. Austin further controls the network by introducing the addition of a second correlation matrix memory [Austin 1986]. This addition of a second matrix allows separation of the matrices with the class pattern. Although at first sight this adds a greater level of complexity it allows a greater control of the matrices as the class pattern is a user controlled parameter. A combination of the two techniques significantly delays the onset of saturation with the network.

3.9 A New Learning Paradigm of Non-Hebbian Learning

This research has investigated whether different Boolean logic operators at the intersection of the correlation matrix memory are plausible and offer any merit. These techniques were proposed to improve the performance of an over-saturated network. The rationale being that a non-association is equally important as an association. Therefore the neurons in the matrices which are not responding are equally important when evaluating a trained network against a given input for a response.

The first trial involved a new implementation of the correlation matrix memory, where the importance of the non-association was given prime focus. This resulted in the development of a correlation matrix memory which associated '0's using a 2 input NOR gate as shown in Table 3.2.

This demonstrated that the use of other Boolean logic operators were plausible and functional such as the EXNOR gate.

Synaptic Input A	Synaptic Input B	Gate	Output
0	0	NOR	1
0	1	$\overline{A} . B$	1
1	0	$A . \overline{B}$	1
1	1	AND	1

Table 3.2 Boolean Neural Logic

The new techniques add an extension in that other 'learning elements' are proposed to that of Hebbian learning. The operation of the matrix remains unchanged, the difference is the new elements of learning. Instead of using the Hebbian 'AND' gate method new learning elements are given below.

3.9.1 Traditional Hebbian Learning

1010 taught with 1001 using the Hebbian learning method on the CMM.

$$\begin{array}{c|cccc} & 1 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

3.9.2 ‘NOR’ Gate Implementation

This method has been identified as the author’s ‘non-Hebbian’ technique. This method logically inverts the inputs, so unlike the Hebbian method the learning is performed on the ‘zeros’ instead of the ‘ones’. This can be implemented by inverting both inputs and then passing this to an ‘AND’ gate or instead using a ‘NOR’ gate.

Example of Non-Hebbian learning: 1010 taught with 1001 using the Hebbian learning method on the CMM.

$$\begin{array}{c|cccc} & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{array}$$

3.9.3 Difference Learning

This method has been identified the author’s ‘Difference’ technique. This method logically inverts one input, so unlike the Hebbian method the learning is performed by ‘ANDing’ an inverted input with a non-inverted input – an A AND (NOT B) gate.

Example of Difference learning: 1010 taught with 1001 using the difference learning method on the CMM.

Example 1

$$\begin{array}{c|cccc} & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}$$

Example 2

$$\begin{array}{c|cccc} & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

This demonstrated that the use of other Boolean logic operators were plausible and functional. It was immediately apparent that this approach was not practical as a straightforward substitution for a conventional correlation matrix memory in the ADAM network [Austin 1986]. As the ADAM network has been optimised to reduce the number of ‘1’s hence increasing the ‘0’s within the matrices causing the immediate over-saturation of the network when using this neural operator.

The network was redesigned to address this by reducing the number of ‘0’s stored within the correlation matrix memory. This necessitated the redesign of several elements of pre-processing to remove the bias to a low level of ‘1’s to a low level of ‘0’s.

The result was that an inverse tuple and inverse class pattern had been created. Functionally it was only another ADAM network all be it focused on ‘0’ instead of ‘1’ but the ability to deal with saturation had not changed.

3.10 A Weightless ADAM Architecture

The key aim was to develop an alternative hardware architecture for Austin’s ADAM which was Boolean logic based instead of the current counter based techniques. This meant that logical implementations of the Willshaw and N point thresholding were required [Willshaw et al 1969].

The following elements perform key functions required in neural networks. It is common for the functions to be implemented using clocked counter based techniques in Austin’s hardware implementations even in weightless neural networks. These techniques are novel in that they allow a completely weightless Boolean implementation of the hardware as well as the function.

3.11 A Weightless Boolean Willshaw Threshold Element

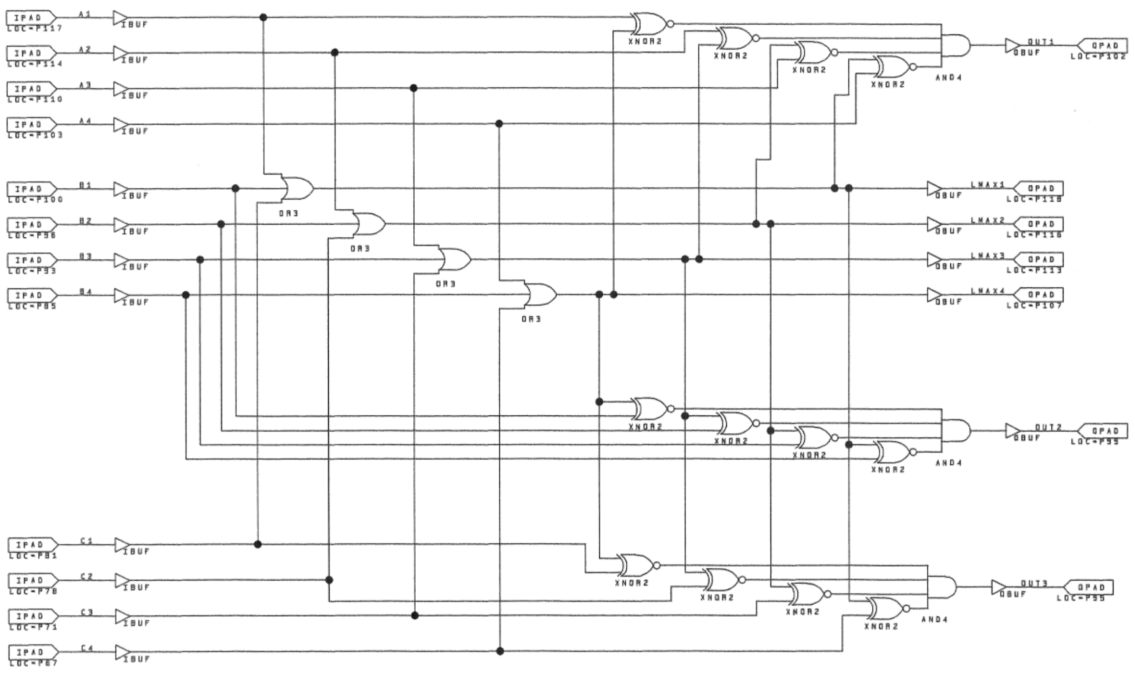


Figure 3.11 Willshaw Thresholding

The Willshaw threshold is a key element in weightless neural networks, often used to select the data stream which most closely represents the desired request. Austin's ADAM is a typical example which can use the Willshaw thresholding technique: it can be used in the class separator pattern to associate the two correlation matrix memories [Austin et al 1987]. The operation of the logic is to take all the separate data streams and bring them together to determine the maximum data response. This is then compared with each of the separate data streams, any data streams which correspond have their output (out *n)* set to '1'. There will be at least one output which is set to '1' due to the way the Willshaw threshold is derived. In practice when used in a neural network the user is unlikely to want to know the Willshaw value, hence the Willshaw outputs (*LMAX n)* would not be brought out from the logical structure. They are only present in this design to show where the Willshaw value can be found and for test and evaluation purposes. The Willshaw threshold is used by Austin in his ADAM network to determine the result from the matrix. It is used to select the rows which have the highest response. Conventionally, Austin performs this function with an arithmetic summation of the columns and storing the highest response which is then applied to all the columns as the threshold.

3.12 A Weightless Boolean N Point Threshold Element

N point thresholding is often referred to as *L-Max* thresholding [Willshaw et al 1970]. N point thresholding is used in the class separator pattern to select the columns with highest correlation score. The architecture in Figure 3.15 allows N point thresholding to be implemented using Boolean weightless logic, instead of using conventional clocked counters as in Austin's implementations. ADAM uses N point thresholding to select the columns of the second matrix using the class pattern as the comparison [Austin 1986].

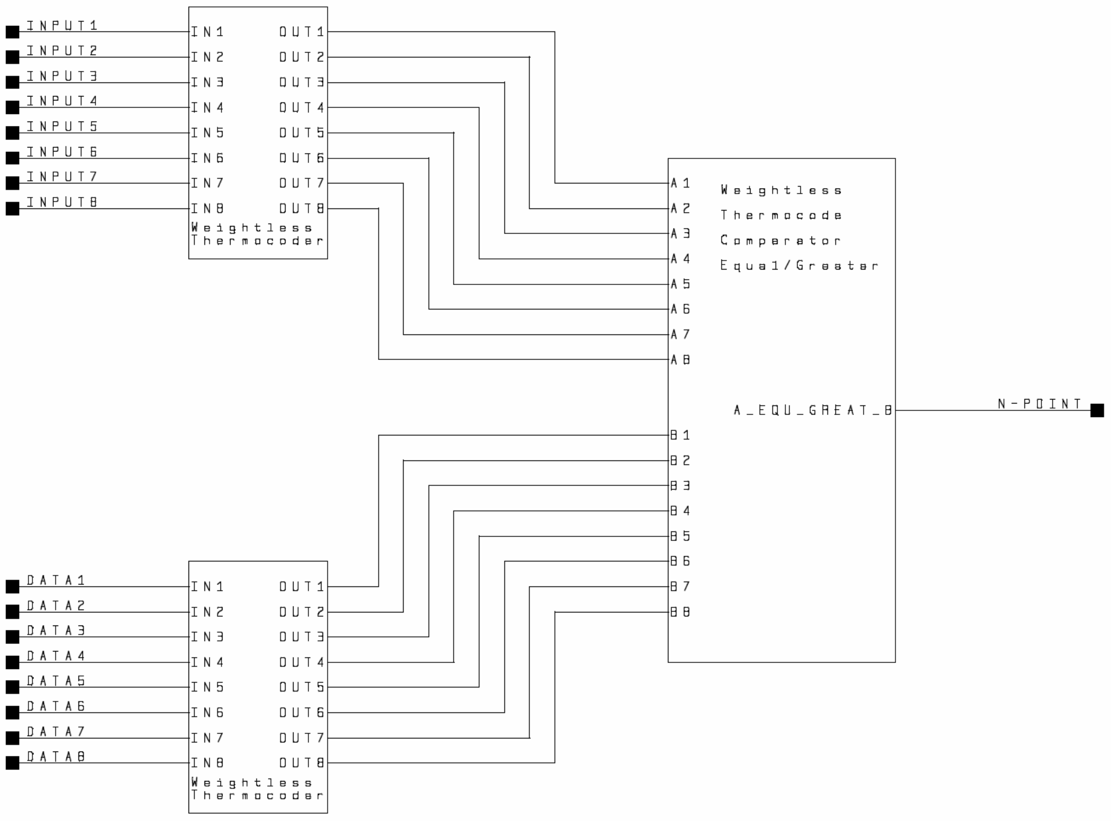


Figure 3.12 Weightless N Point Thresholding

3.13 Implementation of a Weightless Boolean ADAM Architecture

The majority of ADAM is based on weightless Boolean logic including the tupling and the correlation matrix memory. The areas highlighted in Figure 3.13 by the dotted lines show the thresholding elements of ADAM which are not weightless. These elements can be replaced with the elements shown in Figure 3.11 and Figure 3.12 in order to implement a fully weightless hardware Boolean architecture.

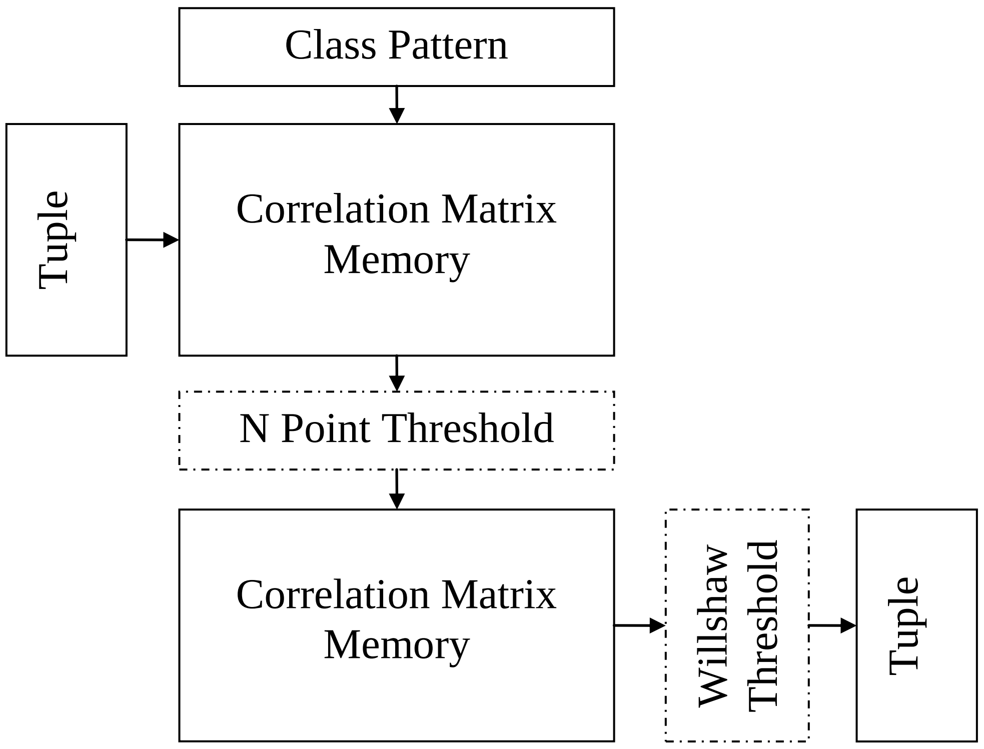


Figure 3.13 A Block Diagram of ADAM

An example of how the Willshaw thresholding weightless Boolean element can be implemented and added to the correlation matrix memory is shown in Figure 3.13. To implement this function within ADAM using the correlation matrix memory each of the rows within the matrices are individually thermocoded. The Willshaw threshold is generated by the logic shown in Figure 3.11, each of the thermocoders is then compared with the Willshaw results using a greater than or equal to function and this determines the output for each row of the matrix.

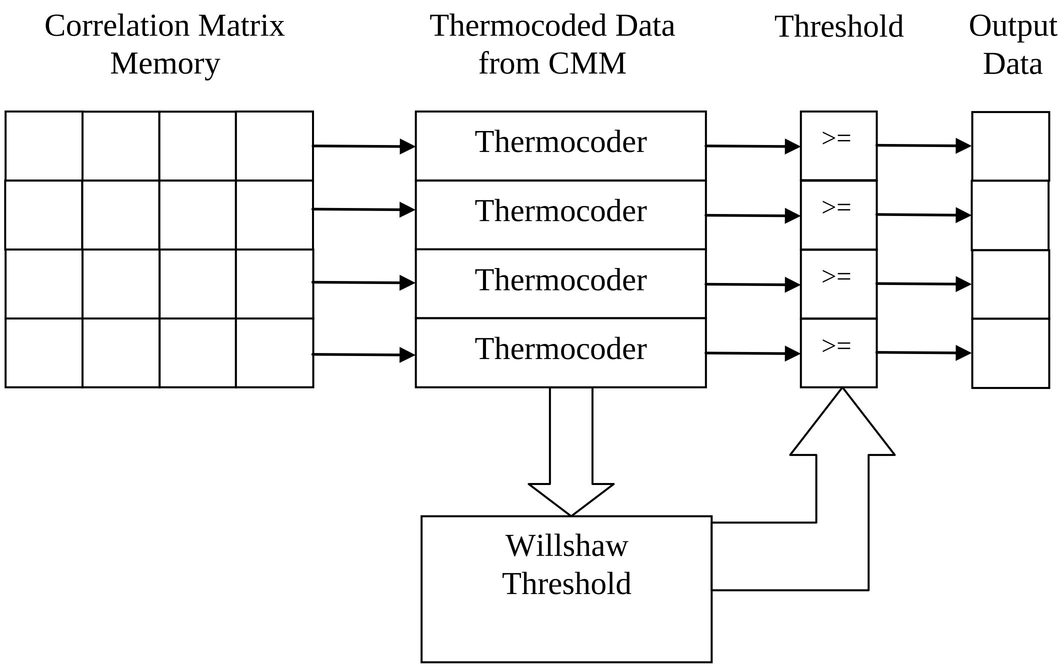


Figure 3.14 A Diagram of the Weightless Boolean Willshaw Threshold on a CMM

Similarly the implementation of the N Point threshold is shown in Figure 3.15. This technique is usually used on the class pattern rather than the input data in the implementation of ADAM however it can be used on the input data as well. The summation is performed by the thermocoder and the threshold applied by the greater than or equal to logic function. A full description of the logic is given in Figure 3.11. Each of the rows in the matrices are also separately thermocoded and individually compared with the thermocode input data each producing a one bit result for the output of either zero or one dependent on the data.

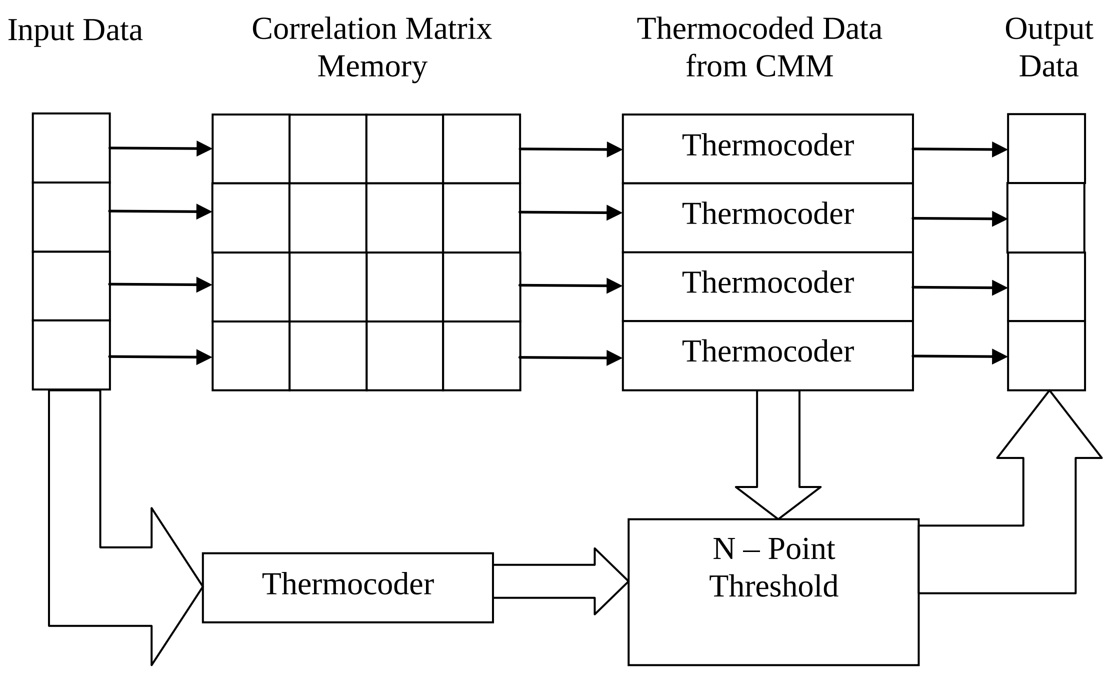


Figure 3.15 A Diagram of the Weightless Boolean N point Threshold on a CMM

3.13.1 Simulation of the ADAM Network

An example of the operation of the weightless Boolean ADAM network is presented in Figure 3.16. The example is based on a pre-trained ADAM network which does not have tupling applied to the inputs. Figure 3.16 shows a pre-trained data pattern being shown to the trained network and the operation of the network in recalling the associated pattern. The weightless thresholding techniques used in the network are shown in more detail in Figure 3.17 and Figure 3.18.

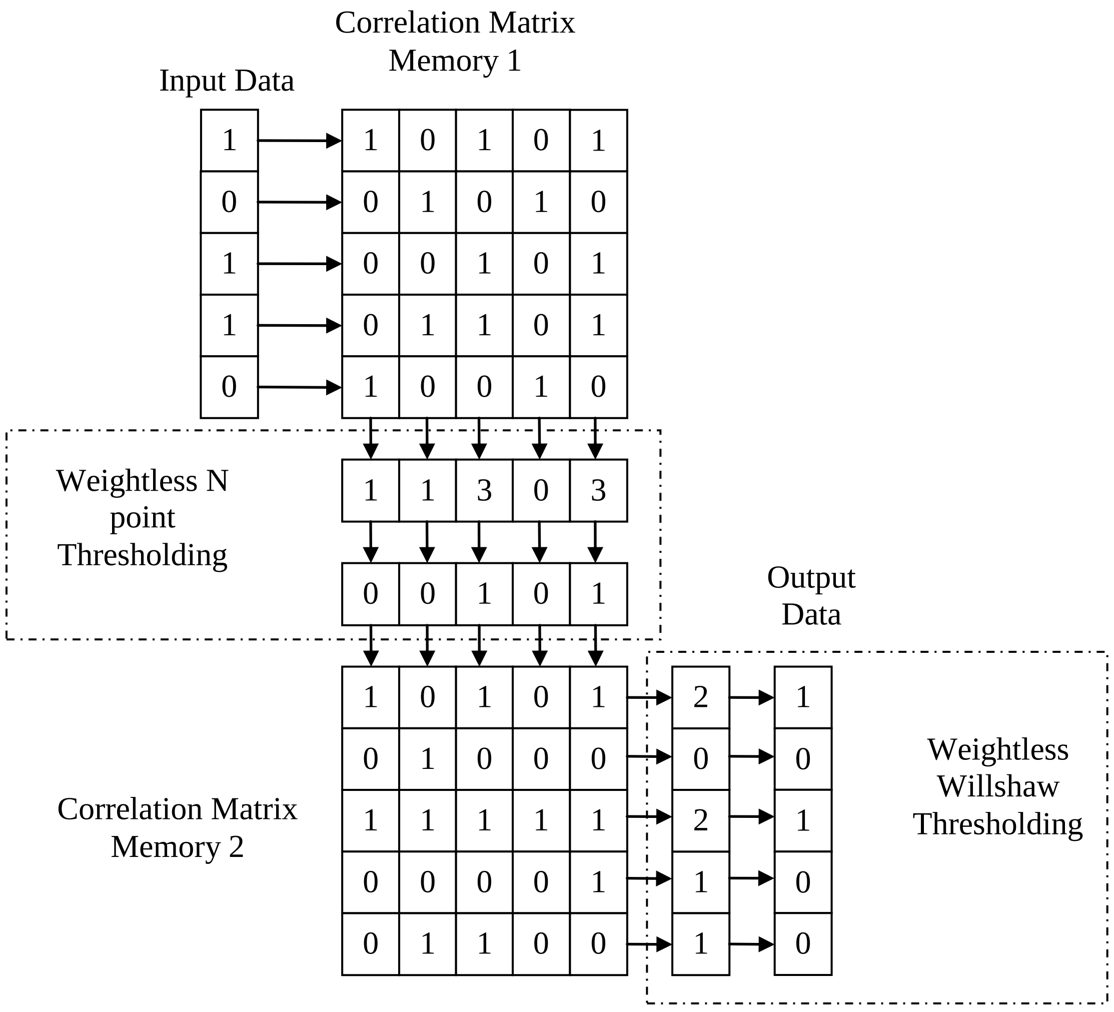


Figure 3.16 A Weightless Example of the ADAM Network

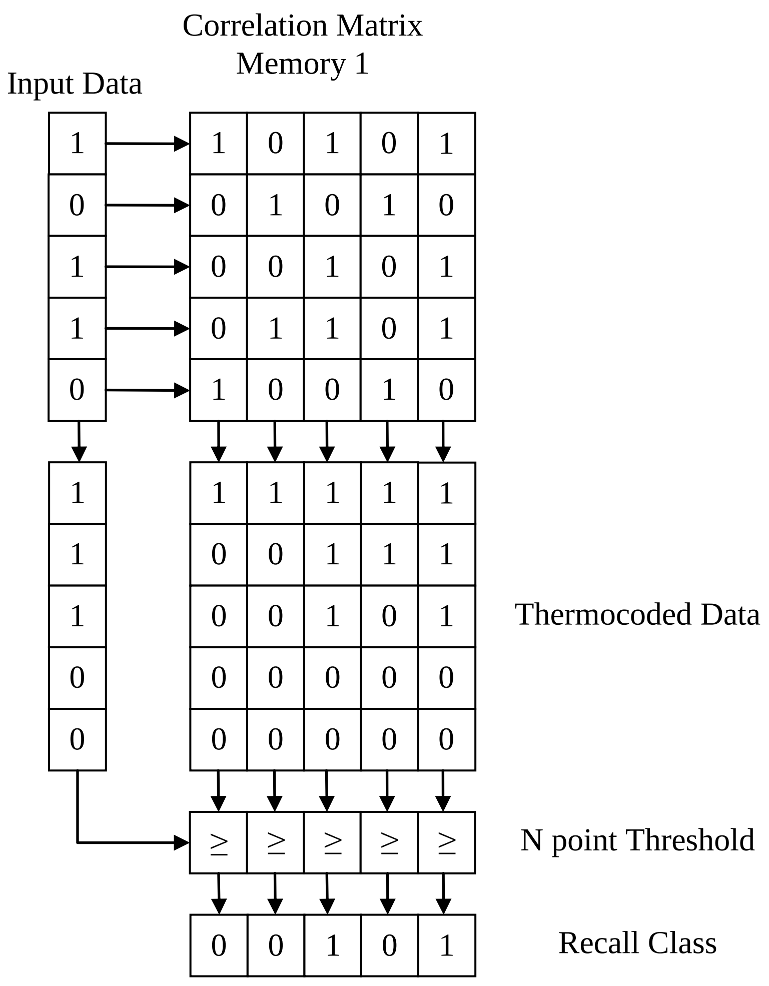


Figure 3.17 A Weightless Example of N Point Thresholding

The implementation of weightless N point thresholding is shown in more detail in Figure 3.17. N point thresholding is used to determine the class pattern from the input data on a trained network. N point thresholding is implemented by thermocoding the input data and comparing this with the thermocoded response from the columns of the correlation matrix memory neurons which have fired as shown in the thermocoded data in Figure 3.17. The greater than or equal to logic is then used to compare the thermocoded columns with the thermocoded input data. In this example this produces the class pattern 00101 for the given input of 10110.

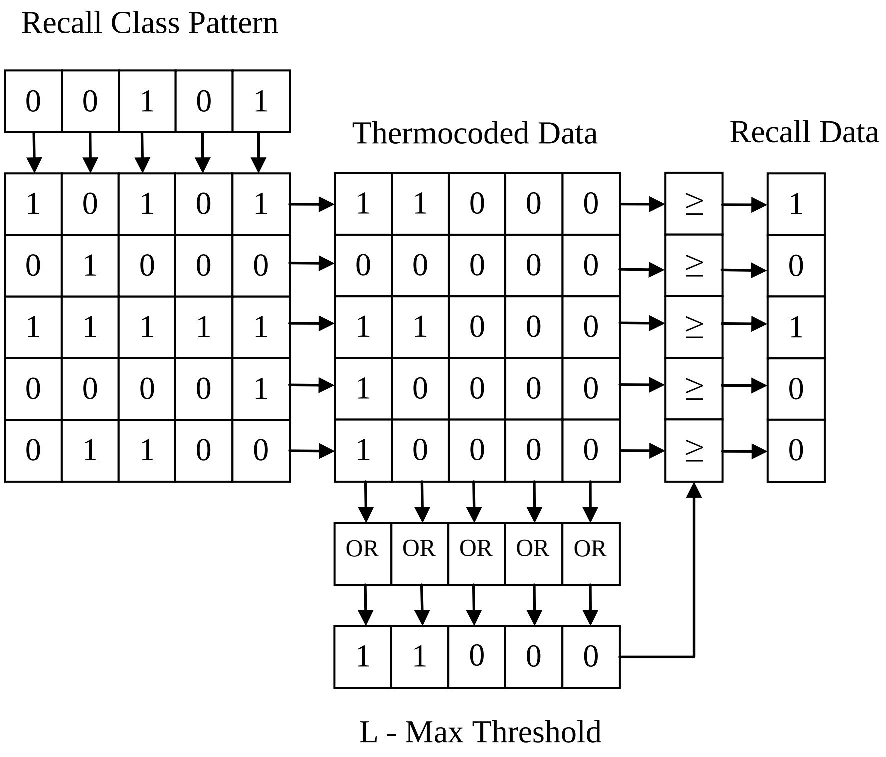


Figure 3.18 A Weightless Example of Willshaw Thresholding

Willshaw thresholding is used on the second correlation matrix memory to determine the output from the presented class pattern as shown in Figure 3.18. The class pattern in this example is 00101 which is presented to the second trained correlation matrix memory. Each of the neurons which respond are then thermocoded horizontally as shown in thermocoded data. The thermocoded data is then Willshaw thresholded by vertically ‘OR’ gating the columns to determine the threshold to be applied which is equal to the maximum response in this case 11000. The threshold value is then used to compare with each of the thermocoded rows using the greater than or equal to logic this produces the Recall Data in this example 10100.

The operation of the non-Hebbian ADAM network is identical apart from the fact that instead of the neuron in the matrices firing and producing a ‘1’ when two inputs fire a ‘1’ is only produced when the two inputs do not fire and both input conditions are ‘0’.

3.13.2 Simulation of the ADAM Network in Appendix D

Attached in Appendix D is a simulation of the new learning elements. Multiple simulations were developed during the research. The simulation program demonstrates the ability of the Hebbian and novel non-Hebbian techniques to implement the ADAM neural network. The simulation produces random matched data words, which are taught into the matrix. Then a random trained word is picked and presented to the trained matrix and its recall word and the output is then scored and thresholded against the trained data. The simulation showed that using both Hebbian learning and non-Hebbian learning the matrices had the same performance. In summary a collection of non-Hebbian learning elements has been proposed. These are based on other logical functions which will operate as the neural function at the intersections within the matrix.

3.13.3 Summary of Chapter 3

A new hardware Boolean weightless implementation of Austin's ADAM neural network has been proposed using a collection of weightless Boolean sum and threshold architectures developed by the author. The main consistuent of these are weightless implementations of decision logic including a Willshaw thresholding element [Willshaw et al 1969]. A novel method for thermocoding weightless serial data which has been patented is demonstrated. The test philosophy used to trial these new elements is presented including a description of hardware used. A novel method for thermocoding weightless serial data which has been patented is demonstrated. This chapter demonstrates how weighted neural networks can be converted in Boolean weightless hardware using a collection of Boolean weightless architectures.

Chapter 4

4 Hardware Weightless Boolean Median Filters

4.1 Overview

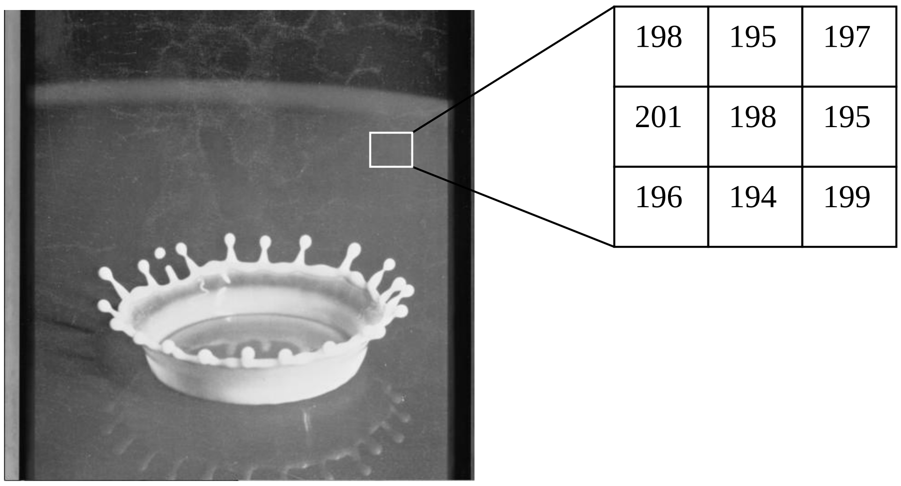
This chapter presents a collection of novel weightless Boolean median filters. A description of the operation of each of the filters is presented. The robustness of a weightless Boolean logic structure used in the weightless median is examined when subjected to a high neutron flux environment, typically experienced at avionics altitudes. The performance of the novel weightless Boolean swap element which is part of the weightless median filter is compared and evaluated with a conventional median filter swap element. Both elements are examined when implemented on a standard FPGA which is subjected to a high neutron flux environment at ground based accelerator facilities. The results from these trials are analysed and discussed.

4.2 Median Filtering

Median filtering is a technique that is used to remove certain types of noise from an image. It is particularly effective at removing outlier noise such as Salt-and-Pepper noise. Salt-and-pepper noise occurs when pixels within the image is corrupted resulting in them becoming either a minimum value or a maximum value. In greyscale images these pixels become either black or white hence the term salt-and-pepper noise.

4.2.1 Operation of the Median Filter

The median filter is implemented by passing a window in which the filter operates over the whole image in sequence as shown in Figure 4.1.



198	195	197
201	198	195
196	194	199

Figure 4.1 Milk Drop Image with 2 Dimensional 3 by 3 Matrix Window

As the filter window is passed over the image, adjacent pixel values are used to calculate a replacement pixel value for the centre pixel. This operation forms the basis of many filtering techniques. The type of filtering executed is dependent upon the replacement algorithm. Median filtering is performed by ordering all the pixel values in magnitude and replacing the centre pixel value with the middle value. A conventional median filter implementation would use an arithmetic logic unit found in a processor to sequentially compare and order the values.

4.3 Novel Techniques for Median Filtering

Two novel weightless means of performing this median filtering using weightless Boolean hardware are described below.

The first technique allows weightless data to be ordered according to magnitude. This technique was developed by King with the addition of a tagging technique proposed by the author. The tagging technique allowed weightless non-thermocode data to be maintained during the ordering technique.

The second technique is a novel weightless Boolean median filter which utilises a weightless Boolean swap block. This technique was developed as part of this research following on from the initial work investigating King's Type 1 Neuroram as an image filter.

Both these techniques have been patented by BAE SYSTEMS. Copies of the original patent can be found in Appendix C.

4.3.1 Weightless Ordering

King and the author jointly hold a patent on weightless ordering and tagging [Armstrong et al 2003]. King's contribution was the ordering method and the author’s contribution was the tagging method. The filter is implemented by first taking the pixel values from the window function and converting them into a weightless binary string. This can be implemented relatively easily in hardware again without the need for a clock or processor. The implementation is simply a case of thermocoding the data horizontally and then thermocoding all the data vertically. Figure 4.2 show three values of weightless data each four bits in length which represent typical input data from a function window.

0101
0100
1101

Figure 4.2 Typical Input Data

The first stage is to thermocode each string of weightless binary data which represents a pixel value from the window function. This operation is performed using King's tuple ordering method [King 1999a]. The thermocoding is then performed on all the pixel values from the processing window this then gives the data as shown in Figure 4.3.

1100
1000
1110

Figure 4.3 Weightless Data after Horizontal Thermocoding

The second stage is to order the thermocode values into magnitude order. This operation can be performed by thermocoding the data in the vertical direction as shown in Figure 4.4.

```
1000
1100
1110
```

Figure 4.4 Weightless Data after Horizontal and Vertical Thermocoding

Once all the vertical lines have been thermocoded the data is now in magnitude order. To perform the median function is now a case of taking the middle value and using this to replace the central value in the window after converting the thermocode value back to binary. This technique is a weightless hardware method which can be further enhanced using the parallel thermocode described later in this chapter.

4.3.1.1 Shortcomings of Weightless Ordering

The disadvantage with this technique is that the correspondence to the original data is lost. The original data is lost as although the Hamming value of the original data is maintained the original order and bit positions are lost through the process.

4.3.1.2 Contributions of the Tagging Method

The tagging method addresses these shortcomings by associating the original data with a tag. These tags were formed from the original data. Once the vertical thermocoding has been performed the resultant ordered thermocodes are compared with tags and where the thermocodes are the same length the original data replaces the thermocode; hence the original data is ordered. The technique does have a drawback that if several numbers have the same thermocode length the original data position for the same string may be lost as shown in
Figure 4.5.

1011

0101

1010

0100

May yield

1011

1010

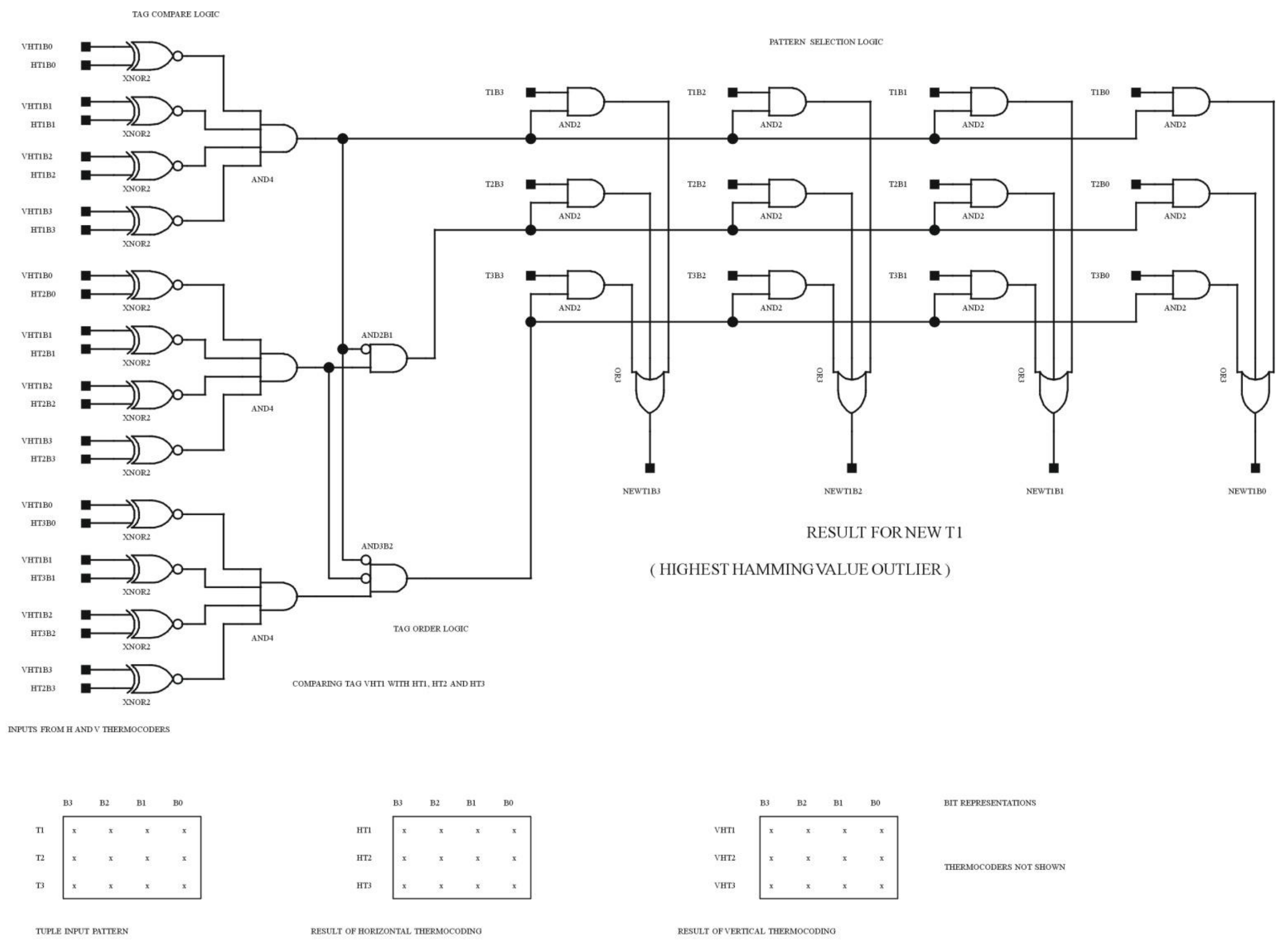
0101

0100

Figure 4.5 An Example of Wrong Positional Assignment

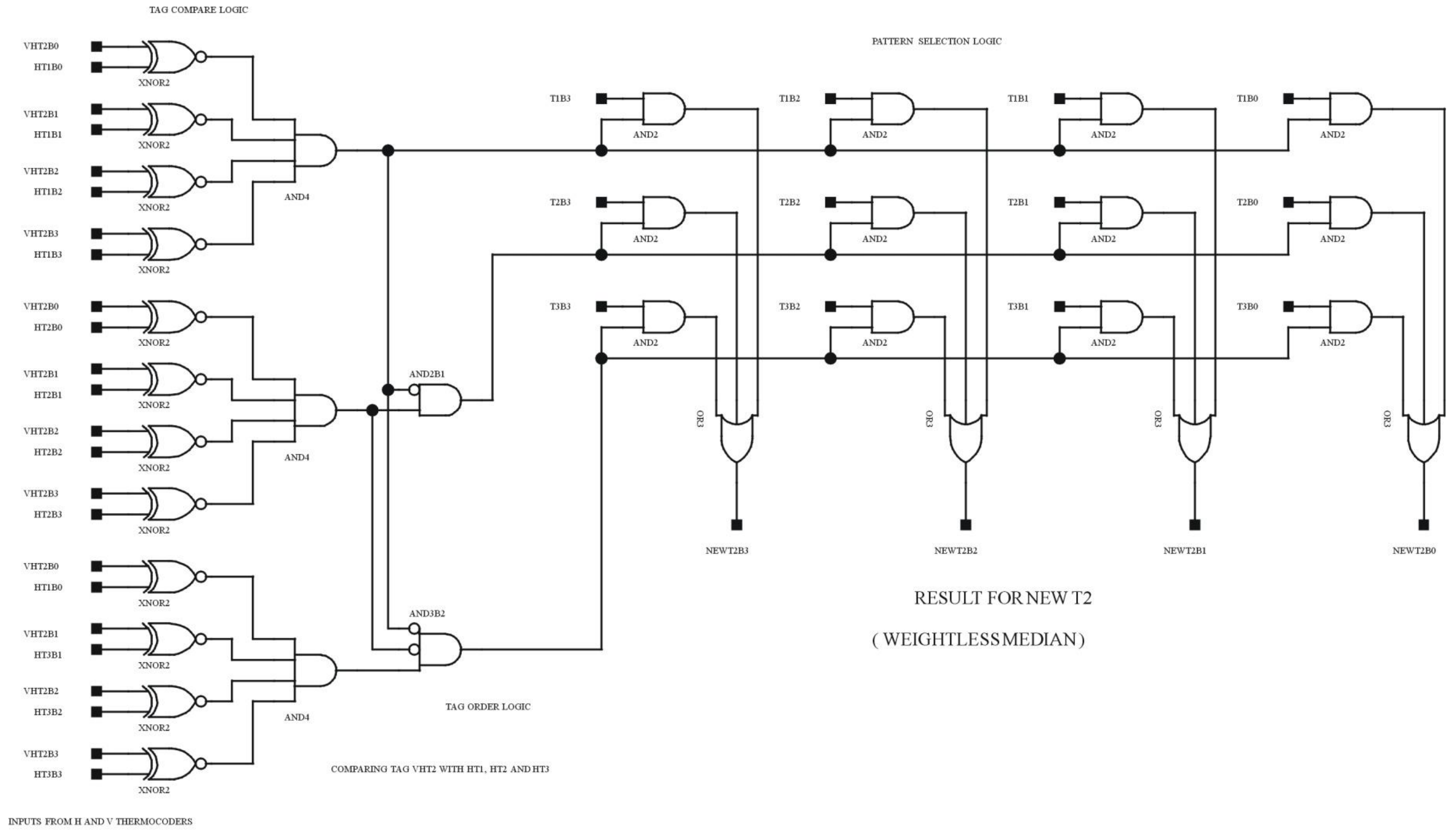
4.3.1.3 The Implementation of the Tagging Technique

The logic for performing the comparison and tagging is given in Figure 4.6, Figure 4.7 and Figure 4.8. The comparison for the tagging technique is made using 'EXNOR' gates to compare the thermocode data from all the vertical and horizontal thermocoders to each of the original dataset. The graphical description of this is given at the bottom of the circuit diagram in Figure 4.6, this also gives a description of the labels used within the circuit diagrams below.



WEIGHTLESS HAMMING VALUE ORDERER, OUTLIER AND MEDIAN EVALUATOR, PART 1

Figure 4.6 New Result for the First Value



WEIGHTLESS HAMMING VALUE ORDERER, OUTLIER AND MEDIAN EVALUATOR, PART 2

Figure 4.7 New Result for the Second Value

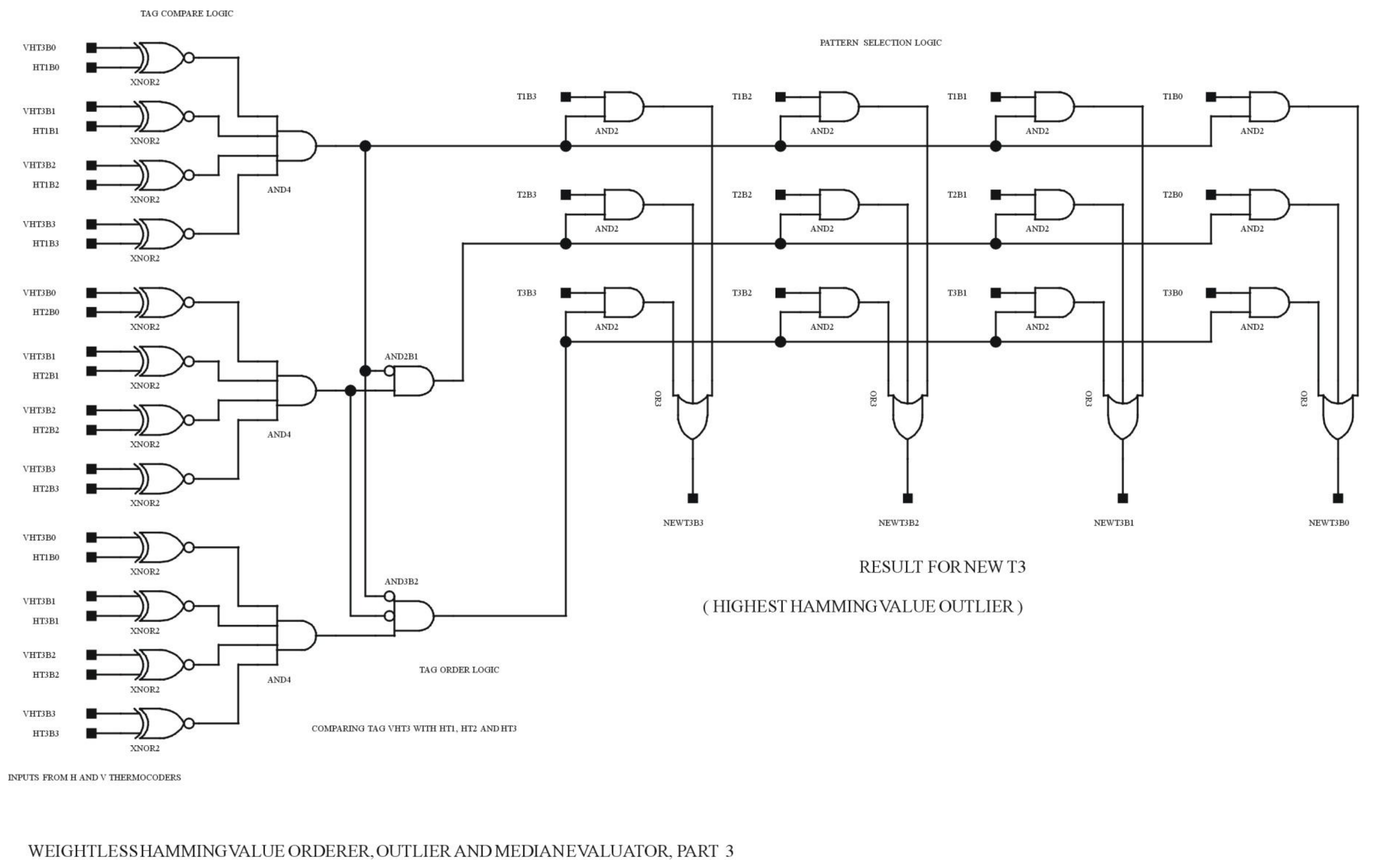


Figure 4.8 New Result for the Third Value

4.4 A Novel Boolean Weightless Median Filter

This technique is based on a more conventional implementation of median filtering. It is also designed to overcome some of the disadvantages faced by the previous ordering and tagging technique. The previous technique requires two stages of thermocoding which with large datasets can be timely. This technique in comparison only requires the data to be thermocoded initially; the only subsequent delay within the system is caused by the swap block. These blocks however are designed to operate in parallel. Although the structure to implement a median filter is still sequential the delays incurred would be less than when a large dataset is used with the previous technique. The operation of this filter also works by the ordering of the original data so the median, highest or lowest values can be found. This is performed without any loss of positional data unlike the previous method. Again the implementation is fully Boolean logic based so the system is totally asynchronous. This weightless median filter operates on similar principles to the previous filter, in that the weightless data is thermocoded and compared. The difference is that the original data is preserved throughout the process. This is achieved by comparing the two weightless values and then thermocoding them whilst still preserving the original values. If the second value has a greater thermocode value than the first of the two original weightless values then they are swapped over. If the first value has a greater thermocode value then the original data it is passed on to the next stage as shown in Figure 4.10. This process deals with the comparison of two weightless codes. If greater numbers of codes need ordering it is the cascade architecture which is important to the operation of the median filter. Figure 4.10 also shows the logical implementation of this method which includes a thermocoder in the structure. However if the data is already thermocoded, from either a previous stage or the input data is thermocoded, this is not necessary. The thermocoder shown in Figure 4.10 is King's thermocoder. The next section discusses the use of an alternative thermocoder which would result in a lower propagation delay. Figure 4.11 gives a typical structure with minimal swap elements to implement a nine input median filter, unnecessary data is removed to reduce the number of elements required. The structure is to be designed so there are enough stages to ensure that the data is completely organised. Figure 4.9 gives an example of a suitable structure given three weightless input strings where all data is retained.

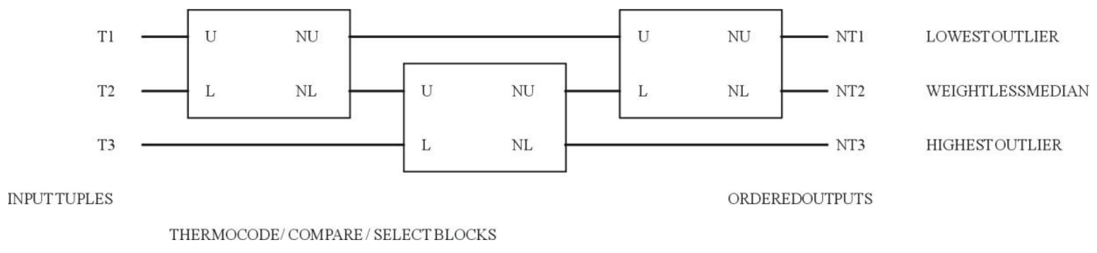


Figure 4.9 An Example of a Weightless Type 2 Median Filter Architecture

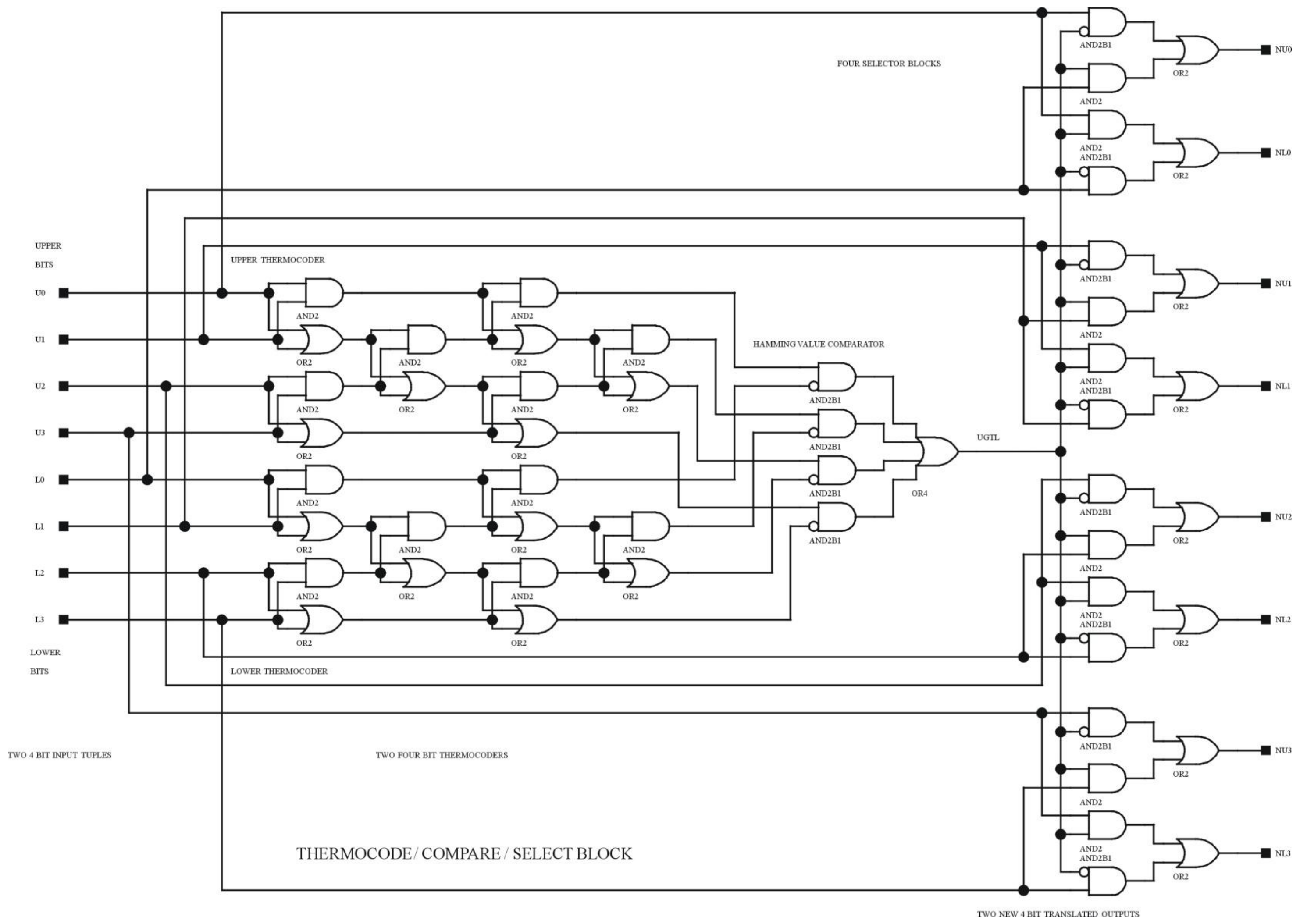


Figure 4.10 Logic Diagram of a Weightless Swap Block

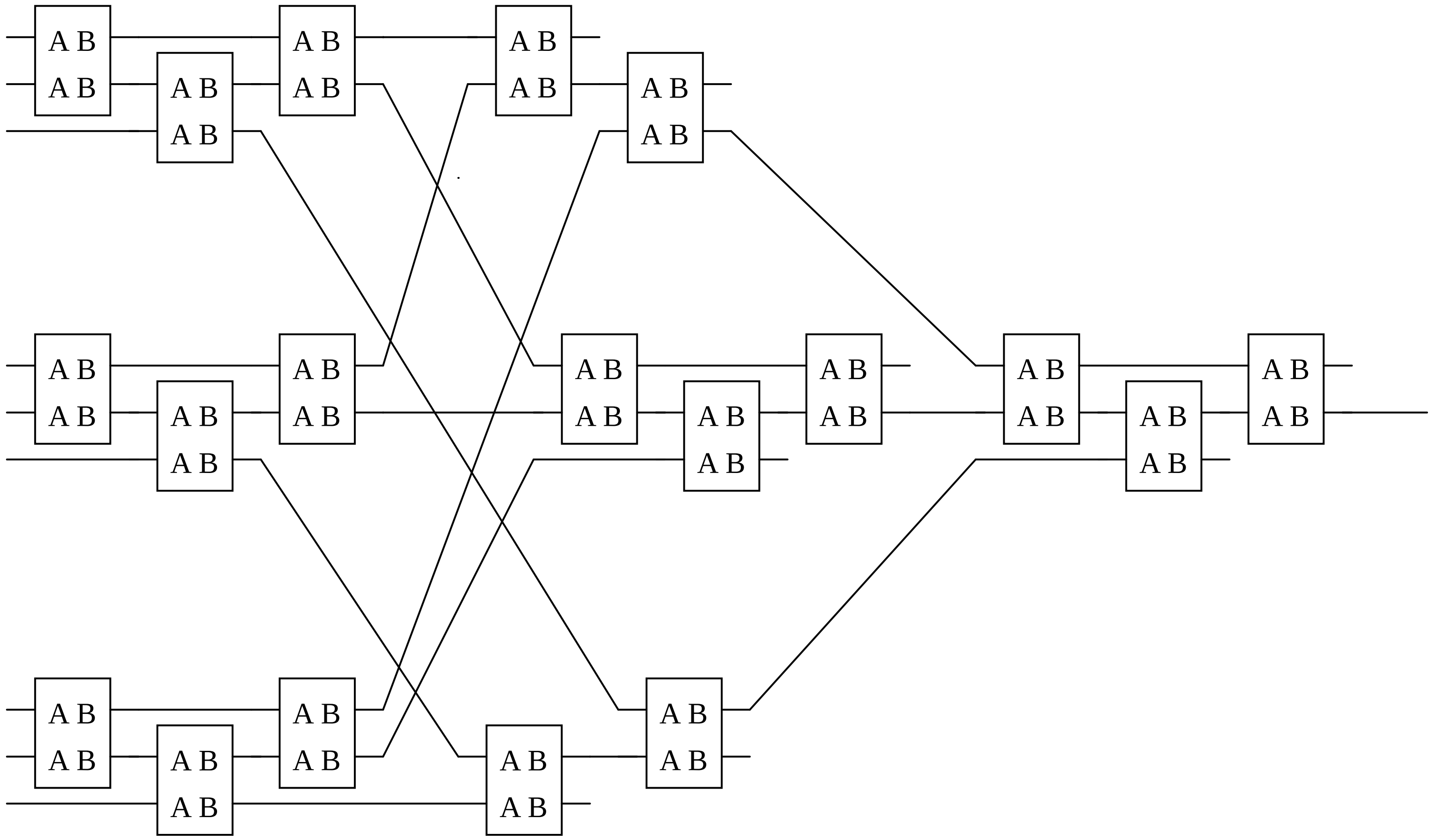


Figure 4.11 Minimal Weightless Median Filter Architecture

4.5 A High Speed Weighted Binary to Thermocode Converter

In order for the weightless median filter to be a practical replacement for a standard median filter it is necessary for it to operate at high speed as many applications which will benefit from its robust properties also require high speed processing such as medical imaging [Hazra et al 2004]. The nature of the median filter prevents it from taking advantage of parallel methods of operation. Therefore it is necessary to reduce the propagation delay of each stage of the filter to a minimum. Modern FPGA's such as the Xilinx Virtex II range have a gate propagation delay of approximately 1 ns. The example in Figure 4.10 shows an implementation of King's weightless thermocoder [King 2000]. This is presented with two 4 bit weightless binary values. These are operated on by two input thermocoders arranged in the necessary structure as defined by King to ensure that all bits are thermocoded [King 2000]. Using two bit thermocoders the logic structure needs the same number of layers as inputs. In this case there are four inputs and hence four layers of logic equating to a total propagation delay for the thermocoder of 4 ns. The subsequent Hamming value comparator and swap block elements would comprise four layers of logic hence another 4 ns propagation delay. The total propagation delay for this weightless Boolean swap block would be 8 ns, subsequent swap blocks do not need thermocoding and hence will only have a propagation delay of 4 ns. If this is applied to the structure shown in Figure 4.10 this would give the overall filter using two sets of four inputs as shown a total propagation delay of 40 ns. This equates to an equivalent clock speed of 25 MHz. This could further be improved by the Xilinx software which would compile the logic reducing the number of logic layers. A similar weighted version of the median filter used in medical imaging is capable of a clock speed of 48 MHz [Bates et al 1997]. The element presented is very basic and a more realistic median filter would be expected to operate with nine inputs all consisting of 8 bit weighted data which requires a 256 bit weightless tuple. If the data is presented in weightless binary format, in order to be thermocoded using an equivalent architecture each input would take 256 ns. This is because each additional weightless input bit increases the number of layers of logic proportionally. All the inputs could be dealt with in parallel. The subsequent swap elements would take approximately 4 ns each, using the architecture in Figure 4.11. The propagation delay of all the swap elements would be 36 ns as there are nine elements in series. The overall propagation delay of the weightless median filter would be 392 ns, equivalent to a clock speed of approximately 2.5 MHz. This is not suitable for the more demanding applications such as medical imaging [Hazra et al 2004]. The main constituent of the delay is the pre-processing, as the original data is likely to be standard weighted binary.

4.5.1 A Parallel Weighted Binary to Thermocoder

Therefore a replacement to King's thermocoder is proposed, that of a simpler parallel conversion method. This method creates a logic array for every bit in the weightless thermocode string. Although the number of layers of logic would increase with larger data widths the rate of growth is significantly less than that proposed by King [King 2000]. This is demonstrated in Table 4.1 below where using King's implementation this would require seven layers of logic in this method only requires one layer.

Value	Inputs			Outputs						
Logic	A	B	C	A.B.C	A.B	(A.C+A.B)	A	(A+B.C)	(A+B)	(A+B+C)
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	0	1	1
3	0	1	1	0	0	0	0	1	1	1
4	1	0	0	0	0	0	1	1	1	1
5	1	0	1	0	0	1	1	1	1	1
6	1	1	0	0	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1

Table 4.1 Operation and Boolean Logic of the Binary to Thermcode Converter

A thermocoder operating on a weightless tuple of 256 bits where the logic gates can cope with 8 inputs would equate to 3 layers of logic with an overall propagation delay of approximately 3 ns. Additionally the swap element logic can be optimised using the FPGA manufacture’s software reducing the swap element propagation delay time to approximately 2 ns per layer. Thus giving the median filter an overall propagation delay of only 21 ns. This would equate to a clock speed of approximately 47 MHz which would be suitable for high speed operations. As new faster FPGA’s are produced this operational speed can increase as device propagation delays through logic gates fall.

4.6 Evaluation of a Weightless Swap Element in a High Neutron Flux Environment

The aim of the testing was to investigate the hypothesis that different Boolean logic implementations of standard elements will have improved abilities to withstand corruption. The logical elements chosen for this trial were a subset of the patented weightless median filter and a compatible weighted element taken from an FPGA implementation of a weighted median filter [Xilinx 1998]. The author understood that the underlying logic implementation is static random access memory based look-up tables.

4.6.1 Initial Trials with the Median Filter

Initial investigations into the performance of the two architectures were focused on the complete median filters. A brief summary of the testing performed on these filters and the information gained which led to the final trial of the sub elements of these filters is presented in this section.

The median filters used were designed with 5 inputs each, 4 of which were fixed values within the FPGA all with a decimal value between 0 and 11. The range of values was limited by the number of inputs and outputs available on the test equipment which was 22. Therefore with the weightless architecture requiring one input and output this limited the maximum number of values to 11. In order to make it a comparative test the values of the weighted architecture were also limited to a maximum value of 11 which translates into 4 weighted binary inputs and outputs.

The values used were carefully chosen to give a selection of results given the input value, so that the filters could be cascaded and the result would ripple down through to the final output. Therefore the filter values were set to 2, 3, 5, 7 this then corresponded to an output of 0-3 for any input value up to 3, 4 for 4 and an output of 5 for any input value from 5 to 11.
The median filters were initially implemented in a Xilinx Virtex IV device on a Xilinx test board but this proved to be more resilient to corruption by neutrons than previous Virtex devices. Therefore no real results were gained at this initial trial.

Following an investigation it was determined by Xilinx that the Virtex II part was nearly twice as susceptible to corruption compared with the Virtex IV part and was the most susceptible device in the Virtex range. Because the experiment was investigating the robustness of the architecture, it required the least robust device in order to increase the data for better statistics. Additionally the new device was filled with as many filters as would fit in, to increase the number of errors and again improve statistics.

The code was recompiled for the Virtex 2 device. The Rutherford Appleton Laboratory offered SPAESRANE the opportunity to evaluate their facility in Harwell to compare it to the other atmospheric neutron simulation facilities. This allowed the new Virtex II architecture to be trialled and acted as a proving ground before the TSL trial.

This immediately exposed one flaw in the test system design, in that once the Virtex IV device became so corrupted that it produced a constant error there was no method of detecting this and reprogramming the device. As the test set was designed to be autonomous the hardware and software were updated with an additional signal to reprogram the device with its configuration, when so many instances of the same error in succession were detected. This latest version of hardware and software was taken to TSL where three median architectures were trialled: weightless, weighted unclocked and weighted clocked. On evaluation of the results it was decided to concentrate on the weightless unclocked architecture versus the weighted architecture.

Median filters have the property to remove outlier noise. In this case this would include spurious results which we would want to capture. This was only compounded by the fact the architecture was made up of a chain of median filters.

4.7 The Weightless and Weighted Swap Element Trial

Therefore for the next trial at TRIUMF it was decided to pick a smaller element of the median filter which would not exhibit the outlier removal properties of the median filter; notably the swap block. This element was also the main part of the median filters responsible for their functionality.

4.8 Test Philosophy

In order to test the architecture it was necessary to cause single event upsets in the FPGA that the architecture was being implemented in. In order to achieve a single event upset suitable ground-based neutron facilities were identified, these being the Theodor Svedberg Laboratory (TSL) in Sweden and the Tri-University Meson Facility (TRIUMF) facility in Canada.

4.8.1 Test Facilities

4.8.1.1 Theodor Svedberg Laboratory

TSL is located in Sweden at the University of Uppsala. It was built in 1947 and is one of the earliest facilities of its kind. A synchrocyclotron is used to generate high-energy protons which are directed onto a tungsten target. This results in a simulated atmospheric spectrum with a maximum energy of 180 MeV. This is also known as a white spectrum beam and is named as the ANITA beam which stands for **A**tmospheric-like **N**eutrons from th**I**ck **TA**rget [Prokofiev et al 2009]. The beam has been characterised by Torok and Platt and compared to similar facilities [Torok et al 2006]. This beam is available in the Blue Hall and locked to prevent exposure to staff and researchers whilst the beam is on.

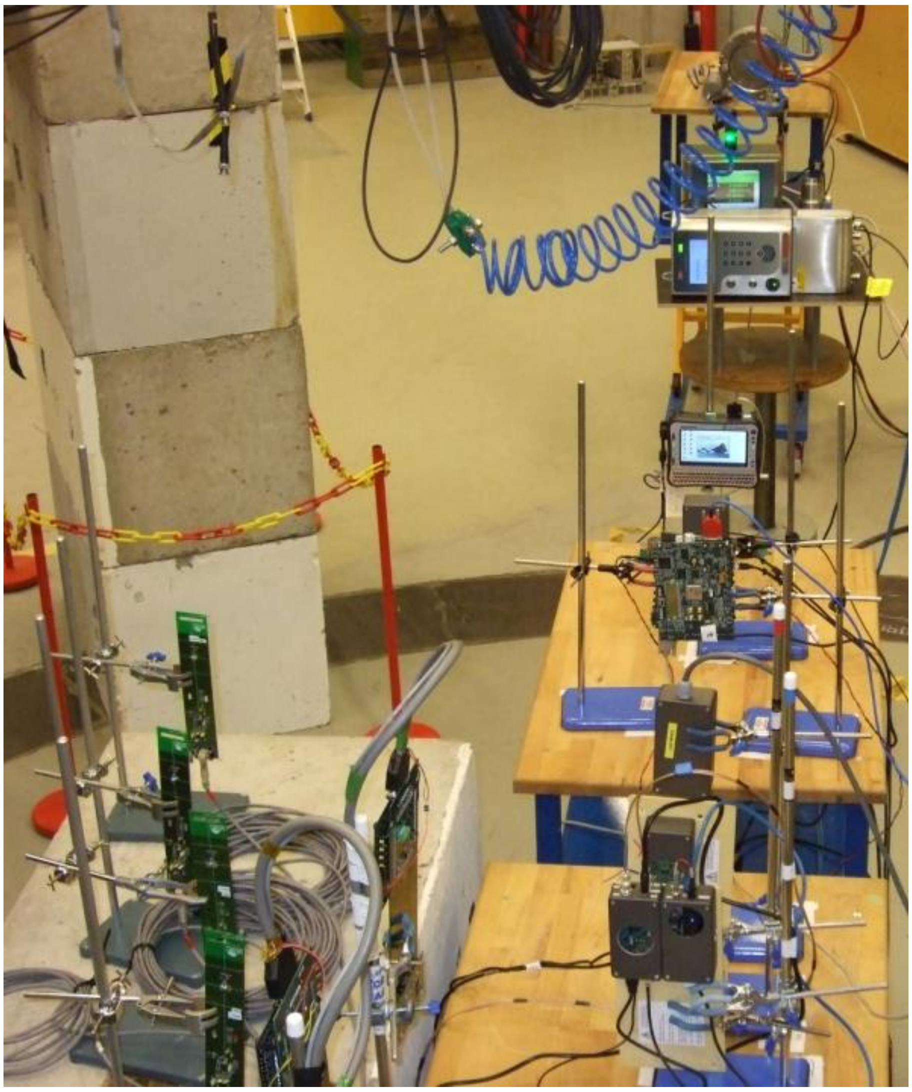


Figure 4.12 TSL Blue Hall – Looking Down the Beam Line

There are two “counting rooms” where experiments can be remotely monitored. The main connections between the two are via BNC connections to the Blue Hall. One counting room is shown in Figure 4.13. The set up of the experiment for this facility is shown in the block diagram in Figure 4.14.

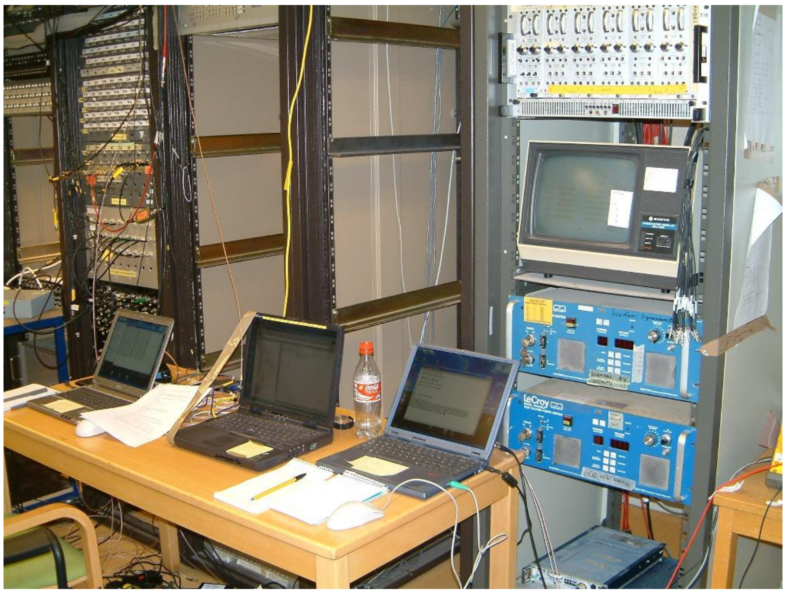


Figure 4.13 One of the Counting Rooms at TSL

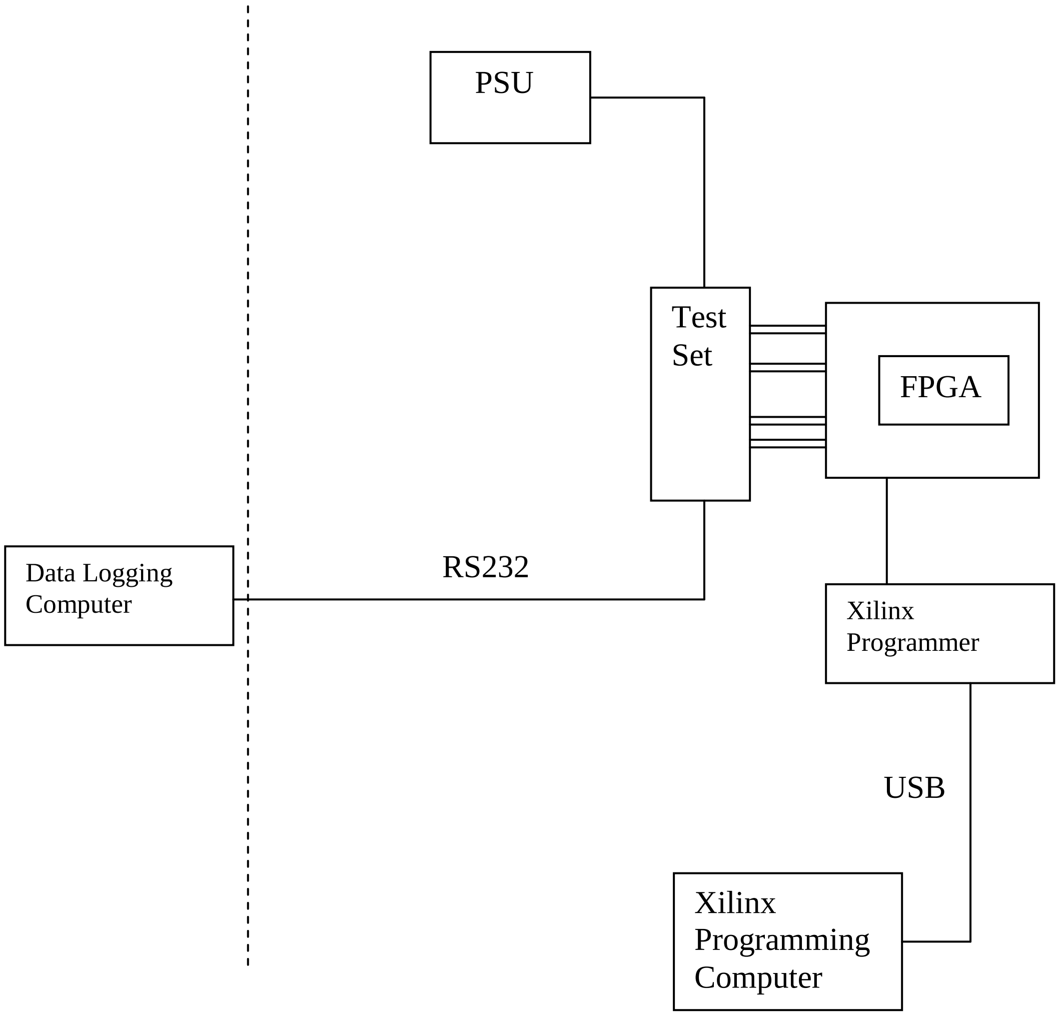


Figure 4.14 A Block Diagram of the TSL Test System Layout

4.8.1.2 Tri-University Meson Facility (TRIUMF)

TRIUMF is located at the University of British Columbia, Vancouver, Canada as shown in Figure 4.15.

Figure 4.15 TRIUMF at the University of British Columbia

The synchrocyclotron is capable of producing, on average, 10^{15} protons every second with energies ranging from 450 to 500 MeV.

Figure 4.16 TRIUMF NIF Buried in the Isotope Production Facility

A well defined neutron beam of approximately 8 by 12 cm is provided in the Neutron Irradiation Facility (NIF) with a similar spectrum to that of the atmosphere. [Blackmore et al 2003]. The neutron flux is one million times that at 39,000 feet for flux >10 MeV and consists of many thermal neutrons. These can easily be removed with cadmium plates; this trial used the thermal neutrons so no cadmium shield was required. This beam is characterised from thermal energy to 500 MeV. The fluence is determined by integrated proton beam current calibrated against activation foil measurements. The beam is located beneath a counting room and accessed using a slider plate via wire and guide rails. The test system having to be lowered on a plate as shown in Figure 4.17. The test set was modified as shown in Figure 4.18. A block diagram for the set up is shown in Figure 4.19.

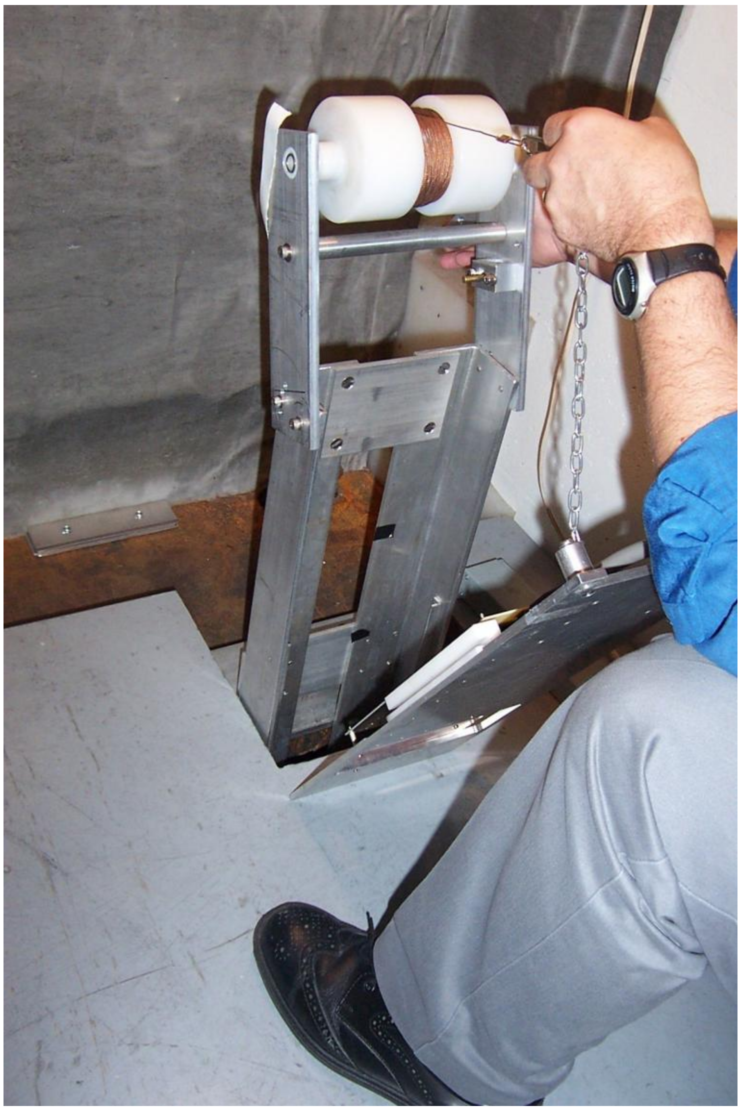


Figure 4.17 TRIUMF NIF Slot and an Example Test Board on Slider Plate

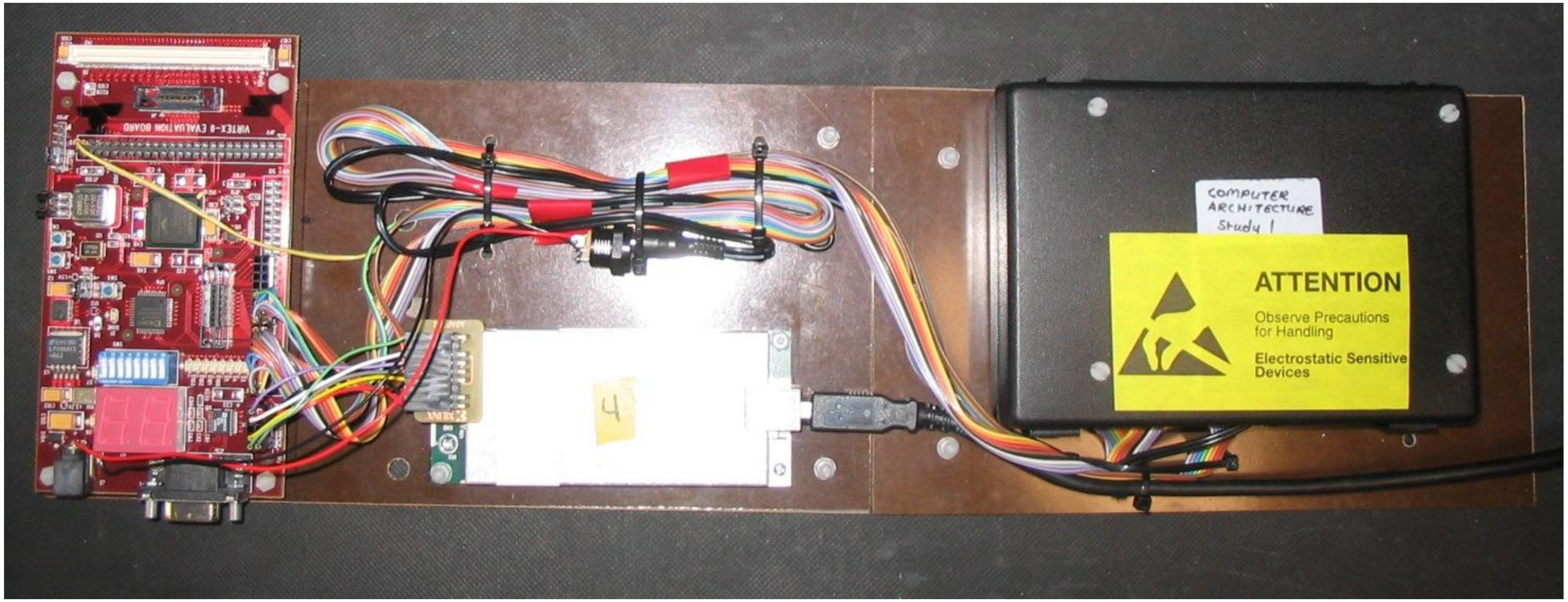

VIRTEX-II EVALUATION BOARD
4
COMPUTER ARCHITECTURE Study 1
ATTENTION
Observe Precautions for Handling
Electrostatic Sensitive Devices

Figure 4.18 The Test System for Mounting on the Assembly for TRIUMF

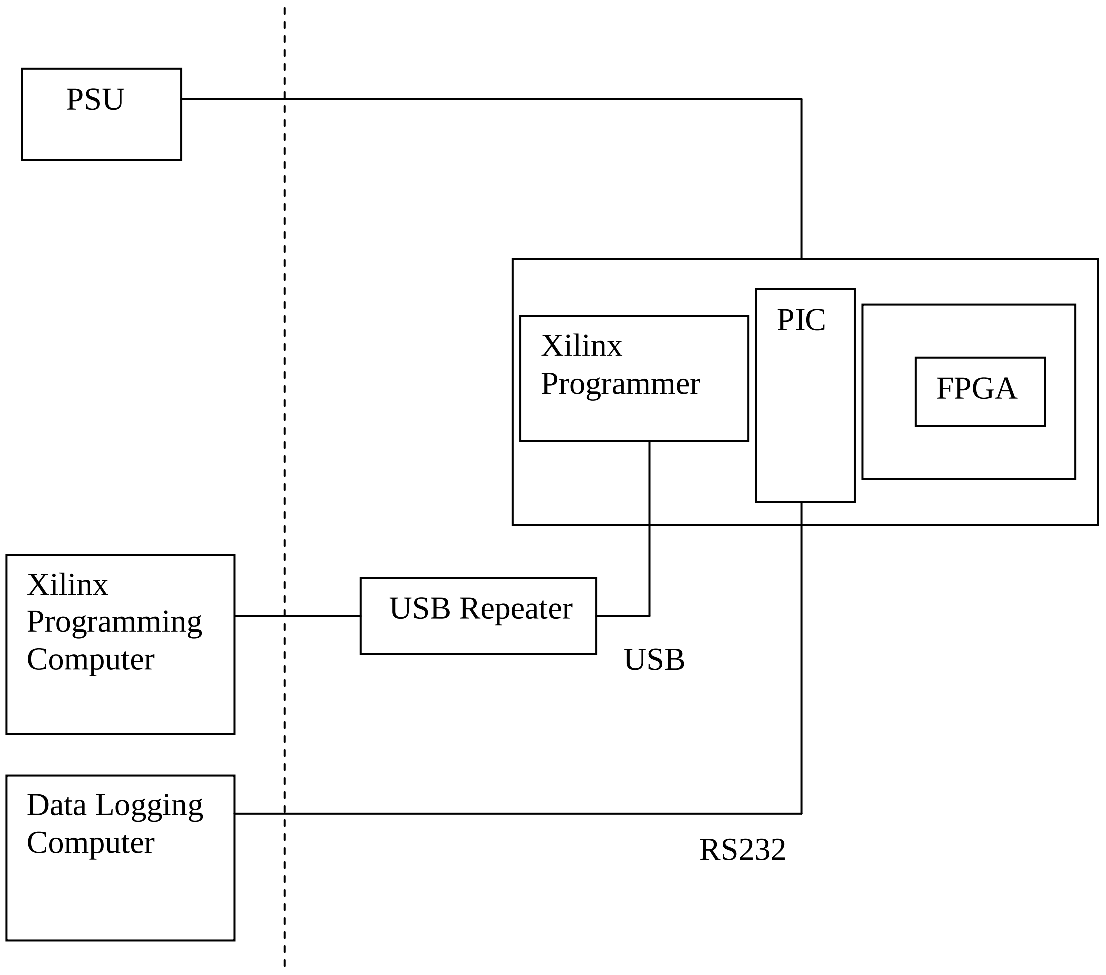


Figure 4.19 A Block Diagram of the TRIUMF Test System Layout

4.8.1.3 The Architecture for Neutron Trials

The device was filled with a chain of swap blocks which constantly swapped all 11 bits as they passed through each unit using decimal values between 0 and 11 as shown in Figure 4.20. The architecture chosen ensured the value present, between 0 and 11 would also be the value returned under correct operation. Therefore any corruption would be immediately apparent and would be passed through the architecture. This trial concentrated on two architectures; that of a weightless swap block and a weighted swap block, and was carried out at TRIUMF. Due to the high flux produced at this facility along with the carefully selected part a good number of results were collected which contradicted the hypothesis that the weightless architectures would be less susceptible. This is due to the fact more signals were needed, as each bit was represented in hardware.

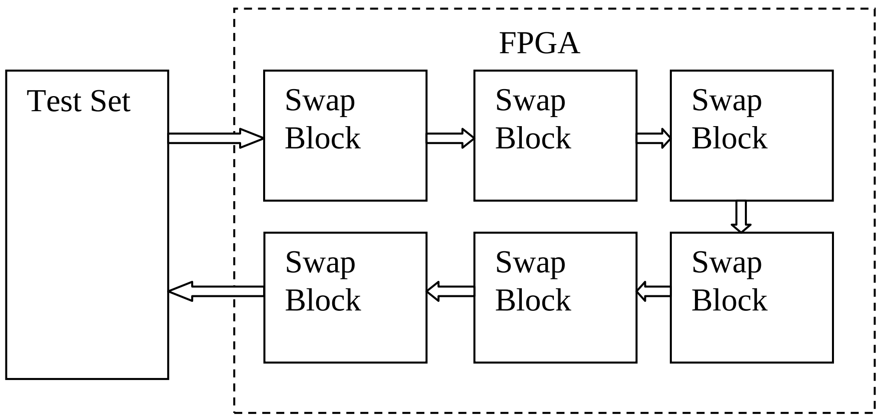


Figure 4.20 Swap Block Architecture with the FPGA Device

4.9 Test Architecture Descriptions

4.9.1 Weightless architecture

The weightless architecture consists of simple Boolean gate structures which manipulate the bits as they propagate through the architecture. The swap block consists of two main elements; a thermocode block followed by the weightless swap block. An example of the swap block structure of this element is shown in Figure 4.22. The thermocoder element is formed by a hierarchical structure of 'AND' and 'OR' gates [King 2000]. Figure 4.23 shows the overall thermocode structure. The complete test architecture is the maximum number of these elements which can be fitted into a Xilinx Virtex II 1000 device, which in this case was 100. Figure 4.21 show the full Xilinx usage report.

Elements	Used	Total	Percentage (%)
Look-up Table 4 input	10,074	10,240	98
Slices (related logic)	5,118	5,120	99
Slices (unrelated Logic	4,891	5,118	95
Total Number OF LUTs	227	5,118	4
Number of bonded IOBs	22	172	
Total Equivalent Gate Count	60,444		

Figure 4.21 Weightless Swap Block Architecture Usage Report

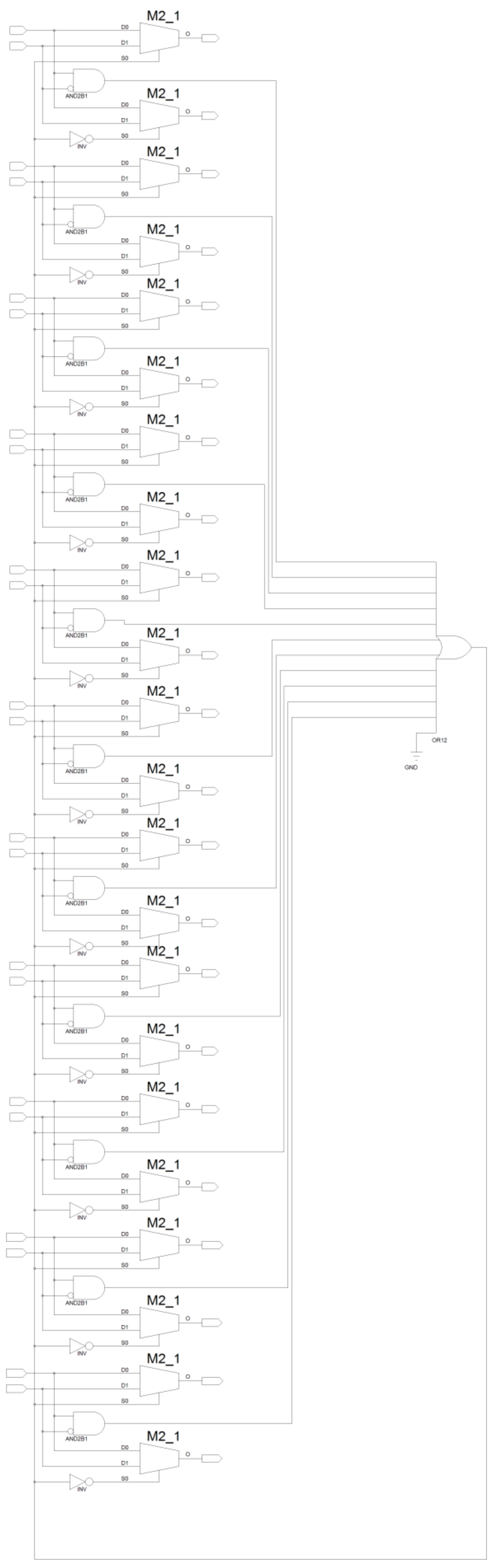


Figure 4.22 Weightless Swap Architecture

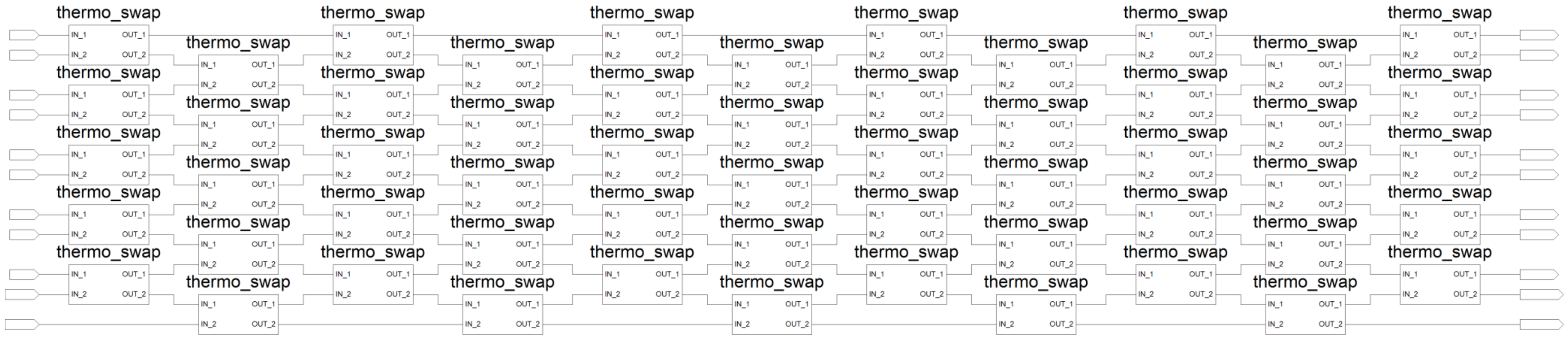


Figure 4.23 Thermocoder Structure

4.10 Weighted architecture

The weighted architecture is much simpler and a more conventional architecture. The swap block consists of a conventional 4 bit input comparator which controls two, four bit multiplexers as shown in Figure 4.24. Figure 4.25 shows an example of the connection architecture clearly showing the crossover between swap blocks.

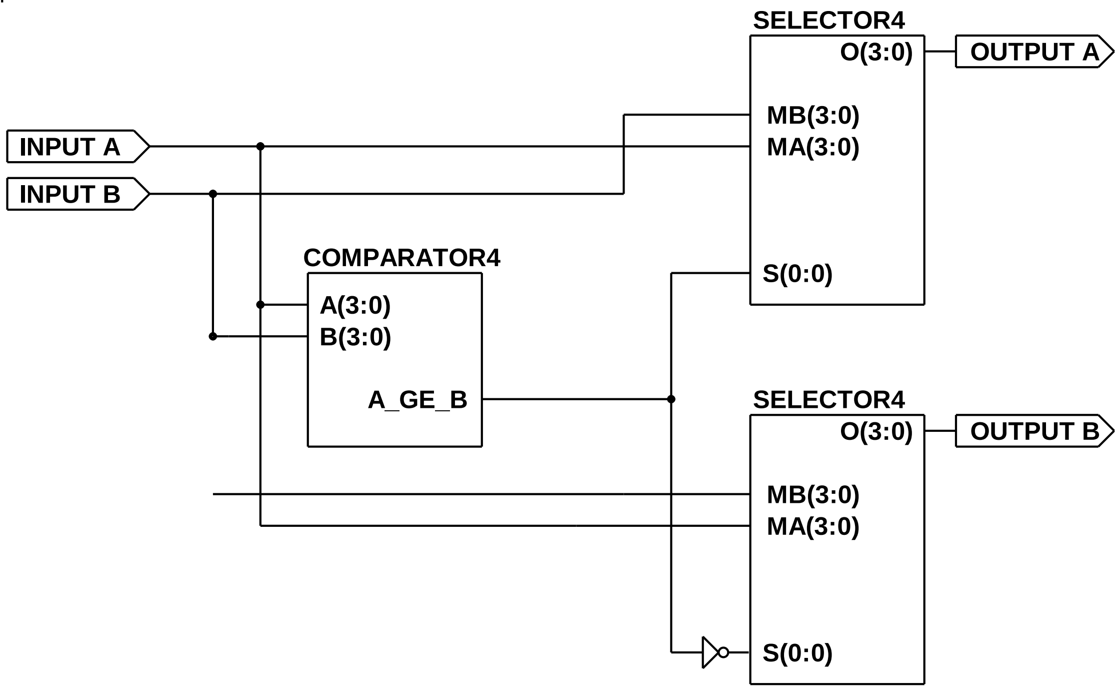


Figure 4.24 Weighted Swap Architecture

With this architecture using less single lines, 4, opposed to the 11 used in the weightless system, more elements could be fitted into a Xilinx Virtex II XCV 1000 device. In this experiment this was 504. Figure 4.26 shows the full Xilinx usage report.

Both architectures were replicated and chained together to utilise the maximum amount of the device in order to maximise the number of errors that would be detected.

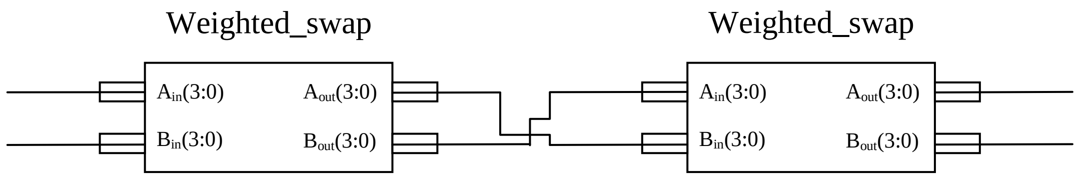


Figure 4.25 Shows the Weighted Swap Block Architecture Mapping

Elements	Used	Total	Percentage (%)
Look-up Table 4 input	6,048	10,240	59
Slices (related logic)	5,040	5,120	98
Slices (unrelated Logic	0	5,040	0
Total Number OF LUTs	6,048	10,240	59
Number of bonded IOBs	12	172	6
Total Equivalent Gate Count	42,336		

Figure 4.26 Weighted Swap Block Architecture Usage Report

4.11 Test Set and Test Program

A test set using a PIC device and a standard laptop was used to stimulate the architecture under test and record the results. The test set was capable of driving and monitoring either 11 weightless bits when running the weightless monitoring code, or 4 weighted binary bits when running the weighted code. The test set was designed to constantly cycle through values 0 to 11. A block diagram of the test set is shown in Figure 4.27 including the board under test with the FPGA in which the architectures are implemented.

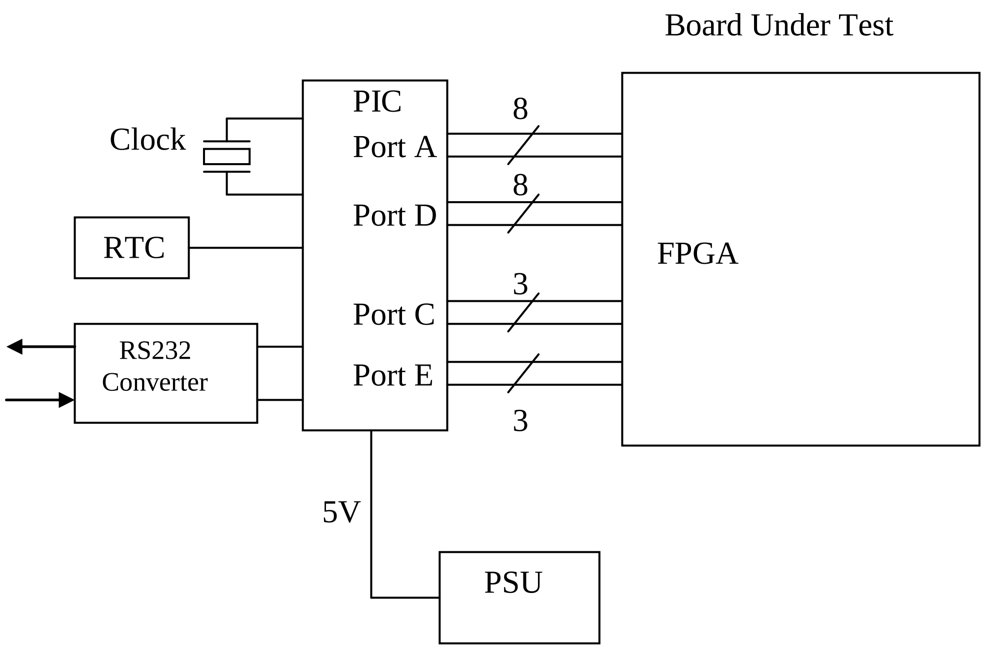


Figure 4.27 Test Set Block Diagram

The Xilinx device was reprogrammed with each architecture in turn using the Xilinx download cable, this was to eliminate the effects of any batch differences between Xilinx devices.

A commercial Xilinx test board was used with a Virtex II 1000 to host the architecture; this was then connected to the test set. Due to the method of access to the beam the whole test system was then strapped to a metal plate which was then lowered into the beam. The results were passed back to the computer for logging via an RS232 cable. A power cable was also sent down to the test set which had two on-board dc-dc converters, one to power the test set and the other to power the board under test. This allowed a higher voltage to power the device to be sent down the cable, and hence negate any effects of voltage drop due to the cable length. A USB cable was sent down via boosters to the Xilinx programmer which was also attached to the plate, so the device could be reprogrammed.

The PIC device within the test set was programmed using PIC C. A copy of the program can be found in Appendix D.

4.12 Analysis of the Data

The raw data collected on the computer was in text format and was saved to disk through hyper terminal. This raw data is manually summarised in Table 4.2. The results were then analysed using chi squared statistics to determine a cross-section rate. A confidence level of 90% was used. An add-in program to Excel was used and is included in Appendix D [Buchan 2004].

4.12.1 Establishing Neutron Fluence

The neutron counter used at the TRIUMF facility is mounted after the test sample. Therefore to calibrate the fluence in neutrons per cm^2 from the neutron monitor counter a correction factor has to be applied to take into account the attenuation of the neutrons caused by the board under test. The correction factor is established by dividing the count of neutrons over a period of time t_1 with no sample, by a similar run with the experiment in the beam line for a period of time t_2. Once these two values and times have been recorded, Equation 4.1 can be used to calculate the number of neutrons/cm^2.

$$Fluence = 3.9 \times 10^{3} \times NmCount \times \frac{\left(\frac{NmCountDUT\_Out}{Beamtime\_Out} \right)}{\left(\frac{NmCountDUT\_In}{Beamtime\_In} \right)} \qquad \textbf{4.1}$$

NmCount is the TRIUMF facility neutron monitor count value during the trial

NmCount DUT_Out is the Device Under Test out of the neutron beam, for calibration purposes

NmCount DUT_In represents the Device Under Test in the neutron Beam

Beamtime_Out is the time out of the beam, for calibration purposes

Beamtime_In is the time the Device under Test was in the neutron beam

Calculation from counts to neutrons/cm^2/s

Calibration flux = (70,649 neutrons over 60 seconds)

This is approximately 1,177 neutrons/cm^2/s without any obstructions or attenuation between the facility counter and the source. A presentation was accepted for publication by the IET [King et al 2008].

Filename	Date	Start	In Beam	End	Counts	Number of Errors	Bit 1 (1)	Bit 2 (2)	Bit 3 (4)	Bit 4 (8)
Wless 1	22/06/2006	13:33:14	13:33:51	13:48:51	910606	13	13			
Wless 2	22/06/2006	14:20:03	14:20:35	14:35:21	923051	18	18			
Wless 3	22/06/2006	15:01:28	15:02:17	15:51:44	2600035	45	45			
Wless 4	22/06/2006	17:28:59	17:29:44	17:53:15	1424339	18	18			
					5858031	**94**	**94**			
Weight 1	22/06/2006	13:55:37	13:56:46	14:11:07	643035	3	1		1	1
Weight 2	22/06/2006	14:41:04	14:42:03	14:53:05	668110	4	1	1	1	1
Weight 3	22/06/2006	16:00:37	16:01:20	17:23:42	3606409	18	6	4	5	3
Weight 4	22/06/2006	17:57:47	17:58:33	18:47:10	2830877	9	3	4	2	
					7748431	**34**	**11**	**9**	**9**	**5**

Table 4.2 Collated Results from Trial

X-SECTION	100				
CONFIDENCE	90	ALPHA		0.05	
	FLUENCE	EVENTS	LOWER LIMIT	BEST ESTIMATE	UPPER LIMIT
	ICM	Errors			
weightless	910606	13	7.60148E-06	1.42762E-05	2.44127E-05
weightless	923051	18	1.15573E-05	1.95005E-05	3.08193E-05
weightless	2600035	45	1.26242E-05	1.73075E-05	2.31587E-05
weightless	1424339	18	7.48975E-06	1.26374E-05	1.99726E-05
weighted	643035	3	9.62113E-07	4.66538E-06	1.36342E-05
weighted	668110	4	1.63127E-06	5.98704E-06	1.53292E-05
weighted	3606409	18	2.95805E-06	4.99111E-06	7.88811E-06
weighted	2830877	9	1.45374E-06	3.17923E-06	6.03516E-06

Table 4.3 Results following Chi Squared Analysis

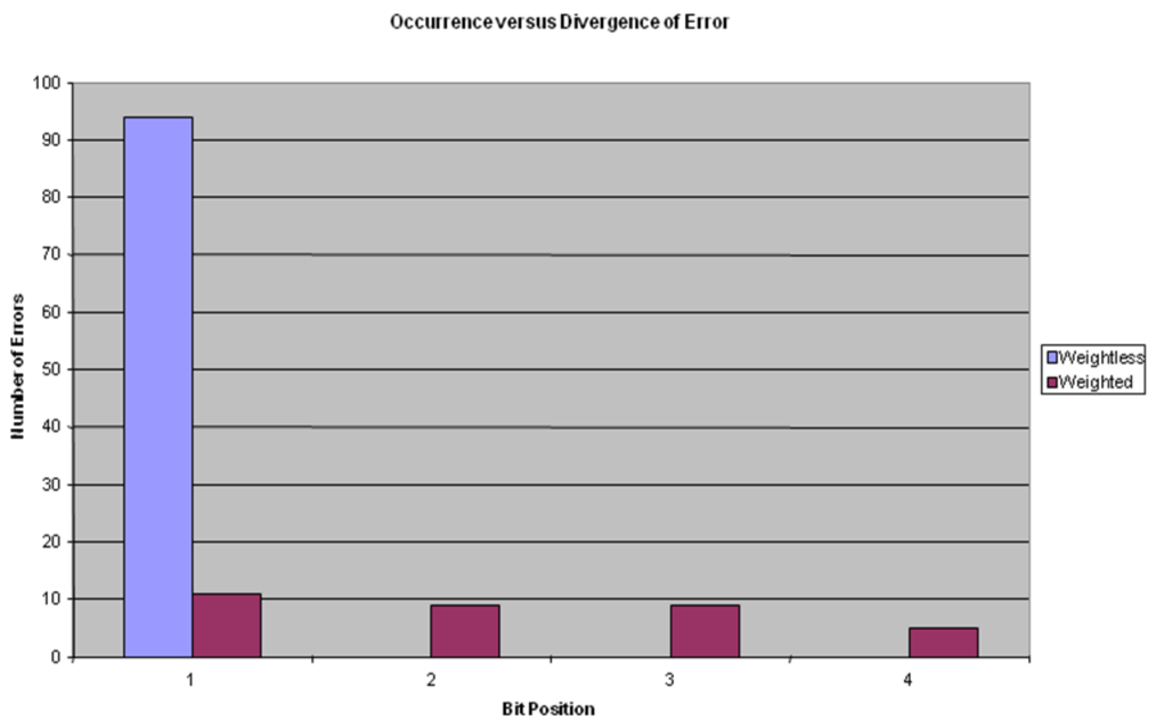


Figure 4.28 Occurrence versus Divergence of Error

4.13 Conclusions

The results show that although the weightless architecture experienced more errors, the magnitude of these errors was much less and entirely deterministic in value. This could have advantages in high reliability systems. Conversely, although the weighted implementation had three times less corruptions the effects of the corruption were much more significant and less deterministic. This was due to the fact each bit represented a weighting meaning the corruption could range from one to the value of the most significant bit. Therefore the percentage error could be up to 50%, but in this experiment all bits were not fully utilized. This increased the maximum potential error to 72% of the maximum value. Whereas the maximum error for the weightless architecture was only 9% of the maximum value. The weighted architecture is more compact due to the reduced number of signal lines meaning that just over five times more gate elements could be fitted into the device under test.

4.14 Summary of Chapter 4

A novel weightless Boolean median filter has been developed which has been patented, this filter has been implemented within an FPGA. The main element of this filter, the swap block, has been compared with a standard median filter swap block. Both filter elements have been implemented in an FPGA and the performance and resilience is determined when subjected to man-made ground based neutron radiation. Trials have shown that weightless architectures experience more errors although the magnitude of these errors is much less and more deterministic. Chapter 5 investigates the performance characteristics of a standard median filter against that of King's Neuroram.

Chapter 5

5 Performance of Weightless Neural Network Image Filters

5.1 Overview

This chapter describes the simulation work performed to evaluate the performance of a neurofilter in comparison to a conventional median filter. Reference images are used with different types of added noise to perform this assessment. The neurofilter used in these simulations was based upon King's Neuroram network [King 2000]. King uses the Neuroram network to filter images with added salt and pepper noise and compares the results to a standard median filter. The work in this chapter is a continuation of King's analysis evaluating the Type 1 Neuroram against a selection of further noise types. The effects of using different thresholds in the neurofilter are also examined. King's thesis showed that a mid point threshold gave the best filtering performance. This hypothesis was tested in particular with the different types of noise. In conjunction with the noise test the effects of filtering multiple times known as 'cascade filtering' was trialed. The results for cascaded neurofilters and the median filters are presented.

Testing the filters with the reference images below evaluates their performance with particular respect to the removal of noise.

5.2 Reference Images

Three reference images were taken for this work from the University of Cape Town, Digital Image Processing standard image library [University of Cape Town 2004]. The three images comprise an image of an F16 aeroplane, a Milk Drop and Los Angeles airport. These images will be referred to as F-16, Milk Drop and LAX and are shown in Figure 5.1, Figure 5.2 and Figure 5.3 respectively. All the images are black and white 512 by 512 pixels 8 bit greyscale.

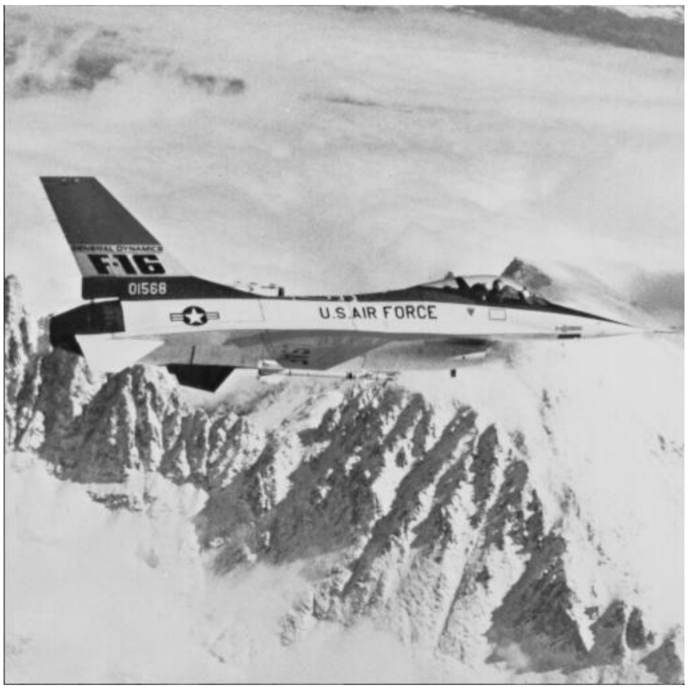


Figure 5.1 Reference Image F-16

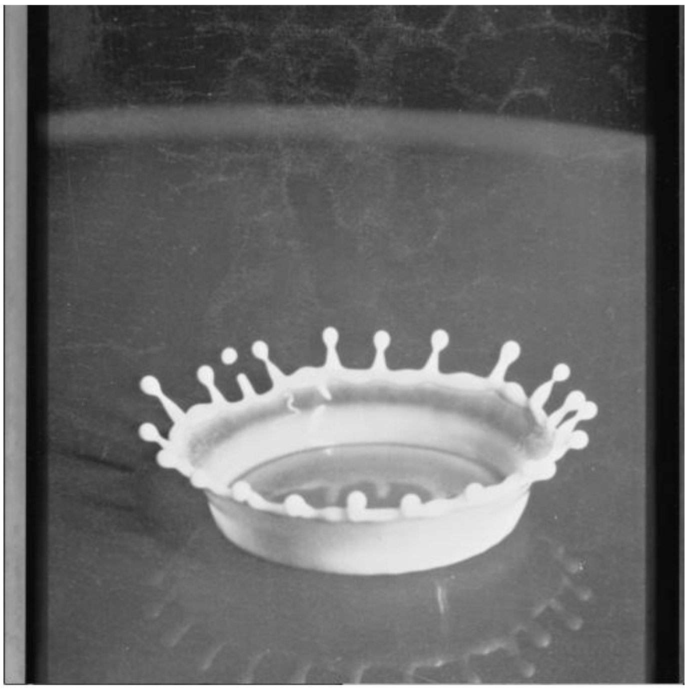


Figure 5.2 Reference Image Milk Drop

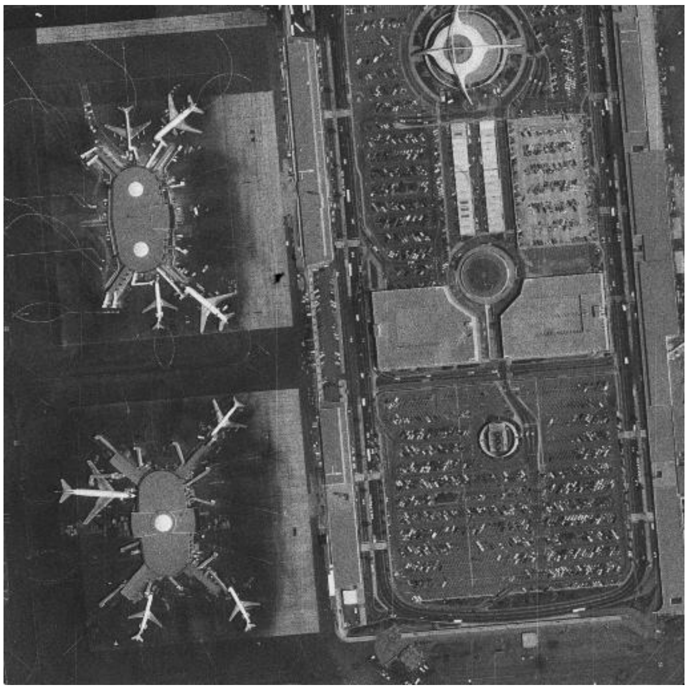


Figure 5.3 Reference Image LAX

5.3 Noise Types

In order to evaluate the performance of the neurofilter and the median filter various noise types were added to the reference images which ranged from 10% to 90% noise in 10% increments. Each of the resultant noisy images were then independently filtered using a simulation of a conventional median filter and a simulation of a Neuroram filter. The resultant performance data was then stored in a MATLAB matrix to aid the graphical presentation. The noise types used to evaluate the performance of the filters were salt and pepper noise, additive Gaussian, additive uniform, multiplicative Gaussian and multiplicative uniform. The noise types were generated using a MATLAB program from the Internet given in Appendix D. This program was secured from the MATLAB exchange as part of a larger Nonlinear Diffusion Toolbox written for MATLAB and produced by Frederico D'Almeida [D'Almeida 2003]. A description of each of the noise types and its effect on the images are described below. The noise was applied to a test image, this image consisted of a 512 by 512 pixels, 8 bit greyscale image generated using MATLAB [The Math Works Inc 1995]. The image was a uniform mid range grey as all pixels were set to 127 except for the first two pixels which were 255 and 0 respectively. This was necessary as the noise generation program used the maximum and minimum values in the image to select the range of noise. Figure 5.4 shows the greyscale image along with the image with the different types of noise added. The accompanying histogram for each of the images given in Figure 5.5.

5.3.1 Additive Gaussian Noise

The first noise type used to distort the image was additive Gaussian. The additive Gaussian noise corruption is implemented by summing a random Gaussian value to the value of each pixel. The size of the Gaussian curve is given by the variance which in this case is the default for the program of 1. The percentage value used in the program represents the percentage of the pixels within the image affected. The histogram Figure 5.5 (b) clearly shows the effects of the additive Gaussian noise and shows the Gaussian curve centred about mid-point.

5.3.2 Additive Uniform Noise

Summing a uniformly distributed random value to each pixel generates the additive uniform noise. The percentage determines the amplitude of the noise added and is based on the percentage between the minimum and maximum pixel values in the image. The histogram in Figure 5.4 (c) clearly shows the additive noise on the original image.

5.3.3 Multiplicative Gaussian Noise

The multiplicative Gaussian noise is generated in a similar way to the multiplicative uniform noise. The noise is added to a percentage of image pixels and follows a standard Gaussian distribution.

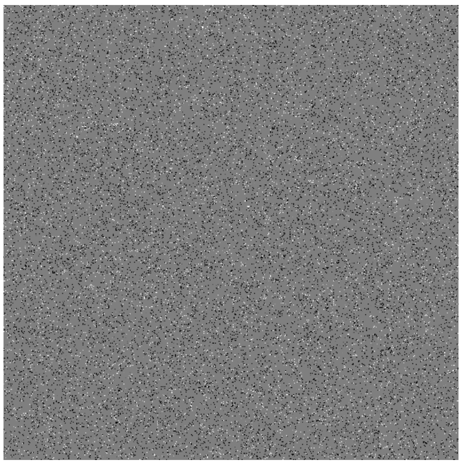
5.3.4 Multiplicative Uniform Noise

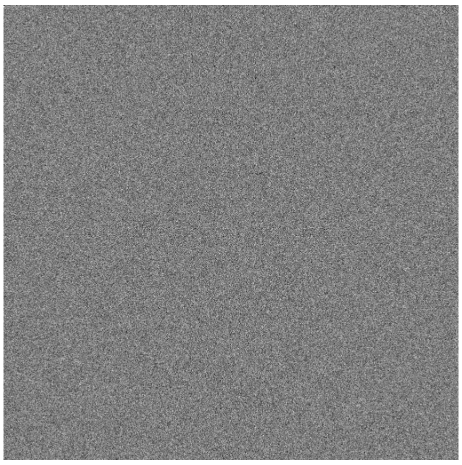
The multiplicative uniform noise is generated by multiplying a random uniformly distributed value to each pixel, the percentage represents the amplitude of noise added to the pixels and is based on the percentage between the minimum and maximum pixel values in the image. The histogram Figure 5.5 (e) clearly shows the uniform distribution of the noise on the image.

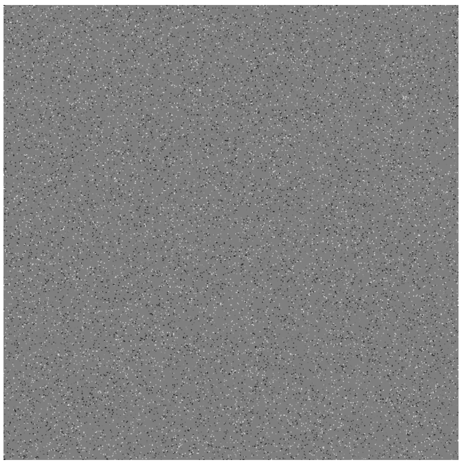
5.3.5 Salt and Pepper Noise

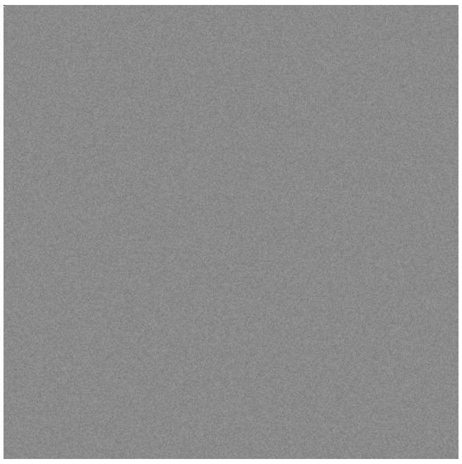
The salt and pepper noise is generated by taking a percentage of pixels in an image and randomly setting them to the minimum or maximum pixel value in this case, either 0 or 255. The histogram in Figure 5.4 (f) clearly shows the generation of black and white pixels.

a) Grey Level Image


(d) 10% Multiplicative Gaussian Noise


(b) 10% Additive Gaussian Noise


(e) 10% Multiplicative Uniform Noise


(c)10% Additive Uniform Noise

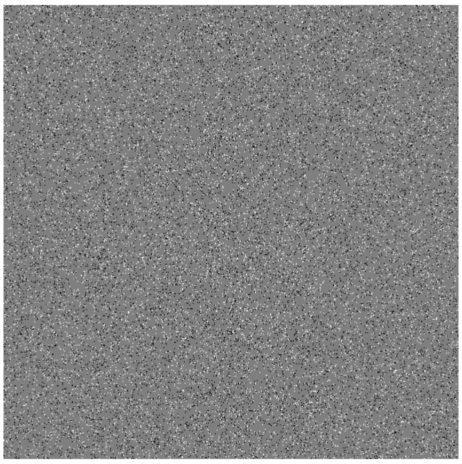
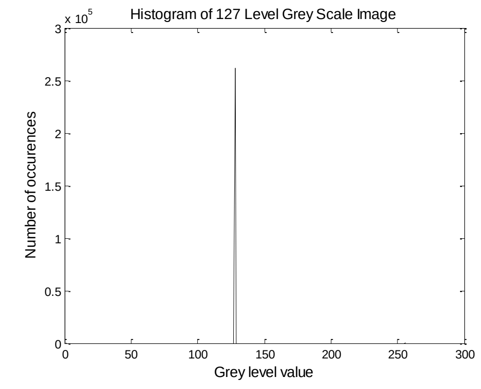
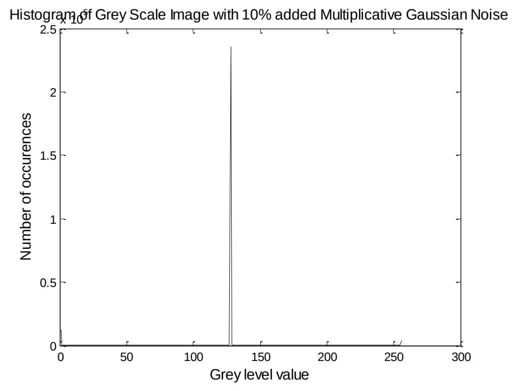

(f) 10% Salt and Pepper Noise

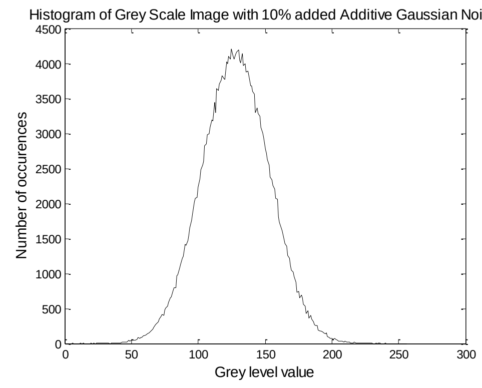
Figure 5.4 MATLAB Greyscale Image with Different Noise Types Added



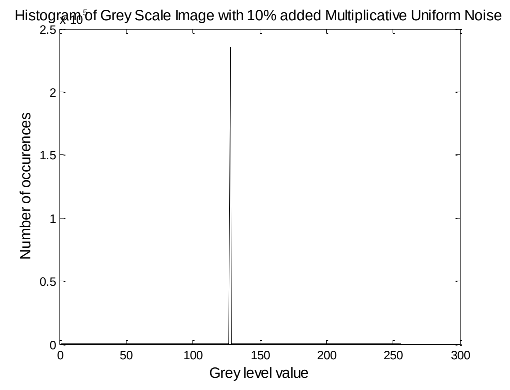
(a)



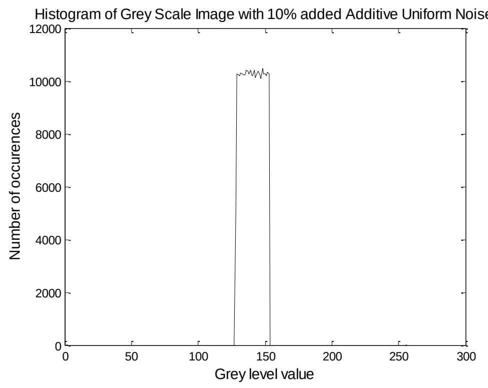
(d)



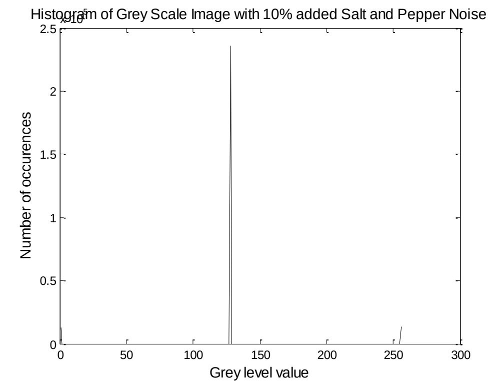
(b)



(e)



(c)



(f)

Figure 5.5 (a) Shows a Histogram of MATLAB Greyscale Image, (b, c, d, e, f) Show Histograms of the Greyscale Following the Corruption with the Stated Different Noise Types

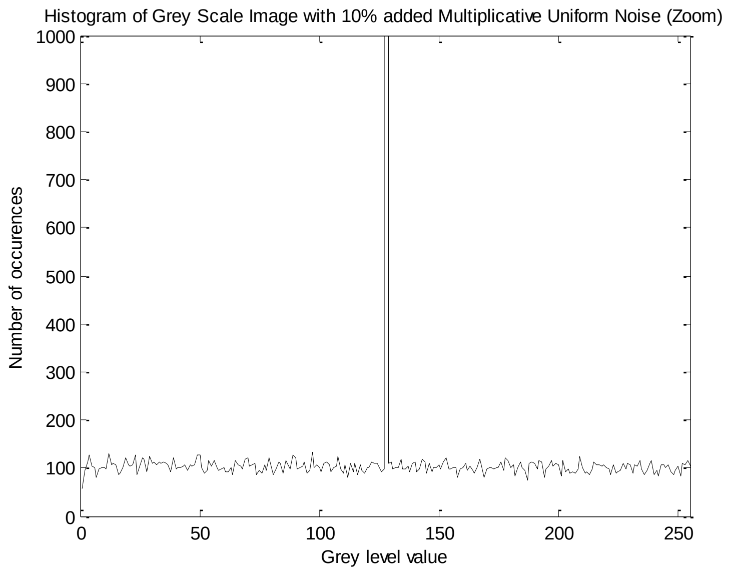


Figure 5.6 A Close Up View of the Multiplicative Uniform Histogram

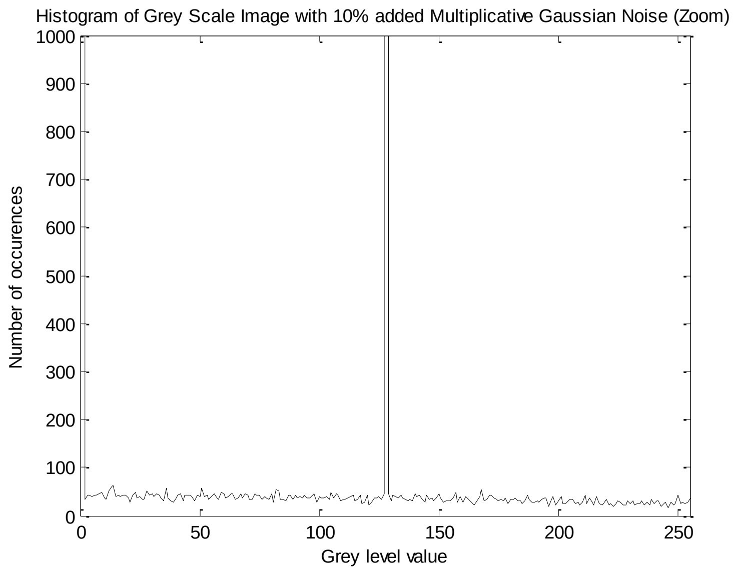
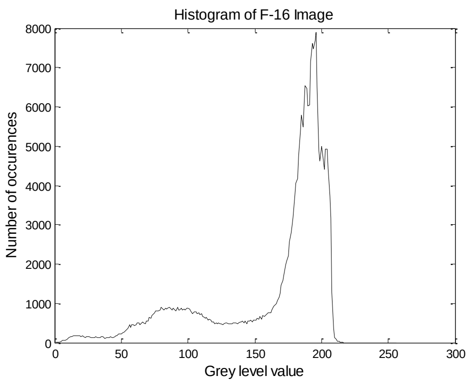
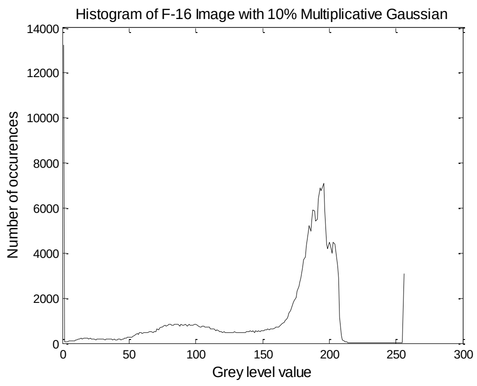


Figure 5.7 A Close up View of the Multiplicative Gaussian Noise Histograms

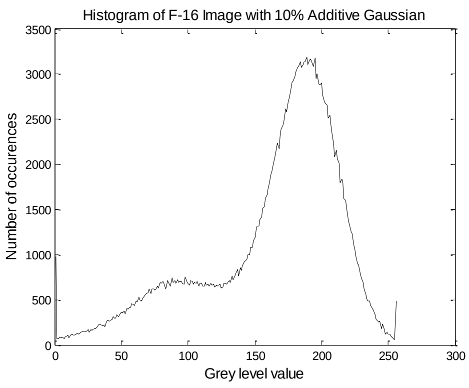
Figure 5.6 and Figure 5.7 shows a close up view of the two histograms for the corrupted images. The multiplicative uniform histogram shows a relatively even level of noise across the bins. The characteristics of the multiplicative Gaussian display a peak of noise at the zero bin level representing all the negative values which are out of range and hence are classed at the minimum level of zero.



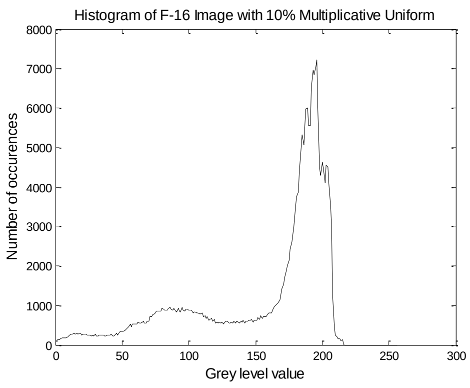
(a)



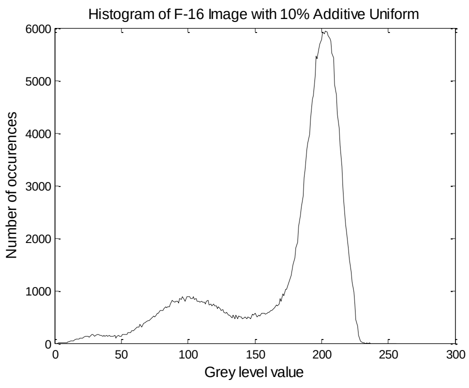
(d)



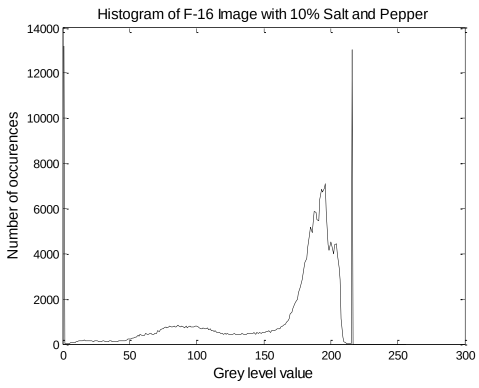
(b)



(e)



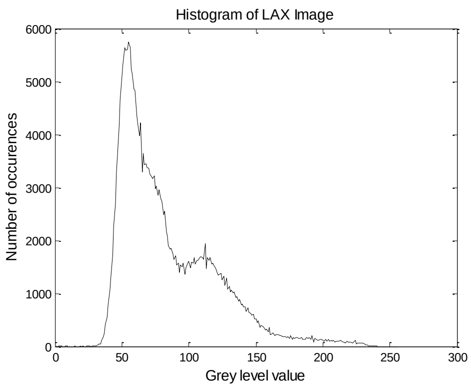
(c)



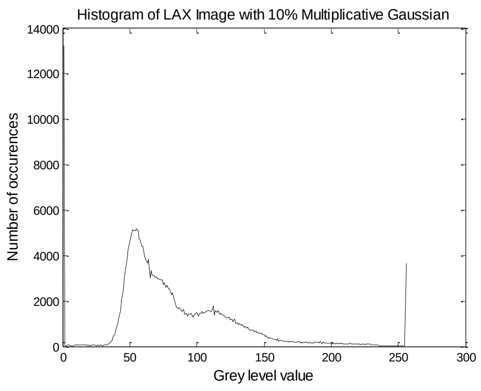
(f)

Figure 5.8 Histogram of the F-16 Image with the Different Noise Types Added

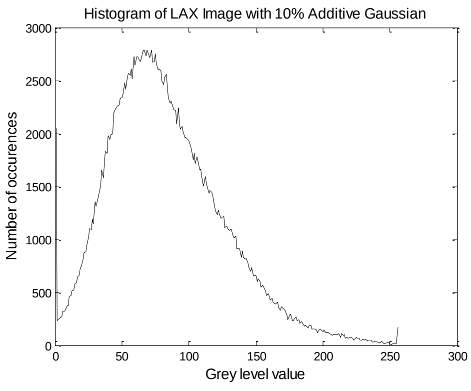
Figure 5.8 (a) shows the histogram of the reference image of the F16. It can seen that the majority of the pixels in the image are around the 200 bin level. Figure 5.8 (b,c,d,e,f) shows the resultant histograms following the addition of the different types of noise. The distortion caused when additive Gaussian is added to the image has severely damaged the image properties as the noise places the pixel values outside the maximum pixel value, thus the majority of picture information is loss.



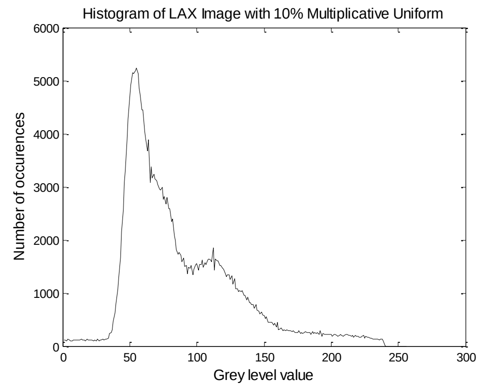
(a)



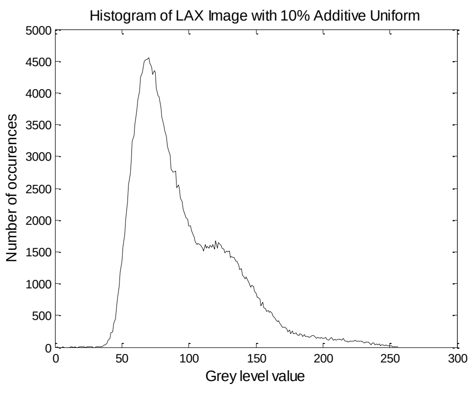
(d)



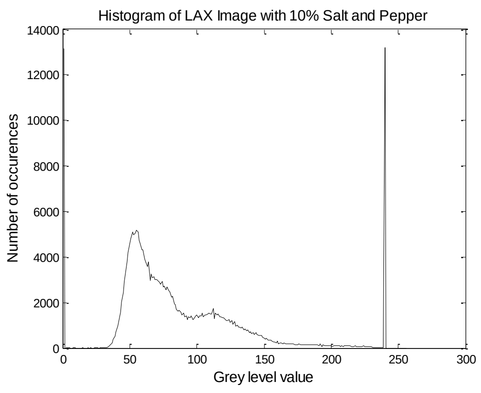
(b)



(e)



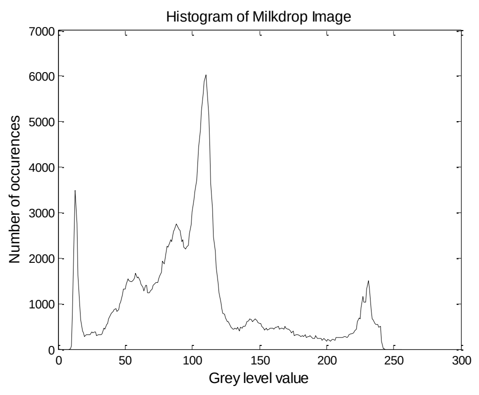
(c)



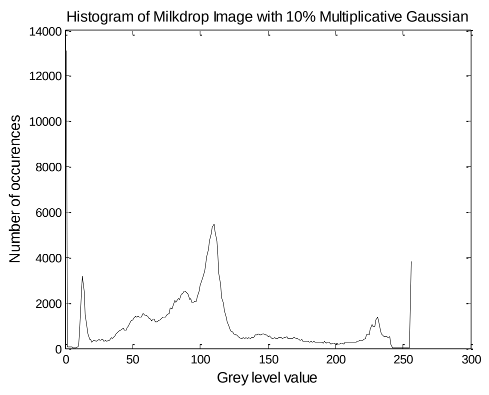
(f)

Figure 5.9 Histogram of the LAX Image with the Different Noise Types Added

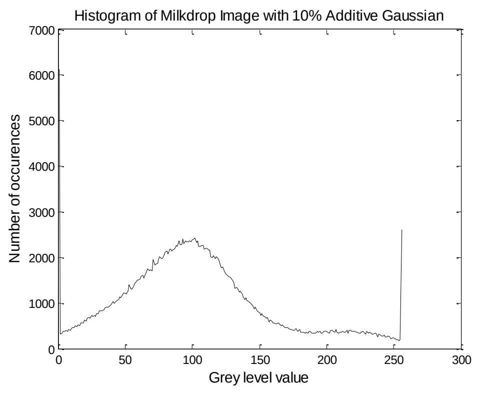
Figure 5.9 (a) gives the histogram for the reference LAX image. Figure 5.9 (b, c, d, e, f) the histogram shows that the majority of pixels in the image are around the 50 to 150 bin region. This means again the severity of the damage caused by the additive Gaussian noise is significantly more than the other noise types.



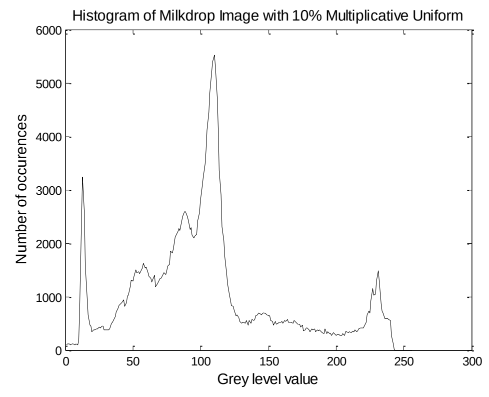
(a)



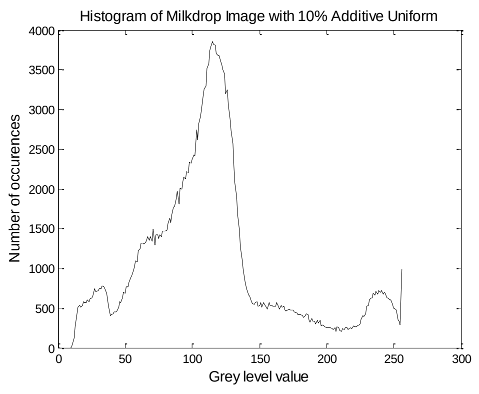
(d)



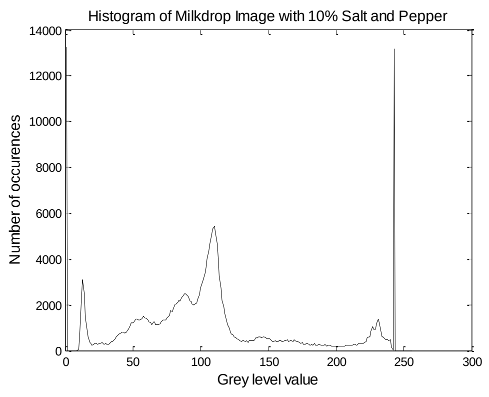
(b)



(e)



(c)



(f)

Figure 5.10 Histogram of the Milkdrop Image with the Different Noise Types Added

5.4 Evaluation Criteria

Several measurement criteria have been used to evaluate the quality of the filtered images. These include the standard measurement techniques of mean squared error, peak signal to noise ratio and finally the more subjective but important criteria of visual inspection. Mean Squared Error (MSE) is used as it measures the differences between two images, in this case the filtered image and the reference image. MSE is used instead of the pure difference because using the difference would yield both positive and negative numbers whereas the squared element of MSE always ensures the result is positive. The equation for MSE is given in equation 3.2.

$$MSE = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left(f(x,y) - f_r(x,y)\right)^2 \qquad 3.2$$

Where M and N define the pixels in the given image and $f_r(x,y)$ represents the reference image and $f(x,y)$ the filtered image.

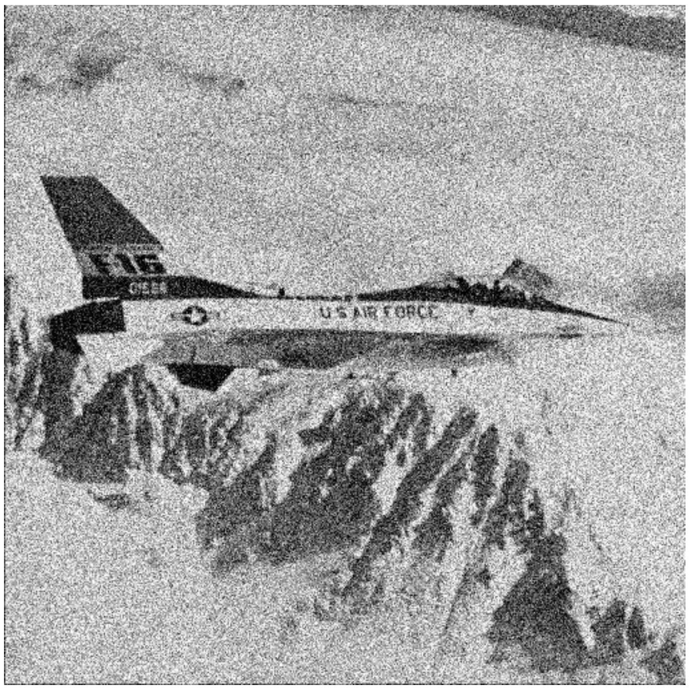
The lower the MSE the greater the similarity of the two images. Therefore it is commonly used to evaluate the performance of any image restoration algorithm or filter. A reference image is used and compared to the processed image. The processed image has had damage introduced and has subsequently been processed to remove the corruption. The effectiveness of the process to remove the noise can then be quantified; the lower the MSE the better the filter or algorithm has restored the image. Another standard measurement used to measure the similarity of images is the Peak Signal to Noise Ratio. This measurement technique is based on the MSE result and hence does not generate negative values. The equation for PSNR is given in equation 3.3. Unlike the MSE the higher the PSNR the better the performance of the filter.

$$PSNR = 10Log_{10}\left(\frac{255^2}{MSE}\right) \qquad 3.3$$

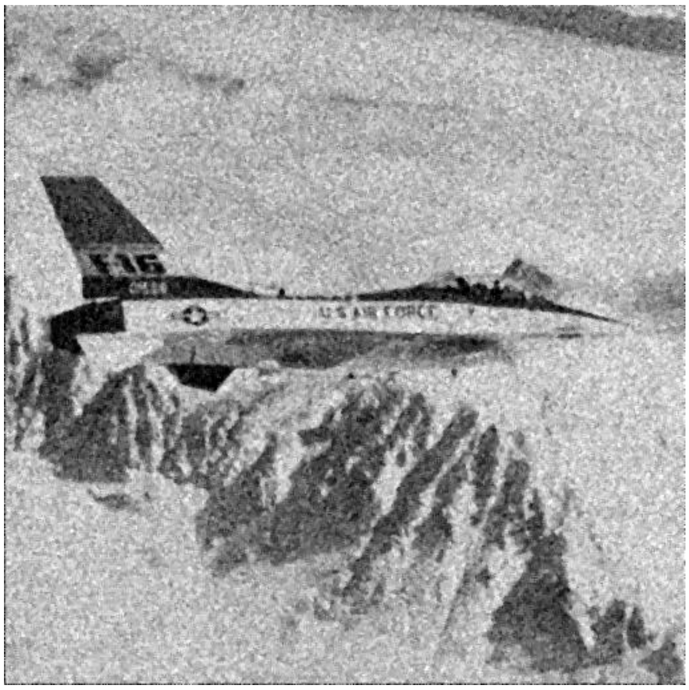
In equation 3.3 the value of 255 is used because it is the peak signal numerical value since the pixel width is eight bits.

The final form of evaluation of the images is probably the most important and yet the most subjective. As this method is visual inspection, this method is important as images are visual representations and the removal of noise is often performed to aid the ability to understand the image.

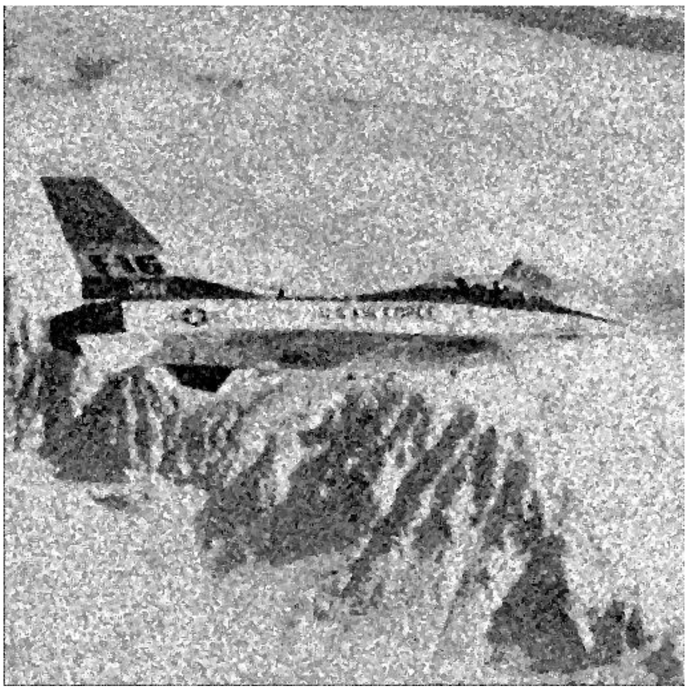
5.4.1 Additive Gaussian Noise Filter Results



(a) F-16 with 20% Additive Gaussian noise



(b) Median Filtered



(c) Neurofiltered

Figure 5.11 Images of F-16 Reference with 20% Additive Gaussian Noise and Filtered Images

This simulation uses additive Gaussian noise. Figure 5.11 (a), shows the F-16 reference image with 20% additive Gaussian noise, images Figure 5.11 (b) and Figure 5.11 (c) show the results after median filtering and neurofiltering respectively. On visual inspection the performance of the median filter and the neurofilter on this image are poor with very little noise removal. It can be seen that the performance of the median filter is better than the neurofilter which causes larger noise clusters causing more distortion to the image. It is clear that both the median filter and the neurofilter are not effective for the removal of additive Gaussian noise. The poor operation of the filters is reflected in both the MSE results and the PSNR results as shown in Figure 5.12.

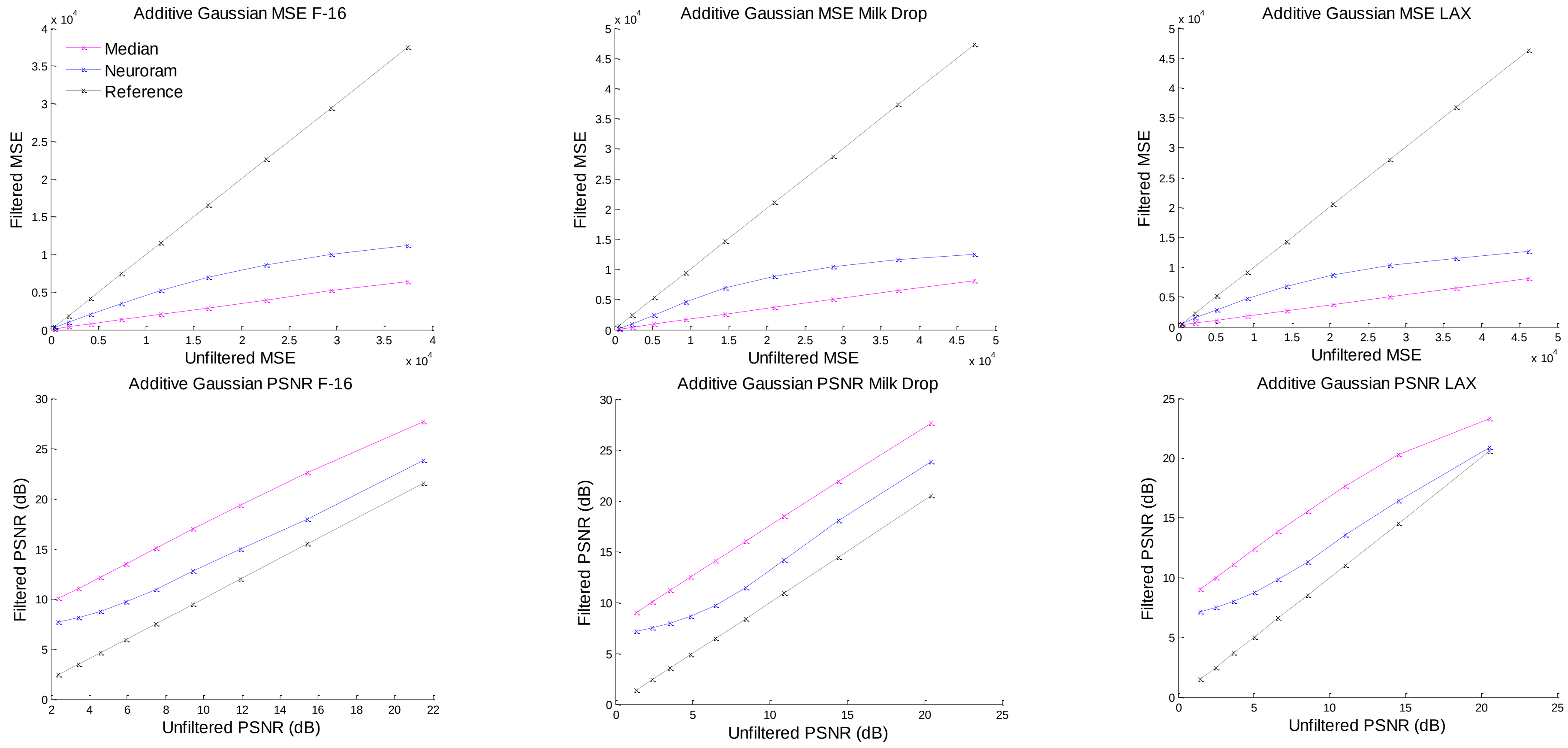
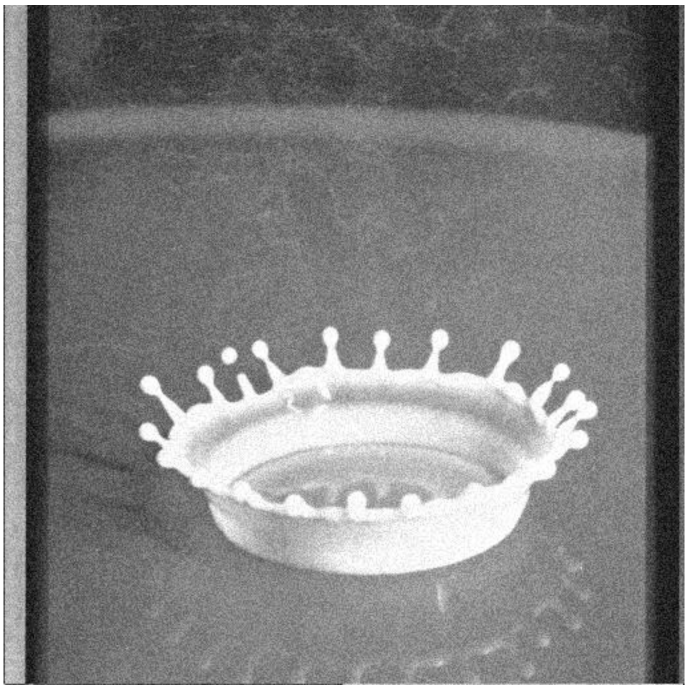
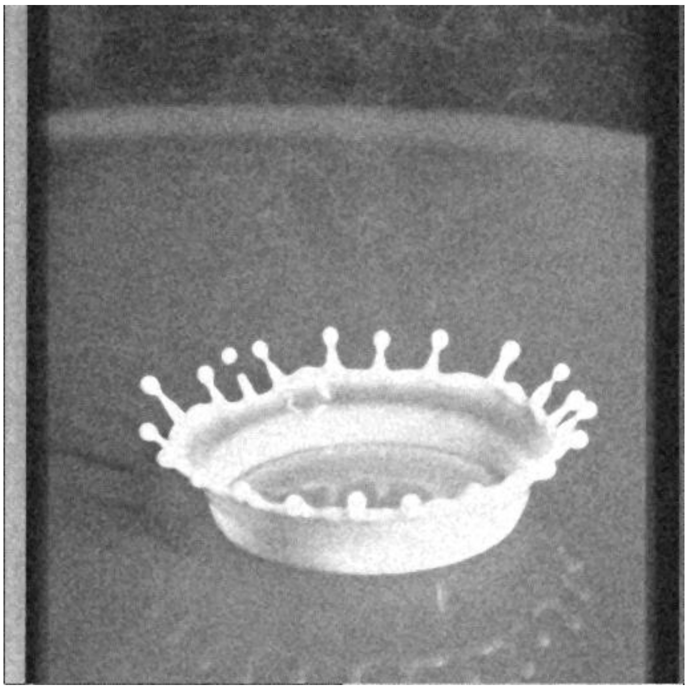


Figure 5.12 Graphs of MSE and PSNR for Median Filter and Neurofilter on Additive Gaussian Noise

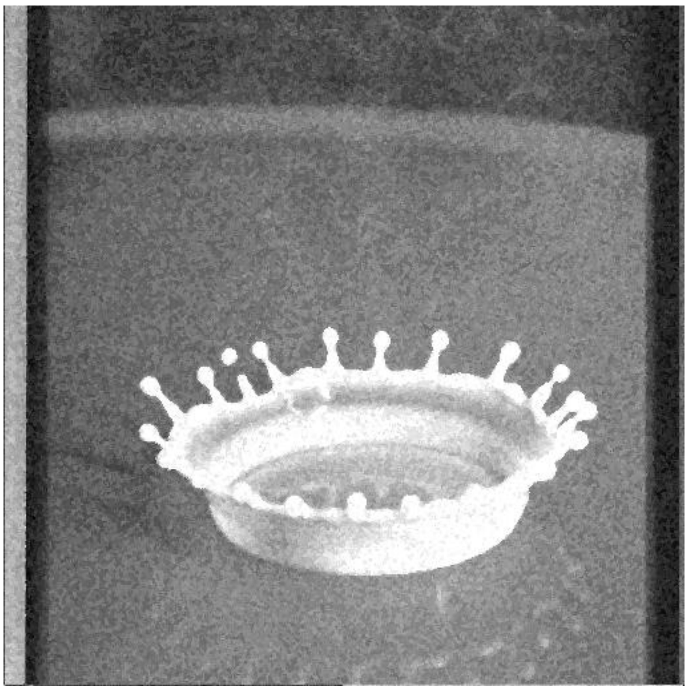
5.4.2 Additive Uniform Noise Filter Results



(a) Milk Drop with 20% Additive Uniform Noise



(b) Median Filtered



(c) Neurofiltered

Figure 5.13 Images of Milk Drop reference with 20% additive uniform noise and filtered images

Following the additive Gaussian noise simulation the filters were further evaluated using additive uniform noise.

Figure 5.13 (a), shows the Milk Drop reference image with 20% additive uniform noise, images Figure 5.13 (b) and Figure 5.13 (c) show the results after median filtering and neuro-filtering respectively. The results from the additive uniform simulation confirm both the median filter and neuro-filters inability to deal with the removal of additive noise. Visual inspection of the images shows little or no improvement in image quality in comparison to the noise images. It can be seen that the median filter has the effect of causing slight blurring to the image. The neurofilter does not cause blurring and hence has a sharper contrast but causes slight pixelisation, particularly on edges. The poor performance of the filters is further reflected in both the MSE and PSNR results shown in Figure 5.14 which show little or no improvement with respect to the noisy unfiltered reference image.

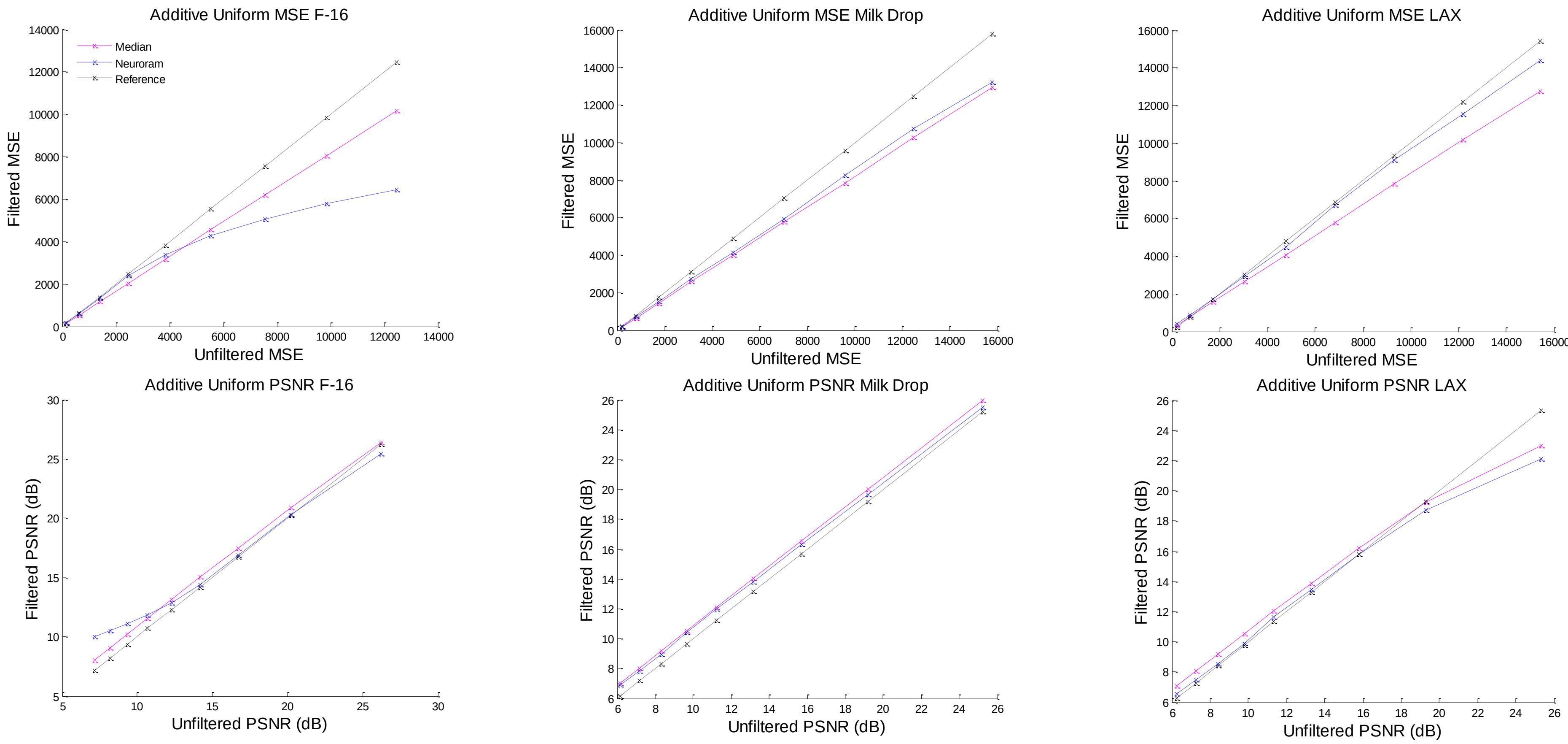
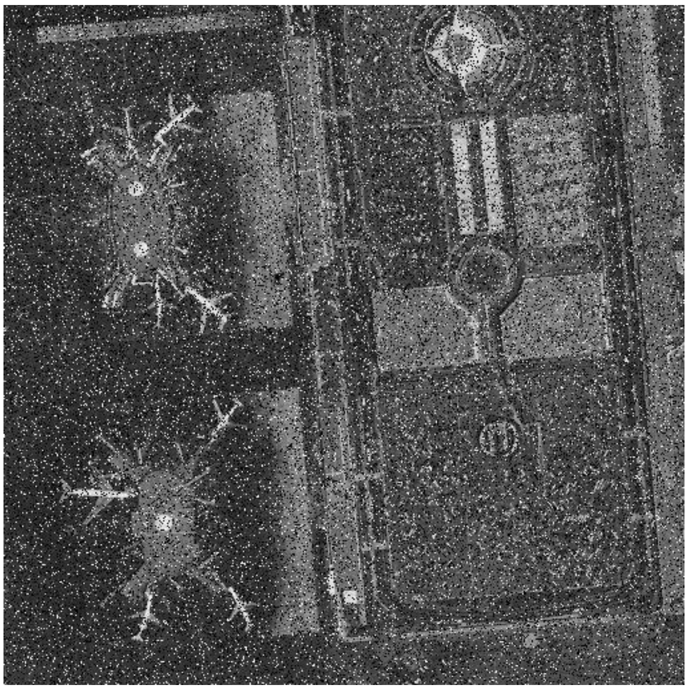
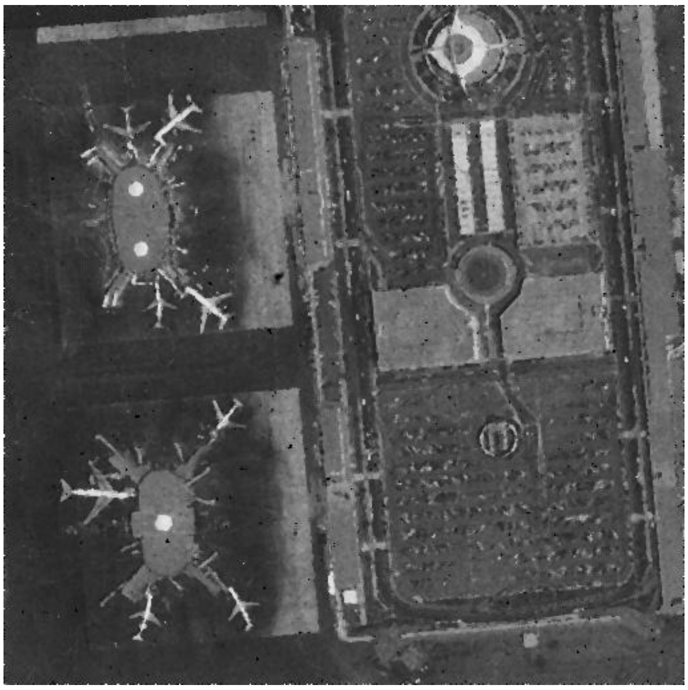


Figure 5.14 Graphs of MSE and PSNR for Median Filter and Neurofilter on Additive Uniform Noise

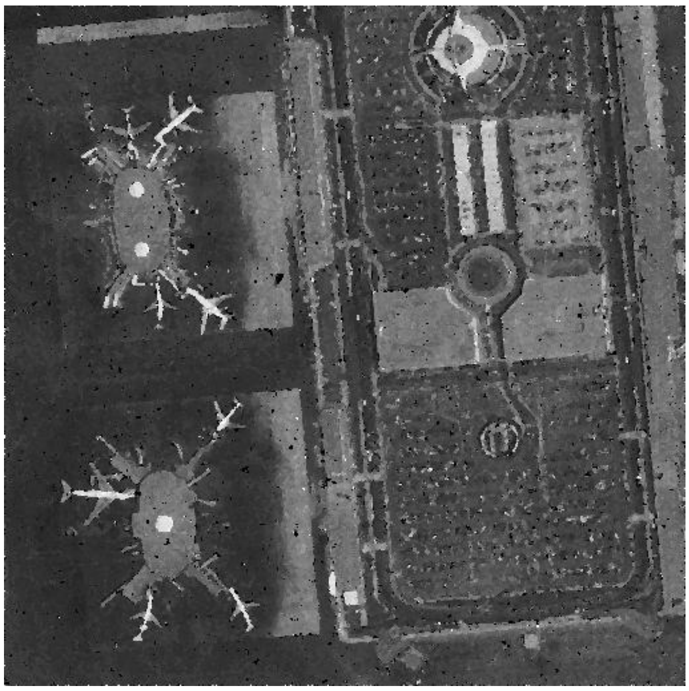
5.4.3 Multiplicative Gaussian Noise Filter Results



(a) LAX with 20% Multiplicative Gaussian Noise



(b) Median Filtered



(c) Neurofiltered

Figure 5.15 Images of LAX Reference with 20% Multiplicative Gaussian Noise and Filtered Images

A simulation using multiplicative Gaussian noise was performed. Figure 5.15 (a), shows the LAX reference image with 20% multiplicative Gaussian noise, images in Figure 5.15 (b) and Figure 5.15 (c) show the results after median filtering and neurofiltering respectively. Visual inspection of the noisy image shows lots of pixel noise which destroys the detail of the original image. It can be seen that from the images in Figure 5.15 (b) and Figure 5.15 (c) that the median filter and the neurofilter perform well at removing this type of noise, however both have introduced distortion. The median filter causes slight blurring causing loss of detail, sharpness and contrast. The neurofilter does not cause blurring but causes slight pixelation but does not affect the contrast and sharpness, if anything it improves them. The effects of damage caused by the filters can be more clearly seen on the writing on the F-16 images. It is noticeable that the damage is worst on the neurofiltered image. This is confirmed with the MSE and PSNR results shown in Figure 5.16 which show improvements in image quality after filtering of the noisy images.

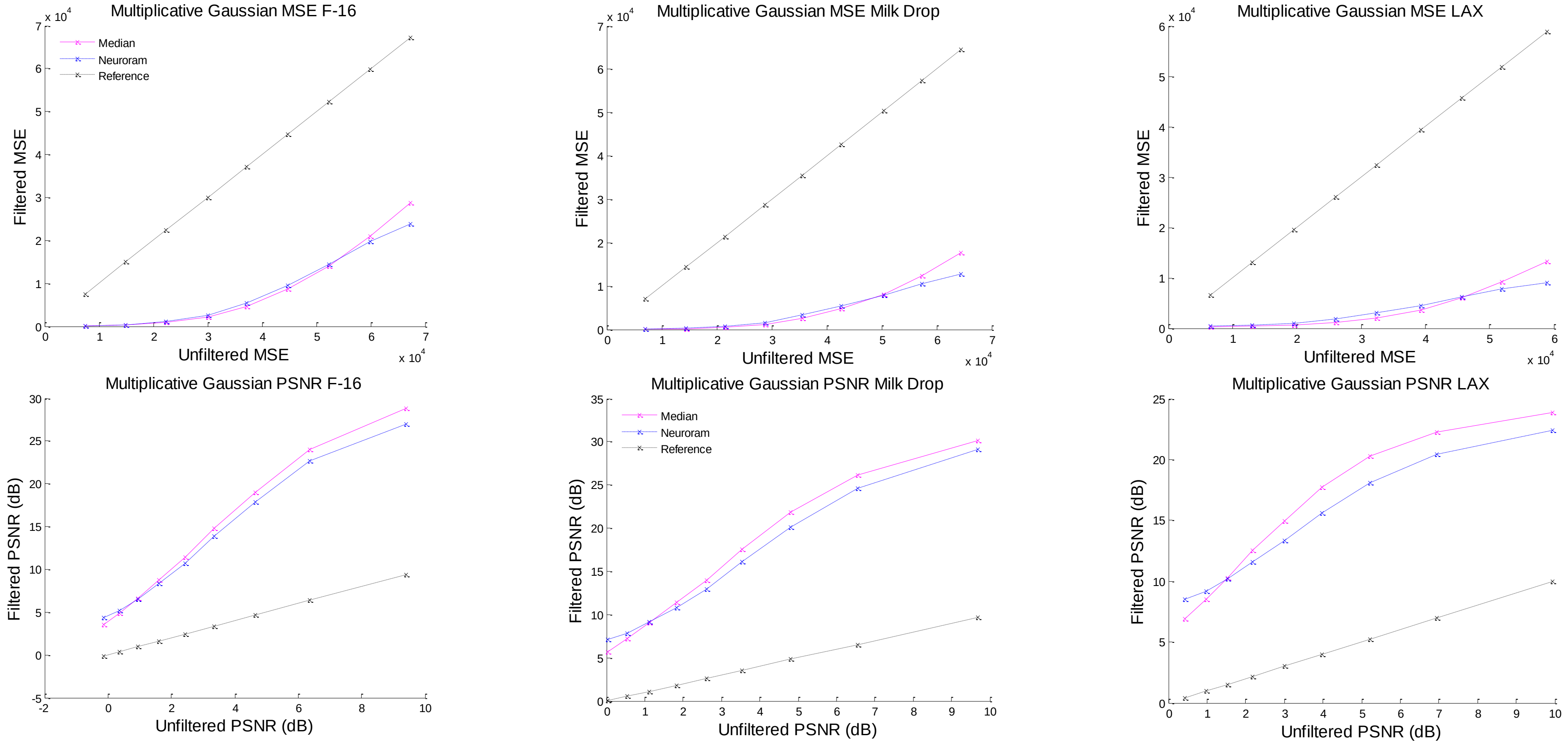
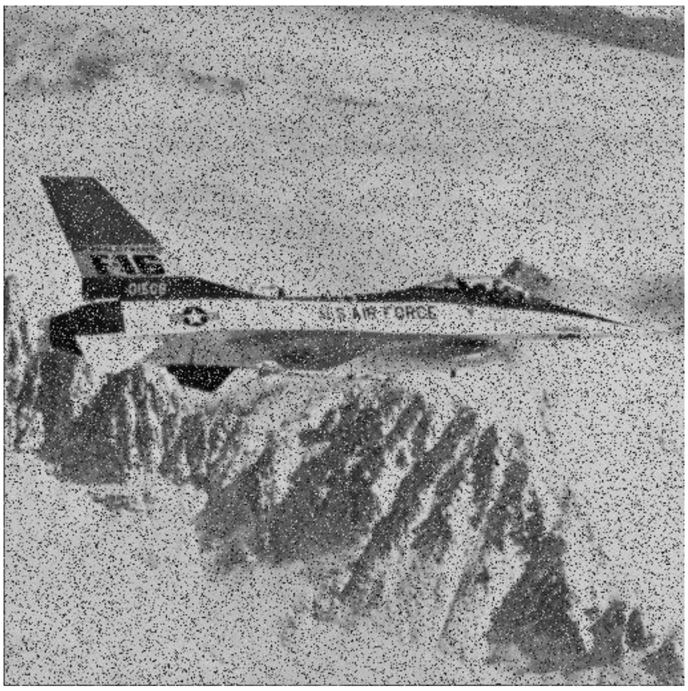
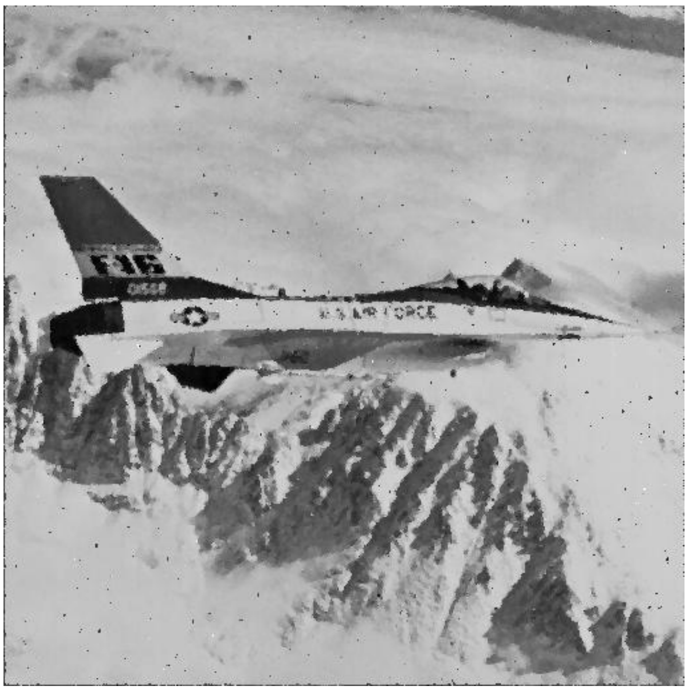


Figure 5.16 Graphs of MSE and PSNR for Median Filter and Neurofilter on Multiplicative Gaussian Noise

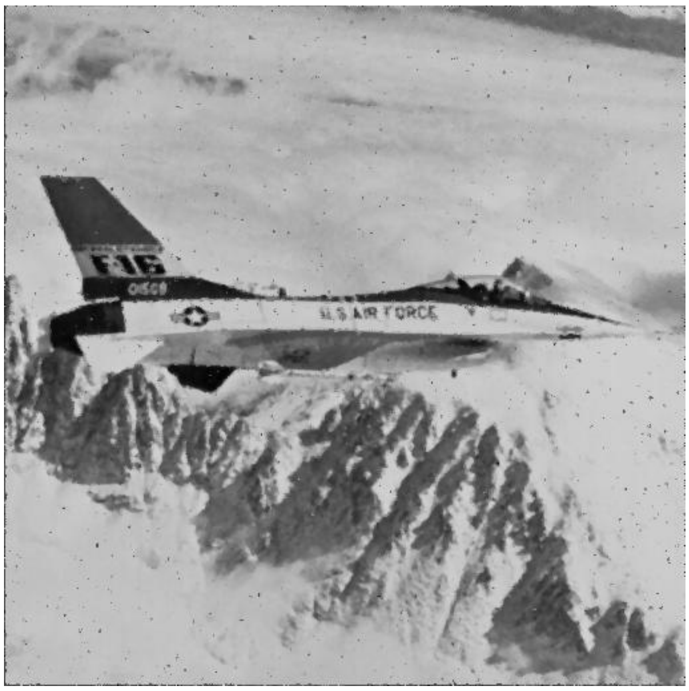
5.4.4 Multiplicative Uniform Noise Filter Results



(a) F-16 with 20% Multiplicative Uniform Noise



(b) Median Filtered



(c) Neurofiltered

Figure 5.17 Images of F-16 Reference with 20% Multiplicative Uniform Noise and Filtered Images

Following the multiplicative Gaussian noise tests the filters were further examined using multiplicative uniform noise to assess their performance. Figure 5.17 (a), shows the F-16 reference image with 20% multiplicative uniform noise. Images in Figure 5.17 (b) and Figure 5.17 (c) show the results after median filtering and neurofiltering respectively. Visual inspection of the noisy images shows lots of pixel noise similar to the ‘pepper’ component of ‘salt and pepper’ noise, which cause a loss of detail in the original image. This simulation confirmed the abilities of both these filters to remove this type of noise effectively. The damaging effects that both filters caused in the multiplicative Gaussian trials are again present. In this simulation the presence of the black pixels (pepper noise) after filtering was more noticeable on the neurofiltered image. It should be noted that the threshold used on all the simulations for the neurofilter was King’s suggested 50%, and later simulations show the ideal value to be less. Probably due to the optimum threshold being dependent on the average greyscale of the image. It is expected that this will improve the performance of the neurofilter. The MSE results in Figure 5.18 show that both filters work but it is clear that the performance of the neurofilter is not as good as the median filter particularly as the noise percentage increases. This is confirmed by the PSNR results.

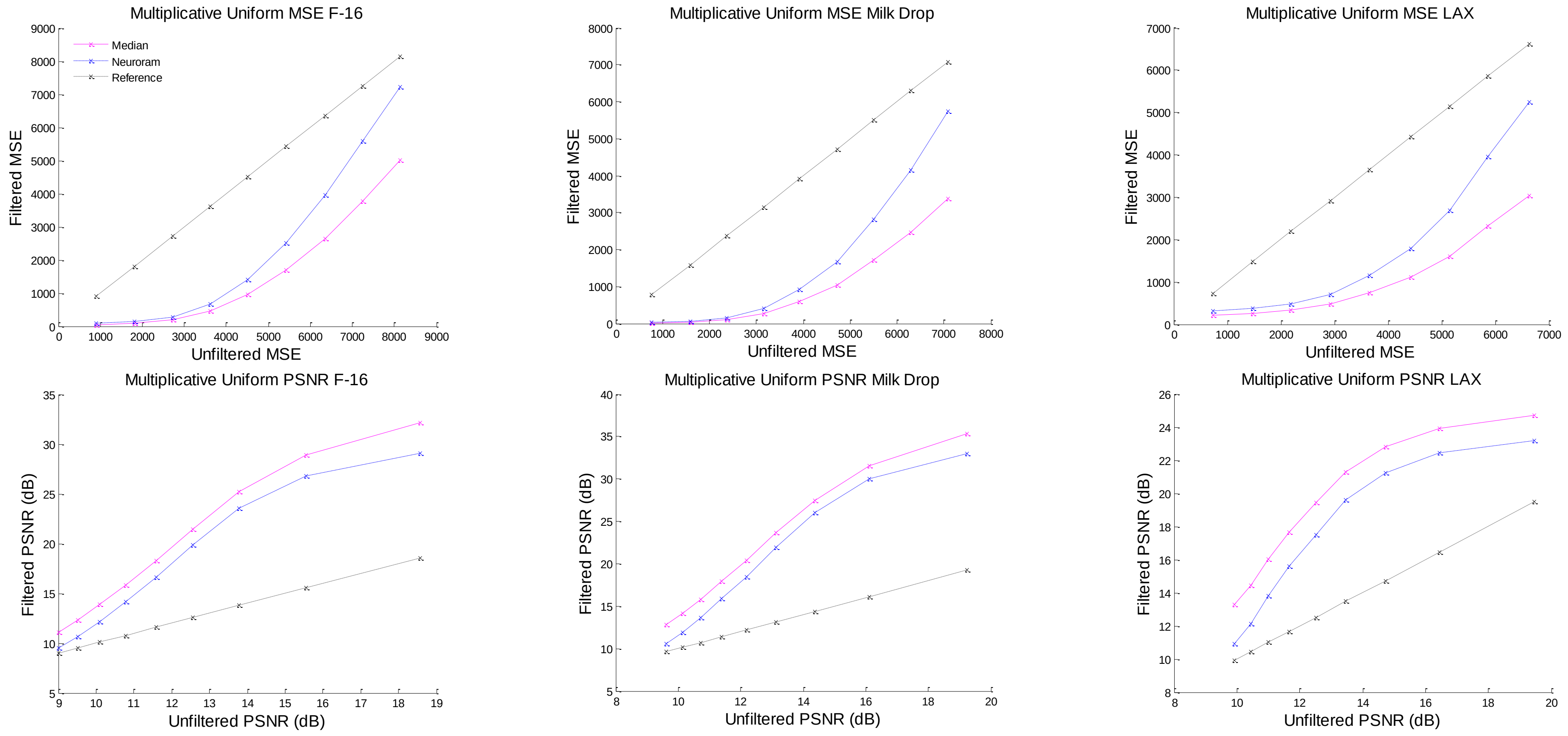
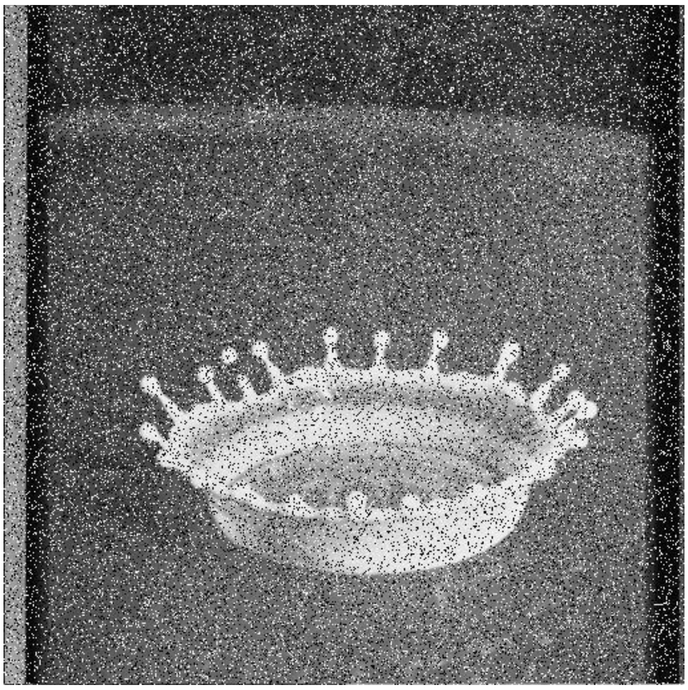
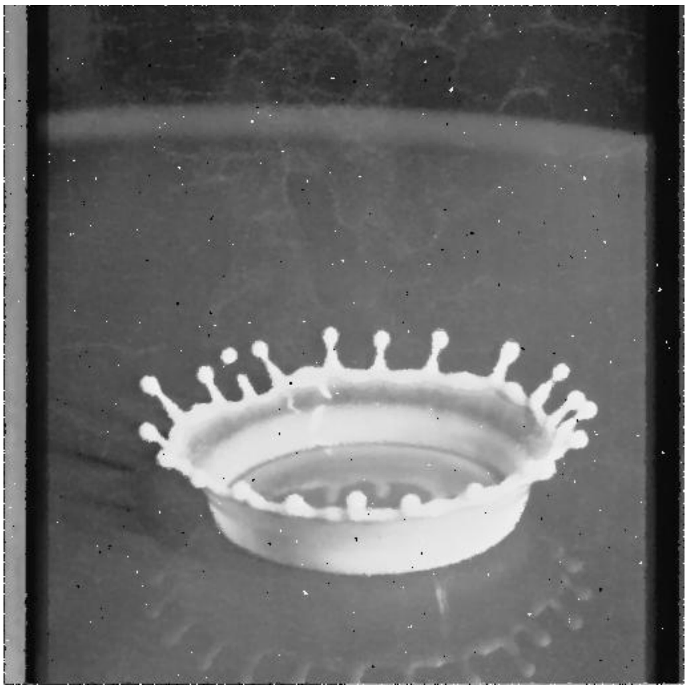


Figure 5.18 Graphs of MSE and PSNR for Median Filter and Neurofilter on Multiplicative Uniform Noise

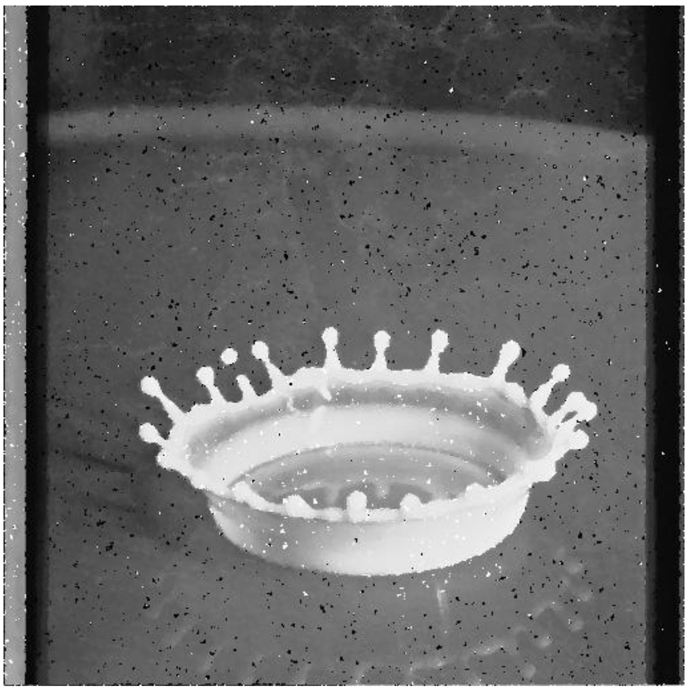
5.4.5 Salt and Pepper Noise Filter Results



(a) Milk Drop with 20% Salt and Pepper Noise



(b) Median Filtered



(c) Neurofiltered

Figure 5.19 Images of Milk Drop Reference with 20% Salt and Pepper Noise and Filtered Images

The salt and pepper noise image processing was repeated in order to evaluate the techniques used with those used by King to demonstrate consistency. Figure 5.19 (a), shows the Milk Drop reference image with 20% salt and pepper noise. Images in Figure 5.19 (b) and Figure 5.19 (c) show the results after median filtering and neurofiltering respectively. It can be seen that both of the filters perform well on the salt and pepper images and the results concur with the findings reported by King. Visually the performance of the median filter gives an improved image with greater noise removal of the salt and pepper noise in comparison to the neurofilter. The neurofilter does have the advantage of maintaining contrast and does not cause blurring which is important in certain applications. The MSE and PSNR shown in Figure 5.20 confirm the visual results showing the performance of the median filter is better.

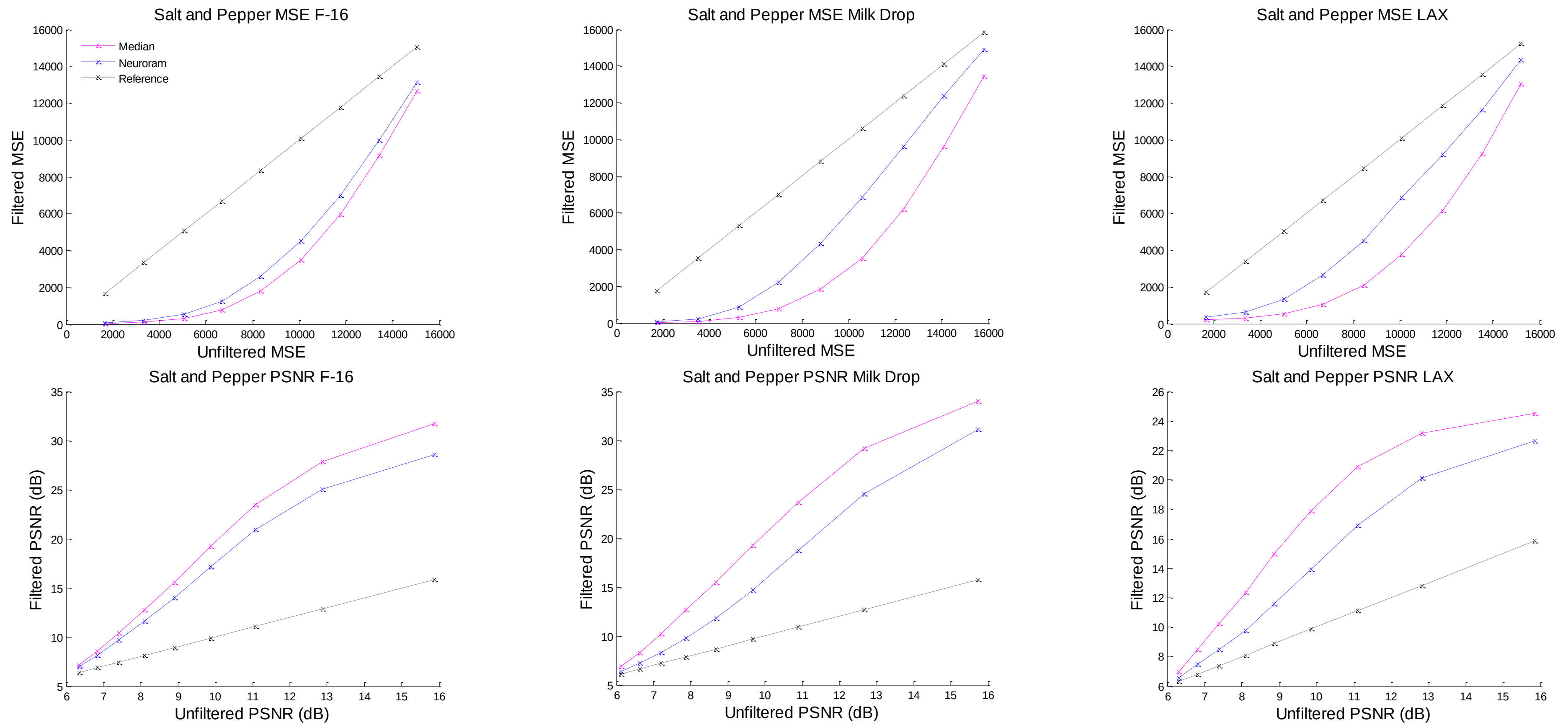


Figure 5.20 Graphs of MSE and PSNR for Median Filter and Neurofilter on Salt and Pepper Noise

It is clear that both the median filter and neurofilter are well suited to the removal of the salt and pepper, multiplicative Gaussian and multiplicative uniform noise. They are not well suited to the removal of additive noise types such as additive Gaussian and additive uniform noise. The MSE and PSNR results show that the median filter outperforms the neurofilter with the exception of lower percentage multiplicative Gaussian noise where the results show the neurofilter performs slightly better. Changing the threshold could further increase this. A visual inspection of the images reflects that the median filter performs better than the neurofilter. Although the distortion caused by filtering could be considered to be less with the neurofilter. The neurofilter causes pixelisation but also improves the edge definition and contrast whereas the median filter just causes blurring.

5.5 Cascade Filter

This novel cascade filter trial was performed to establish the optimal number of filters in line to gain best performance and to identify a balance between performance and real estate. Real estate is the amount of logic required in a semiconductor device. Cascade filtering is implemented by performing multiple sequential filtering on the same image. The image is taken and filtered the resultant filtered image is then taken and filtered again this process is repeated in this case up to 20 times. Both the median filter and neurofilter were tested to establish the difference in their characteristics when cascaded. Both filters used the reference Milk Drop image. Both median filtering and neurofiltering were performed on the reference image using a 3 by 3 filter applied to the whole image. For the neurofilter the threshold level was set to four.

The Milk Drop reference image was used initially with 20% salt and pepper noise and then increased to 30%. This was in order to prove the characteristics of the filters were not noise level related. The simulation was performed using MATLAB simulations. The appropriate level of noise was added to each of the reference images. Each of the noisy images was then processed using either a median filter or the neurofilter. After each level of filtering the measurement criteria of MSE and PSNR were calculated. This iterative filtering was performed using both types of filter, up to twenty times. Each image after filtering was stored along with the measurement criteria data.

Figure 5.21 shows the results of the median filter and neurofilter on the reference Milk Drop image with 20% added salt and pepper noise. From the graphs shown in Figure 5.21 it is clear that the optimum number of filters in cascade for the median filter is three. Although if real estate was a real constraint then two filters would offer a good compromise, with only a slight loss of performance. The optimum level of cascading for the neurofilter is four stages as shown in the figures given in Figure 5.21.

From the median filtered images shown in

Figure 5.22 there is a clear visible improvement in the quality of the images after the second level of filtering. It can be seen that there is no level of improvement from the 20th level of filtering compared to the 3rd level of filtering. The quality of the image after the 20th level of filtering is actually poorer due to the distortion effects caused by the filter. The median filter causes blurring which damages the quality of the image.

The neurofilter shown in Figure 5.21 shows improvements in the image after a greater number of cascade levels in comparison to the median filter. The optimum number of cascaded filters is four, but the damage caused by repeated filtering is less than in the standard median filter. The images in Figure 5.22 show little improvement in the latter images due to the low amount of noise in the images. The neurofilter does not have as profound an effect on the degrading of the image quality as the median filter but does cause pixelisation on the edges. The images in Figure 5.26 show that unlike the median filter where the effect of multiple filtering yields very little improvement in noise removal, after the initial few stages the neurofilter still offers a noticeable difference in noise removal. Although the latter stages of filtering introduce a greater level of inaccuracies which counteract the improvements in noise removal and hence there is little or no improvement in the measurement criteria.

The evaluation of the data shown in Figure 5.21 and Figure 5.24 demonstrates that the performance results of the filters is not noise level related. Although the improvement of noise removal after multiple stages of neuro-filtering is clearer in the latter data set. This is due to the increased distortion in the reference image.

A clear characteristic which is apparent throughout all the trials is that the median filter performs significantly better than the neurofilter. The performance of the two filters differs and the way the filters operate in a cascade formation yields some distinct differences. The main difference being in the optimum number of cascades of filter which yields best effect. Ideally three median filters in cascade offers the most improvement; further increasing the number of filters has a detrimental effect, as this causes blurring of the images and loss of high frequency detail. The optimum number of cascades for the neurofilter is four. Although further increasing the number of filters in cascade removes more noise unlike the median filter where there is little or no improvement after the third stage of filtering. The other difference is the amount and type of damage caused by multiple stages of filtering. The median filter unlike the neurofilter does not cause blurring, although it does cause pixelisation particularly on image edges. In contrast the neurofilter does not cause as much high frequency loss as the median filter.

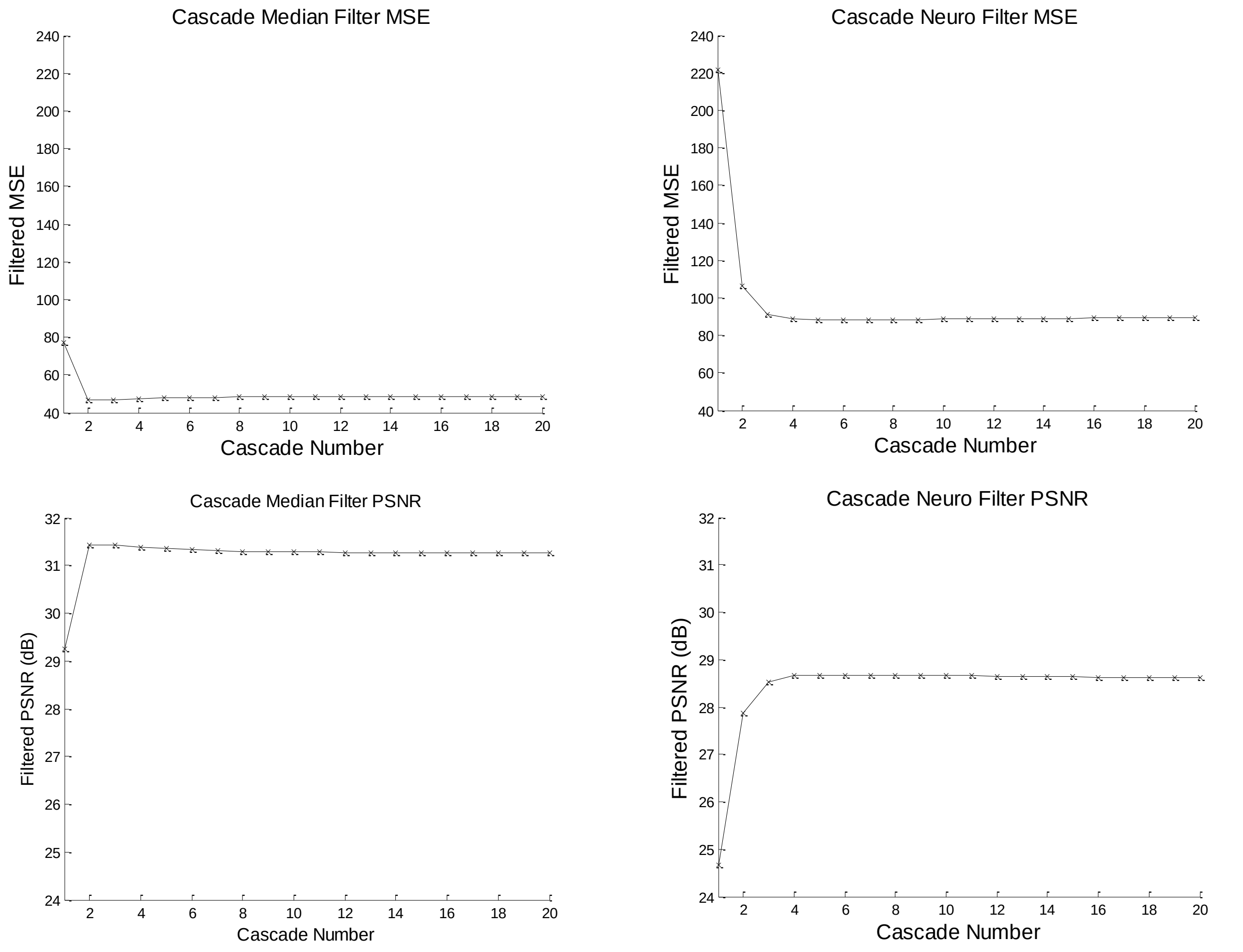


Figure 5.21 MSE and PSNR Results for Cascade of Median Filters and Neurofilters on Milk Drop with 20% Added Salt and Pepper Noise

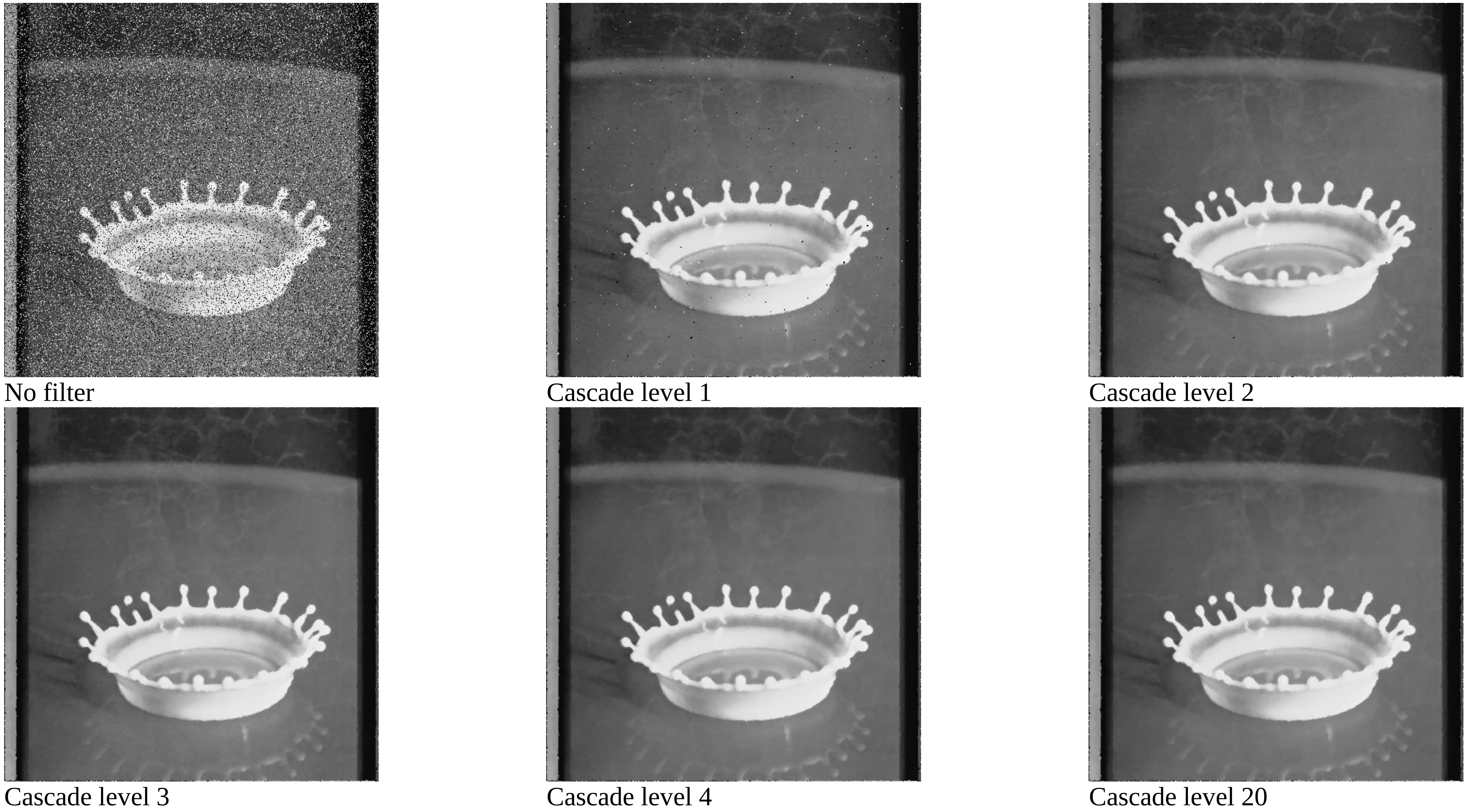
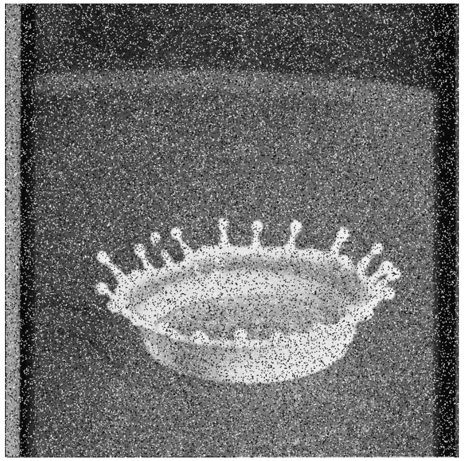
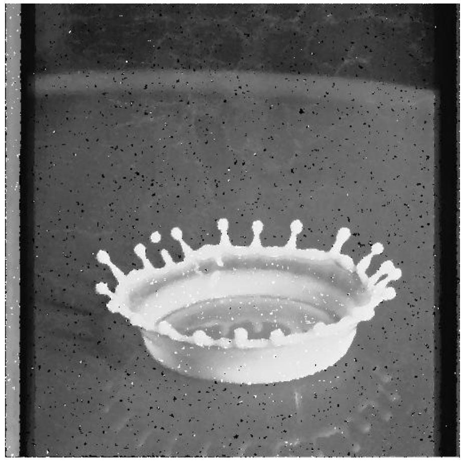
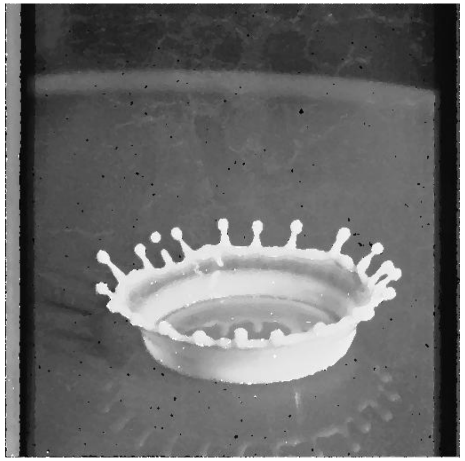
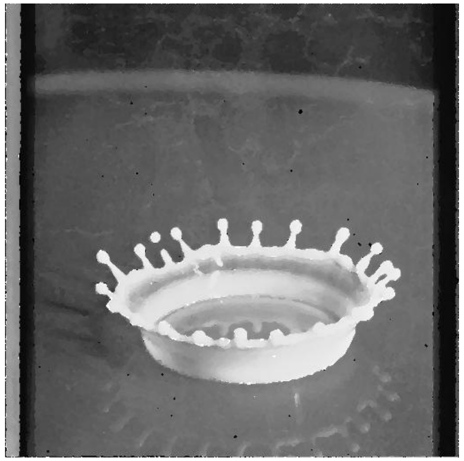


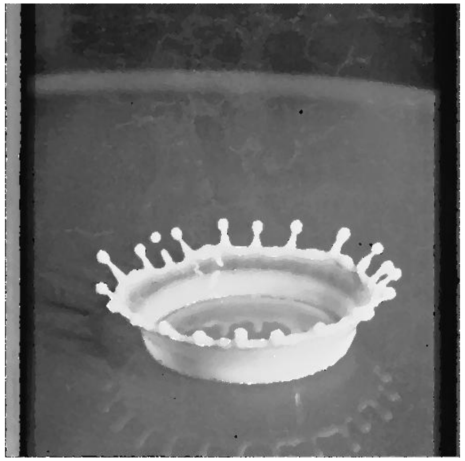
Figure 5.22 Cascade of Median Filters on Milk Drop with 20% Added Salt and Pepper Noise

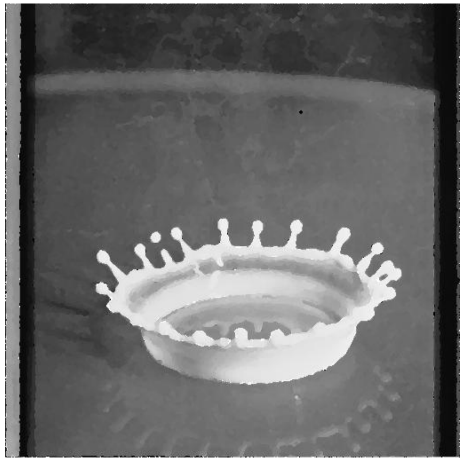

No filter


Cascade level 1


Cascade level 2


Cascade level 3


Cascade level 4


Cascade level 20

5.23 Cascade of Neurofilters on Milk Drop with 20% Added Salt and Pepper Noise

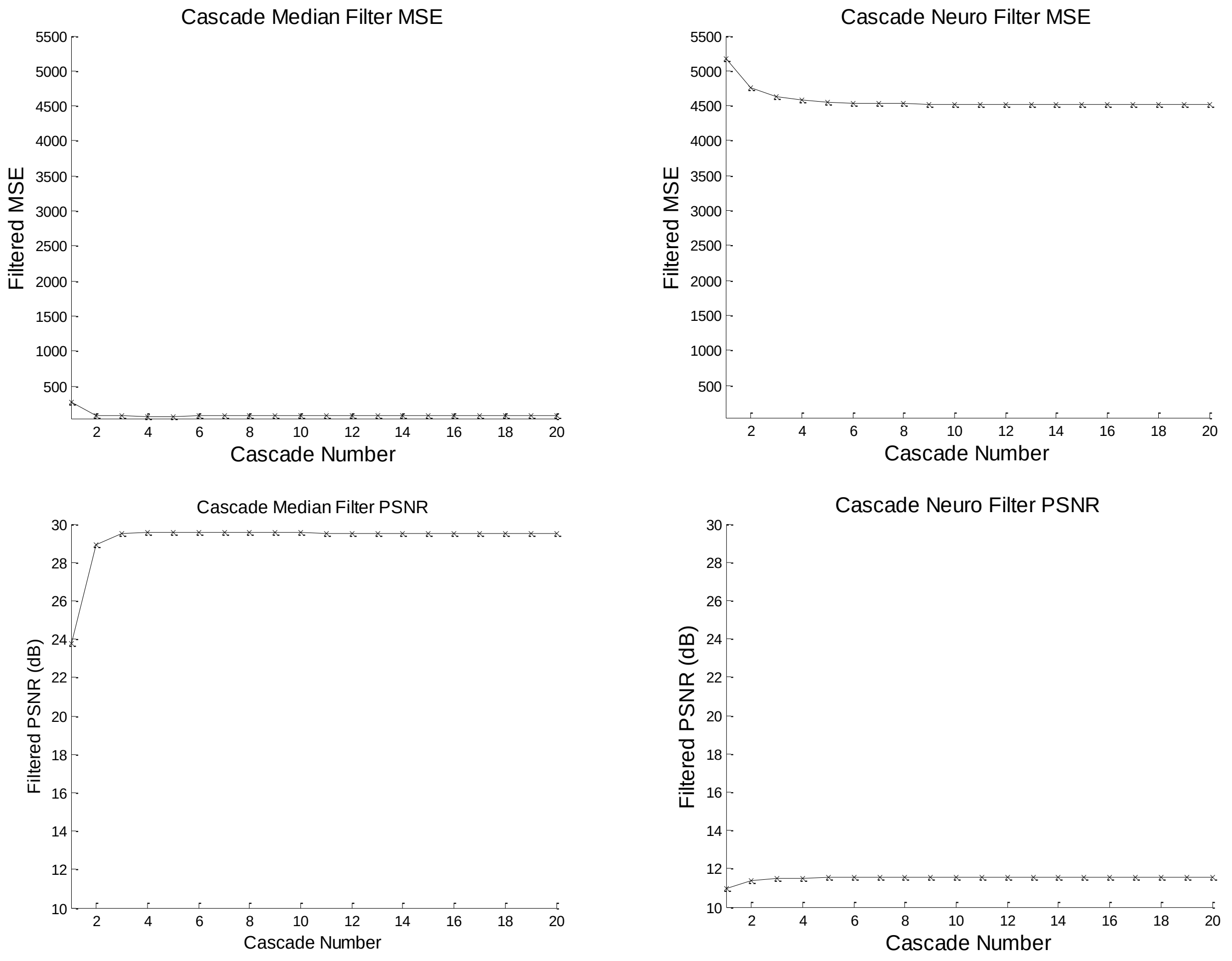


Figure 5.24 MSE and PSNR results for Cascade of Median Filters and Neurofilters on Milk Drop with 30% Added Salt and Pepper Noise

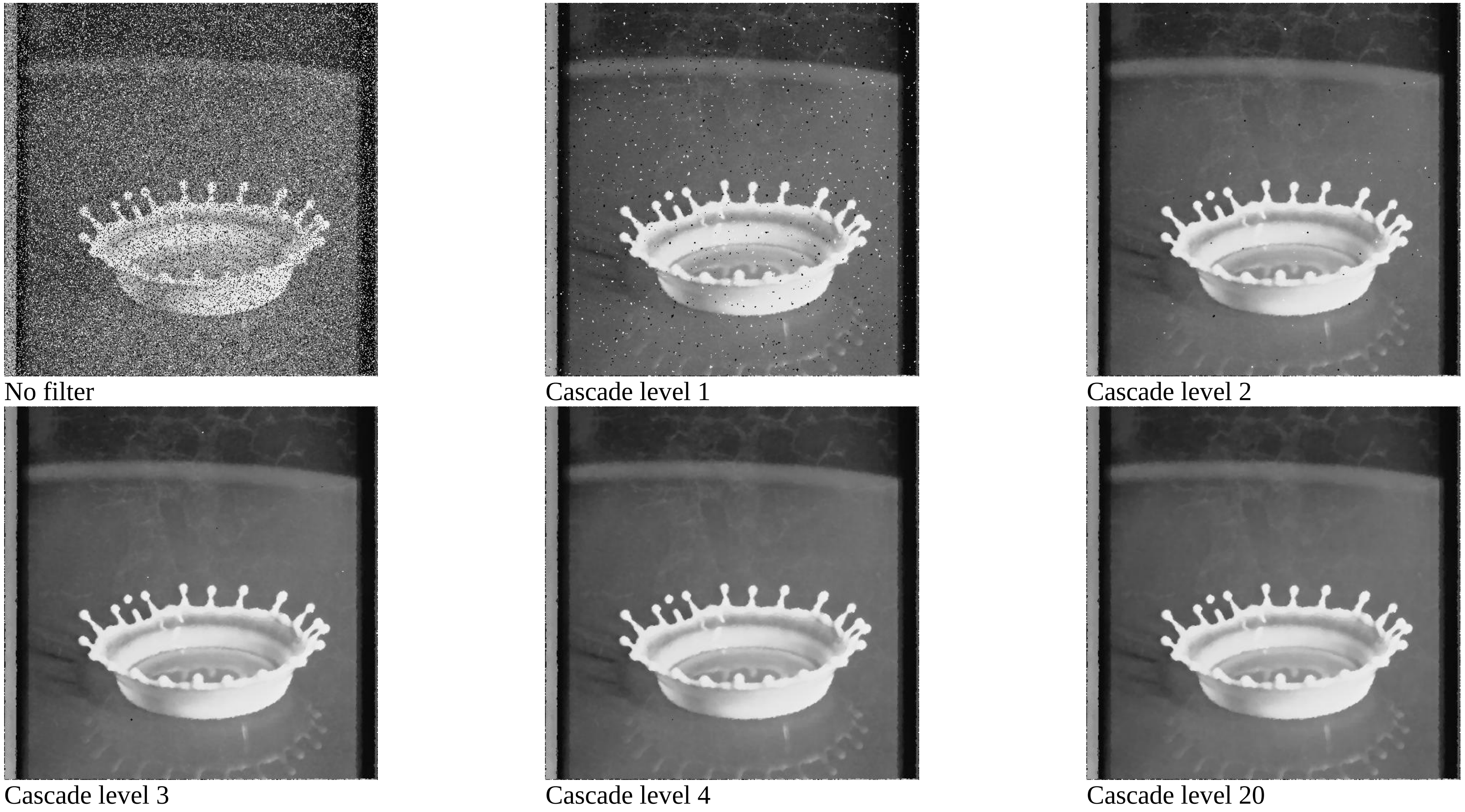


Figure 5.25 Cascade of Median Filters on Milk Drop with 30% Added Salt and Pepper Noise

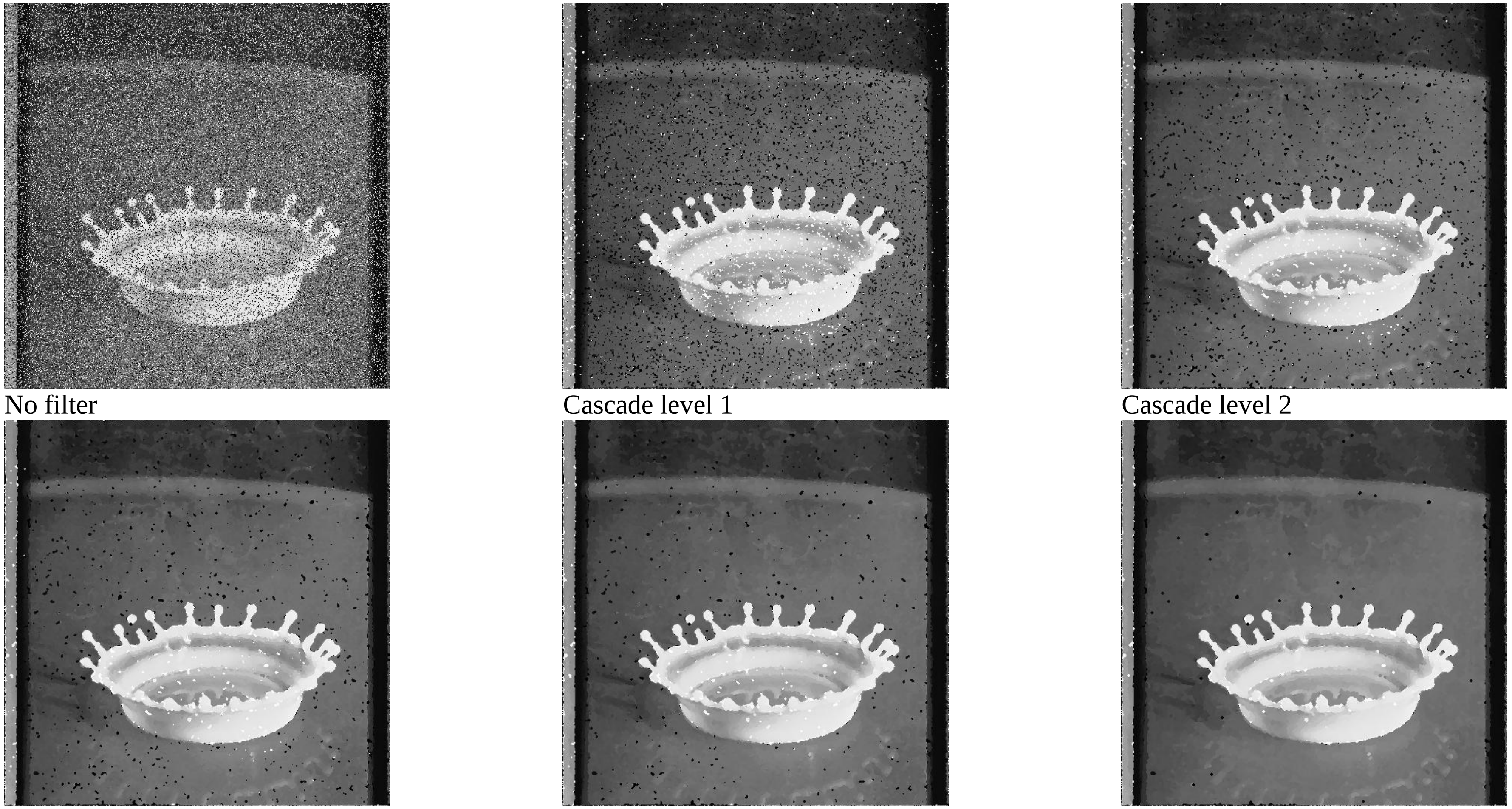


Figure 5.26 Cascade of Neurofilters on Milk Drop with 30% Added Salt and Pepper Noise

5.6 Threshold Assessment

The threshold component was examined to King's prediction, which states the ideal threshold value for optimum performance of Neuroram is approximately half the number of exemplars used, although he qualifies this accepting that this is data dependent [King 2000]. In this case the exemplar size is nine, because the filter is being implemented using a 3 by 3 window on the image. Therefore with this exemplar size we would expect the optimum threshold would be either 4 or 5. The trial was performed again using the standard reference images referred to at the beginning of this chapter. The standard MATLAB implementation of Neuroram was used. In order to prove this hypothesis several different reference images were taken to ensure the results were not image dependent. The uniform multiplicative image was repeated with two noise levels to ensure that the noise value did not affect the results.

The threshold on the images was set between 1 and 8 and the value of the filtered image compared with the reference image using the measurement criteria of MSE and PSNR. The results for each threshold for a given image were plotted for the MSE and PSNR values. The filtered images were also stored for visual comparison.

The first trial was performed on the LAX reference image with 20% additive Gaussian noise. The image was filtered with the neurofilter using thresholds ranging from 1 to 8 and the subsequent PSNR and MSE then plotted as shown in Figure 5.27. The diagrams in Figure 5.28 show the effects of the filter with different threshold. It can be seen that at lower thresholds the definition of the image is low, the contrast is poor only having two levels of grey. As the threshold is increased more detail is shown with an optimum threshold being about three or four. This is reflected in the MSE and PSNR graphs shown in Figure 5.27. It is hard to judge the ideal because the performance of the filter on this type of noise is poor as demonstrated by the results given earlier in this chapter.

The second trial was performed on the Milk Drop reference image with 20% added multiplicative uniform noise. Figure 5.30 shows the filtered images, it can be seen at the low threshold values that the images still contain a lot of white noise. Whereas at the high threshold values the images contain a lot of black noise. The image with the least noise is that which has been filtered with the neurofilter with a threshold value of three. This is confirmed with the results of the MSE and PSNR values given in Figure 5.29.

The third trial performs the same test as the second one but this time with a noise level of 40%. The effect of the neurofilter on the images is the same as the second test as expected and is shown in Figure 5.32. Again the results show the threshold value for optimum filtering of the neurofilter is three. This is confirmed by the MSE and PSNR results given in Figure 5.31. This trial was performed to demonstrate that the filter's operation with respect to its threshold was not noise level dependent.

The final trial was performed on the Milk Drop reference image with 20% added salt and pepper noise. The results for this trial with respect to threshold concur with those of the previous two trials again confirming that the filter operation is fixed and are not noise type or level dependent. Again the low threshold values leave a lot of white noise and the high threshold values leave a lot of black noise. The images and the graphs in Figure 5.33 and Figure 5.34 confirm that the optimum threshold for this neurofilter for this task is three.

Therefore this study demonstrates that the optimum threshold for Neuroram when configured as an image filter operating on a 3 by 3 matrix giving nine data exemplars is three. This gives a threshold percentage of 33%, this differs from the value of approximately 50% claimed by King [King 2000].

The reason for this difference in results from King's original prediction of approximately 50% is due to the limited amount of data that the prediction was made on. The threshold level is only determined by one graph with one dataset. On closer examination of this graph in King's thesis, it shows that the 50% value was not the highest response, however the threshold around 40% gave the highest response. The graph shows how values in the region of between 35% and 55% all give a response significantly higher. The difference between responses is relatively small though. The conclusion from this is that the original prediction was done on a limited data set as well as the prediction is only a rough one rather than an absolute value. In order to determine the true optimum threshold value further research is required on larger data set sizes as this initial research would suggest the value is data dependent.

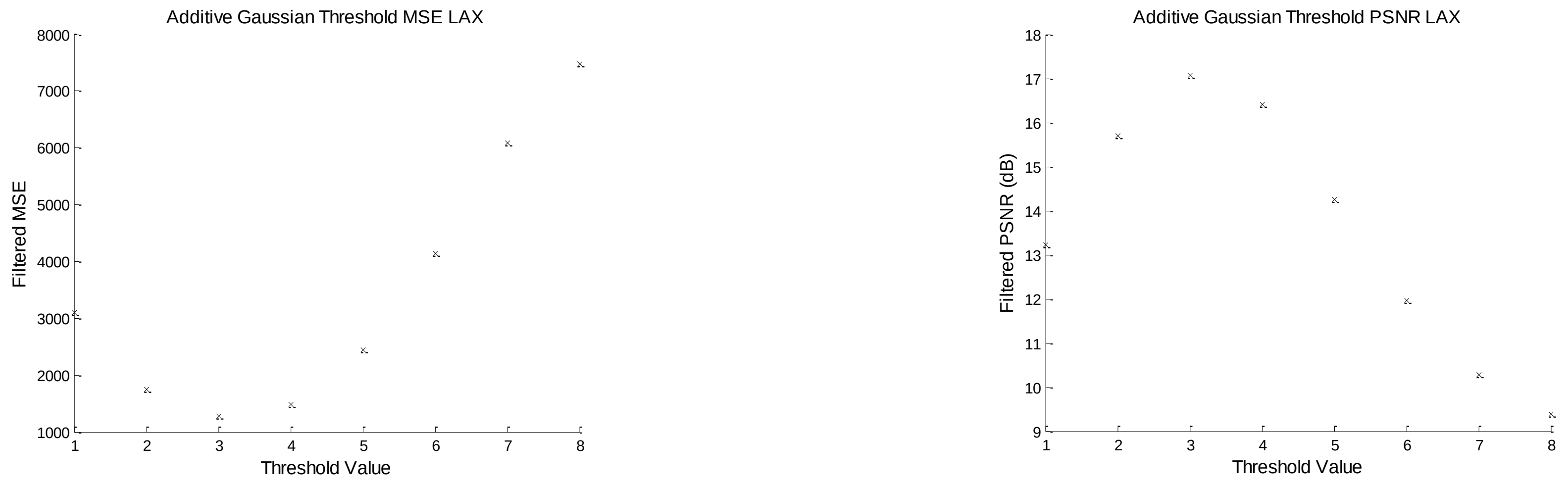
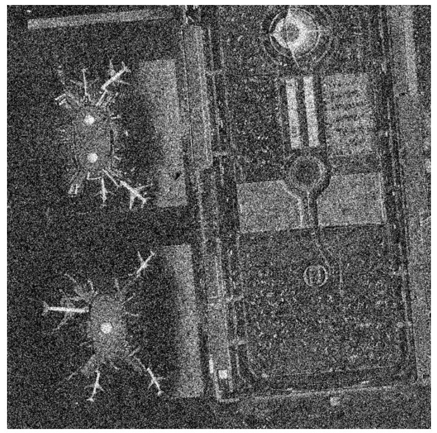
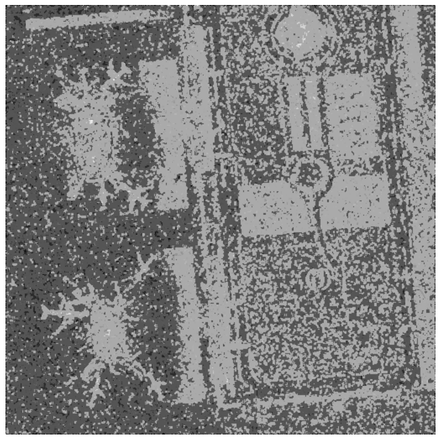


Figure 5.27 MSE and PSNR for Filter Thresholds on LAX Image with 20% Additive Gaussian Noise


No filter


Filtered threshold 1

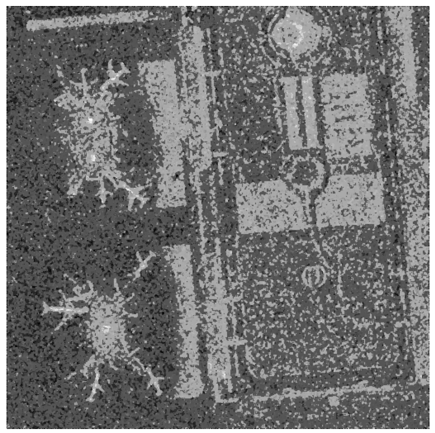
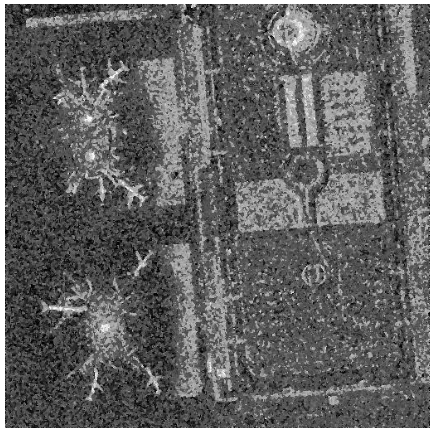
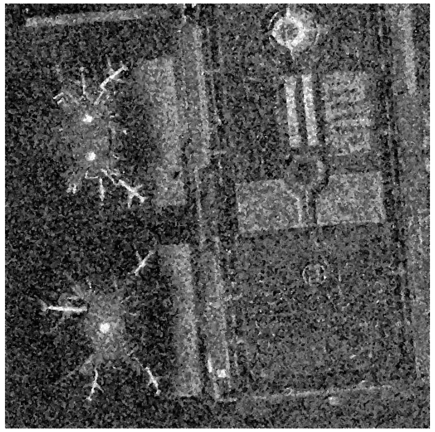
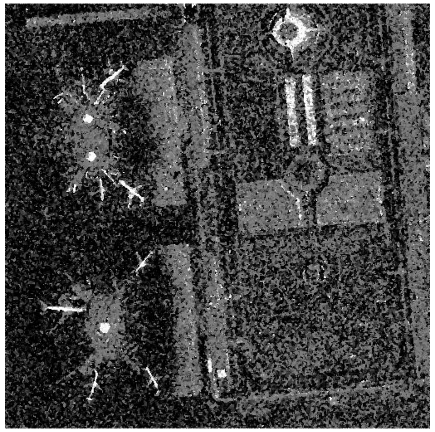
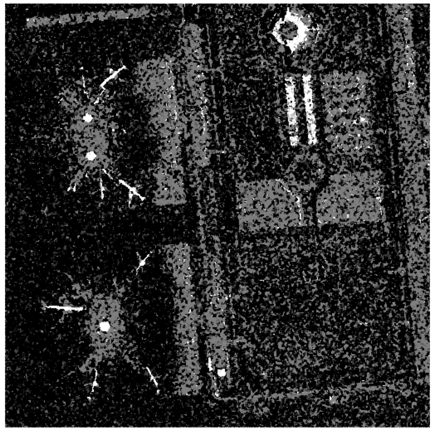

Filtered threshold 2

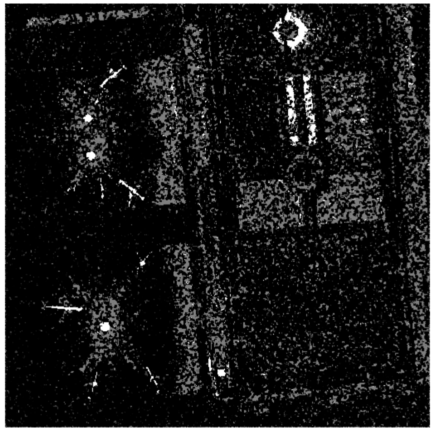
Figure 5.28 Filtered LAX Image with 20% Additive Gaussian Noise with Different Thresholds


Filtered threshold 3


Filtered threshold 4


Filtered threshold 5


Filtered threshold 6


Filtered threshold 7

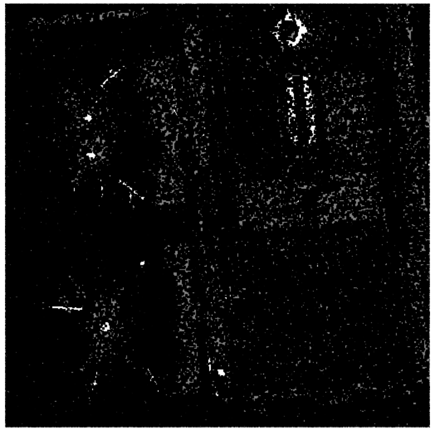

Filtered threshold 8

Figure 5.28 Filtered LAX Image with 20% Additive Gaussian Noise with Different Thresholds

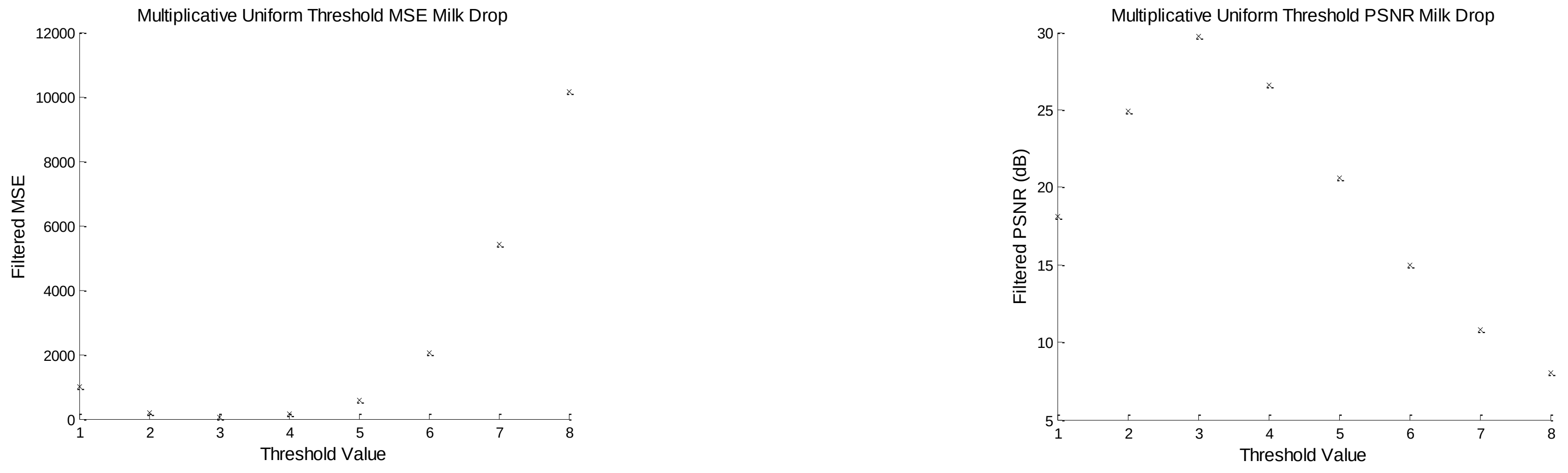
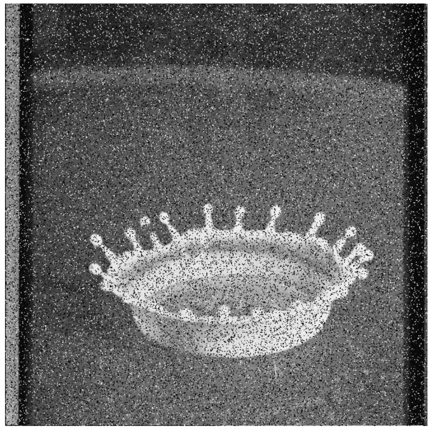
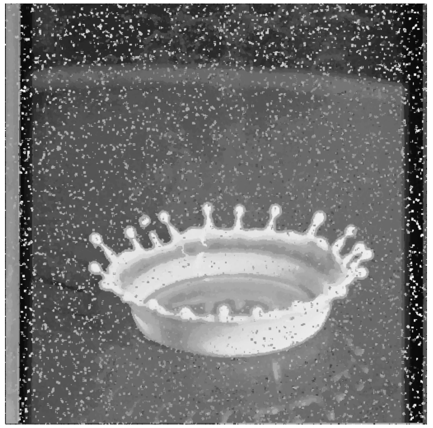


Figure 5.29 MSE and PSNR for Filter Thresholds on Milk Drop Image with 20% Multiplicative Uniform Noise


No filter


Filtered threshold 1

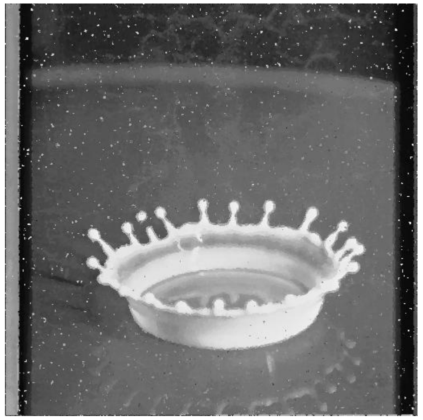
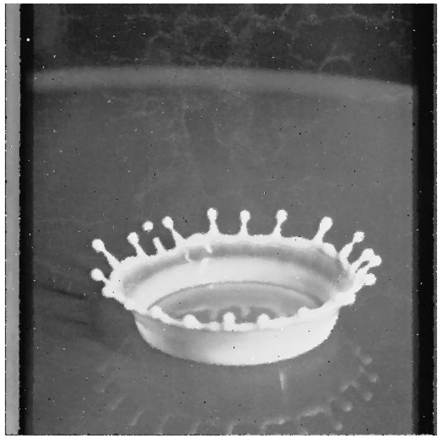
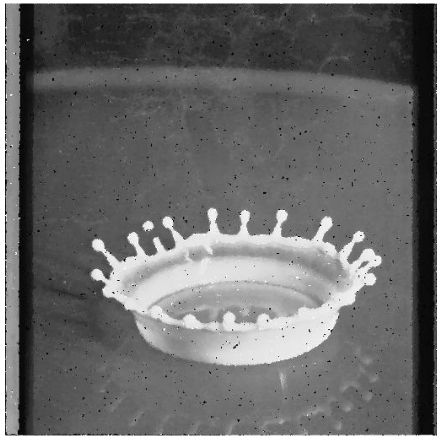
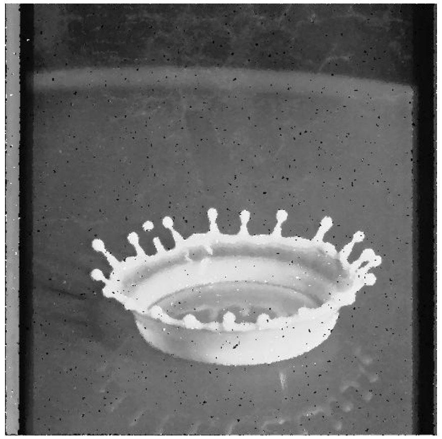
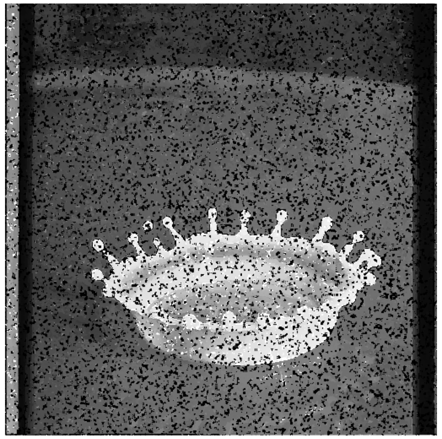

Filtered threshold 2

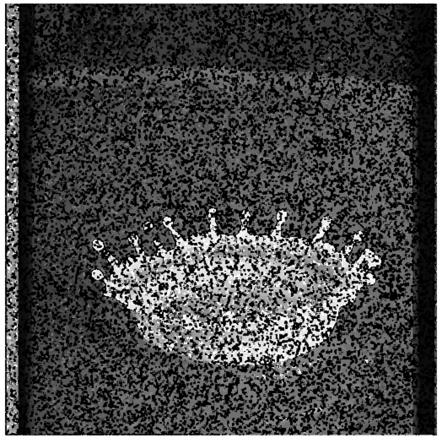
Figure 5.30 Filtered Milk Drop Image with 20% Multiplicative Uniform Noise using Different Thresholds


Filtered threshold 3


Filtered threshold 4


Filtered threshold 5


Filtered threshold 6


Filtered threshold 7

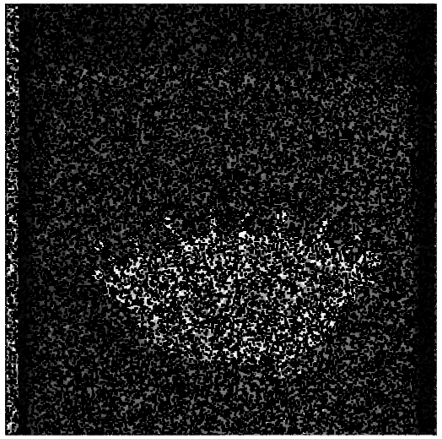

Filtered threshold 8

Figure 5.30 Filtered Milk Drop Image with 20% Multiplicative Uniform Noise using Different Thresholds

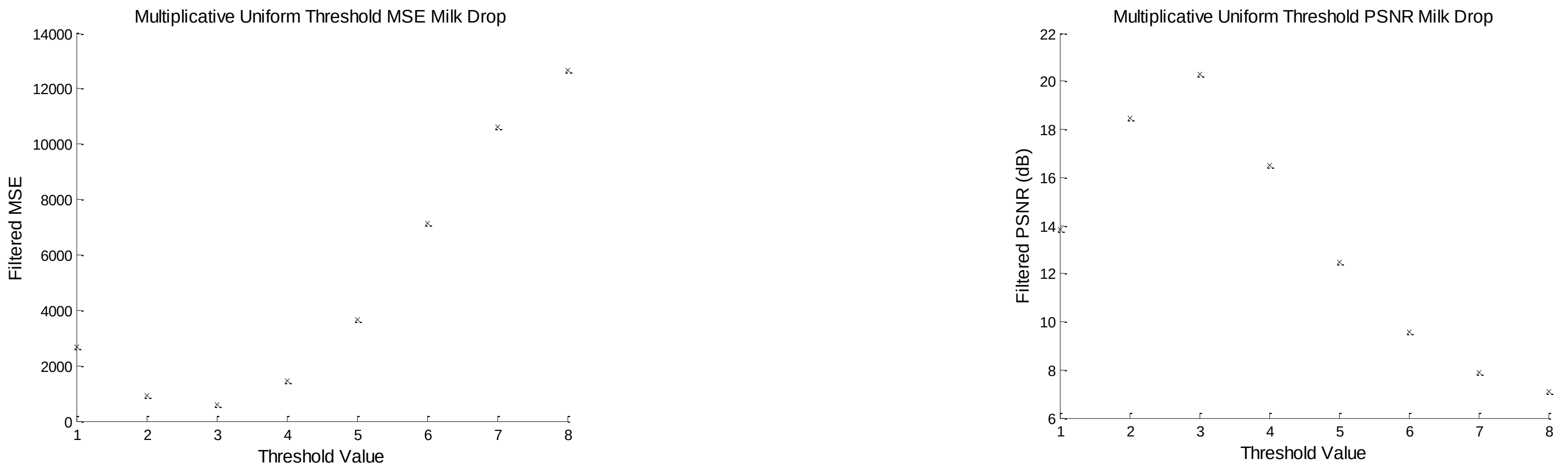
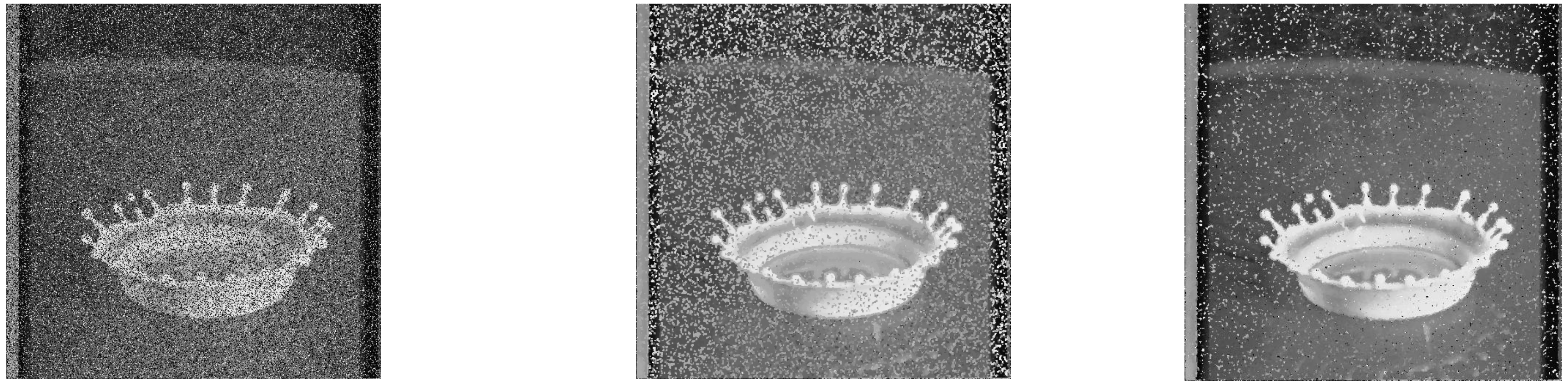
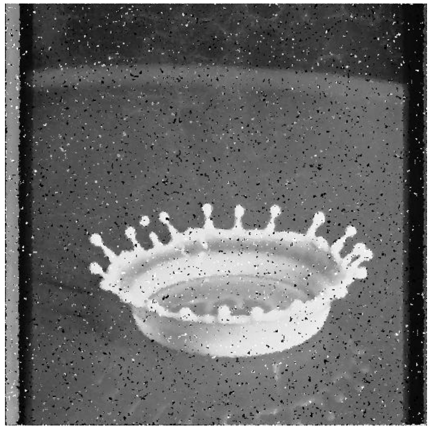


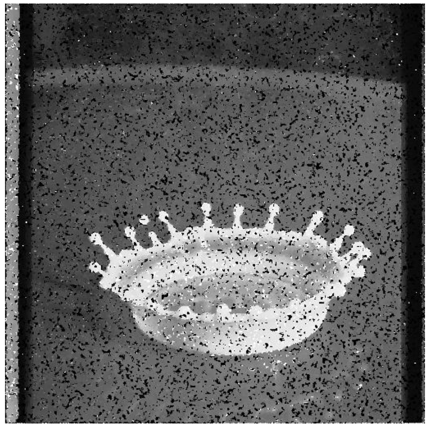
Figure 5.31 MSE and PSNR for Filter Thresholds on Milk Drop Image with 40% Multiplicative Uniform Noise

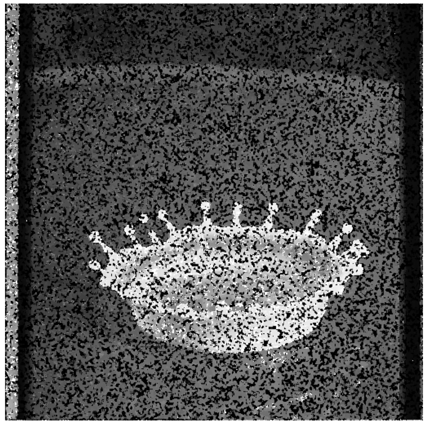


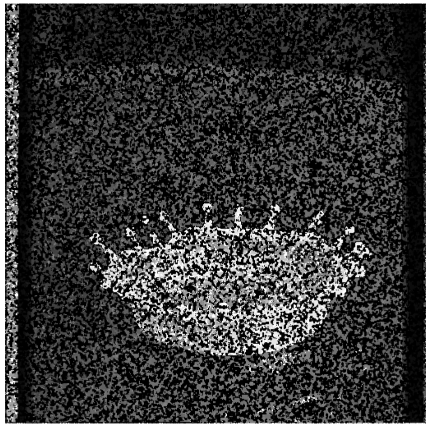
No filter Filtered threshold 1 Filtered threshold 2

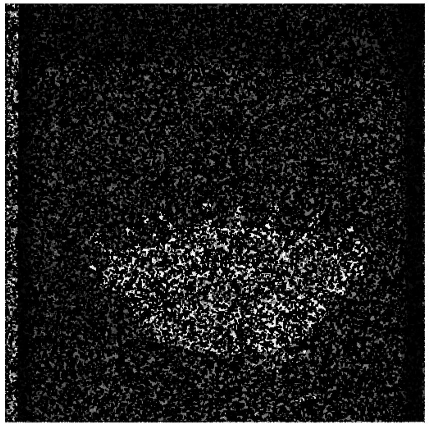
Figure 5.32 Filtered Milk Drop Image with 40% Multiplicative Uniform Noise using Different Thresholds


Filtered threshold 3


Filtered threshold 4


Filtered threshold 5


Filtered threshold 6


Filtered threshold 7

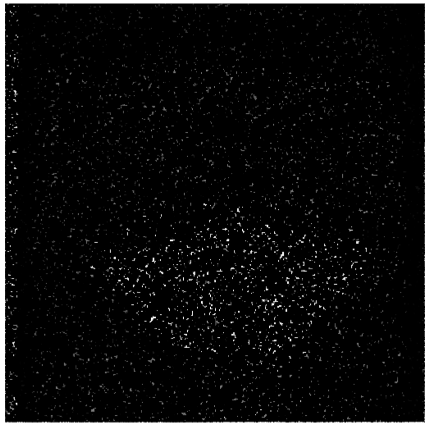

Filtered threshold 8

Figure 5.32 Filtered Milk Drop Image with 40% Multiplicative Uniform Noise using Different Thresholds

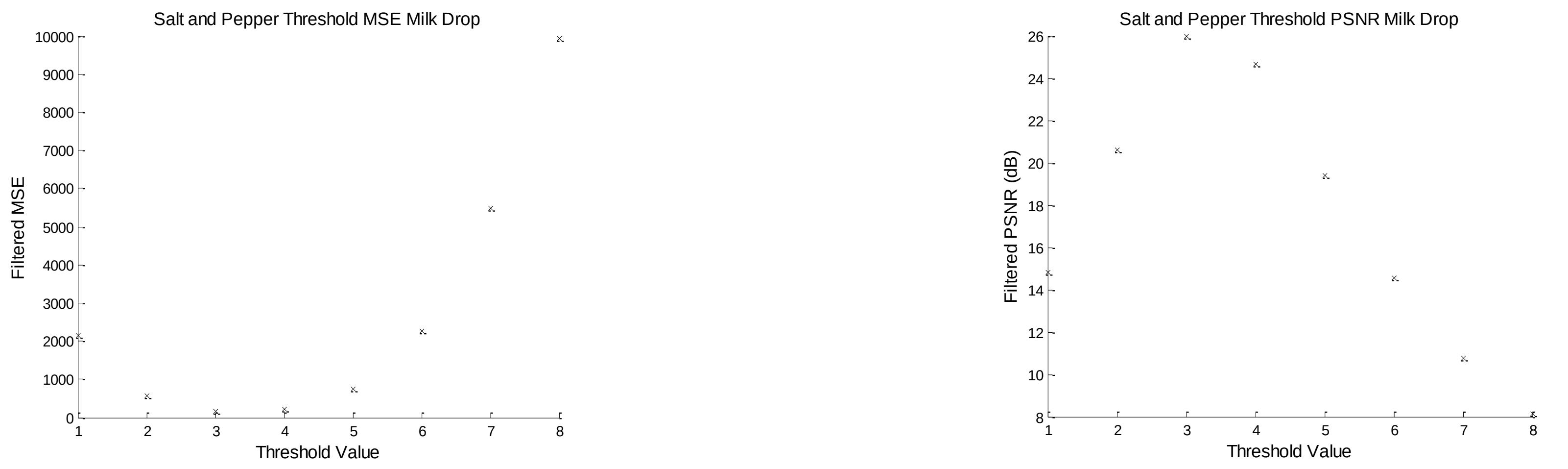
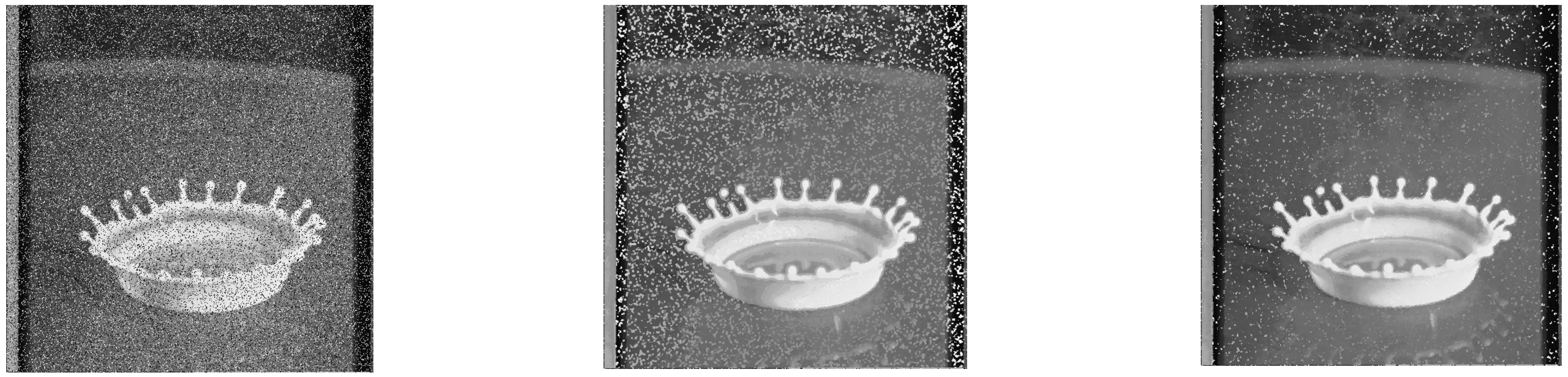
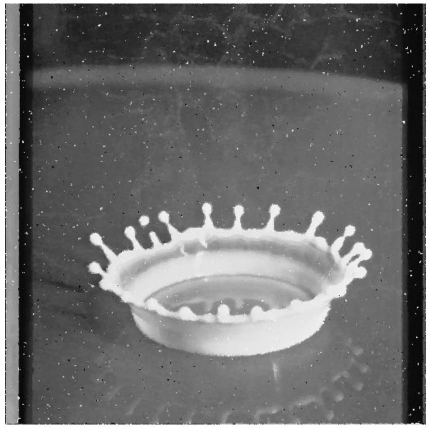


Figure 5.33 MSE and PSNR for Filter Thresholds on Milk Drop Image with 20% Salt and Pepper Noise

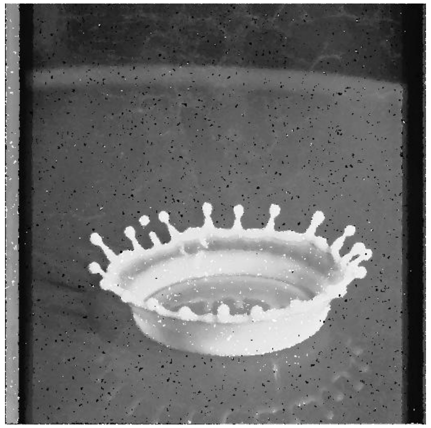


No filter Filtered threshold 1 Filtered threshold 2

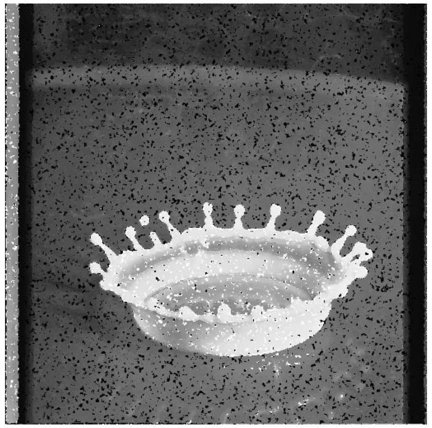
Figure 5.34 Filtered Milk Drop Image with 20% Salt and Pepper Noise using Different Thresholds



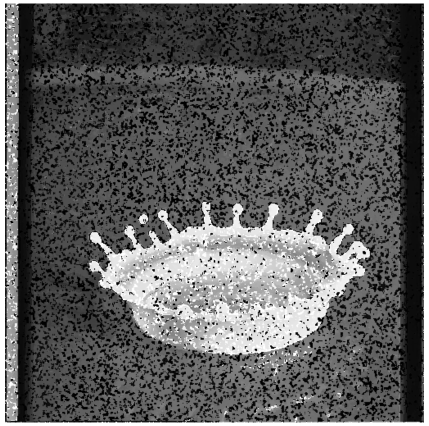
Filtered threshold 3



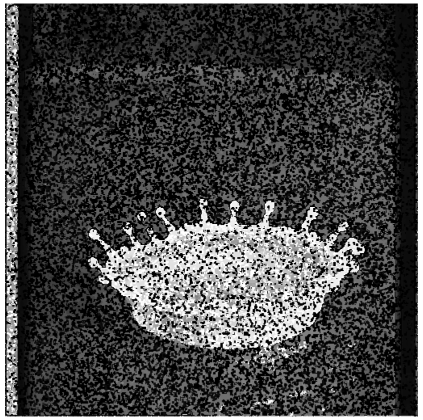
Filtered threshold 4



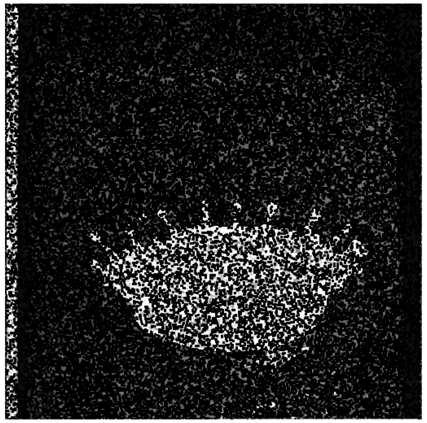
Filtered threshold 5



Filtered threshold 6



Filtered threshold 7



Filtered threshold 8

Figure 5.34 Filtered Milk Drop Image with 20% Salt and Pepper Noise using Different Threshold

Chapter 6

6 Boolean Weightless Self Ordered Map

6.1 Overview

This chapter describes a weightless Boolean implementation of Kohonen's self ordered map. A brief description of a conventional self ordered map and its associated algorithms is given. This has not been covered in the earlier chapters because the self ordered map is traditionally a weighted neural network, often implemented using an algorithmic approach. An overview of the novel weightless Boolean elements used is given proceded by the structures that can be formed to implement the Weightless Self Ordered Map (WSOM). Finally an overall architecture of the WSOM is presented including discussion of the MATLAB simulation.

6.2 The History of the Self Ordered Map

Although the idea of self organisation was proposed in 1973 by Von der Malsburg, it was not until the mid 1970's that Willshaw and Von der Malsburg produced computer models [von der Malsburg 1973, Willshaw et al 1970]. Their inspiration was to produce a biologically plausible visual system that could determine edge orientation and light intensity similar to that found in the visual cortex. The Self Ordered Map (SOM) was finally developed by Teuvo Kohonen a Finnish scientist in 1988 [Kohonen 1984, Kohonen 1988]. He used the earlier concepts of self organisation proposed by Willshaw and Von der Malsburg.

6.3 Learning Types

The SOM is an 'unsupervised learning' neural network meaning the network learns from the frequency of the experienced data rather than having a teacher. This differs from the other neural networks featured so far, as these require input and output data to be trained into the network.

6.4 Operation of the Self Ordered Map

The basic function of the SOM comes from the collection of differing neurons within the network 'firing' in response to different stimuli. A highly simplified example of Kohonen's map, based on Aleksander's description is given [Aleksander et al 1995]. The basic example uses a structure of 12 neurons or nodes each of which consist of 4 bits. The graphical representation shows a '1' state as a black square, conversely a white square represents a '0' state. The input data or input stimuli are presented to all the nodes in the network. The two patterns which the network is going to learn are 0111 and 1001 and are graphically represented in a box, and filled top left to bottom right. Each of the 12 neurons in the network has weights; in this case they are either '1' or '0' also represented by black and white squares. The weights are arranged in a similar manner to the input data. Each of the element's weights correspond to a similar bit in the input pattern and correspond to one of the bits in the input pattern. Initially the weights of the neuron are randomly set. The system 'learns' by allocating areas of the network, known as a neighbourhood which correspond to certain input stimuli. The first input 0111 is presented to the untrained network to assess which node gives the maximum response; in this case it is neuron 1 or node 1 as shown in Figure 6.1.

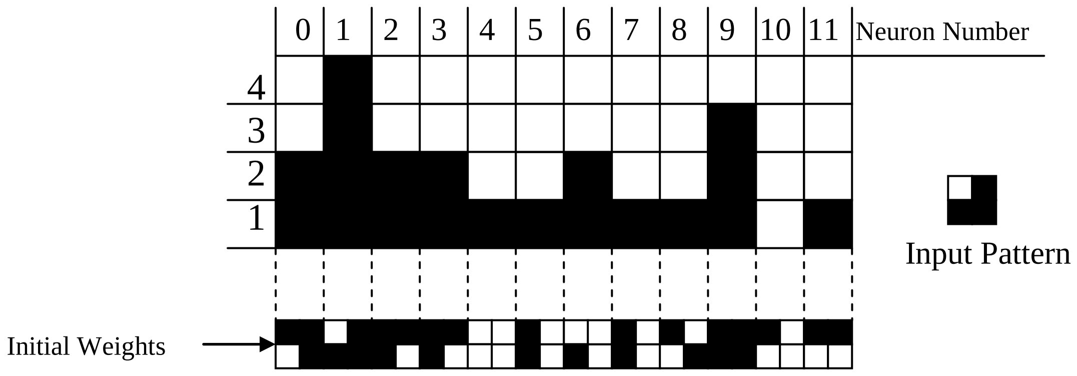


Figure 6.1 Initial Response to Input Pattern 0111

Similarly when the input 1001 is shown to the untrained network, the maximum response is from neuron 8 or node 8 as shown in Figure 6.2.

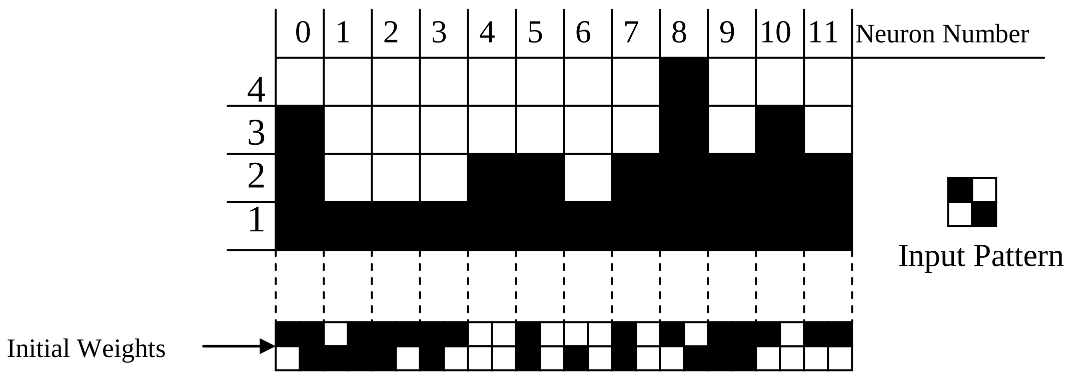


Figure 6.2 Initial Response to Input Pattern 1001

The next step is to identify a suitable neighbourhood around that neuron; this will be the neighbourhood size. In this example the criteria being applied are to strengthen the two neurons either side of the maximum responding neuron; these being 6, 7, 9 and 10. Now the neighbourhood has been identified the next stage is to apply a learning rule, this is usually a strengthening rule. However to apply the standard ‘Mexican hat’ learning parameter the neurons on the periphery will be weakened [Beale et al 1997]. Here we are just applying the strengthening rule to the identified neurons to increase their response to the input stimuli. This is achieved by changing the weights to improve the correlation by reducing the number of bits difference between the identified neurons and the input pattern; in this case the reduction is by one bit. This action is performed on all the identified neurons in the associated neighbourhood as shown in Figure 6.3; this shows the effects of training around neuron 8.

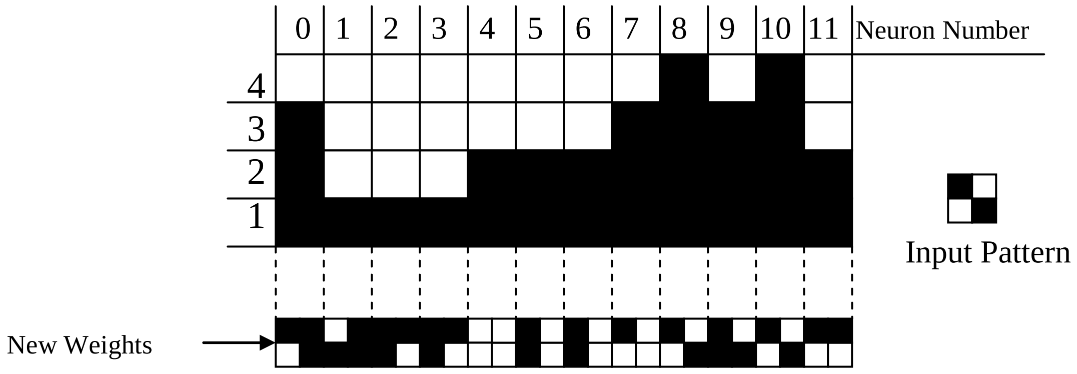


Figure 6.3 Training Neuron 8

The next input 0111 is then tested on the network, where results in a maximum response from neuron 1 as shown in Figure 6.4.

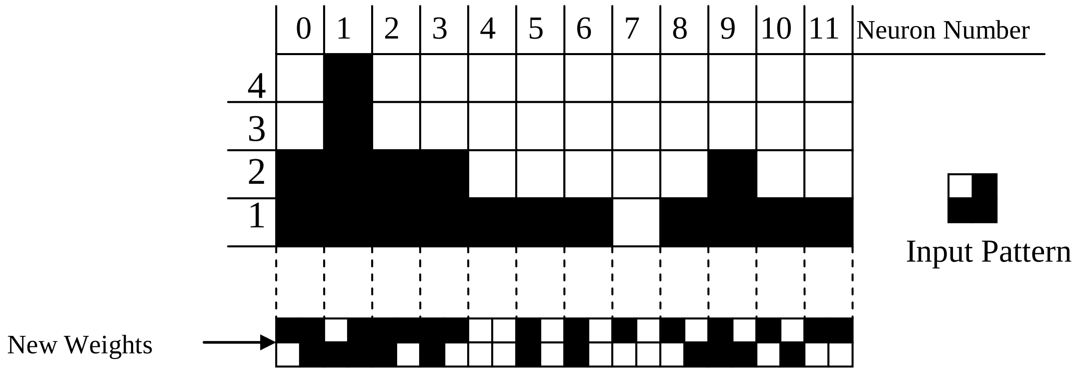


Figure 6.4 Response after Training on Pattern 0111

The same neighbourhood criterion is applied; however this exceeds the edge of the network. Therefore we wrap the data around the edge of the network to avoid creating edge effects, and the necessary special rules which need to be applied to negate them. Therefore the neurons which form the neighbourhood are 0, 1, 2, 3 and 11. The strengthening rule is then re-applied on the identified neighbourhood, the results of which can be seen in Figure 6.5.

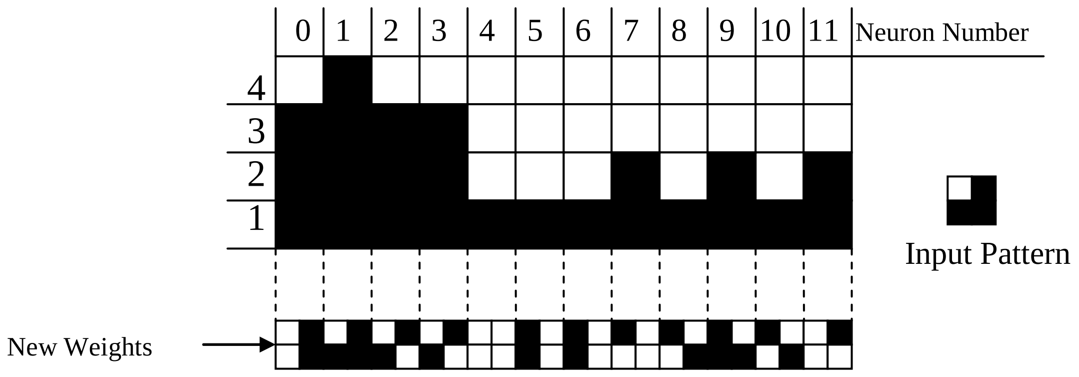


Figure 6.5 Training Neuron 1

The result of the training on the basic network is that two distinct areas of the network respond to the differing inputs, this is the principle of the SOM. This is demonstrated in Figure 6.6 below where the initially trained value 1001 is presented back to the network and the higher numbered node region of the network responds well; conversely when 0111 is presented the lower numbered region of the network responds well.

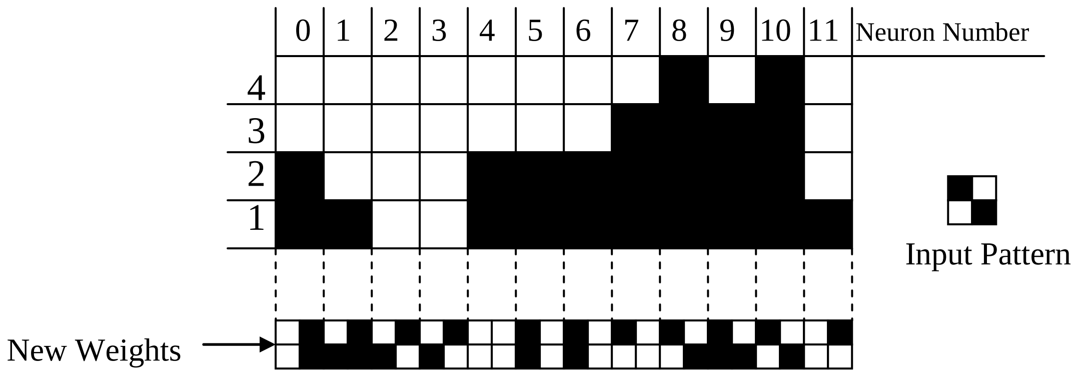


Figure 6.6 Training Neuron 2

6.5 Algorithmic Equations for the Implementation of the SOM

The SOM learning process is often implemented algorithmically using the following equations as described by Beale [Beale et al 1997]. The equations are a list of learning rules applied in stages.

Step 1

This describes the initialisation of the network and is given by Equation 6.1.

$$W_{ij}(t)(0 \leq i \leq n-1) \tag{6.1}$$

Equation 6.1 represents the weight from input *i* to node *j* at time *t*. The weights from the *n* nodes are initially set to small random values. The initial radius of the neighbourhood around node *j* should be large $N_j(0)$.

Step 2

Equation 6.2 shows the next stage where an input $x_{i(t)}$ is presented to node *i* at time *t.*

$$x_0(t), x_1(t), x_2(t)..............x_{(n-1)}(t) \tag{6.2}$$

Step 3

Equation 6.3 is applied to calculate the distance d_j between the input *i* and each output node *j*.

$$d_j = \sum_{i=0}^{n-1} \left(x_i(t) - w_{ij}(t)\right)^2 \tag{6.3}$$

Step 4

The node with the minimum d_j is selected and designated as *j**.

Step 5

Equation 6.4 shows for the new weights within the defined neighbourhood *Nj*(t)* surrounding node *j**.

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t)) \qquad \textbf{6.4}$$

$\eta(t)$ describes the learning rate or gain, where $\eta(t)$ is a value between 0 and 1. It is common to have a large learning rate which diminishes in time, hence slowing the weight changes as the network becomes more stable. Similarly the neighbourhood $N_j{*}(t)$ will also be reduced as time increases.

Step 6

Repeat the process from step 2.

6.6 Applications of the Self Ordered Map

The SOM has been used to address practical problems ranging from image compression, to FPGA routing [Barbalho et al 2002, Maniatakos et al 2008]. Kohonen initially implemented a speech recognition system for his native language, Finnish, producing a phonetic typewriter [Kohonen 2001].

6.7 The Weightless Self Ordered Map

The WSOM works on the same principles as the conventional SOM. Both SOMs require the nodes to be configured with small random values. In the case of the weighted SOM this is the production of a random weighting function for each node in the network. The WSOM consists of a pattern of weightless bits. This random assignment in the WSOM can be achieved by loading the nodes with random strings, in practice pseudo random binary strings. It may be advantageous to initialise the map with graded initial conditions dependent on the data. An example of this would be increasing thermocode. The SOM operates by taking some input stimuli and comparing them with responses from the individual nodes within the network. This evaluation is carried out with an array of EX-NOR gates; the resultant weightless data is the level of correlation between the node and the input data. The XNOR gate has been used in weightless logic as a fast parallel method of correlating weightless data. If two positionally equivalent data bits within a given data stream match then it produces an output '1', but if the outputs don't match then a '0' is produced at the output.

Because the resultant correlated data is weightless it can be re-ordered because the bit positions hold no significance. In order to aid the subsequent stages of processing the data is thermocoded. King's weightless thermocoder described in Chapter 3 is a suitable weightless Boolean element to perform this function. Each of the nodes within the network is designed to contain a resultant weightless thermocoder which represents a correlation score of that node in relation to the input. The SOM operates on a 'winner-takes-all' principle which means the strongest response is selected for further processing. The weightless Boolean L-Max element was originally designed for the replacement of Austin's arithmetic implementation used with the ADAM network in Chapter 3. However it is a simple and elegant method of selecting the highest responding nodes. The L-Max function simultaneously compares all the weightless correlation scores for each of the nodes and sets a threshold equal to the greatest response. Every node within the network is then compared to this response and a weightless bit for each node is set to either '1' if it meets the L-max criteria or '0' if not. This results in the highest responding node being identified; however there may be occasions when more than one node will have the maximum response. To overcome this problem a simple Boolean logic selector was developed to yield only one response.

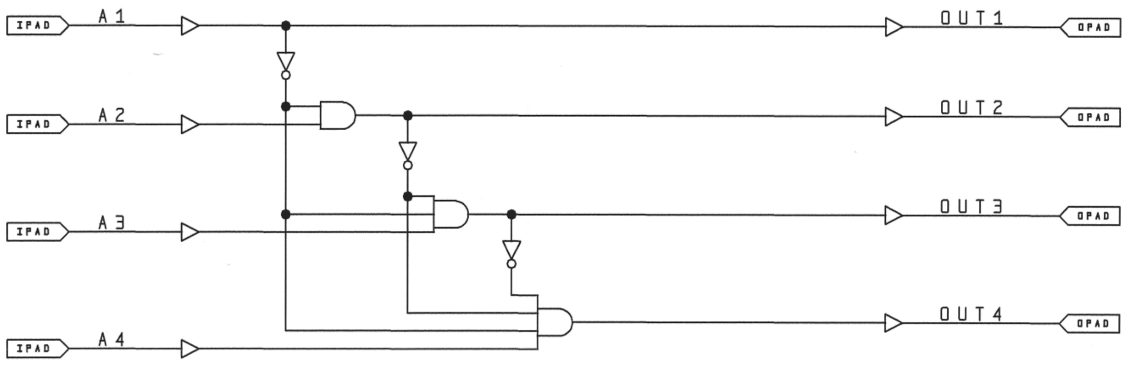


Figure 6.7 Weightless Boolean Selector

The selector is a simple element which is designed to be used on thresholded weightless data. The thresholded data should result with sparsely populated points of interest represented by '1' in the data stream. The function of the selector is to pick out an individual point of interest, when there are several points. The purpose of this is so that in a network an operation can be performed on this one point or region of interest, such as reinforcing or weakening of this element. The selector is used to detect the first bit in the weightless data stream. The selector operates by setting the output (*OUT_n*) to ‘1’ of the corresponding first input (*A_n*) that is set ‘1’ all other outputs are subsequently set to ‘0’. When implemented in the WSOM it was noticed that an offset was needed as the wrap-around effect of the offset skews the data therefore an offset in the selector is needed. Figure 6.8 shows a selector with a fixed offset of one bit. This is a variant of the selector circuit with an offset. This function is equivalent to the standard selector with one minor difference: instead of setting the equivalent bit to ‘1’ it sets each of the adjoining bits to ‘1’ and all other outputs to’0’.

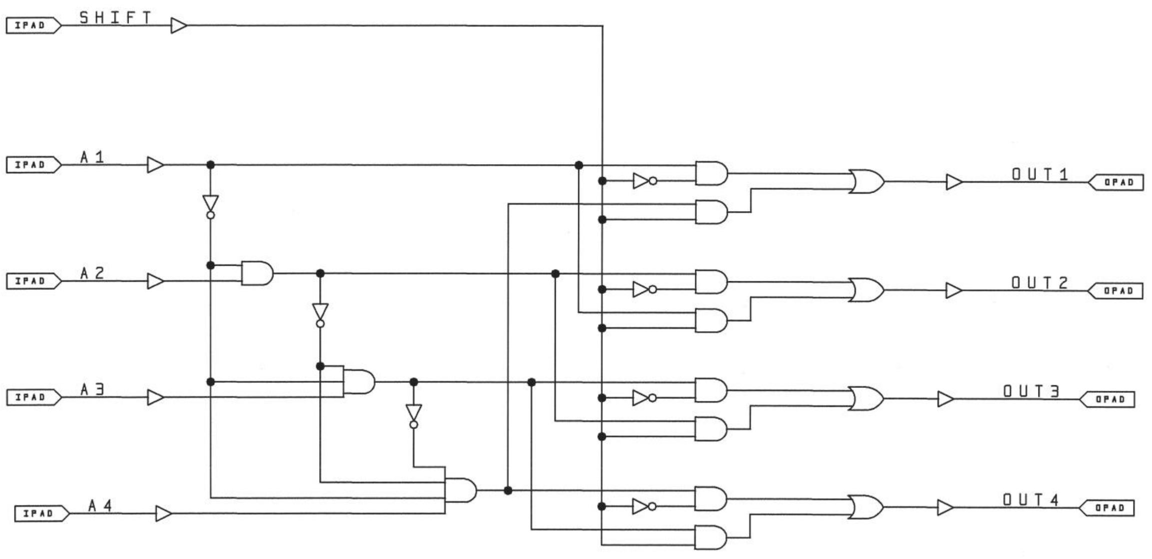


Figure 6.8 Weightless Boolean Selector with Offset

The selector is designed to select the first highest response and discard all the other responses. It is used to identify a point of interest and form a neighbourhood which will be enhanced to respond to the given stimuli which resulted in its selection. Therefore the next stage is to identify the surrounding nodes which will be used to form the neighbourhood. These will also be strengthened to an equal or lesser extent depending on the learning criteria. Figure 6.9 shows a diagram for a fixed ‘expander’. This uses the information from the selector to broaden the field of selected nodes to be operated on.

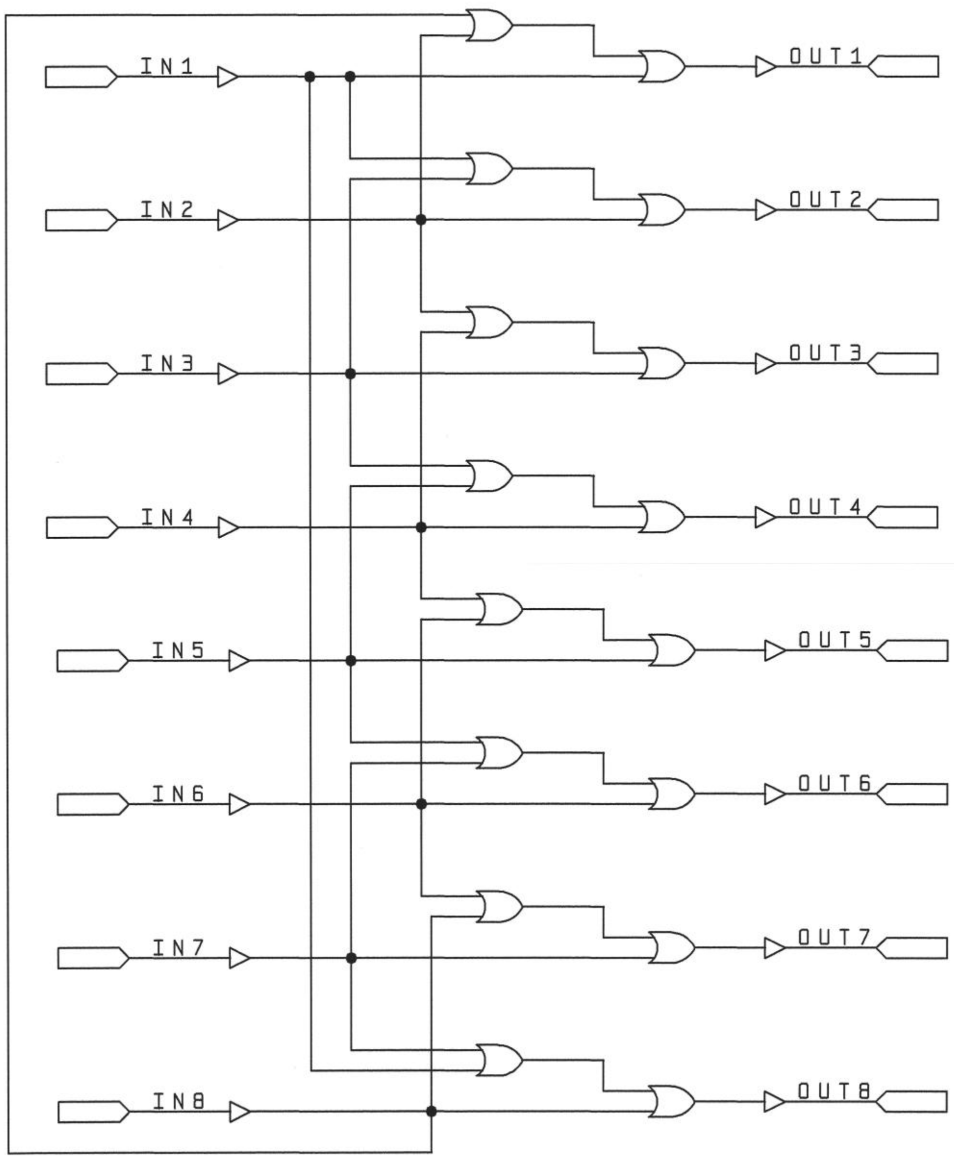


Figure 6.9 Weightless Boolean Expander

The weightless Boolean expander element is designed to be used within a single area of interest such as found in a self learning network like the WSOM. The expander's role is to select a defined surrounding region around the point of interest. Two variations of the expander are shown, the first in Figure 6.9 is a simple Boolean logic implementation which only selects the adjoining bits. Figure 6.10 showing a further variation allows a more adaptive expansion of the surrounding bits. The example given in Figure 6.10 shows where up to an additional two adjoining bits on either side can be strengthened. However the architecture is scalable to allow as many adjoining bits to be set, it could also be designed so it only strengthens one side, or any combination of bits either side.

Figure 6.10 shows a more useful extension to the 'expander' offering variable expanding, this allows the region of interest size to be set and easily varied. This technique is more in-line with the conventional Kohonen topology where the learning rate controls the amount of change as the network evolves giving the network learning true adaptability.

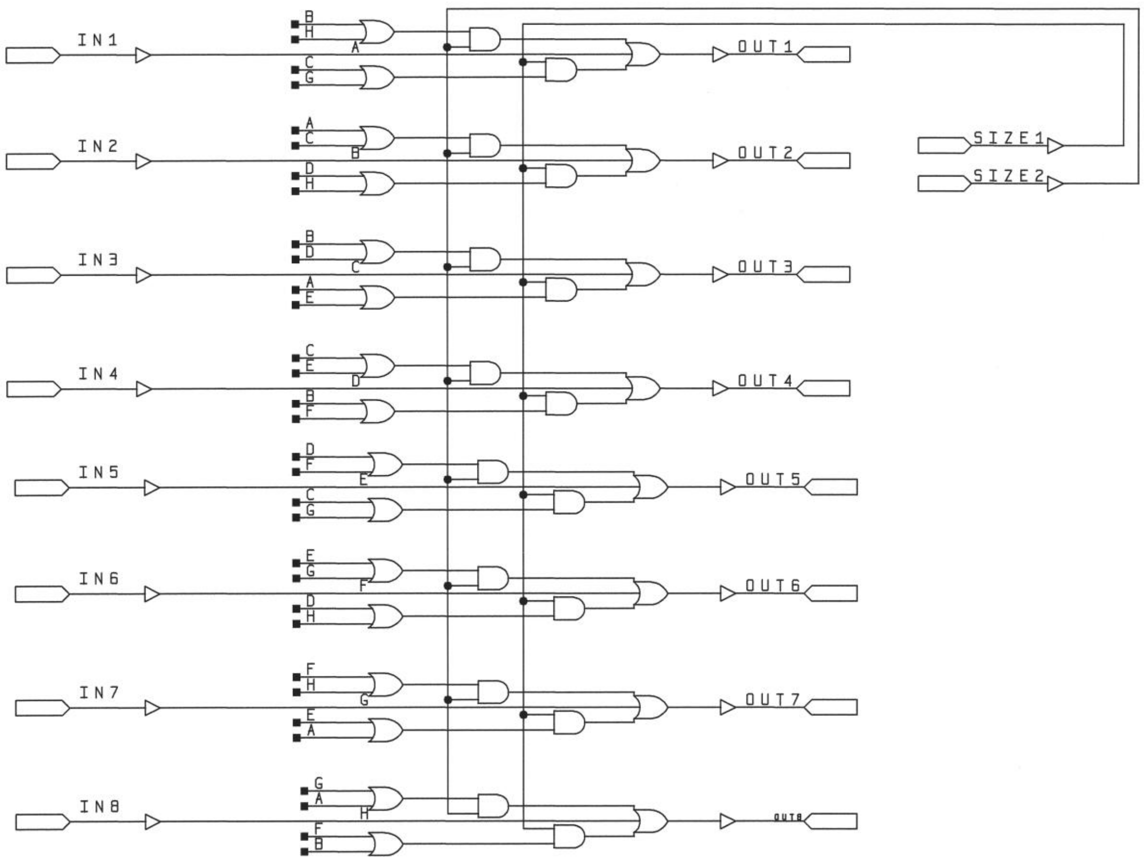


Figure 6.10 Weightless Boolean Multiple Bit Expander

The final stage is to take the nodes which have been highlighted for strengthening and to alter their response to input stimuli. This function is performed by the Hamming distance reducer; the basic variant only reduces the difference between the stimuli and node by one bit, or a set number of bits as shown in Figure 6.12.

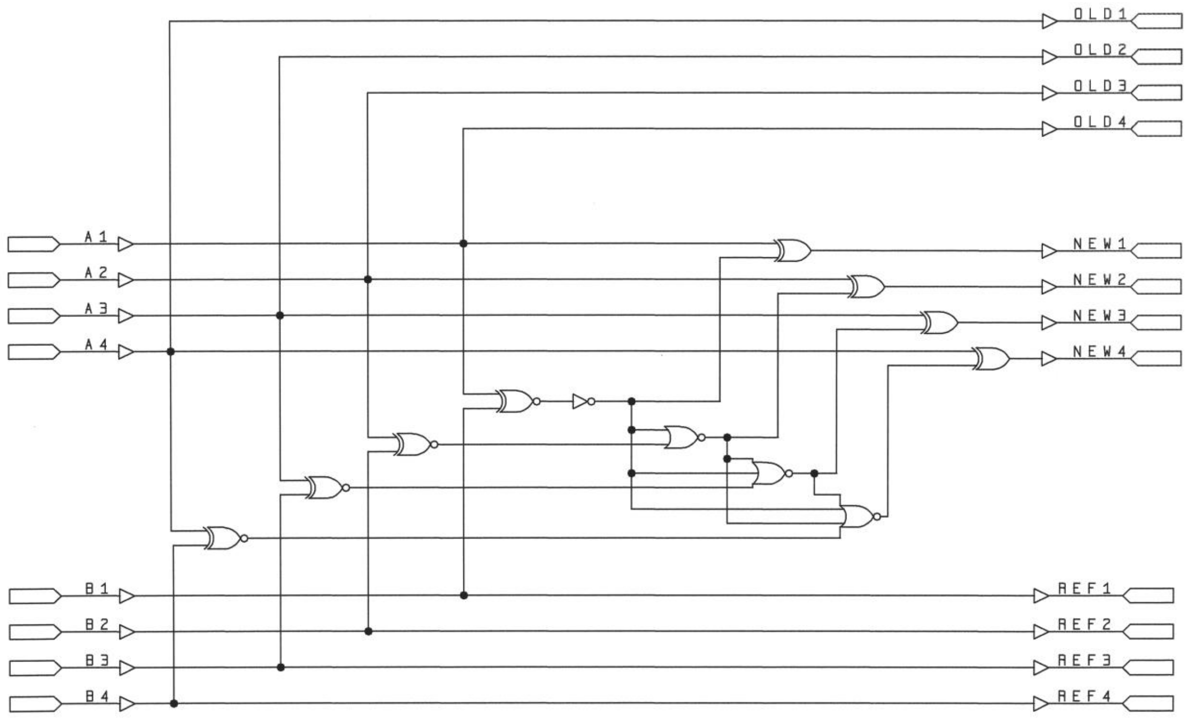


Figure 6.11 Weightless Boolean Hamming Distance Reducer with Test Points

The Hamming distance is the number of bits which differ between two data sets, in this case two data streams (*A1-A$_n$*) and (*B1-B$_n$*). The Hamming distance reducer is designed to alter the status of bits in one weightless data stream to reduce the Hamming distance between the two data streams. Figure 6.11 shows a basic Hamming distance reducer which is a set architecture that only reduces the Hamming distance between the two data streams, by one bit, that being the first different bit in the data stream. Note that outputs (*REF1 - 4*) and (*OLD1 - 4*) would not be used in a real system but are only present for testing. Figure 6.12 shows a large version of the single bit Hamming distance reducer with no test outputs.

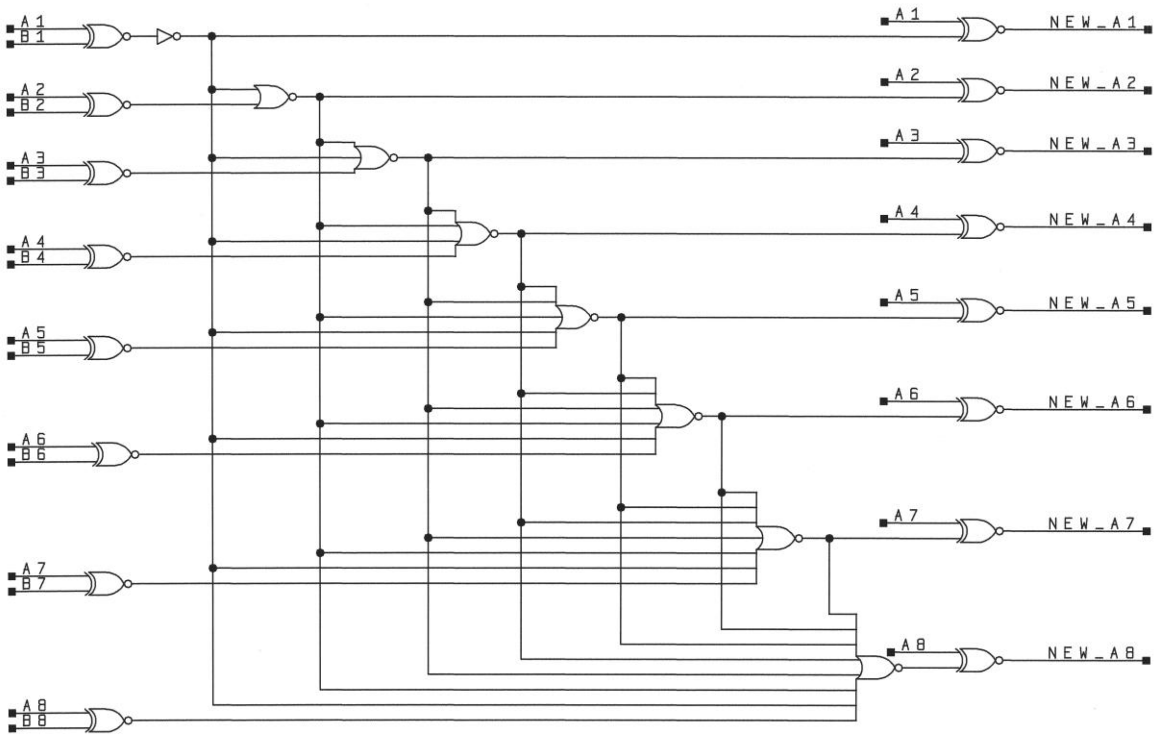


Figure 6.12 Basic Hamming Distance Reducer

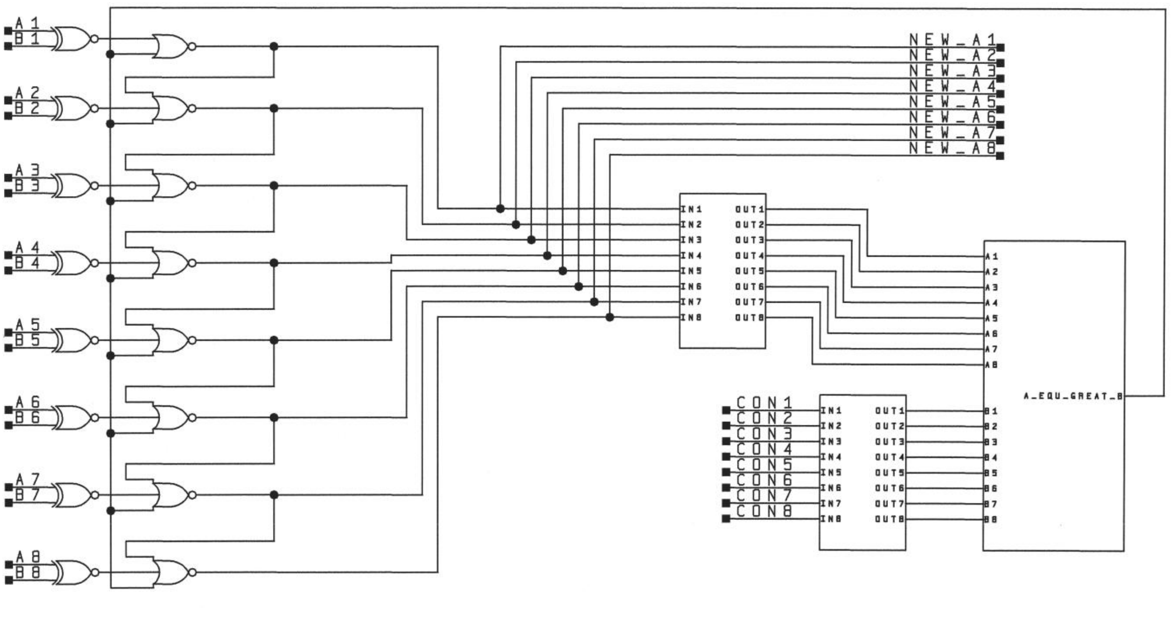


Figure 6.13 The Multiple Bit Hamming Value Reducer

Figure 6.13 shows a multiple bit Hamming distance reducer comprising several earlier elements, these being King's weightless thermocoder and the greater-than-or-equal-to weightless Boolean comparator. The logic shown allows all eight bits to be swapped if required; however in practice this would be unlikely and is only used to demonstrate the flexibility of the element. The multiple Hamming distance reducer has been designed so that the amount by which the Hamming distance is reduced can be varied. This allows another method of controlling the learning rate.

6.8 Self Ordered Map Elements

These elements have all been derived from the quest to design a weightless self ordered map. Although these elements have been designed with this purpose, some have been enhanced, and the intention is that these add to the collection of weightless Boolean building blocks that can be used to further develop existing and new weightless neural networks.

Figure 6.16 shows the overall architecture of the WSOM using the Boolean weightless logic elements described previously within this chapter. All the weightless elements described above have been individually tested using the 'Neuromorph' development board and an array of switches and light emitting diodes to confirm operation.

6.9 Simulation Results

The overall operation of the network has been simulated in MATLAB with two programmes; the first for training the network and the second for evaluating the network once trained. Each of these programmes can be found in Appendix D along with more generic examples where the size of the network can be altered. The logic elements were described as logical functions in MATLAB in order to simulate the network.

The example at the beginning of the chapter is simulated using the two MATLAB programmes which have been pre-loaded with the initial conditions shown in Figure 6.16 and Figure 6.15 The MATLAB Weightless Self Ordered Map is trained via a text file 'data.txt', for this example this file contains the two values 0111 and 1001. Following training the recall can be performed by a second programme shown in Figure 6.15 and a text file 'data2.txt'. This text file allows the user to present data to the network, the network will respond with closest match. If the network has been sufficiently trained and a training data set is given the network will return this as the closest match.

```
% function  Neuron =
somtrain(Elements,NumberofNeurons,FieldofStrength,LearningRate)
% This program performs a weightless
% self ordered map
% James Armstrong 23/02/05
% Filename somtrain.m
% Issue 1


Elements=4;
NumberofNeurons=12;
FieldofStrength=2;
LearningRate=1;

%initalise (NumberofNeurons) with (Elements)
Neuron = [1 0 1 1 0 1 0 1 1 1 1 1;1 1 1 1 0 0 0 0 0 1 0 1;0 1 1 1 0 1
1 1 0 1 0 0;1 1 0 0 0 0 0 0 1  1 0 0];

fid=fopen('C:\data.txt','r');
while(~feof(fid))

    data=(fgetl(fid));
    for Elenum=1:Elements
        Input(Elenum,1)=str2num(data(Elenum));
    end;

    % Xnor the input with all neurons
    for val=1:NumberofNeurons
        Xnored(:,val)=logical(xnor(Input,Neuron(:,val)));

XnorNeurons(:,val)=logical(wtothermo(xnor(Input,Neuron(:,val))));
    end;

    % Determine the Lmax thermocode
    for Elenum=1:Elements
        L_max(Elenum,1)=logical(sum(XnorNeurons(Elenum,:)));
    end;

    % Select the neurons which match the L-max
    for val=1:NumberofNeurons

MatchNeurons(1,val)=logical(greaterequal(XnorNeurons(:,val),L_max(:,1)
));
    end;

    % Selects the first neuron which matches the L-max value
    SelectorPosition=0;
    test=0;
    for val=1:NumberofNeurons
        if(MatchNeurons(1,val)==1&&test==0);
          test=1;
          SelectorPosition=val;
        end;
    end;

    % This takes the selected Neuron and strengths the reponse of the
Neuron
    % surrounding dependant on the value of FieldofStrength

   StrengthenNeurons=logical(zeros(NumberofNeurons,1));
   FieldofStrengthSize=((2*FieldofStrength)+1);
   for Field = 1:FieldofStrengthSize
       StrengthenNeurons(Field,1)=1;
   end;
   StrengthenNeurons=circshift(StrengthenNeurons,-FieldofStrength);
   ShiftNumber=(SelectorPosition-1);
   StrengthenNeuronsShifted=circshift(StrengthenNeurons,ShiftNumber);

StrengthenNeuronsShifted=reshape(StrengthenNeuronsShifted,1,NumberofNe
urons);

    % Teaching the map this function takes the SOM Map and strengthen
the
    % map in the region selected to best match the input

    for val=1:NumberofNeurons
        LearningRateCount=LearningRate;
        if(  StrengthenNeuronsShifted(1,val)==1)
            for bit=1:Elements
                if(Xnored(bit,val)==0&&LearningRateCount>0)
                    Neuron(bit,val)=Input(bit,1);
                    LearningRateCount=(LearningRateCount-1);
                end;
            end;
        end;
    end;
end;

fclose(fid);
save('C:\Weightless Toolbox\matlab_som.mat')
```

Figure 6.14 Weightless Self Ordered Map MATLAB Training Program

```
function  [result,position] = Readtrain()
% This program performs a weightless
% self ordered map
% James Armstrong 25/02/05
% Filename somread.m
% Issue 1


Elements=4;
NumberofNeurons=12;

load('C:\Weightless Toolbox\matlab_som.mat')

fid=fopen('C:\data2.txt','r');
while(~feof(fid))

    data=(fgetl(fid));
    for Elenum=1:Elements
         Input(Elenum,1)=str2num(data(Elenum));
    end;

    % Xnor the input with all neurons
    for val=1:NumberofNeurons
        Xnored(:,val)=logical(xnor(Input,Neuron(:,val)));

XnorNeurons(:,val)=logical(wtothermo(xnor(Input,Neuron(:,val))));
    end;

    % Determine the Lmax thermocode
    for Elenum=1:Elements
```

```
        L_max(Elenum,1)=logical(sum(XnorNeurons(Elenum,:)));
    end;

    % Select the neurons which match the L-max
    for val=1:NumberofNeurons

MatchNeurons(1,val)=logical(greaterequal(XnorNeurons(:,val),L_max(:,1)
));
    end;

    % Selects the first neuron which matches the L-max value
    test=0;
    for val=1:NumberofNeurons
        if(MatchNeurons(1,val)==1&&test==0);
          test=1;
          result=Neuron(:,val);
          position=val;
        end;
    end;
end;
 fclose(fid);
```

Figure 6.15 Weightless Self Ordered Map MATLAB Recall Program

The simulation has shown that it is possible to implement the originally algorithmic SOM in weightless Boolean elements. The implementation of the weightless SOM has led to the development of further weightless Boolean elements. These include the selector and expander including variants. The Hamming distance reducer is also useful for other weightless neural functions where controlled reduction of Hamming distance is required as part of learning. It is envisaged that these elements will be added to the collection of weightless Boolean elements which will allow further non-weightless systems to be implemented weightlessly, particularly neural networks where areas of the network need to be selected for processing or regions defined.

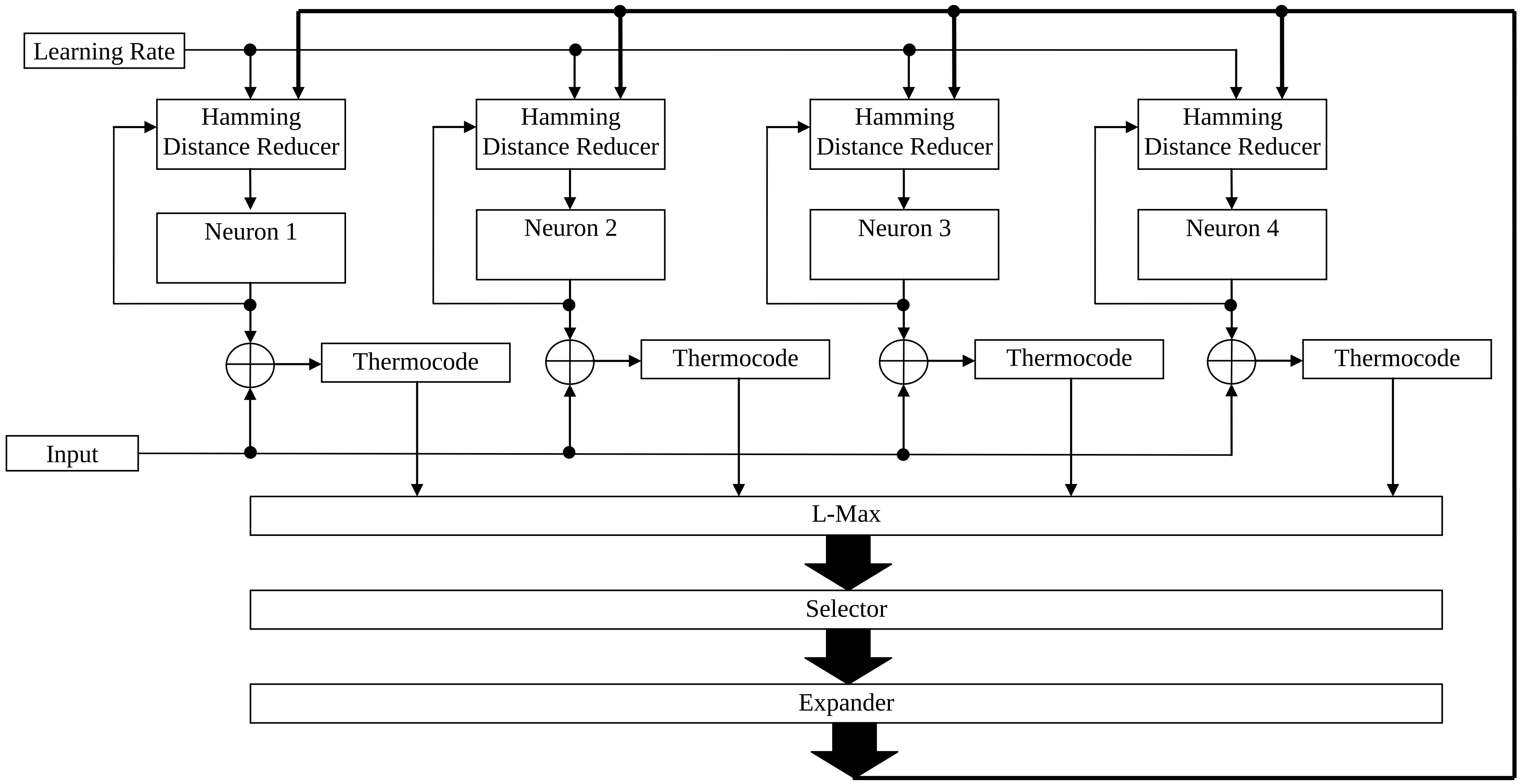


Figure 6.16 Architecture of the Weightless Self Ordered Map

6.10 Summary of Chapter 6

Further weightless Boolean hardware elements are presented which build on the sum and threshold logic described in the earlier chapters. These additional weightless Boolean elements are each described and used to form a weightless Boolean hardware self ordered map. The weightless self ordered map has been simulated in MATLAB using the example in the chapter. It is demonstrated how conventional weighted neural networks can be converted into weightless neural networks using Boolean weightless architectures.

Chapter 7

7 Summary and Conclusions

7.1 Overview

This thesis describes a collection of weightless Boolean elements which are being used to improve both weightless and weighted neural networks and implement novel filters. The robustness of weightless elements has also been demonstrated by trials carried out at ground-based neutron test facilities.

7.2 Objectives

All the objectives described in the introduction were achieved; these are as follows:

- To examine the robustness of weightless elements versus weighted elements when subjected to high-energy neutron radiation mimicking that of the atmosphere through accelerated testing at a ground-based facility.

- To develop a collection of weightless Boolean elements to add to the existing body of elements to further improve weightless neural networks by replacing weighted arithmetic counter units with sum and threshold techniques.

- To further examine King's Type 1 Neuroram filter as an image filter when subjected to an extended range of noise types in conjunction with evaluating threshold properties of the filter.

- To demonstrate how weightless Boolean elements can be developed to implement weightless Boolean hardware implementations of existing weighted neural networks; notably the self ordered map.

- The final objective was to show how weightless Boolean elements could be developed to replace weighted binary counters in the implementation of robust avionic designs.

7.3 Summary of Chapters

The author now briefly summarises the chapters in this thesis focusing on their contribution to the field.

Chapter 1 introduces the research describing the problems faced by the avionics industry caused by atmospheric radiation as well as techniques to understand the interactions between neutrons and semiconductor electronics. Two papers in which the author contributed at the test facility are presented in the appendices. These describe some of the effects observed when using CCD elements to understand semiconductor neutron interaction processes. An overview of the history of Boolean logic and the differences between weighted and weightless Boolean logic is discussed.

Chapter 2 gives a chronological history of weightless hardware neural networks discussing the McCulloch and Pitts model for a neuron through to the modern day approach to weightless neural networks such as ADAM and Neuroram [McCulloch et al 1943, Austin 1986, King 2000]. The history of the division of the McCulloch and Pitts model into both weighted and weightless neural networks is described [Pitts et al 1947]. The operation of ADAM and the correlation matrix memory and associated threshold techniques are discussed [Austin 1986].

Chapter 3 develops a collection of weightless sum and threshold architectures which are formed from simple Boolean logic. Several variations of logic architectures are described; including a serial method generating thermocode which has been patented by BAE SYSTEMS. The patent is included in Appendix C. In addition new weightless Boolean sum and threshold elements are presented which complements the elements proposed by King [King 2000]. These are the greater-than-and-equal and the less-than-and-equal functions given in Figure 3.8 and Figure 3.9 respectively. The advantages of these new architectures are they allow the greater-than-or-equals or less-than-or-equals to be performed in a single stage. Previously King described three elements, greater than, equals to, and less than. In order to implement the combination of these architectures two elements were required, this meant there would be two stages of logic. The method proposed here reduced this to a single stage of logic and hence reduces the propagation delay.

This chapter also describes a new method of learning for the correlation matrix memory. The plausibility of using different logic to implement the learning at the intersections of the correlation matrix memory which have conventionally been formed from 'AND' gates derived from Hebbian learning has been examined [Hebb 1949]. Table 3.2 shows a collection of alternative Boolean logic gates that can be used to implement different learning criteria, these include the 'NOR' gate and half the structure of the 'EXOR' gate. The effects of different learning criteria have been simulated using ADAM network [Austin et al 1987].

In order to design the weightless Boolean hardware architecture of ADAM two new weightless Boolean threshold elements are described which perform the L – Max (N Point) threshold and the Willshaw threshold [Austin et al 1987]. Figure 3.14 and Figure 3.15 show the new weightless Boolean architectures used to form a weightless ADAM.

Chapter 4 describes and evaluates a weightless Boolean hardware median filter when implemented on an FPGA and subjected to single event affects caused by neutron radiation. The weightless Boolean hardware median filter has been patented by BAE SYSTEMS to protect intellectual property rights. Section 5.4 describes the asynchronous weightless Boolean filter and analyses the operational speed in conjunction with conventional weighted median filter implemented in a similar FPGA [Xilinx 1998]. Section 5.5.1 describes a new technique with greater parallelism for converting weighted binary data into thermocode. This is then compared with King's thermocoding technique assessing the speed of operation of both filters when converting 8-bit binary into thermocode. The parallel technique reduces the layers of gates in comparison to King's method drastically reducing the propagation delay of the overall median filter and making the performance comparable to that of a weighted binary median filter implemented in an FPGA.

A study of both filters is made when subjected to neutron radiation using an FPGA which is prone to single event upsets. The results show that the weightless architecture is more prone to upset. However these upsets have a limited affect on the overall result due to the inherent nature of the architecture meaning the failure mode is more graceful and predictable. The increase in upsets within this architecture is due to the number of data channels as each data channel can only carry one bit. Conversely the weighted median filter suffers less upsets but the outcome of these is much less predictable as each of the bits could be affected varying the amount by which the data is changed. The worst-case is that the data could halve or double in magnitude with a single bit failure. This is demonstrated in Figure 4.28 which shows the results of the testing undertaken.

Chapter 5 evaluates of King's type 1 Neuroram filter using three standard images and a collection of further noise types to better characterises the performance of the filter. The noise types that have been applied to the three standard images are additive Gaussian, additive uniform, multiplicative uniform and multiplicative Gaussian as well as Salt and Pepper noise as originally tested by King [2000]. The effects on the images are shown in Figure 5.5 through to Figure 5.10. This analysis was presented in the form of histogram plots of the original image and the resultant images following corruption by the different noise types. The effect of the filter on removing these noise types has been assessed using the evaluation criteria of PSNR and MSE as described in section 4.4. It is clear that the additive Gaussian noise causes severe corruption to the image, this is shown in Figure 5.11(c). The results shown in Figure 5.12, Figure 5.14, Figure 5.16, Figure 5.18 and Figure 5.20 clearly demonstrates the median filter is better for certain noise types. Most types of filter performed well on the original salt-and-pepper noise trials by King, including multiplicative Gaussian noise and multiplicative uniform noise. The filter is not well-suited to the removal of additive noise types such as additive Gaussian and additive uniform noise. It is clear from the images in Figure 5.15, Figure 5.17 and Figure 5.19 the weightless median filter causes less blurring and has improved definition in contrast to the standard median filter.

A novel trial was performed on both the median filter and the neural filter to identify the ideal number of filters in cascade. The images used for these were corrupted with the salt-and-pepper noise as this was the noise type the filters perform the best with. Figure 5.21 shows the optimum number of filters for the median filter was two. However the neural filter continued to show improvement up to five filters with the compromise of performance against number of filters being three. The overall performance of the median filter was better than that of the neural filter.

A final evaluation of the weightless filter examining its properties with various thresholds was performed in section 4.6. King in his thesis describes the optimum threshold being half the number of samples, in this case this would equate to a threshold of either 4 or 5 [King 2000]. King did qualify this statement saying that the threshold would have some data dependency. This research was conducted on two separate images as shown in Figure 5.28 and Figure 5.30. Figure 5.28 shows the effects of the filter with different thresholds and demonstrates that at lower thresholds the definition of the images is poor. However as the threshold increases more detail is revealed showing an optimum threshold of three or four demonstrated by the graph shown in Figure 5.27. The trial was then repeated on the Milk Drop image using multiplicative uniform noise which was added at 20%. This had similar characteristics showing a lot of white noise at the lower thresholds and a lot of dark noise at the higher thresholds as shown in Figure 5.30. This trial was repeated with 40% added noise and the results were the same. This showed the ideal threshold for the filter to be three, against King's original prediction this shows a threshold of 33% performs better in a nine sample window than the originally suggested 50%.

Chapter 6 describes a novel weightless implementation of the self ordered map. This implementation follows the principles of the conventional self ordered map originally developed by Kohonen [Kohonen 1984]. As part of the implementation of the self ordered map several novel weightless elements had been developed. These extend the collection of weightless elements for the development of both weightless neural networks as standard weightless architectures. These include a selector circuit shown in Figure 6.7 which is designed to select the first responding output in a parallel line of weightless data. An expander circuit shown in Figure 6.9 which is primarily designed to take the output of the selector circuit. The operation of this element is to ensure a wider number of outputs responses adjacent to the stimulated input or inputs. This circuit can be used in conjunction with the selector circuit to enhance a region of interest within a weightless network. The final weightless Boolean element developed is the Hamming distance reducer shown in Figure 6.12. This is designed to take two weightless Boolean streams of data and reduce the Hamming distance between the streams of data. Two versions of this are given, one which reduces one of the data streams by one bit and the other which allows multiple bits to be altered in order to reduce the Hamming distance as shown in Figure 6.13.

The culmination of these elements is the implementation of a weightless self ordered map. This demonstrates that a weighted neural network such as the self ordered map can be implemented in weightless Boolean hardware.

7.4 Original Contributions

7.4.1 Robustness to Corruption

It has been demonstrated through ground based accelerator testing in a high flux neutron environment that weightless Boolean implementations respond differently to data corruption compared to standard weighted binary implementations. Although the incidence of corruption is increased in comparison to an equivalent weighted implementation, due to the increased size of the data bus, the resultant effect is less significant. This is due to the removal of the binary weighting function which contributes to a random data corruption. Therefore weightless logic has advantages in avionic hardware which is subjected to atmospheric radiation effects; it has applicability to voting systems and high integrity systems.

7.4.2 Production of a Weightless Median Filter

An aim of the research was to demonstrate that standard functions and elements could be re-designed and implemented in weightless Boolean logic. A novel median filter was produced which does not require any clocked arithmetic elements. The parallel nature of the weightless logic ensures that high speed operation of the filter could be achieved. Coupled with its increased robustness, this makes it an ideal target for medical and avionic applications.

7.4.3 Neuroram as an Image Filter

The research has further defined the properties of Neuroram when configured as an image filter. It has characterised the properties of Neuroram and its ability to filter noise on a greater range of adjustment of the threshold of Neuroram. It has shown its effects are similar to those of a standard median filter although it has some distinct differences. The corruption to the image caused by applying the filter is different, the fact it does not damage the edges of the images. Another distinct difference is the effect of cascading Neurorams.

7.4.4 Sum and Threshold Elements

A further selection of sum and threshold weightless Boolean elements have been designed which contribute to the prior art for use in weightless neural networks. These include Willshaw and N point sum and threshold elements which form an integral part of the ADAM network and new techniques for thermocoding weightless data. A logic element capable of producing combined ‘greater-than-or-equal-to’ and ‘less-than-or-equal-to’ results has been produced.

7.4.5 Non-Hebbian Learning

Conventionally correlation matrix memories have been based on Hebbian learning. A new selection of learning criteria has been suggested, the main example being that of non-Hebbian learning.

7.4.6 ADAM

A completely Boolean implementation of ADAM has been designed and simulated. It is based on the novel weightless Boolean sum and threshold elements. This has resulted in an ADAM network that no longer relies on counters or arithmetic logic, improving its robustness and ensuring the network is not only weightless in operation but also in implementation.

7.4.7 Self Ordered Map

A collection of weightless Boolean elements have been designed which have allowed a weightless implementation of Kohonen’s self ordered map [Kohonen 1984]. These elements include a weightless Boolean expander, selector, and Hamming distance reducer.

Chapter 8

8 Further Work

8.1 Neuroram

A further extension to Type 1 Neuroram particularly with regard to image filtering is to make use of the ability to alter its threshold is suggested. This research has shown the threshold needs to be altered dependent on the data, in this case the greyscale of the image. Two possible extensions are; a filter which sets the threshold for the whole data set in this example the overall average greyscale of the image. A more flexible alternative is a dynamic threshold which is based on the local regional greyscale. It is expected this will have improved characteristics over the fixed threshold models described in this thesis. Closer examination of this property using frequency analysis may define the level of improvement in comparison to traditional techniques. This further work may yield areas where these filters are more applicable; such as medical imaging. It is suggested that an evaluation using a cascade of Neurorams acting as image filters each with differing thresholds may yield some interesting properties.

8.2 Weightless Neural Element Properties

This research has investigated the robust nature and design of weightless Boolean logic elements, and has shown the architectures to be larger, however it is believed that the removal of the high speed clocked elements should also reduce their power consumption and thermal dissipation when implemented in hardware. The reduction of clock speed will also reduce the incidence of atmospheric radiation induced transient single event upsets which are more likely to lead to metastability when high clock speeds are used. A study into these hypotheses may further support the applicability of this technology in avionic systems.

8.3 Non-Hebbian Techniques

The author’s new learning criteria proposed for the Correlation Matrix Memory in this research may offer improvements in other existing weightless neural networks, particularly those based on the CMM such as AURA [Austin et al 1998]. Further analysis of these networks using these new learning criteria is required to study the operation and implementation of them and further characterise their performance.

8.4 ADAM

A further evaluation of ADAM using a combination of non-Hebbian learning and Hebbian learning is required. It is proposed that the addition of non-Hebbian learning in parallel with Hebbian learning may be beneficial on a network that is becoming saturated. This regime would use the same tupling but the final result would be a combination of both learning techniques. The philosophy behind this is that the network would have less 'zeros' and these would hold valuable information which is presently being lost with the current method.

8.5 Weightless Self Ordered Map

The weightless self ordered map has only been simulated so far using MATLAB with a restricted data set. Simulations using larger and varied data sets would further confirm the operation of the weightless implementation. A full hardware implementation of the weightless self ordered map on a typical FPGA architecture would demonstrate a physical implementation.

8.6 Weightless Boolean Elements

The weightless Boolean elements developed during this research are generic. Further investigation into applying them to other weighted and weightless neural networks is suggested to examine if any benefit could be gained. The weightless elements generated also may have merit in the field of robust voting systems for high reliability systems operating in harsh environments; these include space, high altitude, medical and industrial applications where they are subjected to high electro-magnetic fields.

Further research into the robust nature of these elements should be performed in high electro-magnetic fields to evaluate if this technology is more robust due to the removal or reduction of the clock systems. Examination of the Electro-Magnetic Compatibility (EMC) of weightless Boolean logic in comparison to standard implementations may show reduced EMC emissions due to the removal or reduction of the clock frequency.

References

Aleksander, I. and Morton, H., 1995, ‘An introduction to neural computing’, Thomson Computer Press, second edition, chapters 2, 5 and 10, 1995

Aleksander, I., Thomas, W. V. and Bowden, P. A., 1984, ‘WISARD – a radical step forward in image recognition’, Sensor Review, pp. 120-124, July 1984

Aleksander, I. and Stonham, T. J., 1979, ‘Guide to pattern recognition using random access memories’, Computer and Digital techniques, vol. 2, no. 1, pp. 29-40, February 1979

Aleksander, I., 1965, ‘Fused logic element which learns by example’, Electronics Letters, vol. 1, no. 6, pp.173-174, August 1965

Armstrong, J. R., 1999, ‘A RAM based methodology for the implementation of fuzzy thermocode’, Proceedings of the Third International Workshop on Weightless Neural Networks, WNNW99, University of York, paper no. 8, March 30th to 31st, 1999

Armstrong, J. R., 2003a, ‘Serial weightless data to thermocoder coded data converter’, World Intellectual Property Organization, international patent publication WO 03/071684, international application PCT/GB03/00732, filed February 21st, published August 28th, 2003

Armstrong, J. R., and King, D. B. S., 2003, ‘Ordering by Hamming value’, World Intellectual Property Organization, international patent publication WO 03/071683, international application PCT/GB03/00756, filed February 21st, published August 28th, 2003

Armstrong, J. R., 2003b, ‘Ordering weightless binary tuples according to Hamming value’, World Intellectual Property Organization, international patent publication WO 03/071685, international application PCT/GB03/00780, filed February 21st, published August 28th, 2003

Austin, J., 1986, ‘The design and application of associative memories for scene analysis’, Ph. D. Thesis, Department of Electrical Engineering, Brunel University, U.K., pp. 58-88, August 1986

Austin, J. and Stonham, T. J., 1987, ‘Distributed associative memory for use in scene analysis’, Image and Vision Computing, vol. 5, no 4., pp. 251-260, 1987

Austin, J., 1987, ‘ADAM: a distributed associative memory for scene analysis’, Proceedings First Annual Conference on Neural Networks, San Diego, CA, U.S.A, pp. IV-285 to IV-292, 1987

Austin, J., 1993, ‘Rapid learning with a hybrid neural network’, Neural Network World, no. 5, IDG Communications, U.S.A., pp. 531-549, 1993

Austin, J., 1994, ‘A review of R.A.M. based neural networks’ in the Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems, Turin, Italy, pp. 58-66, September 26th to 28th, 1994

Austin, J. and Buckle, S., 1994, ‘The practical application of binary neural networks’, Proceedings of the UNICOM Seminar on Adaptive Computing and Information, Brunel University, U.K., pp. 643-660, 1994

Austin, J., 1997, ‘Associative memory’, in ‘Handbook of neural computation’, release 97/1, Institute of Physics Publishing Ltd. and Oxford University Press, UK, pp. F1.4:1-F1.4:7,1997

Austin, J., 1998, ‘Grey scale n-tuple processing’, Proceedings of the Fourth International conference on Pattern Recognition, Cambridge, pp. 110-119, 1998

Austin, J., Kennedy, J. and Lees, K., 1998, ‘The advanced uncertain reasoning architecture, AURA’, in ‘RAM-based neural networks’, (J. Austin ed.), World Scientific Publishing Co., pp. 43-50, 1998

Barbalho, J.M., Duarte, A.,Neto, D., Costa, J. A. F. And Netto, M. L. A., ‘Hierarchical SOM applied to image compression’, International Joint Conference on Neural Networks, vol. 1, pp. 442-447, July 15th to July 19th 2001, published August 7th, 2002

Bates, G. L. and Nooshabadi, S., ‘FPGA implementation of a median filter’, TENCON '97, IEEE Region 10 Annual Conference, Proceedings of IEEE Speech and Image Technologies for Computing and Telecommunications, vol. 2, pp. 437-440, December 2nd to 4th, 1997

Bedford, D. F., Morgan G. and Austin J., 1996, ‘A draft standard for the certification of neural networks used in safety critical systems’, Artificial Neural Networks in Engineering, pp. 1057-1062, November 1996

Beale, R. and Jackson, T., 1997, ‘Neural computing, an introduction’, Institute of Physics Publishing Ltd., Bristol, U.K., pp. 212-215, 1997

Blackmore, E. W., Dodd, P. E. and Shaneyfelt, M. R., 2003, ‘Improved capabilities for proton and neutron irradiations at TRIUMF’, IEEE Radiation Effects Data Workshop, pp. 149 155, July 21st to 25th, 2003

Bledsoe, W. W., Bomba J. S., Browning, I., Evey, R. J., Kirsch, R. A., Mattson, R. L., Minksy, M., Neisser, U. and Selfridge, O. G., 1959a, ‘Discussion of problems in pattern recognition’, Proceedings of the Eastern Joint Computer Conference, Boston, Massachusetts, U.S.A., pp. 233-237, 1959

Bledsoe, W. W. and Browning, I., 1959b, ‘Pattern recognition and reading by machine’, Proceedings of the Eastern Joint Computer Conference, Boston, Massachusetts, U.S.A., pp. 225-232, 1959

Bolt, G., Austin, J. and Morgan, G., 1992, ‘Uniform tuple storage in ADAM’, Pattern Recognition Letters, no. 13, Elsevier Science Publishers, B.V., pp. 339-344, May 1992

Bolt, G., 1991, ‘Operational fault tolerance of the ADAM neural network systems’, IEEE International Joint Conference on Neural Networks, vol. 1, pp. 1-6, November 18th to 21st, 1991

Boole, G., 1854, ‘An investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities’, Macmillan and Co., Cambridge, 1854

Buchan, I., 2004, ‘Calculating Poisson confidence Intervals in Excel’, Public Health Informatics at the University of Manchester, January 27th, 2004

Chugg, A. M., Jones, R., Jones, P., Nieminen, P., Mohammadzadeh, A., Robbins, S., and Lovell, K., 2002, ‘A CCD Miniature Radiation Monitor’, IEEE Transactions on Nuclear Science, vol. 49 no. 3, pp.1327-32, June 2002

Chugg, A. M., Jones, R., Moutrie, M. J. Dyer, C. S., Ryden, K. A., Truscott, P. R., Armstrong, J. R., and King, D. B. S., 2003a ‘Analyses of CCD images of Nucleon-Silicon Interaction Events’, Proceedings of RADECS 2003 Conf., European Space Agency, Noordwijk, Holland, September 15th to 19th, 2003

Chugg, A. M., Jones, R., Moutrie, M. J., Armstrong. J. R., King, D. B. S., and Moreau, N., 2003b, ‘Single particle dark current spikes induced in CCD’s by high energy neutrons’, IEEE NSREC Conference, Monterey, California, July 2003

Chugg A. M., 2003c, ‘SPAESRANE, Solution for the preservation of aerospace electronic systems reliability in the atmospheric neutron environment’, An aeronautics research programme proposal to the Department of Trade and Industry, MBDA UK Limted, Filton, ED2003-00192, Issue 1, July 2003

Chugg, A. M., 2006, ‘Solutions for the Preservation of Aerospace Electronic Systems Reliability in the Atmospheric Neutron Environment’, SPAESRANE final report, MBDA Filton, DR33269, September 2006, Issue 1

Copeland, B. J. and Proudfoot, D., 1996, ‘On Alan Turing’s Anticipation of Connectionism’, Synthese, Springer Netherlands, vol. 108, no. 3, pp. 361-377, September 1996

D’Almeida, F., ‘Nonlinear diffusion toolbox’, 2003, URL http://www.mathworks.co.uk/matlabcentral/fileexchange/3710-nonlinear-diffusion-toolbox

Dyer, C. S., Truscott, P. R., Sanderson, C., Colwell, B., Chugg, A., Jones, R., MacDiarmid, I. and Johansson K., 2000, 'Cosmic radiation effects on avionics, an increasing hazard in the new millennium?', ICAS 2000, Harrogate, August 2000

Dyer, C. and Lei, F., 2001, 'Monte-carlo calculations of the influence on aircraft radiation environments of structures and solar particle events', IEEE Transactions on Nuclear Science, December 2001

Dyer, C. S., Sims, A. J., Truscott, P. R., Farren, J. and Underwood, C., 1992, 'Radiation environment measurements on Shuttle missions using the CREAM experiment', IEEE Transactions on Nuclear Science, vol. 39 issue. 6, pp. 1809 – 1816, December 1992

Hamming, R. W., 1950, 'Error detecting and error correcting codes', The Bell System Technical Journal, vol. 26, no. 2, U.S.A., pp. 147-160, April 1950

Hazra, A., Bhattacharyya, J. and Banerjee, S., 2004, 'Real time noise cleaning of ultrasound images', Proceedings 17th IEEE Symposium on Computer-Based Medical Systems, pp. 379-384, June 24th to 25th, 2004

Hebb, D. O., 1949, 'The organization of behaviour, a neuropsychological theory', John Wiley and Sons Incorporated., New York, U.S.A., pp. 60-78, 1949

Hess, V. F., 1913, 'Uber den Ursprung der durchdringenden Strahling', Physikalische Zeitschrift, vol. 14, no. 14, 1913

Hopfield, J. J., 1982, 'Neural networks and physical systems with emergent collective properties', Proceedings of the National Academia of Science, U.S.A., 79, pp. 2554-2558, 1982

James, W., 1890, 'Psychology (briefer course)', Holt, New York, U.S.A., pp.253-279, 1890

Kennedy, J. V., Austin, J. and Cass, B., 1995, 'A hardware implementation of a binary neural image processor', Image Processing and its Applications, IEE Conference publication number 410, pp. 465-469, July 4th to 6th, 1995

Kennedy, J. V. and Austin, J., 1994, 'A hardware implementation of a binary neural associative memory' in the Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems, Turin, Italy, pp.178-185, September 26th to 28th, 1994

Kennedy, J. V. and Austin, J., Pack, R. and Cass, B., 1995, 'C-NNAP – a parallel processing architecture for binary neural networks', Proceedings of the IEEE International Conference on Neural Networks (ICNN 95), University of Western Australia, Perth, Australia, pp. 1037-1041, November 27th to December 1st, 1995

King, D. and Armstrong, J., 2008, 'System level prevention: managing SEE using error correction techniques', IET Seminar on Cosmic Radiation, November 18th, 2008

King, D. B. S., Simpson, R. J., Moore, C. and MacDiarmid, I. P., 1998, 'Digital n-tuple Hamming comparator for weightless systems', Electronics Letters, vol. 34, no. 22, pp. 2103-2104, 1998

King, D. B. S., MacDiarmid, I. P. and Moore, C., 1999d, 'Weightless binary n-tuple thresholding hierarchies', World Intellectual Property Organization, international patent publication WO 99/32962, international application PCT/GB98/03837, filed 18th December 1998, published July 1st, 1999

King, D. B. S., 1999c, 'Boolean lattice for implementing sum-and-threshold binary neurons', Proceedings of the Third International Workshop on Weightless Neural Networks, WNNW99, University of York, paper no. 7, March 30th to 31st, 1999

King, D. B. S., 1999b, 'Hamming value comparison for unweighted bit arrays', World Intellectual Property Organization, international patent publication WO 99/32961, international application PCT/GB98/03835, filed December 18th, published July 1st, 1999

King, D. B. S., 1999a, 'Binary code converters and comparators', World Intellectual Property Organization, international patent publication WO 99/33184, international application PCT/GB98/03834, filed December 18th, published July 1st, 1999

King, D. B. S., 2000, ‘Robust hardware elements for weightless artificial neural networks’, Ph. D. Thesis, Department of Engineering and Product Design, University of Central Lancashire, U.K., pp. 41-151, January 2000

Kohonen, T., 1984, ‘Self-organization and associative memory’, Springer-Verlag, 1984

Kohonen, T., 1988, ‘Associative memories and representations of knowledge as internal states in distributed systems’, Proc. European Seminar on Neural Computing, London, UK, pp. 4/1-4/9, 1988

Kohonen, T., 2001, ‘Self-organising maps’, Springer-Verlag, third edition, 2001

MacDiarmid, I., King, D. and Armstrong, J., 2005, ‘BAE SYSTEMS, Air Systems approach to the problem of atmospheric radiation’, IET Seminar on Cosmic Radiation, Savoy Place, London, December 6th, 2005

Maniatakos, M., Xu, S. and Miranker, W. L., 2008, ‘Constraint-based placement and routing for FPGAs using self-organizing maps’, IEEE International Conference on Tools with Artificial Intelligence, vol. 2, pp. 465-469, 20th IEEE International Conference on Tools with Artificial Intelligence, 2008

Moulds, A., Pack, R., Ulanowski, Z. and Austin, J., 1999, ‘A high performance binary neural processor for PCI and VME bus-based systems’, Proceedings of the Third International Workshop on Weightless Neural Networks, WNNW99, University of York, March 30th to 31st, 1999

McCulloch, W. S. and Pitts, W., 1943, ‘A logical calculus of the ideas immanent in nervous activity’, Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133, December, 1943

Picton, P., 1994, ‘Introduction to neural networks’, Macmillan Press, London, pp. 46-60, 1994

Pitts, W. and McCulloch, W. S., 1947, ‘How we know universals: the perception of auditory and visual forms’, Bulletin of Mathematical Biophysics, vol. 9, pp. 127-147, 1947

Prokofiev, A. V., Blomgren, J., Majerle, M., Nolte, R., Rottger, S., Platt, S. P., Cai Xiao Xiao and Smirnov, A. N., 2009, ‘Characterization of the ANITA neutron source for accelerated SEE testing at the Svedberg Laboratory,’ IEEE Radiation Effects Data Workshop, Quebec, Canada, pp. 166-173, July 20th to 24th, 2009

Rochester, N., Holland, J. H., Haibt, L. H. and Duda, W. L., 1956, ‘Test on a cell assembly theory of the action of the brain, using a large digital computer’, IRE Transactions on Information Theory, pp. IT-2:80-IT-2:93, 1956

Rumelhart, D. E., Hinton, G. E. and Williams, R. J., 1986, ‘Learning internal representations by error propagations in parallel distributed processing’, vols. 1 and 2, eds Rumelhart, D. E. and McClelland, J. L., MIT Press, Massachusetts, 1986

Shannon, C. E., 1937, 'A symbolic analysis of relay and switching circuits', Masters Thesis, Massachusetts Institute of Technology, 1937

Steinbuch, K., 1961, ‘Die Lernmatrix’, Kybernetik 1, pp. 36-45, January, 1961

Steinbuch, K. and Zendeh, F., 1962, ‘Self-correcting translator circuits’, Proceedings of the International Federation for Information Processing Congress 62, North-Holland Publishing Company, Munich, pp. 359-366, August 27th to September 1st, 1962

Steinbuch, K. and Schmitt, E., 1967, ‘Adaptive systems and learning matrices’, Biocybernetics in Avionics, (H. L. Oestericicher and D. R. Moore eds.), Gordon and Breach, New York, U.S.A., pp. 751-768, 1967

Stonham, T. J., 1985, ‘Practical pattern recognition’, in ‘Advanced digital information systems’, (I. Aleksander ed.), Prentice-Hall International Inc., Englewood Cliffs, New Jersey, U.S.A., pp.231-272, 1985

Schneur, K. E., 1999, ‘Cosmic radiation and aircrew exposure’, Proceedings of Radiation Protection and Dosimetry Workshop, Dublin, July 1998, Nuclear Technology Publishing, December 15th, 1999

The MathWorks Inc., 1995, ‘The student edition of MATLAB, version 4 user’s guide’, Prentice-Hall International Inc., Englewood Cliffs, New Jerseu, U.S.A., pp. 39-237, 1995

Torok, Z. and Platt, S., 2006, ‘Application of Imaging Systems to Characterization of Single-Event Effects in High-Energy Neutron Environments,’ Nuclear Science, IEEE Transactions on Nuclear Science, vol. 53, no. 6, pp. 3718-3725, December 2006

Turing, A. M., 1936, ‘On computable numbers with an application to the Entscheidungsproblem’, Proceedings of the London Mathematical Society, series 2, vol. 42, pp. 230-265, 1936

Ullman, J. R., 1969, ‘Experiments with the n-tuple method of pattern recognition’, IEE Transactions on Computers, pp. 1135-1137, December, 1969

Ullman, J. R., 1973, ‘Pattern recognition techniques’, Butterworth and Co., U.K., pp. 115-120, 1973

von der Malsburg, C., 1973, ‘Self-organization of orientation sensitive cells in the striate cortex’, Kybernetik, vol. 14, pp. 85-100, 1973

von Neumann, J., 1958, ‘The computer and the brain’, Yale University Press, New Haven, Connecticut, U.S.A., pp. 66-82, 1958

Weeks, M., Freeman, M., Moulds, A. and Austin, J., 2005, ‘Developing hardware-based applications using PRESENCE-2’, Perspectives in Pervasive Computing, IEE London, October 25th, 2005

Willshaw, D. J., Buneman, O. P. and Longuet-Higgins, H. C., 1969, ‘Non-holographic associative memory’, Nature 222, pp. 960-962, June 1969

Willshaw, D. J. and Longuet-Higgins, H. C., 1970, ‘Associative memory models’ in ‘Machine intelligence’, (Meltzer, B. and Michie, D. eds.), vol. 5, Edinburgh University Press, pp. 351-359, 1970

Willshaw, D. J. and von der Malsburg, C., 1976, ‘How patterned neural connections can be set up by self-organisation’, Proceedings of the Royal Society of London, Biological Sciences 194, pp. 431-435, 1976

University of Cape Town, 2004, ‘Digital image processing’, 2004,
URL http://www.dip.ee.uct.ac.za/imageproc/stdimages

Xilinx Corporation, 1998, ‘The programmable logic data book 1998’, part number 0010323, 1998

Xilinx, 2001, ‘Virtex 2.5 V Field Programmable Gate Arrays’, Xilinx Incorporated, DS003-1 (v2.5), April 2nd, 2001

Appendices

Appendix A has the two papers resulting from trials at TSL in using CCD's to capture single event effects.

- Analyses of CCD Images of Nucleon-Silicon Interaction Events.
- Single Particle Dark Current Spikes Induced in CCD's by High Energy Neutrons.

Appendix B contains the two presentations submitted and accepted by the IET for publication.

- BAE SYSTEMS, Air Systems Approach to the Problem of Atmospheric Radiation.
- System Level Prevention: Managing SEE Using Error Correction Techniques.

Appendix C contains three international patents, derived from the author's BAE SYSTEMS invention reports. These have been examined and published.

- Ordering by Hamming Value.
- Serial Weightless Data to Thermocode Coded Data Converter.
- Ordering Weightless Binary Tuples According to Hamming Value.

Appendix D consists of a DVD retained on the inside back cover of this thesis. It contains MATLAB emulations, C code simulations, PIC C code, circuit diagrams and FPGA projects. It represents an archive of electronic data generated during the course of the research.

Appendix A

Appendix B

Appendix C

Appendix D

Contents

- *TRIUMF 2006 CHI Squared Graphs*

CHI Squared Statistics

Chapter 6 Self Ordered Map

- MATLAB Simulations

EDA Weightless Boolean Elements

MATLAB Weightless Toolbox