The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

University of Cape Town

NEUROMORPHIC CROSS CORRELATION OF DIGITAL SPREADING CODES

Mark Philip Vismer

University of Cape Town

University of Cape Town Department
of Electrical Engineering

31 April 2008

DECLARATION

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own.

31 August 2008 – University of Cape Town

Author's Signature

Mark Philip Vismer

University of Cape Town

ACKNOWLEDGMENTS

Special thanks must be given to the following people:

First and foremost, the writer wishes to express his thanks to Jane English for her guidance with writing this dissertation.

Further thanks to Samuel Ginsberg for his support as a co-supervisor, Andrew Wilkinson for his help with signal processing, Prof. John Greene for his help with neural circuits and Prof. Mike Inggs for some advice with regard to GPS systems.

The author would also like to thank Michael Rapson for his thoughts and contributions and his family for their support throughout the past two years.

Finally, and most importantly, I would like to thank my supervisor Professor Jonathan Tapson for his guidance and academic support especially in the field of neural computation.

University of Cape Town

SYNOPSIS

NEUROMORPHIC CROSS CORRELATION OF DIGITAL SPREADING CODES

By
Mark Philip Vismer

The study of neural networks is inspired by the mystery of how the brain works. In a quest to solve this mystery, scientists and engineers hope that they will learn how to build more powerful computational systems that are capable of processing information much more efficiently than today's digital computer systems. This dissertation involves a biologically inspired circuit which can be used as an alternative for a cross correlation engine.

Cross correlation engines are widely used in spread spectrum, wireless communication systems that use digital spreading codes to divide a single communication medium into separate channels. This technology is used in many systems such as GPS, ZigBee and GSM mobile communications. The technology is renowned for its robustness and security since it is highly tolerant to signal jamming and spoofing. Digital spreading in wireless communication is also widely used in military systems and has recently been proposed for use in the medical sector for neural prostheses. A limitation of using digital spreading is that the computational demands on the cross correlation engine are normally quite high and is generally considered to be the limiting factor in designing low-power portable devices.

In recent developments proposed by Tapson, it was shown that a two-neuron mutual inhibition network can be used to generate a cross correlation like function (Tapson *et al.*, 2008).

In this work, the two-neuron cross correlation engine is analysed specifically for application on a particular set of digital spreading codes called Gold codes. Based on the analysis, the neuron's response to an input signal is optimised in favour of yielding a neural cross correlation that resembles the mathematical cross correlation more closely. The aim is to find a biologically inspired computer that is practically viable in an electrical engineering application involving a digital spread spectrum communication system.

University of Cape Town

Simulations of the two-neuron cross correlation engine indicate that its accuracy may be sufficient for detecting a Gold code in a signal that also contains five other Gold codes of equal magnitude. This is comparable to the requirements of a cross correlation engine for a GPS receiver. The architecture of the two-neuron cross correlation engine is then adapted so that it can perform multiple cross correlations in parallel. Finally, the concept is proven in an electronic circuit using several neurons in parallel. Each neuron is implemented entirely from passive components, except for one operational amplifier which is required to apply the noise. An FPGA is used to process the neural spike train and generate the cross correlation functions. With this circuit it is shown that two useful cross correlations can be performed simultaneously on a signal that was generated by summing two Gold codes.

The circuit for the neuron is simple and interfaces directly to the FPGA. It shows how a neural cross correlation engine could be implemented on an Application Specific Integrated Circuit (ASIC). Traditionally, noise is considered to be a nuisance in electronics and is often a limiting factor for most systems. A novel aspect of this approach is that noise is required either in the signal or inherently in the neuron in order for the engine to operate. The circuit therefore encourages the use of lower power circuits and a smaller VLSI production scale, as it may be inherently more tolerant to deep submicron noise.

University of Cape Town

TABLE OF CONTENTS

page
Declaration i
Acknowledgments ii
Synopsis iii
Table of Contents v
List of Figures vii
Glossary and List of Acronyms viii
Chapter 1: Introduction 1
1.1 Introduction 1
1.2 Background to this project 3
1.3 Hypothesis and objectives 6
1.4 Plan of development 7
Chapter 2: An overview of relevant neuroscience 8
2.1 Introduction to biological neural networks 8
2.1.1 Overview of a basic neuron 8
2.1.2 Information processing in a neural networks 10
2.1.3 Noise in neurons 13
2.2 Relevant aspects of computational neuroscience 16
2.2.1 Neural spike coding 16
2.2.2 Modelling neuron behaviour 19
2.2.3 Simulating neural behaviour 21
2.3 Neural correlation of digital signals 23
2.3.1 Autocorrelation properties of a single neuron 23
2.3.2 Digital spreading codes in communication systems 25
2.3.3 Cross correlation techniques for wireless communication 28
2.3.4 Neural cross correlation of Gold codes 28
Chapter 3: Analysis of the neural cross correlation engine 30
3.1 A neural autocorrelation engine 30
3.1.1 Overview 30
3.1.2 Autocorrelation of PRN codes in the ISIH of a neuron 30
3.1.3 Expected density of the ISIH 33
3.1.4 Evaluating the expected density 34
3.1.5 Discussion of results 38
3.2 The two-neuron cross correlation engine 39
3.2.1 Overview 39
3.2.2 Investigation of the two-neuron cross correlation engine. 40
3.2.3 Discussion 44
Chapter 4: Refining the neural cross correlation engine 46
4.1 Overview 46
4.2 Dependence of the ISIH on the input signal 47
4.3 In search of a practical cross correlation engine. 53
4.4 The Multi-code Neural Cross Correlation Engine 59
4.5 Simulation of the MNCCE 62
Chapter 5: An electrical circuit implementation of the MNCCE 66
5.1 Overview 66
5.2 Analogue neuron design 67
5.3 FPGA Configuration 71

University of Cape Town

5.4 The soft-core processor 72
5.5 The PC interface 73
5.6 Results 74
5.7 Discussion 76
Chapter 6: Conclusions 78
6.1 The PSTH needs to resemble the signal for cross correlation to occur 78
6.2 A leaky integrate-and-fire neuron will yield a more mathematically accurate cross correlation. 79
6.3 The Multi-code Neural Cross Correlation Engine 80
6.3.1 The MNCCE can perform multiple cross correlations on a signal. 80
6.3.2 The circuit for the MNCCE works, but its performance is inadequate. 80
6.3.3 There may be a better set of spreading codes for the MNCCE 81
Chapter 7: Recommendations for future work 82
7.1 Improve the performance of the MNCCE circuit 82
7.2 Analyse the power consumption of the neural cross correlation 82
7.3 Demonstrate the MNCCE with the NAMARU V2 development board 83
7.4 Further research must follow before using the neural engine for AER 83
References 84
Appendix A 88
Appendix B 89
Appendix C 90
Appendix D 91
Appendix E 92
The uN1510 GPS down converter 92
The uN8130 GPS Baseband Processor 92
The uN3010 GPS Single Chip Solution 92
Appendix F 93

University of Cape Town

LIST OF FIGURES

page

Figure 2-1 9
Figure 2-2 11
Figure 2-3 24
Figure 2-4 27
Figure 2-5 27
Figure 3-1 32
Figure 3-2 35
Figure 3-3 37
Figure 3-4 37
Figure 3-5 39
Figure 3-6 41
Figure 3-7 42
Figure 3-8 43
Figure 3-9 44
Figure 4-1 49
Figure 4-2 49
Figure 4-3 51
Figure 4-4 51
Figure 4-5 54
Figure 4-6 55
Figure 4-7 58
Figure 4-8 60
Figure 4-9 61
Figure 4-10 62
Figure 4-11 64
Figure 5-1 67
Figure 5-2 70
Figure 5-3 70
Figure 5-4 74
Figure 5-5 75
Figure 5-6 75

University of Cape Town

GLOSSARY AND LIST OF ACRONYMS

AER: Address Event Representation

ASIC: Application Specific Integrated Circuit

CDMA: Code Division Multiple Access

Computational neuroscience: A study of the functioning of biologically realistic neurons (and neural systems) and their physiology and dynamics. The aim is to generate models of biological systems. These computational models are used to test hypotheses that can be directly verified by current or future biological experiments.

EPSP: Excitatory Post-synaptic Potential

FPGA: Field Programmable Gate Array

GNSS: Global Navigational Satellite System

IPSP: Inhibitory Post-synaptic Potential

ISI: Interspike Interval

ISIH: Interspike Interval Histogram

Neural computation: Using models developed in neuroscience to achieve a computational objective.

PDF: Probability Density Function

PMF: Probability Mass Function

PRN Code: Pseudorandom number code

GPS: Global Positioning System

Stochastic: Random, but still conforming to a predictable pattern.

Stationary Process: A stochastic process with a probability distribution that is the same at all times or positions. Thus the mean and variance are constant over time or position.

VLSI: Very Large Scale Integration

VHSIC: Very-High-Speed Integrated Circuit

University of Cape Town

CHAPTER 1: INTRODUCTION

1.1 Introduction

A study of the simplest animals confirms that the biological nervous system is a highly capable signal processing, memory and control mechanism. Today, it is commonly accepted that these computational capabilities of the nervous system are a result of its complex spatial configuration and temporal coordination. The study of brain theory is centred on "computational neuroscience". Here, computational techniques are used to model biological neural networks. This is achieved using artificial neurons that are loosely modelled on biological neurons to create artificial neural networks for various computational tasks.

In digital electronics the NAND* gate is known as the universal gate since it is the building block of all digital logic. Similarly, the neuron is the building block of all analogue processing in the brain, and is therefore considered the basic element of a neural network. In 1989, Mead was amongst those who pioneered a new field of electronic computation inspired by neural systems (Mead, 1989). Later, he pointed out a discrepancy in the computational efficiencies of neurobiology and digital electronics (Mead, 1990). It has been estimated that the human brain performs in the order of 3.6×10^{15} synaptic operations per second while consuming approximately 12 W (Sarpeshkar, 1997). A modern personal computer (PC) can perform in the order of 1×10^{9} floating point operations per second and consumes roughly 100 W while doing so.

Although it is difficult to compare the computational power of a single floating point operation with a single spike of a neuron, this does suggest that the brain is operating many times more efficiently than 'cutting-edge' digital electronics. If one considers the overall computational capabilities of the brain (e.g. auditory processing, visual processing, decision making and motor control), it truly appears to be far superior. Indeed, one would expect nothing less from a system that has been evolving over millions of years. Mimicking the brain by building a network of artificial neurons to imitate its operations may lead to greater computational efficiency. This is a strong incentive for understanding and designing neural networks. However, before proceeding, it is worth considering the construction of artificial neurons as well as their performance and efficiency.

* In actual fact, both the NAND and NOR gates are called universal gates, since they can each be used to build all other types of Boolean logic gates.

University of Cape Town

Artificial neurons with characteristics comparable to biological neurons can be built (Newcomb and Lohn, 1995). The most common platform to be used is analogue electronics, and since millions of neurons would be needed, analogue VLSI is the most appropriate presently available technology for their implementation. Analogue hardware is generally preferred over digital, since biological neurons are fundamentally analogue and this eliminates the need for costly analogue to digital conversion systems. Nevertheless, artificial neurons are often simulated digitally on hardware because they can be easily implemented and reconfigured (Fernadez *et al.*, 2004). This makes it simpler to demonstrate the computational capabilities of neural networks, but limits the possibility of low power and efficient operation.

Our world is increasingly dependent on digital electronics with an increasing demand for portable devices that are smaller and consume less power. It can be shown in certain applications that analogue circuitry can make a more power efficient computer than digital electronics because it consumes less power and requires less hardware (Sarpeshkar, 1998). Further improvements in analogue circuitry are generally limited by noise[†], but what Sarpeshkar proposes is that the most efficient computation in terms of resources is with a mixture of digital and analogue electronics. This is achieved by digitising the data at various stages in the computation to eliminate the noise flaw.

Although the work covered in this dissertation is very different from the architecture proposed by Sarpeshkar, it agrees in the context that it is an analogue-digital hybrid system. It consists of an artificial neural network made from analogue hardware and a digital interface to interpret the results. It is hoped that this system will present a low power and low cost method for implementing a cross correlation engine. In so doing, this application of 'bio-inspired' engineering will further the understanding of biological neural networks and demonstrate yet another application of this technology.

University of Cape Town

[†] This is because noise in analogue VLSI circuits is largely due to 'shot-noise' currents which are an end result of thermal fluctuations. It turns out that the noise problem increases as current flow decreases (Sarpeshkar, Delbrück, & Mead, 1993). This limits the minimum power requirements for analogue hardware. Another type of noise is 1/f noise which is sometimes referred to as 'flicker noise' and is generally believed to be caused by impurities or defect traps in the gate oxide of transistors. This noise increases inversely with the area of the transistor and thus limits its minimum size.

1.2 Background to this project

In 1998, the brain-machine interface became a reality when a patient managed to demonstrate logical control over neural signals by means of an electrode that was able to interface with neurons in the brain (Kenny and Bakay, 1998). Today, a new generation of penetrating microelectrode arrays such as the UTAH Electrode Array (UEA) offer both neural stimulation, recording capabilities and more selective access to the neurons of the nervous system (Norman, 2007). These high density electrode arrays enable the study of groups of neurons which help to understand parallel information processing in the brain. These arrays are also being used for neural interfacing to develop sight restoration and the control of external devices such as prosthetic limbs. As design and manufacturing techniques for micro-electrode arrays are continually improved (Fofonoff *et al.*, 2002; Musalla *et al.*, 2007), so the clarity and quality of the brain interface will also be improved. This will open the doors for a more complex method of handling the neural signals which will require more sophisticated control and data management hardware.

Soon neural interfaces will be able to selectively communicate with a large number of individual neurons. Prostheses will require multiple pathways to communicate with each of these neurons, which may lead to hundreds and thousands of individual data channels. The Address Event Representation (AER) communication protocol is an interrupt driven protocol which is ideally suited to this scenario. It relays analogue data in the timing between unique events which are usually transmitted serially along a single data path. Its asynchronous scheme allows more flexibility and adaptability to different environments.

Most implementations of the AER protocol are severely limited in that they use a wired platform which is unsuited to neural prostheses. Harrison *et al.* (2007) demonstrated the importance and feasibility of a wireless communication system for neural prostheses. Their work reports on the development of a system in which an implanted integrated circuit attached to the Utah Electrode Array (UEA) detected neural spikes and transmitted the data wirelessly to an external station. This reduced medical complications for recording neural signals in the test subject because there was no longer a transcutaneous (through the skin) connector to create a path for infection. Signal processing at the recording site (the implanted electrode array) also reduced noise from external signals. Power was supplied to the system externally via electromagnetic induction meaning that a portable power source could easily be serviced or replaced.

A drawback of implanting neural devices is that it is a difficult and risky procedure that leads to several months of recovery and specialised training before an implant can be adequately used (Bakay, 2006). Neural prostheses must therefore be reliable and capable

University of Cape Town

of functioning for decades with little inconvenience and health risk to the patient. It is now being widely accepted that wireless interfaces are considered to provide more reliable and adaptable communications that are better suited for everyday use in a patient (Norman, 2007).

Although wireless communication is becoming more favourable, there are several limitations which are apparent in systems such as that proposed by Harrison *et al*. Firstly; the system relies on a single communication channel so interference on this channel will result in complete failure. Secondly, it uses a synchronous communication scheme which requires more overhead and data management to convert to and from firing events in the neurons which occur asynchronously. This introduces noise into the timing of the neural firing events and the complexity of the communication network will increase as more transmitting nodes are added. A favourable architecture for future prosthetic devices will consist of multiple nodes for individual motor and sensory 'cells' or groups of 'cells' (Folowosele, Tapson and Etienne-Cummings, 2007). Thus there is a need for a system which can asynchronously relay neural spikes between multiple nodes and neural prosthetic implants in the brain. The design of fault-tolerant active electronics also presents a challenge in that circuits must consume little power and produce little heat since continually heating tissue by just a few degrees can lead to cell death.

A solution to these problems is a wireless AER protocol proposed by (Folowosele, Tapson and Etienne-Cummings, 2007). It uses Code Division Multiple Access (CDMA) spread spectrum techniques to transmit neural spikes as 'events' to and from multiple nodes. Data is spread over the communication channel by using a pseudorandom number (PRN) codes to represent each event making them tolerant to signal jamming and interference. These codes are also known as digital spreading codes and allow multiple events to be sent on the same channel. Their exact timing can be calculated by performing a cross correlation on the received signal with the same PRN code. A significant hurdle in this system is that the cross correlation engine is computationally intensive and requires a large amount of electrical power. A proposed solution to this problem is to use a neural correlation engine implemented using an analogue neural network (Tapson and Etienne-Cummings, 2007).

A low power cross correlation engine would have application to many other CDMA based systems. An attractive option is a Global Navigational Satellite System (GNSS). Most of these systems rely on a constellation of orbiting satellites that transmit PRN codes down to earth. An 'end user' device cross correlates these codes with reference codes in order to calculate the precise time that it takes for them to reach the receiver. Since radio signals travel at the speed of light (299 792 458 m/s), the distance from the

University of Cape Town

respective satellites can be calculated. The orbital paths and position of the satellites are also received in the ephemeris data and together with the calculated distances, the position of the user can be determined (Kaplan and Hegarty, 2006).

Currently, the only fully operational GNSS is the Global Positioning System (GPS). It was developed by the U.S. Department of Defence (DoD) in the early 1970s. Over the years, GPS has been improved and upgraded considerably as our civilisation has become increasingly dependant on it. There is also an increasing demand for portable hand held GPS devices as well as having GPS receivers embedded within cellular handsets. One of the drawbacks of incorporating GPS is that it has a high power consumption which is largely due to the correlation engine that has to operate on the received signals from at least four satellites. Currently, improvements in operating efficiency are dependent on improved VLSI manufacturing techniques and materials, but using a neuromorphic cross correlation engine could lead to an even more efficient solution by making use of analogue hardware.

In 1996, Cariani and Delgutte observed an autocorrelation like pattern in the all-order interspike interval histogram (ISIH) of auditory nerve fibres (Cariani and Delgutte, 1996). Later, Tapson showed that similar properties were also characteristic in a far simpler case: the first order (ISIH) for a single integrate and fire neuron (Tapson, 1998). For the first time, these results hinted at a reasonable biological solution to this task that traditionally requires intense computation in digital systems. A recent breakthrough in this line of research proved that a single neuron could also be used to perform an autocorrelation of digital PRN codes (Tapson and Cummings, 2007). This behaviour was extended to create a neural cross correlation engine that could be used to track Gold codes, which are the 1023 bit digital PRN codes transmitted by GPS service satellites. A practical VLSI implementation of such a system would form a crucial part of a low power GPS receiver which can be used in portable devices. In the field of biomedical engineering, it may be the key to wireless communication in the next generation of neural prostheses.

University of Cape Town

1.3 Hypothesis and objectives

The primary aim of this dissertation is to propose a practical neuromorphic cross correlation engine which can cross correlate an input signal with several other reference signals in real time. Cross correlation engines are already implemented ubiquitously using digital electronics, but are computationally intensive and require large amounts of power. The hypothesis of this thesis is that a neuromorphic cross correlation engine that operates with artificial "integrate-and-fire" neurons can be used practically in an engineering application such as a GPS receiver or wireless AER communication network.

Cross correlation of signals occurs commonly in neural networks of the brain of biological organisms, the most classic examples being in the auditory and visual systems. However, the understanding of this process is severely limited, particularly because of difficulties with interfacing with the brain and because of the vast complexity of the interconnecting neurons. Understanding the process that results in the neural cross correlation will not only aid the development of the cross correlation engine, but also further the understanding of how stimuli may be processed in the brain. Hence a part of this dissertation aims to explain the correlation phenomenon that is apparent in the interspike interval histogram (ISIH) of neurons.

The first objective in this thesis is to review applicable literature in the field of neural computation and computational neuroscience. Secondly, various configurations of neural networks will be simulated and studied with the aim of a suitable cross correlation engine that can be made with neurons. The optimal system is deemed to be the system which will function the most efficiently for a GPS receiver or for a CDMA wireless communication platform for implementing a wireless AER communication protocol.

The third objective is to demonstrate a practical implementation of a neuromorphic correlation by using it to cross correlate digital codes similar to those used in CDMA communication protocols such as GPS. GPS makes an ideal platform for a "test-bed" implementation since the digital codes are repeated and transmitted by the satellites continuously. This means that fewer neurons are needed since they can operate on the iterating codes to continuously improve on the accuracy of the correlation.

University of Cape Town

1.4 Plan of development

The work covered in this dissertation involves a merger of diverse and seemingly unrelated fields in engineering. Hence Chapter 2 will commence with a broad literature review describing biological neurons and some fundamental concepts in computational neuroscience. This is followed by a discussion of the correlation properties of neurons and about the relevance of cross correlation in electrical engineering.

The technique of finding the correlation functions of signals from the ISIH of a neuron is an innovative concept that is new to neural computation and electrical engineering. Little is known about the underlying behaviour of the neurons and the mechanism behind the correlation. Chapter 3 therefore presents an investigation into the autocorrelations and cross correlations which can be performed by compiling an ISIH from various neural circuits. A brief analysis using probability theory is then presented to account for the correlation in the ISIH. This analysis is not intended as a formal model of the neural behaviour, but acts primarily as a guideline for designing a neural correlation engine and anticipating its performance. Chapter 4 presents a neural correlation mechanism that is capable of cross correlating an unknown signal with multiple reference signals using only neurons that are driven by the unknown signal. The concept is implemented and demonstrated in a simulation.

Chapter 5 describes an actual circuit of the neural engine. Several neurons are implemented using active analogue electronics and a FPGA is used to generate a combined ISIH from multiple neurons running simultaneously. The circuit is tested and results are presented and discussed. The final chapter considers a practical implementation of the neural cross correlation for a GPS receiver and discusses some of the major challenges that will need to be overcome. Conclusions are drawn from the outcomes of this dissertation, and recommendations are presented.

University of Cape Town

CHAPTER 2: AN OVERVIEW OF RELEVANT NEUROSCIENCE

2.1 Introduction to biological neural networks

Biological neural networks are found in the central nervous systems of animals. These neural networks perform efficient and powerful processing which enables animals to have intelligent behavioural patterns and sensory perceptions. The basic components of a neural network are neurons which will be described in the following text. This will be followed by a discussion of neural networks in the central nervous systems (CNS) of animals.

2.1.1 Overview of a basic neuron

Neurons differ in structure and operation in order to perform various functions in a neural network. Those found in the CNS form a neural network that processes information from receptors (sensory neurons) and governs the activity of effectors (motor neurons) which control the movement of muscles and the secretion of glands. Information in the brain is communicated by electric charges which flow through the soma (cell body), usually from the dendrites to the synaptic terminals.

Electric charges flow through the cell body either actively or passively as a potential difference across the cell membrane. Standard texts such as *The Handbook of Neural Networks and Brain Theory* (Arbib, 2003) describe these processes. In a motor neuron (similar to that shown in Figure 2.1), the stimulus starts at the *synaptic cleft* which is a small gap between the pre- and postsynaptic cell membrane. The presynaptic terminal of the neuron releases a chemical called the *neurotransmitter* into the synaptic cleft. Receptors in the postsynaptic cell membrane detect the molecules of the transmitter and cause a change in its conductance to various ions. This in turn results in a potential difference across the membrane. In cells such as rods, cones and bipolar cells of the retina, a potential difference can propagate through the cell passively. In most other cells this mechanism is inadequate because the charge decays before reaching the presynaptic terminal. Instead, the potential difference has to be actively propagated along the cell membrane as a pulse known as an *active potential*.

An active potential in a cell begins at the *axon hillock*, which is the point on the *soma* from which the *axon* branches out. Also protruding from the soma are numerous small branches called dendrites which are formed by the cell membrane containing receptors. Stimulation at the dendrites increases the potential in the soma until it reaches a specific

University of Cape Town

threshold at the axon hillock. When the membrane potential reaches this threshold, it triggers a regenerative process which can be abstracted as a wave-like change in conductance of the cell membrane to ions in the cell's soma and surrounding intercellular fluid. A simplified diagram of a neuron with the direction of the flow of an impulse is shown in Figure 2-1. The change in conductance of the cell membrane is more specifically a process which involves ions being transported across the membrane by proteins in the cell membrane which form specific *ion channels*. These channels can actively pump selected ions across the membrane, against the concentration or potential gradient. They can also allow selected ions to passively diffuse across the membrane.

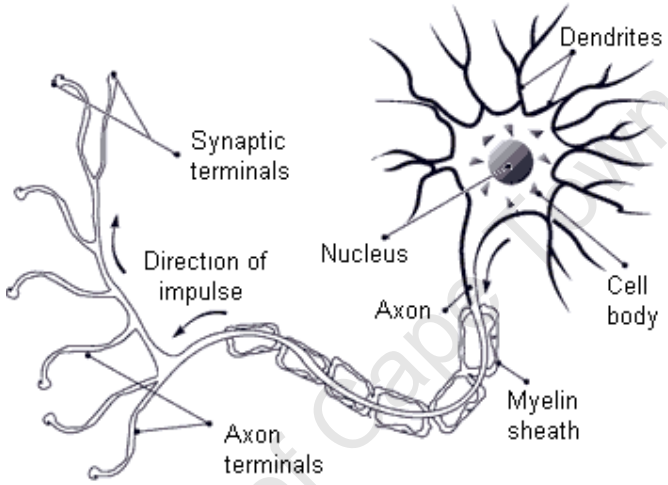


Figure 2-1: The flow of an action potential down the axon of a motor neuron.

University of Cape Town

The controlled flow of ions across the membrane causes a sharp increase in the potential difference, followed by a sharp decrease. This results in a spike or *active potential* which stimulates the same behaviour in the adjacent membrane and so the impulse propagates down the axon. The impulse travels into each of the *axon terminals* and down to the synaptic terminal where the neurotransmitter is released. After an action potential, there is a short *refractory period* during which new impulses cannot be generated even if there is a very strong stimulus. This limits the maximum spiking rate of the neuron.

Mostly sodium and potassium ions are responsible for mediating the change in potential across the membrane. Hodgkin and Huxley (1952) developed the famous equations which describe the action potential based on how the conductance of the membrane to sodium and potassium ions is dependant on voltage and time. These equations have provided mathematical insight into the functioning of neurons. Subsequent research into models of ion channels in the cell membrane account for the various terms in the Hodgkin-Huxley equations and show that small patches of neural membrane may even

act as complex computing elements. This role of ion conductance sets a particular *latency*, *rhythm*, or *firing pattern* of the action potential in different classes of neurons found in different regions of the brain (Bargas and Galarraga, 1998). These neurons respond differently to external stimuli and as a result they process information differently as well.

2.1.2 Information processing in a neural networks

The brain consists of a vast network of neurons. These neurons are interconnected in loops and tangled in skeins so that signals entering the network from the receptors interact with billions of other signals already traversing the system. These signals yield sensory information, make logical decisions and control effectors. They can also modify the very properties of the network itself. This is an example of adaptability in neural networks in that future behaviour reflects prior experience. Understanding how neurons and neural networks process information in the brain involves the vast and nascent field of computational neuroscience. Past work and research in this field that is relevant to this thesis is reviewed in the following literature.

The action potential is the elementary unit of signal transmission in a neural network. Models of action potentials, based on neural behaviour during *in vivo* recordings, generally assume that the form of action potentials is the same and therefore the information content of an action potential must be zero. Unlike action potentials, the form of chemical synapses varies considerably. This is largely dependent on the type of neurotransmitter that is used (Destexhe, Mainen and Sejnowski, 1995).

A particular type of neurotransmitter called a neuromodulator, modulates behaviour of the synapses and the sensitivity of receptors in the neural membrane (Dickinson, 1995). This enables a neural network to switch its overall mode of activity and is partly responsible for the adaptability of neural networks. Two common neurotransmitters are relevant to even the simplest neural models. These neurotransmitters influence the sign of the postsynaptic potential (PSP) resulting in either an *excitatory* (positive) or an *inhibitory* (negative) chemical synapse. An excitatory synapse tends to move the potential difference across the postsynaptic membrane in the direction of the firing threshold. This is known as *depolarisation* and is called an excitatory postsynaptic potential (EPSP). An inhibitory synapse tends to move the potential difference away from the threshold. This is known as *hyper-polarisation* and is called an inhibitory postsynaptic potential (IPSP).

Under some conditions however, the EPSP or IPSP may in fact cause the opposite change in membrane potential. In mathematics, this would relate to the output of a multiplication being either positive or negative depending on the sign of the first multiplicand (synapse) and the second (post synaptic potential). This is discussed in

University of Cape Town

Gerstner and Kistler (2002). For an inhibitory synapse, the sign of the postsynaptic potential can change to positive given certain conditions of hyper-polarisation of the membrane potential. Similarly, Dickinson also found that the sign can change, but in this case, due to the effect of other neuromodulators which reduce the efficacy of the chemical synapse, allowing a direct electrical synapse to dominate it (Dickinson, 1995).

Postsynaptic potentials are further influenced by the shape, size and *location* of synapses on a dendrite. Synapses at the distal end of a dendrite are expected to evoke a smaller postsynaptic response at the soma and will naturally have a lower 'weight', than a synapse that is located directly on the soma. This can lead to a phenomenon called "shunting inhibition" where the input from hundreds of excitatory synapses can be drowned out by an inhibitory synapse on the soma (Borg-Graham *et al.* 2002; Gerstner & Kistler, 2002).

The geometrical relationship of synapses is also significant when considering the cooperative effect of many sub-threshold postsynaptic potentials. When these accumulated potentials yield a potential change at the axon hillock which exceeds the threshold, an action potential is generated. In this way, synchrony between two or more neurons which signify special events can be detected. Similarly, synapses from different distances along their axon hillock can lead to a type of spatiotemporal cross correlation. This is due to phase shifts as a result of action potentials having to propagate down different lengths of the axon. This phenomenon is illustrated in the simple figure below:

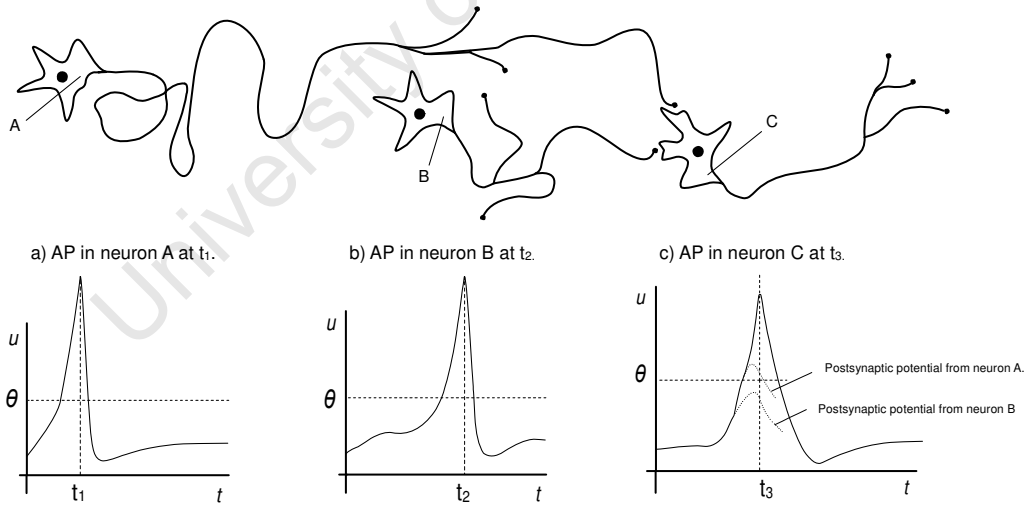


Figure 2-2: Neuronal processing as a result of length-dependant conduction time and the position of neurons. A stimuli occurring at neuron A before occurring at neuron B, causes neuron C to fire.

This is a model similarly represented by Rall (1964) and then later by Borst (1989). It can easily be related to a common scenario involving two light sensors which are each represented by neuron A and B. When a body passes the sensors, it casts a shadow

University of Cape Town

which generates a stimulus first in A, then in B. As a result of the stimulus, neuron A fires and generates an action potential which has to travel a long distance before it arrives at the synapse at C. The action potential from B firing travels over a shorter distance to the synapse such that both the synapse from A and B occur simultaneously. This causes C to spike as well. In a large neural network consisting of millions of neurons, a single neuron in the vertebrate cortex often connects to more than 10^4 postsynaptic neurons. Thus multiple neurons, which each receive synapses from different lengths along axons of many sensory neurons, could detect various time differences between an event or stimulus. This process has been previously linked to pitch perception in the auditory system (Licklider, 1951) and motion detection in the visual system (Hassenstein and Reichardt, 1956).

An alternative idea regarding neural computation was presented by McCulloch and Pitts in 1947 and was the first to link neural networks with digital computation by using a 'high-level' binary model. McCulloch and Pitts showed how excitation, inhibition and threshold could be used to create discrete-time logic elements such as 'AND' or 'OR' gates. This effectively proves that neurons can perform any task that can be performed by a digital computer.

In a McCulloch-Pitts neuron, an input or output is represented as logic '1' with a spike or logic '0' represented by the absence of a spike. Each connection or synapse, from the output of one neuron to another has an attached weight (w). An excitatory synapse represents a positive weight and an inhibitory synapse represents a negative weight. If the potential increases above the threshold (ϑ) then the conditions for the gate are met and the neuron outputs a spike. This idea can be extended to describe neurons with multiple inputs. If the probability of the neuron firing is dependant on the sum of multiple inputs, its behaviour relates to that of an 'AND' gate in that it requires a PSP from many neurons before it will fire. A similar mechanism in which the coincident arrivals of spikes from just two input neurons can function as a multiplier has also been described (Srinivasan and Bernard, 1976). An alternative architecture to perform an "OR" function could involve a sensitive neuron that needs only one or very few PSP's before it fires.

These models provide elementary ideas for information processing in the brain. Another example of information processing can be shown based on the assumption that neurons are also subject to a large amount of noise. This would mean that the firing event of a neuron is a stochastic process and therefore no longer deterministic in time. In a network of many neurons in parallel, each receiving an identical unknown stimulus, the probability of each neuron firing can be determined by observing the total number of neurons that fire. In such a scenario, each neuron could then synapse to an 'AND'

University of Cape Town

type neuron which will only fire when the combined input stimuli reach a certain threshold. The result is a threshold detector for an input stimulus. This is an example of how important noise could be in a neural network. In fact, it is generally accepted that noise may play an important role in neural processing. It will be discussed in further detail in the next section.

2.1.3 Noise in neurons

In vivo experiments of neurons during spontaneous activity, have found highly irregular and unpredictable neural spike patterns (Shadlen and Newsome, 1994; Softky and Koch, 1993). Repeated experiments under similar conditions also vary considerably (Hubel and Wiesel, 1977, 1959 cited in Gerstner and Kistler, 2002). More simply, one concludes that for a naturally functioning biological neuron there is a broad distribution of the intervals between one spike and the next and these intervals appear to be unrelated to external stimuli. A simple explanation for this obviously complex scheme of information encoding is that most of the spikes are influenced by noise. This is a widely accepted phenomenon, because noise occurs on a molecular level (Kramers, 1940) and is thus ubiquitous. Further evidence of noise in biological neurons is that it can be linked directly to processes in the cell (Shadlen and Newsome, 1998).

Since a neuron in the cortex receives synapses from thousands of other neurons it is impossible to accurately map how a neuron spikes in response to its input. As a result, it is difficult to determine whether the observations are just noise or a highly efficient method of coding information. When neurons are stimulated internally by driving them with a known time-dependent intracellular input current, a pattern is recognisable in the spike intervals (Zador, 1998; Bryant and Segundo, 1976; Mainen and Sejnowski, 1995 cited in Gerstner and Kristler, 2002; Cariani and Delgutte, 1996). This indicates that the behaviour of neurons is largely dependant on their input, but that noise still plays an important role in the output pattern of the inter-spike intervals.

The experiments by Cecchi *et al.* (2000) provide some insight into the effect of noise on the intervals between spikes. They found that the variation of noise in spike intervals is dependent on the input signal and follows from the theory that uncertainty in the firing (spiking) time is a result of noise in the neuron. Cecchi *et al.* conclude that noise can affect the generation of an action potential in two ways: (i) it introduces randomness into the membrane potential; and (ii) it introduces randomness into the firing threshold of the neuron. Thus, when the membrane potential is near the firing threshold (so that the difference is smaller than the noise voltage), it is impossible to predict whether a neuron will fire or not. This uncertainty is called the *spike initiation variability* and is as a result of what is commonly known as *escape noise* (Gerstner and Kistler, 2002).

University of Cape Town

Intuitively, spike initiation variability should decrease if the time period over which the neuron's membrane potential is near the threshold is reduced.

Cecchi *et al.* claimed that this can be achieved by increasing the rate of change of the membrane potential ($\Delta v/\Delta t$) over the time period. The change in a neuron's membrane potential is dependent on the intensity of its input. Thus the variation in the firing time, which equates to noise in the inter-spike interval, is dependant on the input as well. This theory was verified by running simulations on various models of neurons. It was also verified with *in vivo* experiments. Cecchi *et al.* analysed recordings from the *visual pathway* in the cortex of a cat and found consistent results of variation in inter-spike intervals. When shown a moving bar with low contrast and low speed, the inter-spike intervals were more distributed than when exposed to a moving bar with high contrast and high speed. Cariani and Delgutte (1996) found similar results when cats were exposed to acoustic waveforms. An analysis of spike patterns of the *auditory* nerve fibres found that interval distributions had high peak-to-mean ratios in response to salient pitches. Correspondingly, low peak-to-mean ratios were reported for weak pitches. A logical conclusion from both these results is that the efficiency of a neural network that relies on information encoded as interspike intervals depends on the intensity of the input signal and the magnitude of the noise.

The most significant source of noise that is specific to neurons and especially those in the cortex arises from ion channels in the membrane of the neuron (Faisal, White and Laughlin, 2005). One cause for this *channel noise* is a fluctuating, finite number of open ion channels in the membrane that varies the conductivity of the membrane to arriving synapses (White et al., 1996, 2000; Schneidman et al., 1998 cited in Gerstner and Kistler, 2002). It is also widely accepted that ion channels are in part responsible for *electrical noise* in neurons. Ion channel noise results from thermodynamic fluctuations which influences the gating behaviour of the ion channel. This contributes to the variations in the membrane potential discussed earlier. In addition, Faisal and Laughlin (2007) proved the significance of spike *interval noise* from channel noise which perturbs the action potential as it travels down the axon. In other words, unreliability of the ion channels varies the propagation of the action potential along the axon which affects its arrival time at the synapse. This noise can alter the spike timing in the order of milliseconds over propagation distances of millimetres in thin axons. As one would think, these stochastic effects are also larger for longer axons, where the distance the action potential has to travel is larger as well. Faisal and Laughlin found that in thin axons such as those in the cortex, this propagation variability can in fact exceed the spike initiation variability discussed previously.

University of Cape Town

It seems that there is no agreed upon theory of the role of noise in neural networks. However, the presence of noise is certain and there are clear indications that noise does play an important role in biological neural networks. Since the brain is a powerful and highly evolved processing system, it seems unlikely that the noise is a defect. This suggests that noise may be a key element for the neural processing and in fact, recent work supports this theory (Salinas, 2006; Ma, Beck and Latham, 2006). We should thus look to designing systems that rely on noise as a key component for operation.

University of Cape Town

2.2 Relevant aspects of computational neuroscience

2.2.1 Neural spike coding

It is generally accepted that information is carried in the number, timing and location of neural spikes (Gerstner and Kistler, 2002) and that little information is portrayed by the form of the action potential since all action potentials of neurons appear to be the same. Consequently, the trend is to refer to action potentials as simple spikes modelled as Dirac δ-pulses and then investigate the encoding in their spatial-temporal patterns. Some techniques of neural spike encoding will be reviewed.

A well known concept of information encoding in neural networks is *rate coding* and is discussed by Gerstner and Kistler (2002). In this scheme, information is believed to be encoded in the *mean firing rate* of the neuron. A means of expressing this information is the Peri-Stimulus Time Histogram (PSTH). This graph reports on the neural response of an individual neuron or a population of neurons at time t. To generate the PSTH, spikes are counted over period's Δt in length, to form bins at various points along t. In order to generate a meaningful histogram, the experiment has to be repeated many times under identical conditions so that the number of spikes that occur for each Δt can be accumulated.

The PSTH is a very useful plot in neuroscience since it can be used to derive the mean spike rate over an interval or the probability of a neuron firing over an interval. It is used frequently during *in vivo* experiments to record the response of a single neuron to a time referenced stimuli. In such experiments, a neuron is stimulated in independent tests and the spike response times are recorded on the same time scale with respect to the stimulus starting at t=0. The number of spikes during different periods Δt are then counted to form $n_K(t\,;\,t+\Delta t)$ where the *spike density* of the PSTH as defined by Gerstner and Kistler (2002) is

$$\rho(t) = \frac{1}{\Delta t}\frac{n_K(t;t+\Delta t)}{K} \qquad 2\text{-}1$$

In the above equation, $n_K(t\,;\,t+\Delta t)$ is the number of spikes accumulated over K independent repetitions of the experiment for a single neuron. This technique of using the sliding window Δt, is also discussed in Dayan and Abbott (2005). Equation 2-1 can be extended for a population of neurons by redefining K as the product of the number of neurons in each experiment and the total number of experiments.

Probability theory will be used throughout this dissertation. The probability mass function (PMF) will be used to describe the response of a practical neuron in similar

University of Cape Town

context to the PSTH. When derived from a recorded spike train, the PMF ρ_j is defined as:

$$\rho_j = \frac{n_{\mathrm{K}}\left((j-1)\Delta t; j\Delta t\right)}{n_{\mathrm{K}}\left(0;T_K\right)}, \quad j \in \mathrm{N} \qquad 2\text{-}2$$

This equation is used with spike data from a simulation or recording where j represents the index of the time interval which relates to a bin in the PSTH. The discrete time axis j is referenced to when the experiment started at $t = 0$. Thus given that *one* spike occurred in some bin j of the set N, then $P\{N=j\} = \rho_j$ represents the probability of that spike having occurred at bin j of the PSTH. Note that the length of each bin is Δt and T_K is the length of time over which the experiment is recorded. Thus ρ_1 represents the probability of a spike occurring in the first "bin" time interval of the experiment. If a neuron is being stimulated by a signal x, which is a known function of time, t, then the PMF of the neuron firing is denoted by $P_x\{N=j\} = \rho_{j|x}$. Bear in mind that x must occur with respect to the same time reference and time scale for each experiment, but there is *no restriction* on the state of the neuron when the experiment starts.

A special case of the PSTH is when the neuron *is restricted* to starting from its reset state at $t = 0$ and only the first spike is counted. This response is called the *time-to-first-spike* (Gerstner and Kristler, 2002). Using $n^*_{\mathrm{K}}(t;t+\Delta t)$ to describe this special case of the PSTH, the PMF of the time-to-first-spike in terms of a discrete function of each bin interval at interval i from when the stimulus started at t=0, can be expressed as:

$$f_i = \frac{n^*_K\left((i-1)\Delta t; i\Delta t\right)}{K}, \quad i \in \mathrm{N} \qquad 2\text{-}3$$

Thus f_i is also the density of the time-to-first-spike with a measure of spikes per bin length Δt. Note that i is the discrete time axis with reference to when the neuron started, whereas j is used to denote the discrete time relative to the start of an experiment at t=0. Sometimes, we may wish to consider f_i for a neuron that is stimulated by an input signal x that is at a particular phase α when the neuron starts. The notation $f_{i|(x,\alpha)}$ is used to denote this scenario. When derived from spike trains (from either *in-vivo* experiments or a simulation), $f_{i|(x,\alpha)}$ describes the probability of an interspike interval falling into a bin of the PSTH that was generated from a neuron when the stimulus to the neuron was at a particular phase (α) when that neuron started. Later a more general definition as a continuous function of τ will be introduced.

In the study of neural computation, the time-to-first-spike coding scheme generally relies on the assumption that neural information is transmitted in the first spike. As a realistic model, it is often useful for when dealing with stimuli that undergo abrupt

University of Cape Town

changes. Such behaviour is often observed in real world scenarios where neural systems have fast response times.

Before concluding this section, another method for analysing spike patterns needs to be discussed. It is called the Interspike Interval Histogram (ISIH) and applies to a scenario in which a single neuron is reset to its initial condition and continues to operate immediately after it fires. The ISIH is compiled by recording the spike times of a single neuron with respect to the time of *previous spikes*. A *first order* ISIH is the time between each spike and its previous spike only and is denoted by $ISIH^1$. In this dissertation a first order ISIH will be used and can be assumed when no superscript is specified. A first order ISIH can be compiled for a given recording of neural activity by measuring the intervals between successive spikes and then sorting the intervals into separate bins that make up the histogram.

Under certain circumstances, the PMF for the time-to-first-spike is identical to that of an interval of the ISIH. Consider a regime in which a neuron is reset immediately after it fires. If its behaviour is assumed to remain the same throughout an experiment and its input is zero (or constant), then each spike interval is independent of all other spike intervals. The spike interval is therefore analogous to the time for the neuron to fire from when it last started. The first order ISIH is therefore identical to the PSTH compiled for the time-to-first-spike and can be defined by f_i. However, if a neuron is stimulated by a time dependent signal, then each spike interval becomes dependent on when the previous spike interval ended. This is because it defines the phase of the stimulus when the neuron starts. The probability of a particular spike interval is therefore defined by $f_{i|(x,\alpha_j)}$.

The ISIH is important in neuroscience since it can be used to calculate the mean firing rate for a neuron by taking the inverse of the mean interspike interval. A subject of much debate is the usefulness of the ISIH for describing brain activity. The importance of the ISIH for the analysis of motor control can be traced back to the pioneering work of Adrian in 1926 and 1928 (cited in Gerstner and Kistler, 2002). However, when analysing computational properties of the brain, one of the main arguments is that reaction times in behavioural experiments are often too short to allow for analysis over the long periods necessary to generate the ISIH. Nevertheless, evidence suggests that the central auditory processing mechanism analyses the ISIH to determine pitch (Cariani and Delgutte, 1996). Noting how quickly the auditory system perceives sound, it seems plausible that there must be another method to generate the equivalent information in the ISIH more quickly - most likely by running multiple neurons in parallel.

University of Cape Town

2.2.2 Modelling neuron behaviour

Modelling neurons is important for understanding their behaviour, and predicting their response to inputs under various conditions. These models can then be used to predict the behaviour of more complex neural networks which are aimed at achieving some or other computational task. Many neural spikes are required to achieve a computational objective such as recognising a pattern or simply generating an ISIH. In a digitally based simulation, the computations for modelling the behaviour of neurons can become quite demanding. The challenge is to limit the computational requirements of the simulation by balancing a trade off between the accuracy of the neurons and the operational requirements of the neural network.

In this thesis, a simple integrate and fire (IF) model will be used to simulate each neural operation. This model is identical to that used in the paper by Tapson (1998) and is sometimes criticised because the behaviour of the neuron has been stripped down to a bare minimum. The main reason for using this model is its simplicity. It is also the only diffusion model which can be solved exactly for all parameter values (Feng, 2004). Although highly simplified, the IF model has become widely accepted for the study of neural spiking patterns since it describes synaptic integration which is the most distinguished behaviour of a neuron. It has consequently become a foundation for other more complex models and has been well documented in well-known texts on neural computation (Gerstner and Kistler, 2002; Dayan and Abbott, 2005). Burkitt (2006a) provides a brief background to this model which involves an analysis of its behaviour and the model's relevance to a real neuron.

The IF neuron is a *perfect integrator* (or leakless integrate and-fire neuron model), in which the decay of the membrane potential over time is neglected. Essentially, the IF model involves integrating all inputs until the neuron's potential reaches a threshold after which it fires. The process then starts again from the initial potential of the neuron. Sometimes the refractory period of the neuron is also modelled with a delay τ_r before the neuron starts again.

A fundamental property of the IF model is that all the random changes in the membrane potential are modelled as continuous white (Gaussian) noise with drift to form a continuous time *stochastic process*. In other words, the IF-neuron is a random process whose condition can not be prescribed with certainty. One therefore needs *probability theory* which predicts the state of the IF-neuron with a degree of certainty. A good introduction to probability theory is presented in Bertsekas and Tsitsiklis (2002). Further references for more intricate concepts can be found later in this section.

University of Cape Town

To adopt a common convention, a particular stochastic process will be described by a *random variable* which can take on numerical values with certain probabilities. The *random variable* represents the state of the process and is a real valued function defined on the elements of a sample space. A random variable will be denoted with a capital letter, and an element of its sample space will be denoted by the equivalent letter in lower case. In the context of an IF neuron, the process describing the neuron's potential is represented by the family of random variables, $U(t)$, for $0<t<T_\vartheta$, where T_ϑ is the time at which the neuron first fires. If $U(t)=u$, then the process is said to be in state u at time t, where $u \in U$ is the potential of the neuron. Next we express a fundamental property of the neuron's potential as an element of the state space of U. The IF neuron is an integrator so the rate of change of the neuron's potential is:

$$\dot{u}(t) = \mu + \sigma\eta(t) \qquad \text{2-4}$$

where μ is the drift, and $\sigma\eta(t)$ is the Gaussian noise with standard deviation of σ. The equation is characteristic of a *diffusion process*, namely a continuous-time, continuous-state Markov process described in Bharucha-Reid (1960). It also relates to Brownian motion with drift (Ross, 1983) and thus U can be expressed by using the *Wiener process:*

$$U(t) = u_0 + \mu t + \sigma W(t) \qquad \text{2-5}$$

where $W(t)$ is a standard Wiener process such that U is a Wiener process with initial value $u(0) = u_0$, drift μ and infinitesimal variance σ^2.

A primary discussion about using the Wiener process to model the potential across a neural membrane can be found in Tuckwell (1998). For simplicity, the IF model will be used with the assumption that $u_0 = 0$ at $t=0$, which means that the neuron starts with its potential at zero. Using stochastic theory and the definition of the Wiener process, the probability density function $P\{U(t)=u\}$ can be solved. It is defined by the function $f(u,t)$ as

$$f(u,t) = \frac{1}{\sqrt{2\pi\sigma^2 t}} \exp\left(-\frac{(u-\mu t)^2}{2\sigma^2 t} \right) \qquad \text{2-6}$$

and is known as a *normal distribution* of u with variance $\sigma^2 t$ and mean μt at time t. The next step in describing the IF model, is to formalise the spiking event which is characterised by a firing time denoted by $t^{(f)}$ and defined by the threshold criterion:

$$u(t^{(f)}) = \vartheta \qquad \text{2-7}$$

Here, $t^{(f)}$ is the time for u to reach the threshold ϑ for the *first time* since the neuron was last reset. The interval between when the neuron starts and when it fires is thus called

University of Cape Town

the *first passage time* of the membrane potential, $u(t)$, across the threshold (Burkitt, 2006a). Previously, the spike interval histogram, which represents the probability mass function of the time-to-first-spike, was discussed in 2.2.1. Relating this to the ideal IF model, the *density of the first passage time* is the continuous-time, theoretical probability density function (PDF) of the time-to-first-spike. The first passage time density function to ϑ is denoted by $f_\vartheta(t)$. Finding it analytically is important for deducing the information encoding and processing properties of an IF neuron.

The solution for $f_\vartheta(t)$ can be obtained by using the renewal equation and Laplace transforms (Tuckwell, 1988). It was first introduced to neuroscience by Gerstein and Mandelbrot (1964) and is defined by the equation below provided the neuron has the initial state of $u_0 = 0$ at t=0.

$$f_\vartheta(t) = \frac{\vartheta}{\sqrt{2\pi\sigma^2 t^3}} \exp\left(-\frac{(\vartheta - \mu t)^2}{2\sigma^2 t} \right) \qquad 2\text{-}8$$

A limitation of this function is that it can only be used when the input stimulus to the neuron is stationary. Since the ISIH and the time-to-first-spike PSTH are identical for stationary input (as discussed in 2.2.1), it can be said that $f_\vartheta(\tau)$ describes the ISIH for the IF neuron under these conditions. Here τ was used instead of t and denotes time referenced with respect to the last spike which occurs at τ=0. If a neuron is stimulated continuously by a time dependent input signal x, then $f_\vartheta(\tau)$ no longer holds. The function $f_\vartheta(\tau|x,\alpha)$ will be used to describe this behaviour. The function is the probability density function of a neuron firing when stimulated by the signal x which was at phase α when the process started (or when the neuron last fired). This function is synonymous to interspike interval densities (ISI) derived from *constrained stimuli* as described in Plesser (1999).

2.2.3 Simulating neural behaviour

In computational neuroscience, it is necessary to be able to predict the response of a neuron or set of neurons to a particular input signal. The predicted responses can be compared with those of *in vivo* experiments and can be analysed in search of clues as to how information is processed and represented in neural networks. A problem is that the neural dynamics are usually too complex to solve for analytical models. A simpler method for determining a neuron's behaviour is to run a discrete simulation of the neuron and approximate the integration.

For the perfect integrate and fire model just described, the discrete behaviour of the neuron's potential can be described by

University of Cape Town

$$u_{n+1} = u_n + \mu T_s + \eta_{n+1}\sqrt{\sigma^2 T_s}\ , \quad u \in \mathrm{N}_0 \qquad 2\text{-}9$$

where the sample time T_s is defined as the time in seconds between each interval. The change in voltage per sample as a result of the drift is μT_s and the deviation of the noise in the sample space is $\sigma T_s^{-1/2}$ such that σ is the deviation of the noise in the real process. When u_n reaches the threshold value ϑ, a spike event is generated and the neuron's potential (u) is reset to its initial value u_0=0. In this discrete version of the process, η_n represents a randomly generated variable with the standard Gaussian distribution.

PSTH's and ISIH's are generated from the spike data. However, the simulation is of a stochastic process and thus the histograms can only provide an approximation of the corresponding distribution. An analytical model to predict the spike interval of a neuron is often difficult to find. Instead, discrete methods can be used by first approximating the process which describes the potential of the neuron as a Markov Chain (discrete in space and time). The Chapman-Kolmogorov equation can then be used to determine the state of the potential from its previous state and transitional probabilities (Bharucha-Reid, 1960). To model the behaviour of a neuron, one must also account for the survival probability S_n which can be interpreted as probability that the potential has not yet reached its threshold. The absolute probability for the state of the neuron at interval n+1 is therefore:

$$p_{n+1}(j) = S_n \sum_i p_{\mathrm{n}}(i) q_{i,j} \qquad 2\text{-}10$$

In equation 2-10, S_n is the survival probability of the neuron (i.e. the probability that the neuron has not fired by interval n). The function $p_n(i)$= P{U_n=i} is the probability that the neuron's potential will be in state i at interval n. The last parameter, $q_{i,j}$, is the transitional probability of the membrane potential from state i to a state j and has the form of the Gaussian distribution with mean i - (j+μT_s) and variance equal to the variance of the white noise in the neuron (η_n). The probability of the neuron firing at interval n is thus:

$$\mathrm{P}\{U_n \geq \vartheta\} = \sum_{i=\vartheta}^{+\infty} p_n(i) \qquad 2\text{-}11$$

The survival probability can therefore also be calculated from:

$$\mathrm{S}_{n+1} = 1 - \mathrm{P}\{U_n \geq \vartheta\} = \sum_{i=-\infty}^{\vartheta} p_n(i) - p_n(\vartheta) \qquad 2\text{-}12$$

A MATLAB function that uses these equations to calculate the probability function of the time-to-first-spike is attached in Appendix A.

University of Cape Town

2.3 Neural correlation of digital signals

2.3.1 Autocorrelation properties of a single neuron

The idea that a neuron can be used to do an autocorrelation was first presented by Tapson in 1998. The work followed on from the analysis of temporal discharge patterns by Cariani and Delgutte where an autocorrelation like pattern in the all-order interspike interval histogram (ISIH) of auditory nerve fibres was observed (Cariani and Delgutte, 1996). Tapson found that he could replicate the effect in a simulation with a single IF neuron. The behaviour of the neuron is identical to the "perfect integrate-and-fire neuron" described previously.

The input to the ideal IF-neuron is drift and diffusion noise (the Wiener process). This models the effect of an ensemble of randomly occurring EPSPs and IPSPs (Tuckwell, 1998). The drift could be said to account for either a slightly higher arrival rate of EPSP or the natural tendency of the neuron to charge itself. Noise, which occurs inherently in the neuron as a result of thermal fluctuations and chemical processes in the cell, is also modelled by the diffusion process. Tapson adapts the model to allow for an additional, larger stimulus that is still small relative to the firing threshold of the neuron. From equation 2-4, the potential on the neuron can be described by:

$$\dot{u}(t) = \mu + \sigma\eta(t) + h(t) \qquad \text{2-13}$$

To elaborate, the rate of change of membrane potential is broken up into three components: $\sigma\eta(t)$ is a random rate of change with a Gaussian distribution and models *diffusion noise* in the neuron; $h(t)$ represents the net rate of change in membrane potential as a result of an input stimulus, and μ represents the mean drift rate of the neuron.

Both $\eta(t)$ and $h(t)$ have a mean of zero so μ ensures that the neuron will eventually fire. Input $h(t)$ is calculated for the discrete-time case by scaling an input signal $x(t)$ by a gain factor g. Adapting equation 2-9, the behaviour of the neuron can be simulated by

$$u_{n+1} = u_n + \mu T_s + \eta_{n+1}\sqrt{\sigma^2 T_s} + g x_{n+1} T_s \qquad \text{2-14}$$

As before, when u_n reaches the firing threshold (ϑ), a spiking event is generated and the process restarts from u_0=0.

The key to achieving the autocorrelation with the IF neuron, lies in the method of handling the temporal spiking information. In Tapson (1998), the data is presented as a 1st order Inter-Spike Interval Histogram (ISIH) which is compiled from the spikes of a single neuron with a *continuous* input signal. The neuron starts immediately after it spikes, but the input stimulus is not restarted. Each bin of the histogram is compiled by

University of Cape Town

accumulating the set of spike intervals that fall into that bin. Thus bin i would be the number intervals that were in the range $[(i-1)\Delta t;\ i\Delta t)$. For a histogram that shows the maximum resolution, the length of each bin is set to $\Delta t = T_s$. The result extracted from Tapson (1998) shows an ISIH generated for a periodic input signal to the neuron:

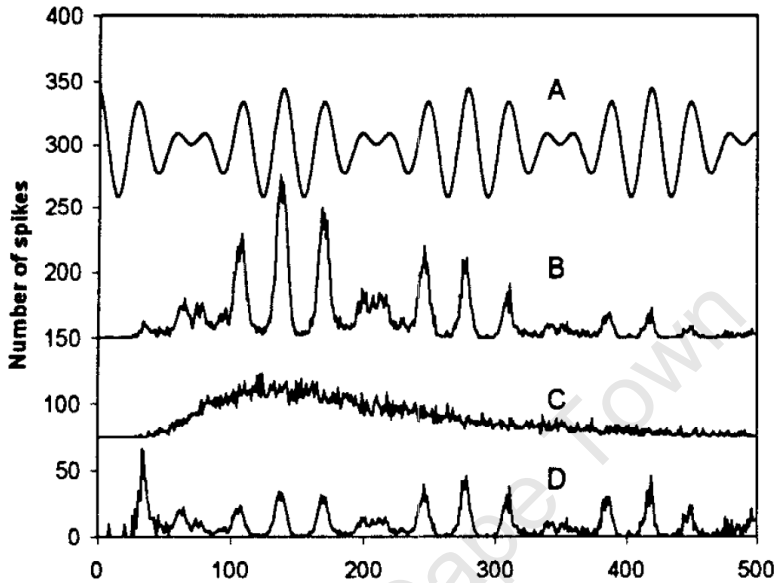


Figure 2-3: Extracted from Tapson (1998). The 1st order Interspike Interval Histogram (ISIH) for an IF neuron that restarts immediately after it fires and has a periodic input signal $x(t) = \sin(4\omega t) + \sin(5\omega t)$ that is input continuously. The drift of the IF neuron was set to ω. Curve A shows the calculated autocorrelation of $x(t)$. Curve B shows the ISIH generated from the spike events of the neuron. Curve C shows the ISIH for the same neuron in a separate experiment with no input signal and has the shape predicted by Gerstein and Mandelbrot (1964). Curve D is simply curve B divided by a smoothed version of C so that the autocorrelation is more apparent.

University of Cape Town

In Figure 2-3, an autocorrelation is clearly evident in the ISIH histogram. This is an extraordinary phenomenon if one considers the amount of computation that is required to generate an autocorrelation mathematically. The definition of a cross correlation function of two signals $x(t)$ and $y(t)$ is:

$$R_{x,y}(t) = \int_{-\infty}^{+\infty} x(t')\, y(t'+t), dt' \qquad 2\text{-}15$$

An autocorrelation is simply the cross correlation of a signal with itself, denoted by $R_{x,x}(t)$. For the results in Figure 2-3, discrete signals were used so that the ISIH could be generated from a simulation. The cross correlation function for a discrete signal is

$$R_{(x,y)n} = \sum_{m=0}^{\infty} x_m y_{n+m} \qquad 2\text{-}16$$

Similarly, the autocorrelation of a discrete signal is defined by $R_{(x,x)n}$. Tapson (1998) also demonstrates that the autocorrelation properties in the ISIH for a neuron holds for a digital signal. Although this is a highly non-biological scenario, it does present a novel idea for performing a valuable task in digital systems.

Computing the cross correlation function is a computationally intensive task. Since the autocorrelation of a signal is apparent in the ISIH of a single neuron, one raises the question: is it possible to use an analogue neuron or neural network to practically compute a cross correlation for a digital system? Furthermore, would this approach be more efficient and less computationally intensive?

2.3.2 Digital spreading codes in communication systems

Before considering further correlation properties of a neuron, the author will introduce a set of digital spreading codes which will be used through out this thesis for stimulating the neurons. These codes are now almost ubiquitously used in spread spectrum digital communication systems. A significant hurdle in these systems is overcoming the high power consumption consumed by performing cross correlations to decode the transmitted data.

The value of digital spreading systems can be understood by considering the problem in which multiple data streams need to be sent simultaneously over the same medium. To solve this problem, spread spectrum techniques can be used which separate each data stream onto its own frequency band or channel. This method is often unfavourable when using wireless communications, since it means that in order to receive all the data streams, one must tune into and listen on each channel (Verdu, 2003). This requires a considerable amount of hardware (or processing in the case of software radio) which in turn increases cost, complexity and the power consumption of a wireless system. Another type of spreading technique is time division multiple access (TDMA). This is a multiplexing technique such as that recently used in neural implants by Harrison *et al.* (2007). The significant disadvantages of this spreading technique are that it prevents true asynchronous communication and provides little protection against jamming (Glisic and Vucetic 1997).

A modern solution is to use a logical pseudorandom number to encode each bit so that many bits can be sent simultaneously. Pseudorandom number (PRN) codes enable several 'virtual' asynchronous data channels to be introduced onto a single communication medium. This is a particular scheme of Code Division Multiple Access (CDMA) that is known as a Direct Sequenced Spread Spectrum (DSSS) system and is discussed in many texts such as *Spread Spectrum CDMA Systems For Wireless Communications* by Glisic and Vucetic (1997). It is used frequently in wireless systems

University of Cape Town

since it has good antijamming capabilities, and allows a limited number of data bits to be sent on a particular carrier frequency at the same time. Each bit can be sent asynchronously and independently of the other bits. The DSSS spreading technique relies on PRN codes which are an ordered stream of binary ones (+1) and zeros (-1) referred to as *chips*. A chip should not be confused with a bit. A *bit* is the element in an information carrying stream. In a DSSS system, each bit consists of many *chips* which make up the full length of the PRN code. Pseudorandom number codes are appropriately prefixed 'pseudo' since they are actually deterministic and can be generated by using a configuration of multi-bit shift registers. These are known as linear feedback shift registers (LFSR). In a wireless network consisting of several transmitters, the LFSR can be carefully configured for each transmitter so as to generate a particular PRN code from a special set which have distinguishable properties.

The principle of encoding data with a PRN code can be more easily understood by first considering a set of PRN codes that are of a suitably large length. Each code of the set is made up of a randomly generated binary sequence and is guaranteed to be very different from the other codes such that their binary sequences are uncorrelated. If each of the PRN codes is assigned to an individual transmitter node, then every transmitting node can each independently and asynchronously send a data bit on the communication medium by sending each data bit as the PRN code. Note that an inverted data bit is represented by the PRN code with each of its chips (individual logic states) inverted.

To a receiver, all data on the DSSS system appears to be noise. However, if the individual PRN codes are known, then they can be generated internally by the receiver and cross correlated with the received signal to find the various data bits. This phenomenon can be more easily understood if one considers the cross correlation properties of a noise signal which has a mean of zero. Using equation 2-15 to cross correlate two independent white noise signals yields:

$$\mathrm{R}_{\eta 1,\eta 2}(t) = \int_{-\infty}^{+\infty} \eta_1(t+t')\eta_2(t'), dt' \qquad \text{2-17}$$

For noise signals that are independent and therefore uncorrelated, by the Central Limit Theorem, $\mathrm{R}_{\eta_1,\eta_2}(t)$ evaluates to zero for all t. If instead, one considers the cross correlation of η_1 with the signal ($\eta_1 + \eta_2$), then $\mathrm{R}_{\eta_1,(\eta_1+\eta_1)}(t)$ evaluates to:

$$\mathrm{R}_{\eta 1,(\eta 1+\eta 2)}(t) = \mathrm{R}_{\eta 1,\eta 1}(t) + \mathrm{R}_{\eta 1,\eta 2}(t) \qquad \text{2-18}$$

$\mathrm{R}_{\eta_1,\eta_1}(t)$ is the autocorrelation of η_1 and evaluates to the total energy in the signal for t=0. Since the noise is independent of time, it is uncorrelated with all other phase shifts of itself. Thus once again by The Central Limit Theorem, $\mathrm{R}_{\eta_1,\eta_1}(t)$=0 for $t \neq 0$.

University of Cape Town

The result of 2-18 is 0 for all t except for t=0. A similar property holds if digital PRN codes are used, except that the spike at R(0) is finite, and for $t \neq 0$, R(t) is non-zero, but generally of a much smaller magnitude than R(0).

In this dissertation, we will be dealing with a particular set of finite length PRN codes called Gold codes. These codes have a length of 1023 chips and are often used because their autocorrelation function is guaranteed to have a distinct peak at t=0 and a much smaller value for all other t. A plot of the autocorrelation function for 3 epochs of a Gold code is shown in Figure 2-4 below.

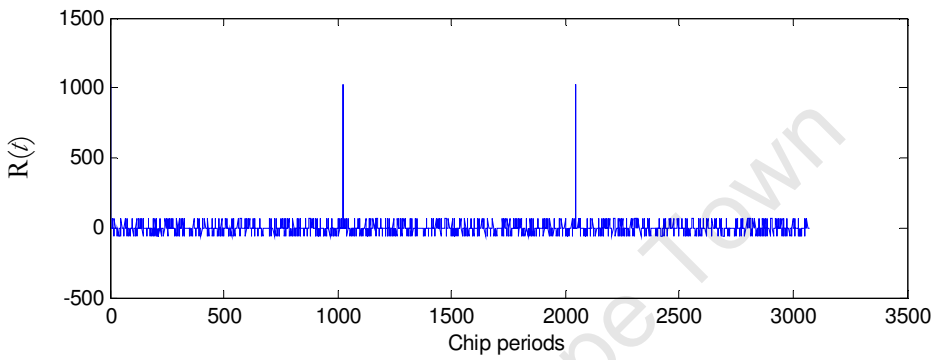


Figure 2-4: Autocorrelation function of 3 periods of a Gold code.

Another important property of Gold codes is that the cross correlation function between different codes has a small range of values with no distinct peaks as shown below.

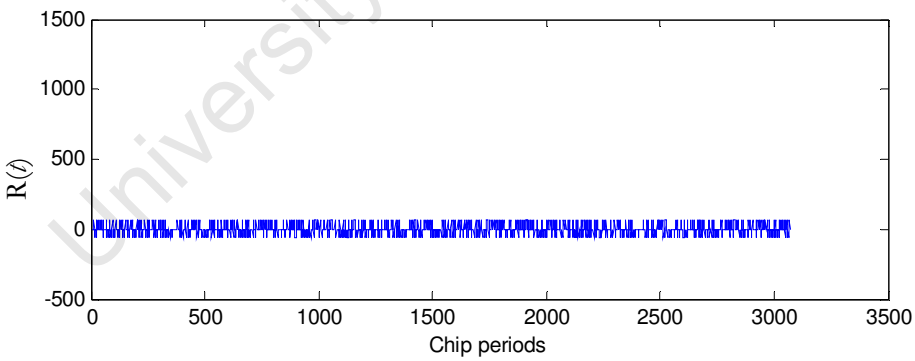


Figure 2-5: Cross correlation between two different Gold codes.

University of Cape Town

2.3.3 Cross correlation techniques for wireless communication

Cross correlating a received wireless signal is a computationally intensive task, especially when the function needs to be implemented for multiple CDMA codes. There are many techniques which have been developed for implementing the function and most of them rely on a digital front-end (Pace, 2000). When high sensitivity is required, the received signal is sampled with a high quantisation resolution and is often correlated using Fast Fourier Transform (FFT) techniques which significantly enhance the computational efficiency (Psiaki, 2001).

A time domain correlation engine multiplies the received signal by an internally generated reference signal usually after the demodulation process. The received signal is then integrated until it reaches a predetermined threshold which indicates detection of a bit. This signal detection engine requires that the internally generated code be in phase with the received signal. For asynchronous spreading codes, multiple engines are required with a reference code generated for each possible phase shift.

This is known as a RAKE architecture and was proposed many years ago (Price and Green, 1958). A novel method for cross correlating CDMA codes transmitted at a high data rate using impulse radio technology was recently presented (Sverre Lande and Hjortland, 2007). This method is an improvement on the RAKE detector in that instead of clocking the codes at different phases, it uses the propagation delays from inverted logic to create the different phase delays for the multiplication. Where a traditional RAKE detector would rely on multiplications with different phases of the reference signal, this approach multiplies the same reference code with different phases of the received signal and in essence is just a digitally based cross correlation engine.

2.3.4 Neural cross correlation of Gold codes

Recently, in a paper presented by Tapson and Etienne-Cummings (2007), the task of performing a cross correlation using IF-neurons was demonstrated with an electronic circuit. The circuit accepts an 'unknown' signal, $y(t)$, which is the received signal containing a mixture of Gold codes at different phases. It also accepts the generated reference Gold code, $x(t)$, with which to perform the cross correlation. Essentially, the cross correlation is achieved by compiling the ISIH for an IF neuron (as described in section 2.3.1) with input $y(t)+x(t)$ and subtracting a suitably scaled ISIH for an IF neuron with input $x(t)$.

The underlying principle for using this method lies in the basic properties of the correlation function. The autocorrelation of $y(t)+x(t)$ can be expressed by:

$$\mathrm{R}_{(y+x),(y+x)} = \mathrm{R}_{yy} + \mathrm{R}_{y,x} + \mathrm{R}_{x,y} + \mathrm{R}_{x,x} \qquad 2\text{-}19$$

University of Cape Town

Since $y(t)$ contains only periodic Gold codes and a little noise, its autocorrelation is a closely uniform signal with a spike that occurs at t=0 and again every 1023 chip periods. This is also true for $x(t)$ and thus the autocorrelation is easy to extract, leaving only the cross correlation function, $R_{y,x}(t)$ and its mirror image, $R_{x,y}(t)$.

In the circuit presented by Tapson and Etienne-Cummings, the ISIH for the neurons corresponds reasonably well with the mathematical autocorrelation of the input signal to the neuron and thus equation 2-19 holds. Since both $R_{y,y}$ and $R_{x,x}$ have the same form, these components can be extracted from the first ISIH since they will occur at multiples of the period of the reference Gold code.

It is evident from the neural correlation circuit that the IF neuron is not only capable of performing an autocorrelation of sinusoidal stimuli, but also digital PRN codes. The question is: Is it possible to build a neural cross correlation engine that is more efficient than performing the equivalent operation digitally? Furthermore, how could this engine be used in a practical engineering application such as in a GPS receiver or for a wireless AER protocol in neural prostheses?

University of Cape Town

CHAPTER 3: ANALYSIS OF THE NEURAL CROSS CORRELATION ENGINE

3.1 A neural autocorrelation engine

3.1.1 Overview

Past work on the autocorrelation properties of a single IF neuron was reviewed in section 2.3. There was however, no literature which analyses this process. The ISIH from an IF-neuron stimulated by a Gold code will be investigated by running a simulation in MATLAB. The aim is to observe the relationship between the ISIH for an IF-neuron stimulated by the Gold code and the autocorrelation function of the Gold code itself. The performance of this system will also be investigated with regard to the number of spikes that are required to generate a useful ISIH.

In addition, the expected form of the ISIH will be expressed in terms of the probability density of the time-to-first-spike and the probability density of the PSTH. These methods of neural spike coding were discussed in section 2.2.1. The histograms compiled from the spike trains of the simulated neuron will be used to approximate these functions. Note that in some cases, for clarity the histograms are plotted as line graphs connecting adjacent peaks of each bin of the histogram.

3.1.2 Autocorrelation of PRN codes in the ISIH of a neuron

The equation used in the MATLAB simulation of the behaviour of the IF neuron is described by equation 2-13 which is identical to the model used by Tapson (1998). For simplicity, the sample time of the system is set to the chip period of the Gold code used in the stimulated neuron. For GPS, the chip period is $T_c = 1\mu s$. Equation 2-13 is simplified for a set of constant parameters of each interval as follows:

$$u_{n+1} = u_n + drift + gain_noise \times \eta_{n+1} + gain_sig \times y_{n+1} \qquad 3\text{-}1$$

The parameters are defined as follows: *drift* represents the mean change in potential; *gain_noise* represents the deviation of the potential over that interval; and *gain_sig* is the weight of the input stimulus to the neuron. These parameters are varied depending on the experiment. In general, *drift* is in the range of 0.00025 to 0.001 $V\mu s^{-1}$ so that if noise and the input signal were ignored, the neuron would fire with intervals of 1 to 4 ms respectively. Parameters *gain_noise* and *gain_sig* must be small relative to the threshold of the neuron and are generally in the range of 0.01 to 0.04.

University of Cape Town

A simulation was performed with a Gold code as the input signal to an IF-neuron in a similar manner as presented by Tapson and Etienne-Cummings (2007). When the neuron started, its potential was set to the initial value u_0=0. When u reaches a threshold of ϑ=1 V, the neuron generates a spiking event. Every time the neuron spiked, its potential was reset to its initial value of zero. The Gold code was however continuously fed into the neuron as a periodic signal such that the code restarts every 1023 chip intervals. A first order ISIH was then compiled from the simulated IF-neuron, by compiling a histogram of intervals between successive spikes. The MATLAB function for performing the simulation and generating the ISIH is called `ncorr1` and is listed in Appendix C. The code that can be used to reproduce each of the figures that will be shown has also been provided in Appendix E.

Plots of the ISIH output for three experiments are shown in Figure 3-1. In each experiment the neuron was stimulated with a different input. These plots were compiled from approximately 67 000 spike intervals occurring over 10^5 iterations of the input signal. For 10^4 iterations of the signal, the pattern deviated only slightly for separate experiments under identical conditions. When fewer iterations (in the order of 10^3) were used the ISIH for separate experiments was inconsistent. This uncertainty in the ISIH is expected since the spike intervals are perturbed by noise which leads to a deviation in the spike interval (Cecchi et al. 2000). One can however assume that for 10^5 iterations the randomness will be smoothed and the variation of plots over separate experiments will be small relative to the structure of the ISIH.

Plot A in Figure 3-1 shows an ISIH derived from the spike train of the simulated IF-neuron with no input signal. The *drift* parameter for the neuron was 1/1500 Vs^{-1} and the *gain_noise* parameter was 0.01. Since the input to the neuron is stationary, the probability density, $f_\vartheta(t)$ in equation 2-6, predicted by Gerstein and Mandelbrot (1964) can be applied. From equation 2-9, we can solve for μ and σ as follows:

$$\mu = \frac{drift}{T_c} \qquad\qquad \sigma = \frac{gain_noise}{\sqrt{T_c}}$$

Substituting the respective parameters yields μ= 666.67 Vs^{-1} and σ=10 $Vs^{-½}$. Using these values and a threshold of ϑ=1 V, a discrete version of the curve $f_\vartheta(t)$ was calculated on the time frame of seconds. It was then multiplied by T_c to generate the PMF and then scaled up by the total number of spikes used to generate the ISIH. The final curve is drawn in green in Plot A and shows that the ISIH has the expected shape.

University of Cape Town

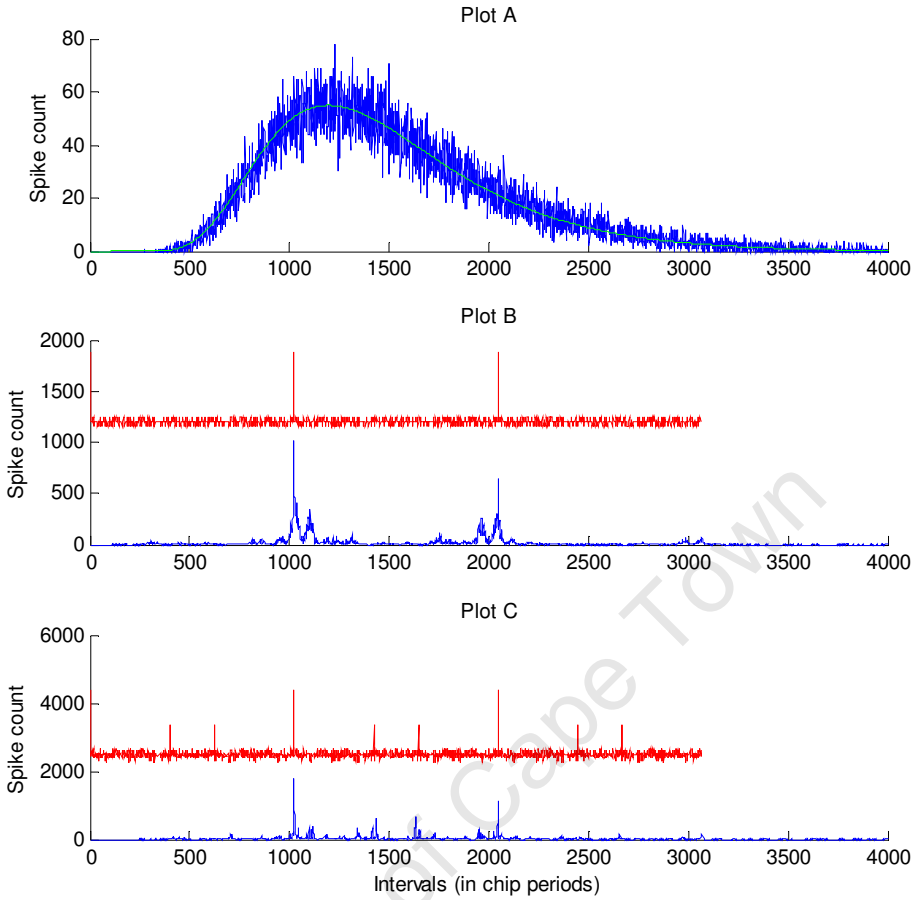


Figure 3-1: ISIH's for an IF neuron with various input signals. Plot A is an ISIH compiled from the IF neuron with no input signal. The blue curve in plot B is an ISIH compiled from the IF neuron with a Gold code as the input signal. The blue curve in plot C is an ISIH compiled from an IF neuron with the input signal being a mixture of the Gold code and a 400 bit phase shift of itself. In both plot B and C, the red curve is the actual autocorrelation function of the input signal which has been vertically offset.

University of Cape Town

In Plot B of Figure 3-1, the blue curve shows the ISIH for the neuron, with a pseudorandom number (PRN) as the input signal. The PRN signal was a Gold code with logic '1' represented by 1 and logic '0' represented by -1. The signal was first normalised to ensure that its power can be compared for various experiments. It was then multiplied by *gain_sig* of 0.03 so that the stimulus to the neuron is small relative to the firing threshold. The curve in red above it is the mathematical cross correlation and shows that the characteristic autocorrelation spikes line up with the peak of the ISIH.

The blue curve in Plot C shows the ISIH response for a Gold code input signal added to a 400 bit phase shift of itself. As in B, the red curve above it shows the mathematical

autocorrelation of the input. If we call the two signals x_0 and x_{400} then the autocorrelation of their sum can be expressed as:

$$R_{(x_o+x_{400})(x_o+x_{400})} = R_{x_0x_0} + R_{x_{400}x_0} + R_{x_0x_{400}} + R_{x_{400}x_{400}} \qquad 3\text{-}2$$

All the components of the autocorrelation line up with peaks in the ISIH. These peaks are also proportional in height which means that not only phase, but also magnitude can be correctly interpreted from the ISIH. Furthermore, as demonstrated by Tapson and Etienne-Cummings (2007) the ISIH can be used to correctly detect phase shift by subtracting the autocorrelation.

This method is a novel approach for autocorrelation of digital PRN codes, though a significant limitation is the poor accuracy which is clearly evident from the misleading spikes on the ISIH. Although the spikes are of a lower magnitude, they reduce the robustness of the engine by increasing the risk of incorrect phase detection when interference is present in the input signal. There is a clear need to improve the performance of the engine if it is to be of practical use. To do this, we need to understand the process of generating the ISIH which leads to the autocorrelation.

3.1.3 Expected density of the ISIH

An approximation to the density of the ISIH can be derived by normalising the ISIH compiled from a spike train. The normalised ISIH is a realisation of the probability density function (PDF) for a random spike interval in that spike train. This PDF for when the neuron is stimulated by a *continuous* input signal (as in the simulation shown previously) will be denoted by $\psi(\tau)$. In other words, if T is the set of all possible spike intervals in the experiment, then $\mathrm{P}\{\mathrm{T}=\tau\}=\psi(\tau)$ is the probability density function for a spike interval. The aim of this section is to express $\psi(\tau)$ in terms of other neural encoding density functions.

Consider the ideal IF-neuron modelled by the Wiener process with drift as reviewed in section 2.2.2. If a neural experiment or simulation involved a *constrained stimulus* to the neuron (i.e. input signal to the neuron has the same phase restart for each interval), then $\mathrm{P}\{\mathrm{T}=\tau\}$ is defined by $f_{\vartheta}(\tau|x,\alpha)$ where x is the input signal and α is the phase of x at the time that the neuron started. In other words, this is the first passage time density function that is also dependent on an additional input signal to the neuron.

Next consider, $\rho(t)$ defined by equation 2-1. As defined in Gerstner and Kistler (2002) and similarly in Dayan and Abbott (2005), this is the density of the PSTH. Taking the limit $\Delta t \rightarrow 0$ and $K \rightarrow \infty$, and normalising yields the probability density function *for an arbitrary neural spike* in the experiment. This PDF for the ideal IF-neuron is denoted by

University of Cape Town

$\rho^*(t)$. The two functions $\rho^*(t)$ and $f_\vartheta(\tau|x,\alpha)$ are important since they define the expected outcome of two simple spike encoding methods, namely the PSTH for unconstrained neural input and the PSTH generated from only the first spike of an experiment in which the neuron is stimulated by the signal x. It will be shown that these two functions can be used to express $\psi(\tau)$.

A particular interspike interval of a neuron with a continuous stimulus is conditioned on two events. Firstly, a spike must have occurred at some time t and secondly, it must be followed by another spike after an interval τ. This can be expressed as the joint probability density function of $\rho^*_x(t)$ and $f_\vartheta(\tau|x,\alpha)$.

$$\mathrm{P}(t,\tau) = \rho^*_x(t) f_\vartheta(\tau \mid x,t) \qquad 3\text{-}3$$

To clarify, equation 3-3 is the probability that during the entire experiment, a spike will occur at t and then the next after an interval τ (at time $t+\tau$). Note that t is relative to when the entire experiment started at t=0, $\rho^*_x(t)$ is the probability that any spike in the experiment will occur at time t and $f_\vartheta(\tau|x,t)$ is the probability that a neuron starting at t will fire at $t+\tau$ given that it was driven by signal x. The substitution for $t=\alpha$ applies if one assumes that the neuron starts immediately after it spikes. The phase of x when the neuron starts is therefore equal to the time of the last spike.

Using the fundamental property of a joint probability density function, the PDF for an interspike interval with a continuous stimulus is defined as:

$$\psi(\tau) = \int_0^{T_x} \rho^*_x(t) f_\vartheta(\tau \mid x,t).dt \qquad 3\text{-}4$$

where T_x is the time period over which the neuron is stimulated by x. This equation can be used to predict the expected density of the ISIH. A minor technicality which should be mentioned is that this definition does not apply to the first interval timed from when the experiment started to the first spike. Strictly speaking, this "first interval" is not actually a spike interval since it does not occur between two spikes.

3.1.4 Evaluating the expected density

A problem with using equation 3-4 to evaluate $\psi(\tau)$ is that the functions $\rho^*_x(t)$ and $f_\vartheta(\tau|x,\alpha)$ are non-linear and exceedingly difficult to calculate. Instead, a straightforward approach will be followed which uses ρ_j and f_i. As reviewed in 2.2.1, ρ_j can be derived from the PSTH. The PMF $f_{i|(x,\alpha)}$ can be derived by repetitively stimulating a neuron by x at phase α and compiling the PSTH for only the first spike. Equivalently one could derive it from an ISIH generated from a neuron stimulated by a signal with the same

University of Cape Town

phase restart for each interval. This has also been previously referred to as "constrained stimulation" by Plesser (1999). From equation 3-4, it follows that

$$\psi_i = \sum_j \rho_{j|x} f_{i|(x,j)} \qquad 3\text{-}5$$

The first step in using equation 3-5 to find ψ_i, involves finding $\rho_{j|x}$ from the spike train. In order to produce an accurate PMF for $\rho_{j|x}$, many repetitions of the experiment will be required. Instead, an analytic method is used which takes advantage of its periodicity. In so doing on a small part of ρ_j is created which can be scaled and duplicated to define ρ_j over the entire experiment. The `ncorr1` function in Appendix C shows the method more clearly. The Figure 3-2 below shows $\rho_{j|x}$ in blue which was derived from the same spike train used in Figure 3-1 C. The integral of the input signal is plotted in red above it and shows that there is a strong correlation between the two curves. The peaks of the spike distribution seem to emphasise those of the integrated Gold code.

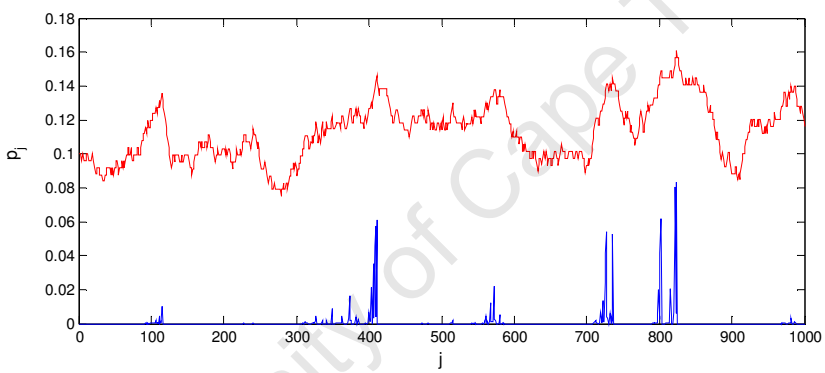


Figure 3-2: The graph of ρ_j over the first 1023 chip periods is shown in blue. The integral of one epoch of the input signal (the Gold code) is drawn in red above. It has been scaled and vertically offset for clarity.

The function $f_{i|(x,\alpha)}$ can be found by using the MATLAB function `calc_fptd` in Appendix A. This MATLAB function modifies the set of transitional probabilities in equation 2-10 in accordance with the input stimulus to the neuron. Another method for evaluating $f_{i|(x,\alpha)}$ is to compile a PSTH for the special case where only the first spike is counted, and the input stimulus to the neuron starts at phase α when the neuron starts. This is performed by the function `nttfs` which is listed in Appendix B.

Both methods discussed above are time consuming and may require several days to compute for all possible phases shifts of α. Furthermore, the aim is to find a guide for the relationship between the ISIH and the autocorrelation function. A rough approximation is made:

University of Cape Town

$$f_{i|(x,\alpha)} \approx \rho_{i+\alpha|x} f_i c \qquad 3\text{-}6$$

In equation 3-6, c is a scalar to ensure that the normalisation axiom still holds and f_i defines the probability of the time to first spike for a neuron with no input stimulus. It can be generated as described in section 2.3.1. This approximation holds with reasonable accuracy when the input signal to the neuron is a Gold code and the same neuron model as in the previous simulation is used. From equation 3-5 the density of the ISIH can be approximated by:

$$\psi_i \approx f_i c \sum_j \rho_{j|x} \rho_{i+j|x} \qquad 3\text{-}7$$

The autocorrelation function of ρ is clearly present in the summation term. If equation 3-7 holds as a good approximation for the density of the ISIH, then it will be a useful guide for optimising the neural correlation function. Shown below is the ISIH from plot C of Figure 3-1. Below the ISIH is the graph of ψ_i calculated using equation 3-7 with ρ_j derived from the same spike train as that used to compile the ISIH (as shown in Figure 3-2). Figure 3-4, shows the same graphs but from a simulation in which *gain_noise* was 0.02. Further simulations were also performed with various selections of Gold codes over a wide range of phase shifts. In all cases, the calculated density for the ISIH agreed with the measured ISIH.

University of Cape Town

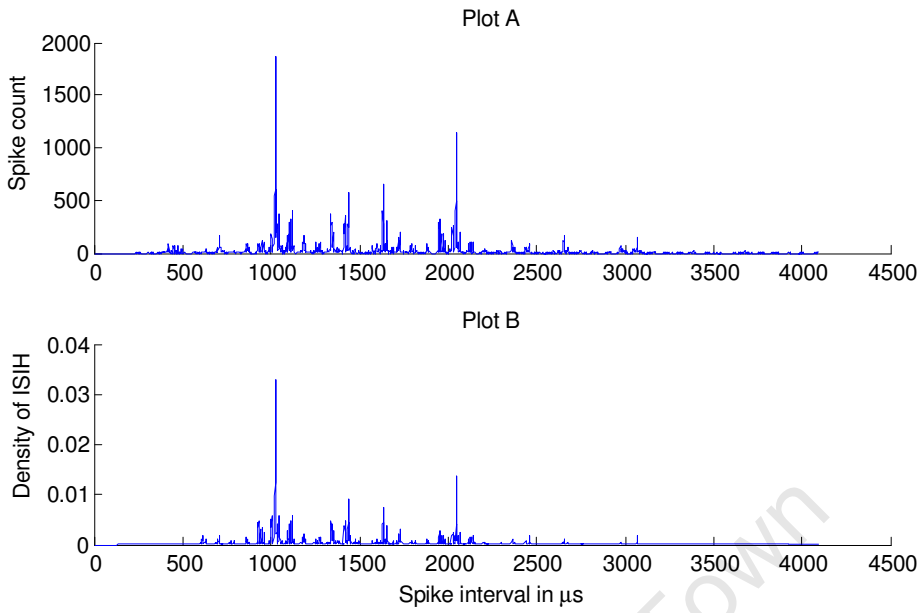


Figure 3-3: Plot A is the ISIH for an IF-neuron. The noise deviation of the sample space was 0.01 V and an input signal gain of 0.03. Plot B is the corresponding probability density function calculated using equation 3-7.

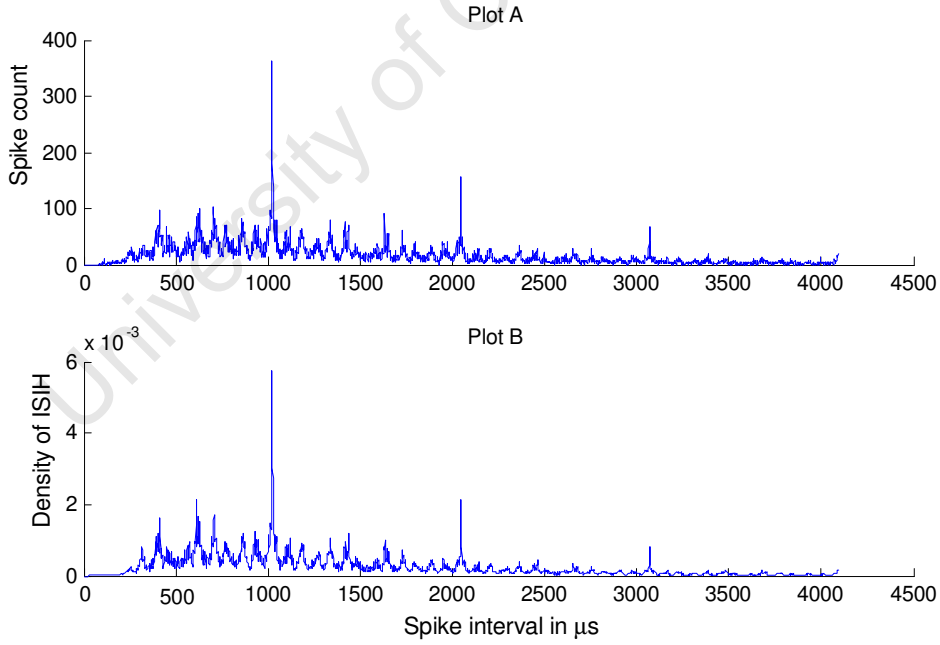


Figure 3-4: A repeat of the simulation shown in Figure 3-3, but with a sample space noise deviation of 0.02 V and an input signal gain of 0.02.

University of Cape Town

3.1.5 Discussion of results

The simulations of a single neuron verified that there is a relationship between the ISIH compiled from the spike train and the autocorrelation of a Gold code as the input signal. A further investigation found that this relationship was in fact strongly related to the autocorrelation of the spike density. The significance of these results is that it suggests one direction to follow for optimising the performance of the neural engine would be to aim at making the spike density, ρ_j, as closely related as possible to the desired signal to autocorrelate. Interestingly, the graph of ρ_j in Figure 3-2, shows that ρ_j is very different from the input signal, and is in fact is more closely related to the integral of the input signal.

Inspection of Figure 3-3 finds that the cross correlation spikes in the ISIH are less distinct than those from a simulation in which the noise in the neurons was greater and the magnitude of the signal was less. This indicates that the response is strongly dependent on the parameters of the neuron and that there must exist some optimum set of parameters which will yield the most desirable correlation. Gold codes were designed with a specific purpose. This purpose is to allow the phase of multiple Gold codes to be detected in the same signal. Furthermore, Gold code encoded bits are robust against noise and interference since they can still be detected by performing a cross correlation. Figure 3-2 shows that ρ_j is different from the input signal and that therefore the autocorrelation that occurs in the ISIH in clearly not of the input signal. This could mean that these properties of Gold codes will not apply to the autocorrelation in the ISIH

Close inspection of the ISIH's reveals that there are often local maxima that surround the desired peaks of the histogram. These smaller peaks occur consistently for simulations with the same type of input signal. They are small relative to the autocorrelation peaks of the ISIH, but less so for the peaks corresponding to the cross correlations. As is evident from Figure 3-4, these 'pseudo-peaks' could easily introduce ambiguity into the peak detection for the cross correlation.

Assuming a neuron can be implemented such that its power consumption is negligible, then in order for the neural engine to be more efficient, the computations required to generate the ISIH would have to be less than those used by a traditional numerical cross correlation engine. A digital front-end for the neural network that processes each neural spike will most likely consume the most power in the engine. A design goal for the neural correlation engine is therefore to require as few spikes as possible unless of course an alternative low-power (and perhaps non–digital) method of compiling an ISIH from neural spikes can be found.

University of Cape Town

3.2 The two-neuron cross correlation engine

3.2.1 Overview

The method discussed previously demonstrates a novel method for performing an autocorrelation on a signal. A cross correlation of two signals can be extracted from the autocorrelation under certain conditions (Tapson and Etienne-Cummings, 2007). The auto correlation function of all Gold codes has the same general form with a single narrow peak. Thus when the sum of two signals is applied to the neural autocorrelation engine, the autocorrelations can be removed from the final plot leaving behind only the cross correlation peaks. A limitation of this method is that there is ambiguity between the two cross correlation peaks that remain. The method can also be considered inefficient since much of the neuron's energy and processing time is spent generating spikes that form the autocorrelation which is discarded. This is evident in Figure 3-1 B and C in which the spike intervals at multiples of T_c clearly dominate the ISIH.

To overcome these problems, a method to cross correlate two input signals using two neurons was proposed by Tapson *et al.* (2008). The key to achieving the cross correlation in this circuit is to compile a histogram of intervals *between* the spikes of two separate IF-neurons where each is stimulated by its own input signal. Note that the resulting ISIH is different from those conventionally used in computational neuroscience, since the intervals are not between spikes of the same neuron. The diagram below in Figure 3-5 shows the configuration of the neural circuit.

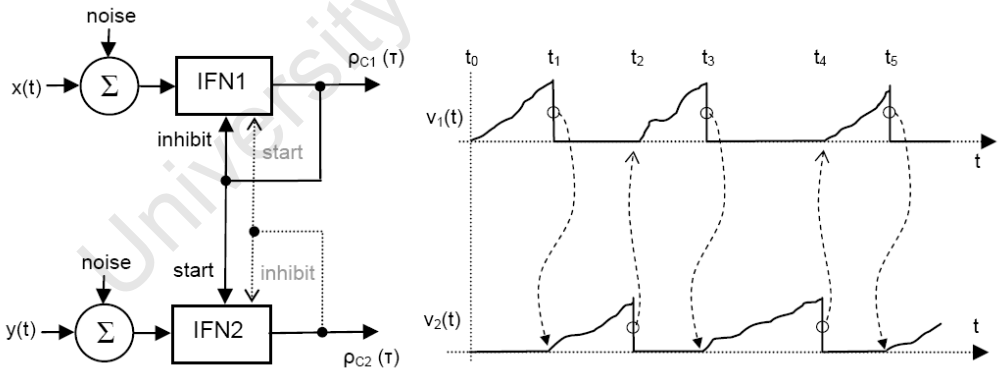


Figure 3-5: Circuit diagram of a two-neuron cross correlation engine. In this engine, the two neurons operate alternatively such that when one fires, it inhibits itself and activates the other. An ISIH is compiled seperately for each neuron. Source: Figure 1 from Tapson et al. (2008).

University of Cape Town

3.2.2 Investigation of the two-neuron cross correlation engine.

In this circuit, when one neuron fires, it inhibits itself and starts the other neuron. Thus only one neuron can operate at a time. A spike interval for IFN1 is defined by the time from when IFN2 spikes to when IFN1 spikes. In the same sense, a spike interval for IFN2 is the interval from when IFN1 spikes to when IFN2 spikes. Following a similar procedure as in sections 3.1.3, the PDF for the spike interval of IFN1 can be modelled by:

$$\psi_{IFN1}(\tau) = \int_0^{T_x} \rho_y^*(t) f_\vartheta(\tau \mid x,t).dt \qquad 3\text{-}8$$

Similarly for IFN2:

$$\psi_{IFN2}(\tau) = \int_0^{T_x} \rho_x^*(t) f_\vartheta(\tau \mid y,t).dt \qquad 3\text{-}9$$

Note that $\rho_x^*(t)$ is the PDF for the neuron IFN1 spiking at time t and $\rho_y^*(t)$ is the corresponding PDF for IFN2. To demonstrate that these equations accurately approximate the ISIH, the equivalent discrete equations for 3-8 and 3-9 were used (as similarly performed in section 3.1.4). This time however, no approximation was used and the discrete versions of the functions were tediously calculated from simulation. A simulation of the two-neuron cross correlation engine was then performed using the MATLAB function `ncorr2` which is listed in Appendix D. Plot A of Figure 3-6 shows the ISIH compiled from the spike train of IFN1. Below it in plot B is the expected density of the ISIH for IFN1 calculated using a discrete version of equation 3-8 which is defined as:

$$\psi_{IFN1} = \sum_j \rho_{j|y}^* f_{i|(x,j)} \qquad 3\text{-}10$$

The PMF for $\rho_{j|y}$ was calculated from simulation as in Figure 3-2. The function $f_{i|(x,\alpha_j)}$ was calculated using `nttfs` in Appendix B. The input stimulus to each of the neurons was a Gold code. The signal $x(t)$ represents the unknown signal and was the same Gold code as $y(t)$, but phase shifted by 200 chips. The noise gain for both the neurons was 0.03 and the signal gain was 0.01. The calculated RMS error between the normalised ISIH compiled from the spike train and the calculated PMF of the ISIH was

$$RMSE \approx \sqrt{\frac{\sum_i^N (\psi_{compiled} - \psi_{calculated})^2}{N}} = 8.44 \times 10^{-5} \qquad 3\text{-}11$$

University of Cape Town

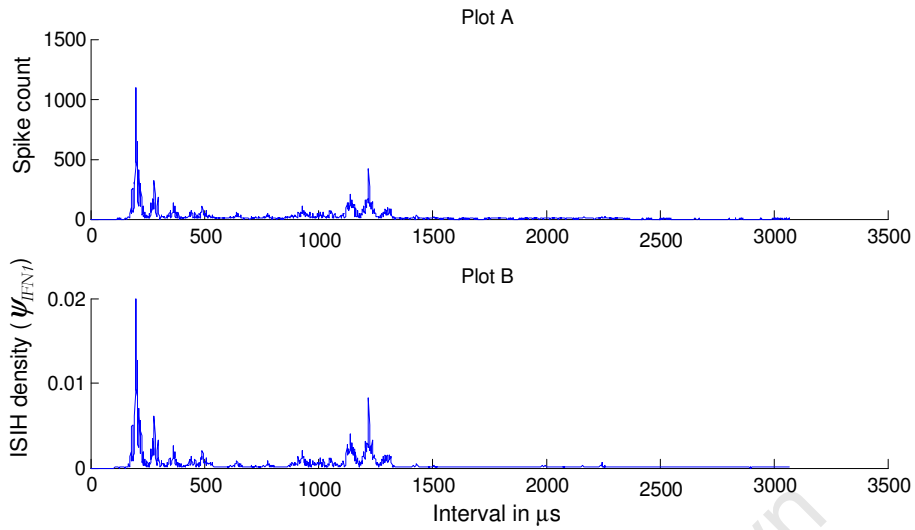


Figure 3-6: Plot A shows the ISIH for IFN1 generated from a simulation of the two-neuron cross correlation circuit. Plot B shows the expected ISIH density. The same Gold code was input to each neuron, except that it was first phase shifted by 200 chips for IFN1.

Tests with different Gold codes and various phase shifts also found a close fit between the two graphs. The approximation for $f_{i|(x,\alpha)}$ made in equation 3-6 can also be applied to this scenario. The ISIH density would thus be approximated by equation 3-12 and shows that the cross correlation is roughly present:

$$\psi_i \approx f_i c \sum_j \rho_{j|y} \rho_{i+j|x} \qquad 3\text{-}12$$

Simulation similar to that of Figure 3-6 but using 3-12, found that the approximation agreed with the density of the ISIH, but with a larger margin of error than when used with the autocorrelation engine.

An analysis of Figure 3-6 finds that the peak in the ISIH occurs at the correct phase difference of 200 bits between the two signals. A simple peak detection algorithm on the ISIH would therefore be able find the correct phase difference between $x(t)$ and $y(t)$. Measuring the phase difference between two signals is a useful capability. In a GPS receiver the phase difference between an internally generated Gold code and a received Gold code is used to calculate the distance between the transmitter and the receiver.

It is also useful to be able to detect the signal in the presence of noise or interference. An example of such an application would be in Code Division Multiple Access (CDMA) communication. In GPS, this functionality is critical since the signal-to-noise ratio is often very low and there can be as many as 6 other signals being transmitted on the same carrier frequency at any given time (Rabbany, 2006). The main objective for the neural cross correlation is therefore that it be able to detect the presence and phase of a

University of Cape Town

signal which is subject to interference. The two-neuron circuit was tested with one input signal consisting of two Gold code signals that were phase shifted and added together. The input signal to the other neuron was just the reference Gold code. The results are shown for several simulations in which the reference Gold code had a phase difference of 200 chip periods. In Figure 3-7, three ISIH's compiled from IFN1 are shown. For the simulation, each neuron had a *gain_sig* of 0.02 and *gain_noise* of 0.015. As before, the drift parameter was 1/1500 Vs^{-1}. Each simulation used a different phase shift of the interference Gold code. In these plots, 1023 bit slices of the ISIH were phase-wrapped and added together to form a plot over just the possible phase shifts of the input signal. More simply, this is just a modification of the ISIH generator to have a 1023 chip envelope. This is valid, since the phase difference between the two input signals remains constant throughout the simulation. As a result, fewer spikes are required to generate the plot.

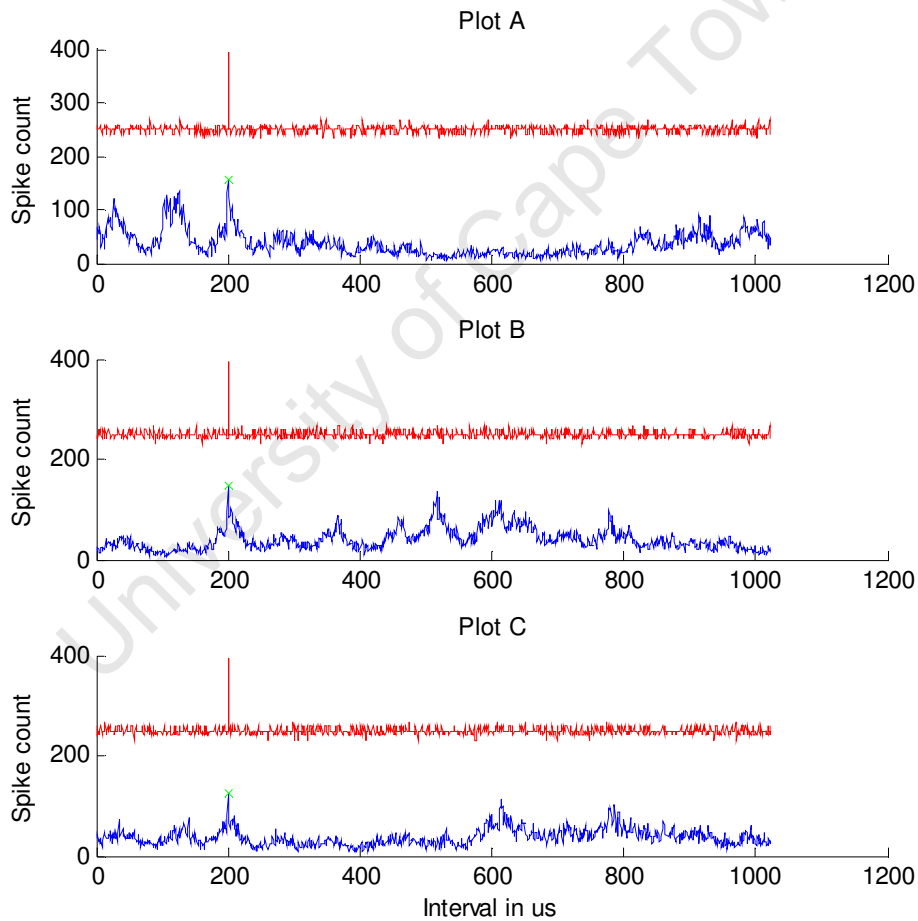


Figure 3-7: ISIH's for IFN1 of three separate simulations of the two-neuron cross correlation engine. The signal $y(t)$ was a reference Gold code and $x(t)$ was the reference Gold code phase shifted by 200 chips and added to an interference Gold code. In each simulation, a different phase shift was used for the interference Gold code.

University of Cape Town

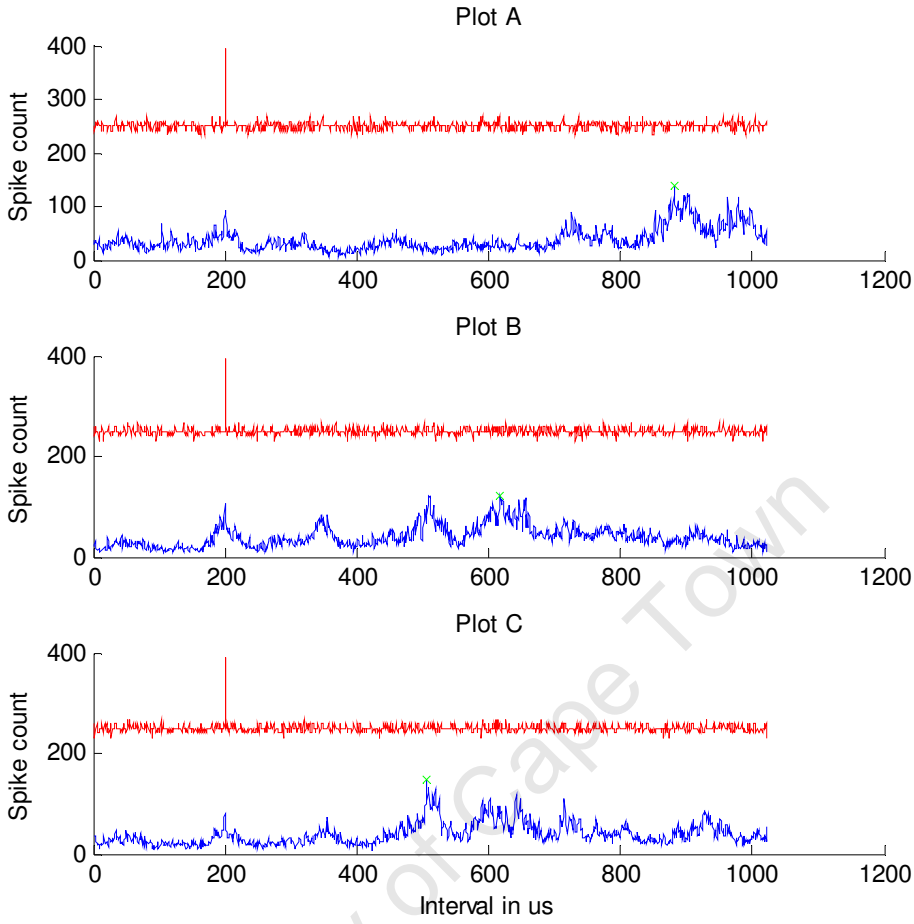


Figure 3-8: As in Figure 3-7, three ISIHs are shown. Different phase shifts of the interference Gold code were used which results in a misleading peak.

The peak of each enveloped ISIH in Figure 3-7 is indicated with a green cross which shows that they correctly corresponds to the phase shift of the reference Gold code. However, these results are not consistent for different phase shifts of the interference codes. In Figure 3-8, the same interference Gold code was used but with a different selection of phase shifts. The phase difference of the reference Gold code was still 200 chips. The peaks of the plots of the ISIHs of Figure 3-8 do not correspond to a 200 chip phase difference.

University of Cape Town

3.2.3 Discussion

Based on the test results in Figure 3-8, it is clear that the two-neuron cross correlation circuit performs poorly when there is an interference Gold code added to the signal. Equation 3-8 suggests the reason for this. It shows that the ISIH is dependent on two PDFs: that of a neural spike occurring at a particular instant in the experiment and that of neural spike interval having a particular length. Although these functions are influenced by the input signal, it was observed that they differ in form considerably and do not have the fine logical structure of the Gold codes. It is this fine structure which is fundamental for the distinctive cross correlation and autocorrelation properties of the Gold codes.

The fact that these properties do not hold for in the ISIH is further evident if one inspects the ISIH from a two-neuron cross correlation between two *different* Gold codes. These Gold codes are uncorrelated, yet the ISIH appears to indicate otherwise.

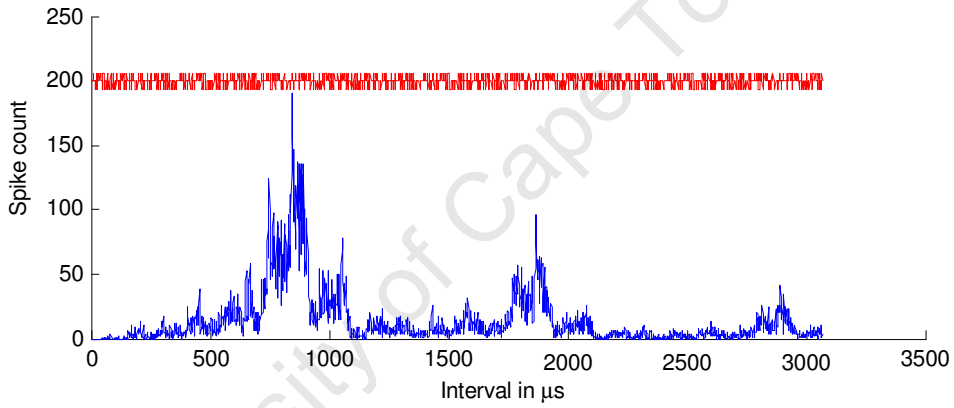


Figure 3-9: The ISIH of IFN1 from a two-neuron cross correlation between two different Gold codes. The ideal mathematical cross correlation is vertically offset and drawn in red.

A simple peak detector algorithm will be unable to determine if the two input signals were the same Gold code or not. Hence, the neural engine may only be useful in determining the phase of a signal when that signal is definitely present.

Bear in mind that instead of analysing the ISIH with a peak detector, one could consider using a more complex algorithm that analyses the pattern of the ISIH to extract signal information. This would rely on the expected ISIH being unique to each combination of gold codes. Although this may be plausible, a suitable detection algorithm is likely to be computationally intensive. A neural network may be well-suited to dealing with this problem and perhaps should be considered in a future line of research.

University of Cape Town

The rest of this dissertation will however make the assumption that a simple peak detector will be used to analyse the ISIH. The two-neuron cross correlation circuit as it stands will be inadequate for such a system if it is to be used in a CDMA communication system or a GPS receiver unit. Equipped with a broad understanding of the properties of the ISIH, this dissertation will proceed with the aim of making a neural cross correlation engine that produces more distinct cross correlation peaks in the ISIHs of neurons that are stimulated by Gold codes. In light of equation 3-8 (and 3-9), a good starting point would be to modify the neuron such that $\rho^*_x(l)$ and $f_\vartheta(\tau|x,\alpha)$ resemble the signals to cross correlate as closely as possible.

University of Cape Town

CHAPTER 4: REFINING THE NEURAL CROSS CORRELATION ENGINE

4.1 Overview

This chapter will refine the architecture of the neural cross correlation engine to make it more practical for cross correlating digital PRN spreading codes. The primary focus will be on Gold codes which are commonly used in wireless CDMA systems. The outcome of this chapter is the MATLAB class called `neural_engine` which performs a simulation of a neural processing system that can cross correlate an unknown signal with a digital reference code. This class was designed to simulate the behaviour of a neuron that is stimulated by an input signal, and coordinate its operation as part of a neural network. The `neural_engine` can be configured to run multiple neural cross correlations in parallel and therefore requires only a few iterations of an input signal to produce a meaningful result. It can be used as a real time cross correlation engine in a simple simulation for a GPS receiver. Similarly, it can also be used in a simulation of a CDMA communication system. The simulation is most useful in that it demonstrates a proof of concept and provides a platform for roughly evaluating the system's performance.

Cross correlation engines in CDMA systems aim to achieve a cross correlation as defined by the mathematical cross correlation function (see equation 2-14). The method presented here aims to manipulate the neural behaviour and configuration in favour of yielding a neural engine that performs as closely as possible to the mathematical cross correlation. In so doing, the author aims to find a 'bio-inspired computer' that is practically viable in an electrical engineering application.

The first part of this chapter will motivate an important design decision which influences how an unknown signal drives the neuron. Following this, the architecture of the neural engine will be presented. Towards the end of this chapter, the capability of the engine is demonstrated with a simulation that demonstrates how the phase of multiple Gold codes can be detected in an unknown signal. This is the task required by GPS receivers during acquisition mode. This simulation also demonstrates a starting point for an engine that may pose a solution for a wireless implementation of an AER communication protocol for neural networks. Such a protocol was first proposed by Folowosele *et al.* (2007).

University of Cape Town

The notation $R_{x1,x2}(\tau)$, will refer to the mathematical cross correlation function of the Gold code signals x_1 and x_2 and is defined by:

$$R_{x1,x2}(\tau) = \int_0^{T_0} x_1(t)x_2(t+\tau).dt \qquad 4\text{-}1$$

The derivative of x is denoted by $\dot{x}$, thus $R_{\dot{x}_1,\dot{x}_2}(\tau)$ refers to the mathematical cross correlation function of the derivative of two Gold codes, x_1 and x_2.

4.2 Dependence of the ISIH on the input signal

In Chapter 3 it was suggested that the shape of the ISIH is strongly dependent on the PSTH and the time to first spike. Thus the first step in improving the accuracy of the cross correlation should involve adjusting the form of the input signal or the behaviour of the IF-neuron such that if a PSTH or first passage time distribution were to be calculated it would correlate as strongly as possible to the signal on which to perform the correlation. The same argument holds for the time to first spike histogram.

An initial step in making the neuron more suitable for PRN codes, involved investigating the ISIH for when the input Gold code was first modulated. Frequency Shift Keying (FSK) modulation and Amplitude Modulation were used, but the resulting ISIHs showed little information. Better results were obtained when Manchester encoding was used. Manchester encoding a bit stream involves assigning two levels to each binary value. In the case of a Gold code, a chip in a logic '1' state is represented by a '1' followed by a '-1' and a chip in a logic '0' state is represented by '-1' followed by '1'. As a result, the mean of any Manchester encoded signal is zero.

This encoded Gold code was scaled and used as the input for a simulated IF-neuron in an identical configuration to the tests performed in section 3.1.2. The ISIH generated from this neuron had similar characteristics to the mathematical autocorrelation of the Gold code. The distinguishable narrow autocorrelation spike at R(0) was present and the rest of the ISIH was similar to that generated from an IF-neuron with no input. This agreed with the shape predicted by Gerstein and Mandelbrot (1964) which is described in section 2.2.2. This was notably different from the ISIH shown in Figure 3-1 for which the normal (not encoded) Gold was used. Figure 3-1 shows 'stray' spikes which do not agree with the mathematical autocorrelation of the Gold code. These were no longer present.

The nature of the ISIH generated from a neuron with a Manchester encoded digital signal can be more easily understood by examining Figure 3-2. This plot showed that the spike density is strongly depended on the *integral* of the input signal and thus by

University of Cape Town

equation 3-4, this also influences the ISIH. The integral of a Manchester encoded digital signal remains closely uniform since the integral of a bit will be the same whether the bit is 1 or 0. Consequently, the probability density of spike times will have a more uniform distribution. If we apply equation 3-4, we expect $\psi(\tau)$ to be more closely related to the mathematical definition of the autocorrelation of the Gold code. This is because the form of $\rho^*(t)$ and $f_\vartheta(\tau|x,\alpha)$ are now more closely related to the input signal.

Manchester encoding the signal before applying it to the neuron provides some insight into the effect of the integral of the input signal on the ISIH. It is however, an inconveniencing requirement for the neural engine. This is because the signal we are aiming to correlate is a Gold code which is not inherently Manchester encoded. Furthermore, the signal to be processed may be at an unknown phase or contain a large amount of interference which means that it will not adhere to the strict digital structure necessary for Manchester encoding. Theoretically a similar effect can be achieved using other types of modulation which have a mean of zero for both states of the chip. For example, Bipolar Phase Shift Key (BPSK) modulation which is the type of modulation used in some CDMA protocols including GPS.

The reset of this thesis will however use the derivative of the digital signal as the input to the neuron. This is because the resulting PSTH was found to have the best correlation with the original signal. Taking the derivative of the signal has a similar property to Manchester encoding it in that the integral of the input signal remains closely uniform since its integral is a digital signal. The integral is the original Gold code signal (provided the correct initial value is selected). Thus intuitively, the PSTH would correlate with the digital code, because the majority of the spikes would occur when the code is logic high.

The effect of using the derivative of the Gold code as the input signal to a neuron will be demonstrated with the two-neuron cross correlation engine. Some results from a simulation of the engine are shown in Figure 4-1. This simulation has the same architecture as that described in 3.2 except that in this case a different form of the input signal drives the neuron. In these experiments, two neuron input signals called $\dot{x}_1$ and $\dot{x}_2$ were used. The signal $\dot{x}_1$ was the input to neuron IFN1, and was the *derivative* of an arbitrarily selected Gold code. The signal $\dot{x}_2$ was identical, but with a phase-lead of 200 chips and was the input to IFN2. For these neurons, the standard deviation of the noise in the sample space ($\sigma T_s^{-1/2}$) was 0.01 V, and the *drift* parameter which is used to control the mean firing rate was set to $1/1500\text{Vs}^{-1}$. The gain of the input signal was 0.03 so that it is small relative to the firing threshold of the neuron which is 1 V. Figure 4-1 shows a separate ISIH generated from the spikes of each neuron over 10^6 iterations of the input signal. The mathematical cross correlation of the Gold code is also plotted and is offset

University of Cape Town

vertically above the respective ISIH. As in Chapter 3, one sample represents *one chip period* of the Gold code. The neuron's potential is calculated after each input sample of the input signal. A spike interval is quantified as the number of samples that occur between successive spikes and can therefore be expressed in chip periods.

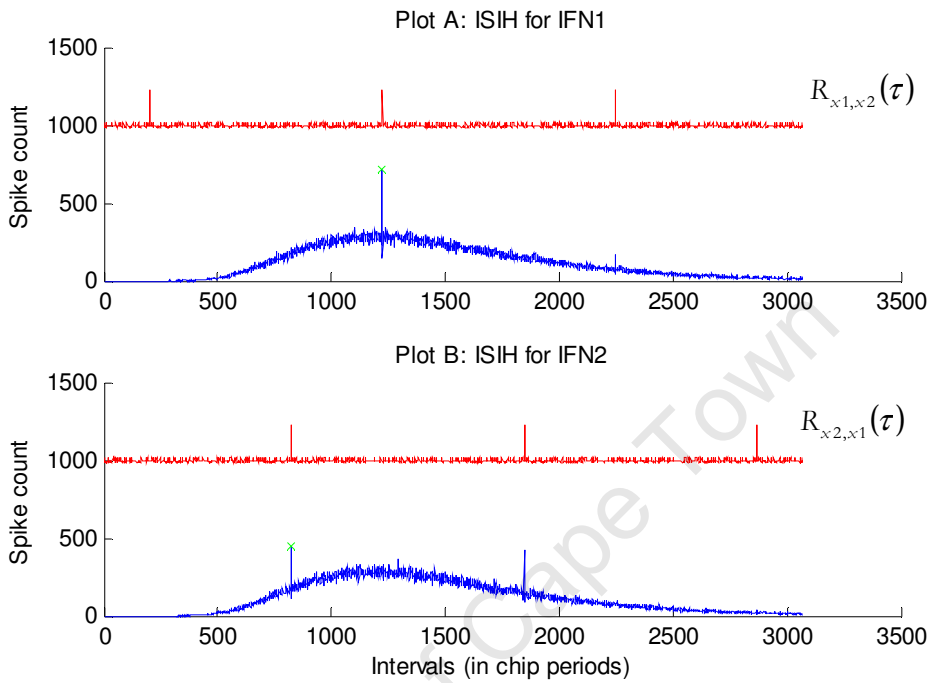


Figure 4-1: ISIH for each neuron in the two-neuron cross correlation engine. Plot A shows the ISIH for IFN1 which had the derivative of an arbitrary Gold code as the input signal. Plot B shows the ISIH for IFN2 which had an identical input signal, but with a phase lead of 200 chips. In each plot, the mathematical cross correlation of the Gold codes are shown in red and are vertically offset by 1000 for clarity.

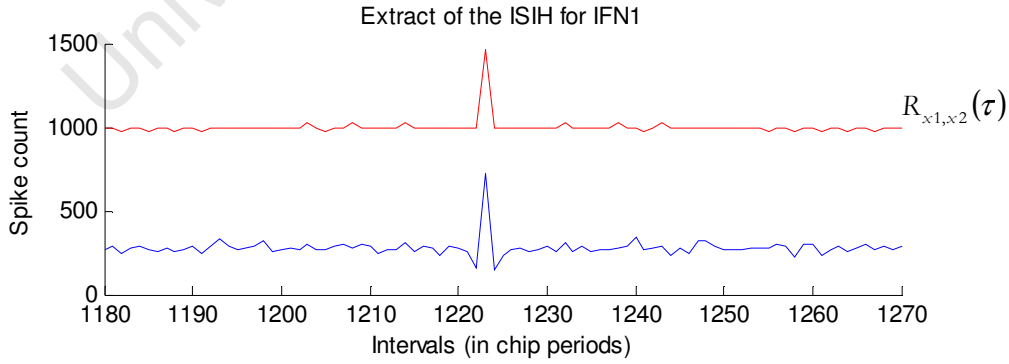


Figure 4-2: This plot shows the peak in the ISIH more clearly. The spike occurs at an interval of 1223 chips and coincides with the spike in the mathematical cross correlation which is plotted in red.

University of Cape Town

In Figure 4-1 the characteristic narrow peak in the autocorrelation function of the Gold code is clearly present in the ISIH. Figure 4-2 shows the alignment of the peak occurring at interval 1223 more clearly. Since a Gold code has a chip length of 1023, this spike correctly indicates a phase difference of 200 chips between the two periodic signals.

The ISIH generated here differs considerably from that shown in Chapter 3. Whether it will be useful or not will be investigated later. For now, it may be of interest to the reader why the ISIH generated in Figure 4-1 tends to relate more closely to the mathematical definition. We will take a look at this briefly by following on from the equation which suggests that the density of the ISIH is a function of the density of the PSTH and the density of the fist passage time.

$$\psi_{IFN1}(\tau) = \int_0^{T_x} \rho_y^*(t) f_\vartheta(\tau \mid \dot{x}, t).dt \qquad 4\text{-}2$$

To investigate, a PSTH is generated from exactly the same data that was used in Figure 4-1. It is shown for IFN1 for one period of the input signal in Plot A of Figure 4-3. Plot B shows a bar graph (in blue) of the PSTH for 100 chips. Offset above it and plotted in red is the Gold code. In green is the derivative of the Gold code which is the input signal to the neuron.

A separate experiment was used to compile Figure 4-4 which shows the time-to-first-spike for an IF-neuron with the derivative of a Gold code as the input signal. The neuron was identical to those used previously to generate the histograms for IFN1. As discussed in section 2.2.1 this histogram was created by performing many independent simulations in which the neuron fires only once and always starts from its reset state with the input signal at the same phase. In this particular example, the reset phase was zero and a total of 10^5 spikes were used. The MATLAB function `nttfs` in Appendix B was used to simulate the neuron.

From the graphs in Figure 4-3 and Figure 4-4 it appears that the probability of a neuron spiking is related to both the Gold code and the derivative of the Gold code. This can be more easily understood if we consider how the various inputs to the neuron affect its probability of firing. Recalling equation 2-13, the inputs to the neuron which are integrated are the drift term, μ; the noise term, η; and the stimulus to the neuron $h(t)$. The drift and noise are small relative to the firing threshold and influence the trend of the ISI density as modelled for no input stimulus by 2-8. Note that, the two parameters have little influence on the fine structure apparent in the above histograms compiled from 10^5 spikes. The input stimulus to the neuron is also small relative the threshold of the neuron and has a mean of zero. The stimulus is the derivative of a Gold code so that

University of Cape Town

when it is integrated, the potential of the neuron experiences a modulation which is small relative to the firing threshold. This small modulation only influences the probability of the neuron firing when its potential is near the threshold.

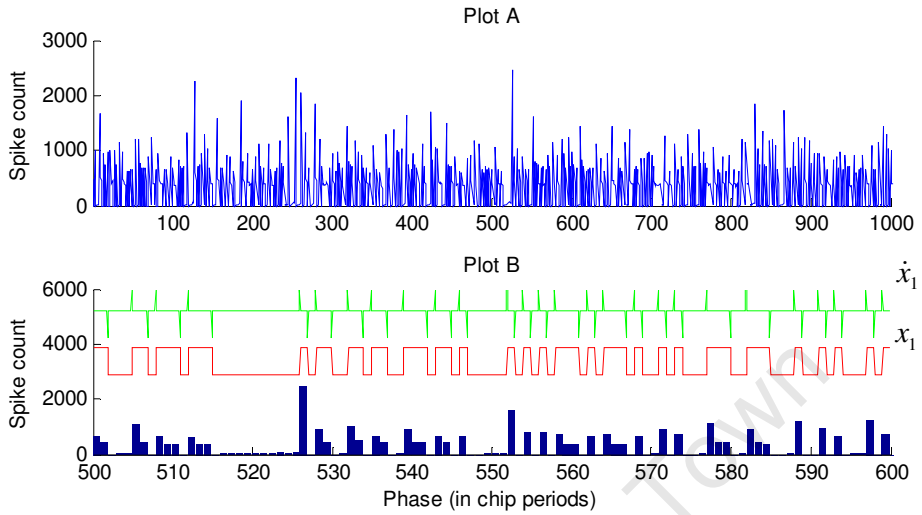


Figure 4-3: Plot A shows the PSTH over one period of a Gold code for the neuron IFN1. Plot B shows a bar graph (in blue) which is an extract of plot A. The Gold code (red) and input signal (green) are vertically offset above it.

University of Cape Town

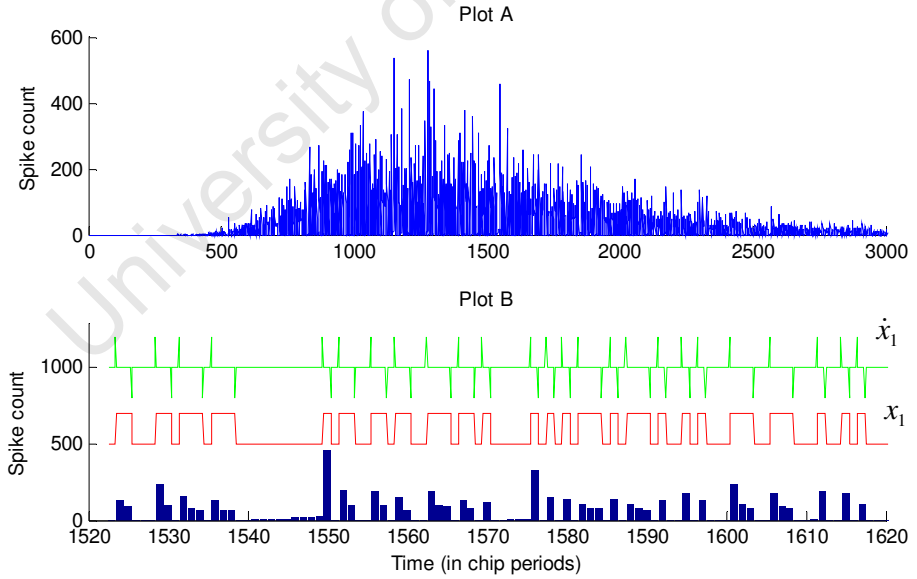


Figure 4-4: Plot A shows the time-to-first-spike density function for a neuron with the derivative of a Gold code as the input signal. The input signal was reset to a phase shift of zero every time the neurons started. In plot B, the blue bar graph is the curve in plot A over the 1520 to 1620 chip periods. Vertically offset above it is the Gold code (red) and the derivative of the Gold code (green) which is the input signal to the neuron.

The length of time for when the potential is near the firing threshold is very short relative to the length of an entire spike interval. If for example we were to assume that this period is in the order of 5 to 10 chips, then we could say that for the entire interspike interval, the neuron behaves as though its input were stationary and only during the 5-10 chip periods before it fires is the probability of the neuron firing influenced by the input signal. On the contrary when the input signal is not the derivative, but the original Gold code, then the integral of the input signal has a much larger influence on the potential of the neuron (see Figure 3-2) and thus a much larger length of the input signal influences the probability of the neuron firing.

In order for equation 3-8 to perform more closely to the mathematical definition of the cross correlation function, the PSTH of one neuron and the first-passage-time density function of the other must resemble the respective input Gold codes as closely as possible. This optimal resemblance would be possible if the dependence of the neuron firing on a particular chip of the input signal was restricted to only the current chip at which the neuron fires. This is because the probability of a spike occurring at a particular interval would be solely related to the magnitude of the input signal at that chip interval.

In a mathematical cross correlation, the scalar multiplication between the two signals means that each chip of one Gold code is multiplied with a respective chip of the other, and then the results are added up to make one sample of the cross correlation function. In the neural cross correlation, two consecutive spikes are equivalent to one particular scalar multiplication between one bit from the reference code, and one bit from the input signal. This is achieved by 'ANDing' the probability of a spike occurring at a particular chip input to one neuron, and then again at a particular chip for the other neuron. This is shown by function 3-8 and 3-9. The respective spike pairs then get accumulated to form a bin in the histogram.

Figure 4-3 and Figure 4-4 shows that each neural spike is still influenced by previous chips of the input signal that stimulate the neuron. The histogram bins in plot B of each figure show that there is a strong dependence on previous values of the Gold code signal as well as the signal's current change in state. The highest peaks align with positive edges of the derivative of the Gold code and these peaks are proportionally higher after periods of weak or negative stimuli when the Gold code is logic low. This response is expected because if the Gold code is low, then the neuron has a *small* chance of firing, and if it does not fire, the drift will cause its potential to increase towards the threshold. There is thus a higher chance of it firing at the next bit period. This means that when the Gold code eventually goes high, the neuron has a higher chance of firing since the potential is close to the threshold.

University of Cape Town

We desire that when the Gold code goes low, the probability of the neuron firing remains low, thus the drift term for the neuron must be small relative to the input stimuli so that the neuron is unlikely to reach the threshold before the Gold code goes high again. Consequently, when testing with various drift parameters of the neuron, the ISIH correlated the most strongly with the mathematical cross correlation when the drift was small. Note that this is strictly for the case in which Gold codes are being cross correlated.

In conclusion, the dependency of $\rho^*_{\dot{x}}(t)$ and $f_\vartheta(\tau|\dot{x},\alpha)$ on previous states of the input signal, weakens their resemblance to the Gold code. Using the *derivative* of a Gold code as the input signal to the neuron reduces the number of previous chips on which the neuron is dependent on for firing. The probability densities, $\rho^*_{\dot{x}}(t)$ and $f_\vartheta(\tau|\dot{x},\alpha)$, therefore relate more strongly to the respective Gold code. Unless this problem of spike dependence on previous chips can be overcome, one should expect that no matter how many neural spikes are used, the neural cross correlation will deviate from the mathematical cross correlation even when the ISIH is corrected for the non-uniform distribution for the first passage time density function (see parameter f_i in equation 3-7).

4.3 In search of a practical cross correlation engine.

It is clear from the previous section that using the *derivative* of the Gold code as the input to the neuron results in the PSTH relating more to the digital nature of the Gold code. The relation however, is far from perfect. Furthermore, it is reasonable to assume that since the behaviour of the neuron is a stochastic process, the resulting cross correlation will always be inferior in accuracy to that of a deterministic cross correlation based on the mathematical definition of the function. There is thus a trade-off, in that by using a neural correlation engine which may have other advantages (i.e. low power), there is a distinct disadvantage in that it may have less functionality.

In the case of a correlation engine for a GPS receiver, the tradeoffs of the neural cross correlation engine may be of little concern. Gold codes are designed so that many bits of data can be transmitted on a data channel at one time. The GPS service network has nominally 24 operational satellites transmitting Gold codes (Kaplan and Hegarty, 2006). However, the satellites are in individually controlled orbits around the earth so that at any given moment a GPS receiver on earth can receive a signal from at least four satellites and at most six. This means that a correlation engine need only be able to distinguish a particular Gold code from a signal consisting of at most 5 other Gold codes.

University of Cape Town

A simple experimental evaluation will be presented to show that the two-neuron cross correlation engine is indeed suitable for a GPS receiver. Using the derivative of a Gold code as the input signal means that $\rho^*_{\dot{x}}(t)$ and $f_{\vartheta}(\tau|\dot{x},\alpha)$ are now more closely related to the respective input signal and one can expect the correlation properties of Gold codes to hold. Figure 4-1 shows that the autocorrelation property holds. To test the cross correlation property, we need to investigate the ISIHs from the two-neuron cross correlation engine when a derivative of a *different* Gold is used as the input signal to the other neuron. This is shown in Figure 4-5 from which it is clear that there are no consistent peaks. This holds true for experiments with other Gold codes tested at various phase shifts and with artificially generated white noise which resulted in a similar ISIH to that shown below.

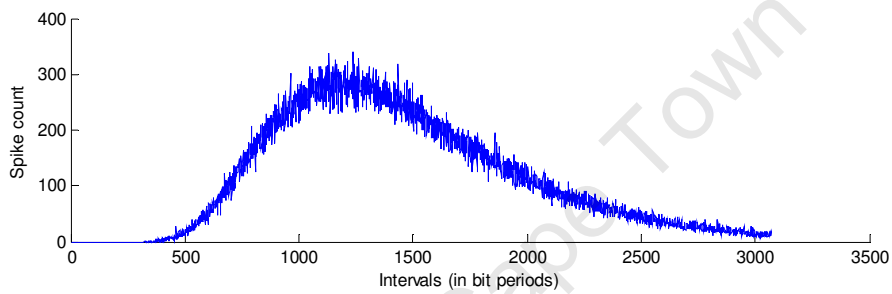


Figure 4-5: ISIH for IFN1 with the derivatives of different Gold codes as the input signals for each neuron.There are no distinct spikes which could be falsely interpretted as the presence of a signal.

To summarise the results shown up until now, it can be said that using the derivative of the input signal may result in a useful neural cross correlation engine for Gold codes, because:

- The neural cross correlation between a Gold code and a phase shifted version of itself yields a peak in the ISIH that corresponds to the phase difference.
- The neural cross correlations between different codes, and noise, yields no distinct peaks in the ISIH that may be misinterpreted as phase shifts of the same code.

A final experiment suggests that the neural correlation engine will be capable of processing a signal for a GPS receiver. It involves a very simplified simulation of a demodulated GPS signal which consists of 6 Gold codes at arbitrary phases that were added to white Gaussian noise. The derivative of this signal was then used to drive IFN1. The signal for IFN2 was the derivative of the *reference* Gold code, which was one of the Gold codes in the simulated GPS signal that was phase shifted by 200 chips. The simulation of the correlation engine was run for 10^5 iterations of the input signal. The results are shown in Figure 4-6 below.

University of Cape Town

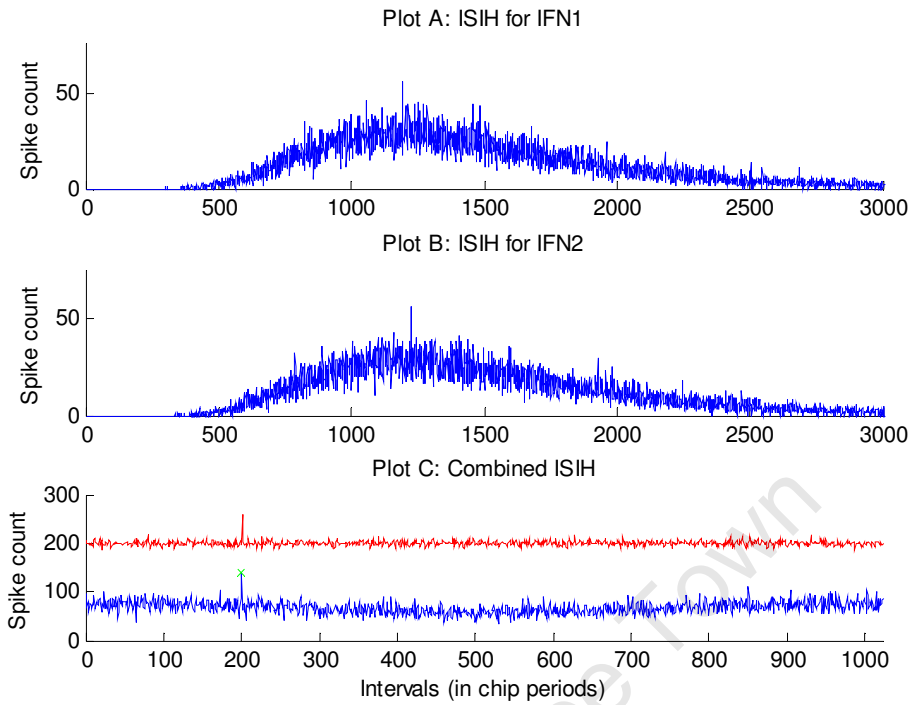


Figure 4-6: ISIH generated from neurons of the two-neuron cross correlation engine. Plot A shows the ISIH from IFN1 which had the derivative of a simulated GPS signal as its input. Plot B shows the ISIH for IFN2 which had the derivative of the reference gold code as its input signal. In Plot C the curve in blue was compiled from the data in Plot A and B by reversing A, adding it two B and then enveloping the final curve at intervals of 1023 chip periods. The curve in red is the offset mathematical cross correlation between the simulated GPS signal and the reference Gold code.

University of Cape Town

Plots A and B show the ISIHs for IFN1 and IFN2 which appear to consist mainly of noise. Running the simulation over many more periods of the input signal would provide more data to improve the signal-to-noise ratio and thereby show the cross correlation peaks. This is however unnecessary if the two ISIHs are combined by first reversing one adding it to the other and then enveloping the result at 1023 chip periods. In the simulation, the effect of Doppler shift on the GPS signal was not modelled. Thus both the input signals have an equal period length, but unequal phase shift, meaning that successive cross correlation peaks occur at multiples of 1023 chip periods. Furthermore, if $\alpha+n1023$ (for $n \in \mathrm{N}$) represents the intervals at which the cross correlation peaks occur in Plot A, then $n1023-\alpha$ represents the intervals at which the peaks occur in Plot B. This is why we can combine all the ISIH data into one plot over the range of 1 to 1023 as shown in Plot C. This plot shows that more of the noise has been smoothed and that a peak detector could easily detect the correct phase difference of the reference Gold code.

The simulation just described, was repeated with an arbitrary selection of different phase shifts and different Gold codes. An important finding from these tests was that in each case the peak was clearly identified and accurately represented the correct phase shift of the Gold code in the simulated GPS signal. For tests in which the reference code was not present in the simulated GPS signal, the combined ISIH showed no distinct peaks.

The results of the above experiment are promising and indicate that a practical implementation of the above architecture may indeed be capable of adequately cross correlating real GPS data. There is however still one problem that makes the neural engine as described impractical. The simulation to generate Figure 4-6 required 10^5 iterations of the input signal. At a bit rate of 1.023MBits/s, a Gold code is transmitted every 1ms, so it would take 100 seconds to generate the ISIH for a single reference Gold code during acquisition mode using one pair of neurons. The usual time over which to cross correlate a received GPS signal is 5-20ms depending on the position of the respective satellites and the requirements of the GPS receiver (Kaplan and Hegarty, 2006). This is because there is a 50bit/s data stream which is combined with the Gold code using a logical XOR function. This means that every 20ms the Gold code stream may or may not be inverted. The GPS service satellites are also continuously orbiting the earth which means that the phase of the transmitted Gold codes is continuously changing.

These problems can be over come by using other broadcasts which can help the receiver to predict the dynamics in the satellite signal. A straight forward solution however, is to run multiple cross correlation engines in parallel over a shorter period of time, and then combine the resulting ISIHs. It was discussed earlier in section 4.2 that when the input signal to the neuron is the derivative of a Gold code, then the PSTH and time-to-first-spike density function are dependent on the input signal for only a very short period of the interspike interval. Thus while the neuron is operating, little or no processing is performed on the greater part of the input signal. Having many neurons operating independently, but in parallel will process more of the input signal provided the spiking of the neurons is evenly spread out. This will ensure that more information is extracted from the received signal.

Another impediment for a GPS receiver is that the correlation engine is also required to cross correlate at least 4 different Gold codes at the same time, so that the position can be calculated. The most obvious method of addressing this problem is to have multiple neural correlation engines running in parallel with different reference codes. Equation 3.8 shows an important property that can be used to derive a far more efficient method. The equation shows that the cross correlation occurs between the probability density of

University of Cape Town

the PSTH and the time-to-first-spike density. The same PSTH from the neuron with the unknown signal could be used for each correlation as follows:

$$\psi(\tau) = \int_0^{T_{\dot{x}}} \rho_{\dot{y}}^*(t) f_{\vartheta}(\tau \mid \dot{x}_i, t).dt \qquad 4\text{-}3$$

where $\dot{x}_i$ indicates the Gold code for which the ISIH is to be generated.

This equation could be implemented as a neural correlation engine using the architecture shown in Figure 4-7. Start with one stochastic IF-neuron that is driven by the derivative of an unknown signal which may or may not contain several Gold codes which are each at some arbitrary phase shift (i.e. a simplified GPS signal). This neuron must be operated continuously, similarly to the regime described in section 3.1. For each Gold code, add a neuron which is driven by the derivative of that Gold code. These additional neurons immediately inhibit themselves when they fire, and may only activate when the neuron driven by the unknown signal fires. The ISIH compiled from each of the additional neurons would show the cross correlation function for the respective Gold code with the GPS signal.

This concept of a neural correlation engine will not be demonstrated. It was mentioned only because is elaborates on the idea that the cross correlation is performed by compiling the ISIH of a neuron whose *start time* is controlled by another neuron stimulated by the other input signal. The diagram of this engine attempts to present the concept more simply. This concept is important for understanding the neural correlation engine presented in the next section.

University of Cape Town

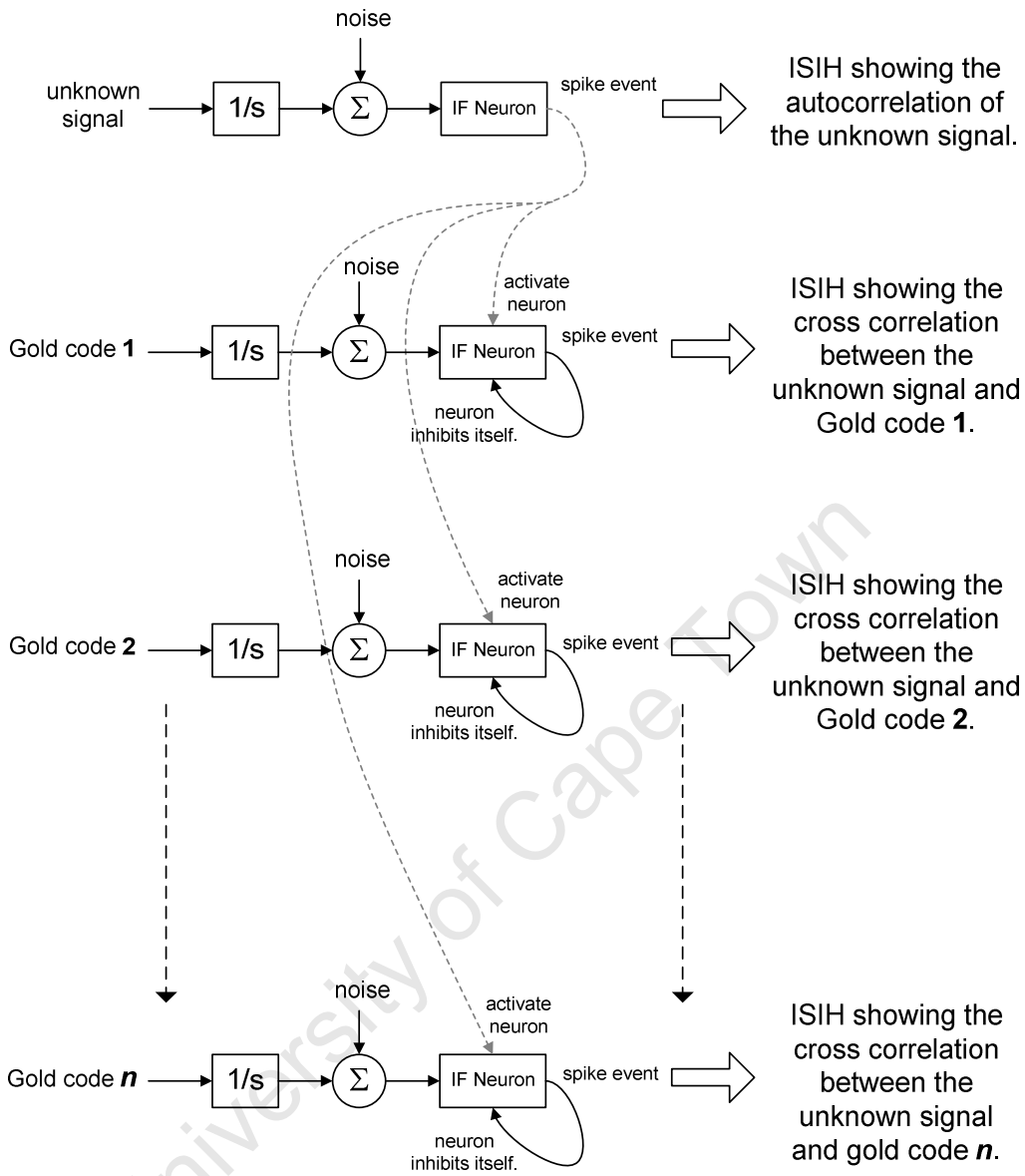


Figure 4-7: A diagram showing the architecture of a single neural cross correlation engine that can cross correlate an unknown signal with multiple signals by using only one additional neuron per signal.

University of Cape Town

4.4 The Multi-code Neural Cross Correlation Engine

The significance of a Direct Sequenced Spread Spectrum technique for CDMA wireless communication was introduced in section 2.3.2 of this dissertation. The writer now wishes to expand on this idea by considering a simple star topology wireless regime. In such a communication topology, DSSS is particularly useful because it allows the central node (or server) to communicate privately with multiple wireless nodes which are operating on the same carrier frequency, but on separate asynchronous data channels. An important capability of the server is therefore to be able to cross correlate the demodulated data stream with the DSSS code for each of the nodes that may be on the network. This would be typical of an AER wireless communication protocol.

In a similar type of network, a GPS receiver acts as the central node for the GPS satellite network. When a GPS receiver powers up from its reset state, it first has to 'study' the GPS spectrum in search of any satellite positioning signals that may be within range. This is usually achieved by sampling the received signal and then performing a computationally intensive digital cross correlation with each possible Gold code that could have been transmitted. This initial mode of operation is known as "acquisition mode" and is usually the most computationally intensive task of a stand alone GPS device. After successfully locking onto at least four positioning signals, the GPS receiver enters a "tracking mode" during which it locks onto the phase and phase drift of the transmitted Gold codes. In this mode, the GPS receiver must continue to perform a cross correlation so that it can 'keep an eye out' for other Gold codes. When the satellite goes out of range, the receiver can quickly switch to another signal and maintain a position fix.

These are only two of many scenarios in which multiple cross correlations with a single unknown signal must be performed in parallel. To address this problem in the neural correlation engine the writer will present a concept which develops on the neural architecture presented previously in Figure 4-7. The aim is to extend a single neural cross correlation engine to a regime in which an unknown received signal can be cross correlated with multiple reference signals in a manner that is more computationally efficient than using multiple two-neuron cross correlation engines. In general one can assume that a practical implementation of a computationally more efficient system will require less energy and hardware 'real estate'. These are the primary incentives pushing development of this system.

Return briefly to Figure 4-7 as the neural architecture on which to develop the proposed method for cross correlating multiple signals. In this case, one could consider the neuron to be acting as the front end of an analogue to digital converter (ADC). This is

University of Cape Town

because it takes an analogue signal and from it derives an event which may be digitally timed to generate an ISIH. Note also that the additional neurons are driven by the reference codes which are generated by the local digital circuitry and converted to analogue signals to drive the neurons. From a general perspective, it appears that these neurons are performing a redundant task in that they are converting an inherently digital code back into a digital format. A more efficient method may try to perform the same transformation without the data conversions. The proposed method relies on the assumption that generating the ISIH is a post processing task which will be assigned to a digital system.

This is demonstrated for one reference signal in Figure 4-8. To understand it requires a slight change in perspective. Consider that the PSTH generated from the known reference code will always be the same and therefore need not be generated. In fact, the probability density function of the PSTH could simply be replaced directly by the reference Gold code as follows:

$$\psi(\tau) = \int_0^{T_s} x_i(t) f_{\vartheta}(\tau \mid \dot{y}, t).dt \qquad 4\text{-}4$$

As before, the reference Gold code is represented by x_i and the unknown signal is y. This is the same equation which represents the PDF for the ISIH generated from IFN2 in the two-neuron cross correlation demonstrated in section 3.2.2 except that here the spike probability density function $\rho^*(t)$ has been substituted with the actual digital signal. A block diagram indicating how this process can be implemented is shown in Figure 4-8. The reference Gold code x_i controls both incrementing and decrementing a bin in the histogram generator. A cross correlation of two input signals is shown and requires that the reference signal must be digital. Note that this method uses a different concept to some previously presented cross correlation architectures (Sverre Lande and Hjortland, 2007; Price and Green, 1958).

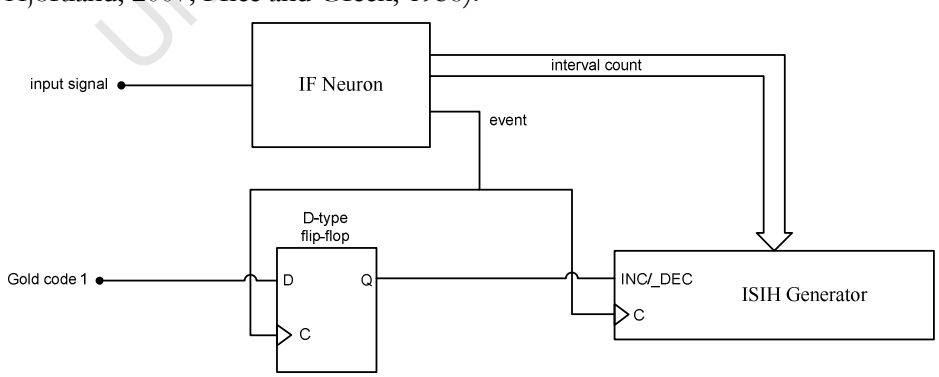


Figure 4-8: A diagram showing a neural cross correlation method that uses one D-type flip-flop instead of a neuron driven by a reference code.

University of Cape Town

In comparison to the two-neuron cross correlation engine, the IF-neuron that is driven by the Gold code is replaced by a D-type flip-flop (DFF) which sets its output (Q) to its input (D) when a positive clock edge occurs at C. The result is that every time the neuron fires the logic state of the Gold code at that instant is recorded. The next time the IF neuron fires, this logic state is used to control whether the ISIH generator increments or decrements the ISIH bin selected by the "interval count". One advantage of this method is that the mean of the resulting function would be constant which means that a peak detector would not have to compensate for the uneven trend of the ISIH. This method would require less hardware real estate. In comparison to the architecture shown in Figure 4-7, each neuron that is driven by a reference code is replaced by a single DFF. In an electronic implementation, this may be far simpler than a neuron and also more power efficient if one considers the post processing that is required to compute the interval for each neural spike. The improvements in efficiency of using this method become more significant when multiple reference codes need to be cross correlated simultaneously.

The Multi-code Neural Cross Correlation Engine (MNCCE) extends the above method to perform multiple cross correlations with a single neuron. This is achieved by adding an additional D-type flip-flop and an ISIH generator for each Gold code with which to cross correlate the received signal. An array of *n* D-type flip-flops is used where *n* corresponds to the number of Gold codes to cross correlate with the unknown signal. This 'array' of flip-flops is also known as an *n* bit synchronous register. Figure 4-9 shows a diagram of the MNCCE.

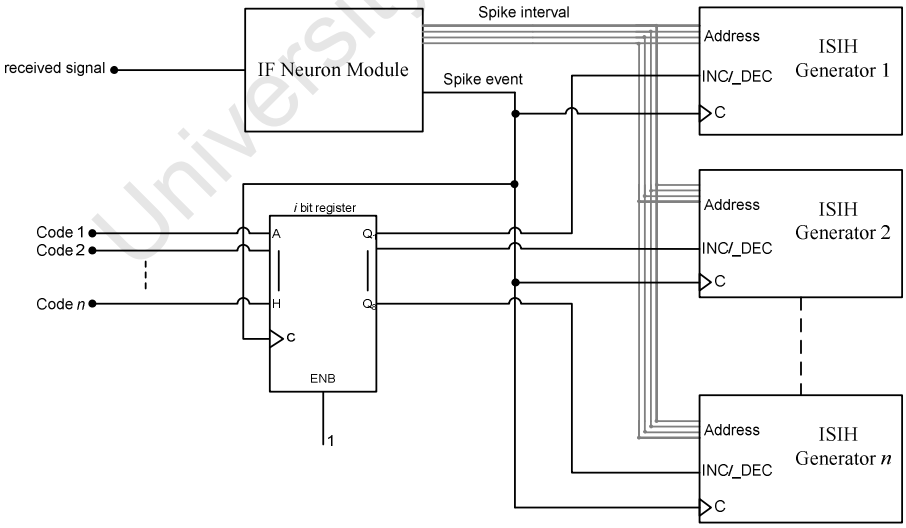


Figure 4-9: Architecture of the Multi-code Neural Cross Correlation Engine which uses a single neuron to cross correlate a received signal with many other known digital signals.

University of Cape Town

4.5 Simulation of the MNCCE

The previous section introduced the concept behind the Multi-code Neural Cross Correlation Engine. This section will conclude this chapter by presenting the method and results of simulating the MNCCE. The next chapter will report on an electronic circuit and post-processing hardware that is capable of using this method to implement a GPS receiver and a wireless AER network.

The simulation of the MNCCE was performed in MATLAB using a specifically designed class called `neural_engine` which encapsulates its functionality. Before it can be used, it must be initialised with several parameters to define its behaviour. The function "run" can then be called which accepts a vector of samples of the input signal to cross correlate, which in this case is the demodulated received signal. As shown in Figure 4-10, the function `run` returns an array of vectors where each vector is the ISIH for each Gold code that the engine was assigned to cross correlate. The index of each axes of the vector corresponds to the spike interval which is in units of 1/(*sample rate of input signal*).

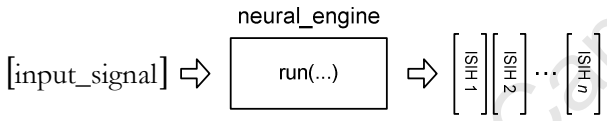


Figure 4-10: A diagram showing the data flow when using the MATLAB `neural_engine` class.

The `neural_engine` class was designed so that it could easily be configured for different modes of operation. It can implement multiple neurons so that many MNCCE can be run in parallel. The IF-neuron model is identical to that used previously, and requires that the input signal, noise and drift are small relative to the threshold. Table 4-1 shows a summary of the parameters that can be used to configure the engine.

University of Cape Town

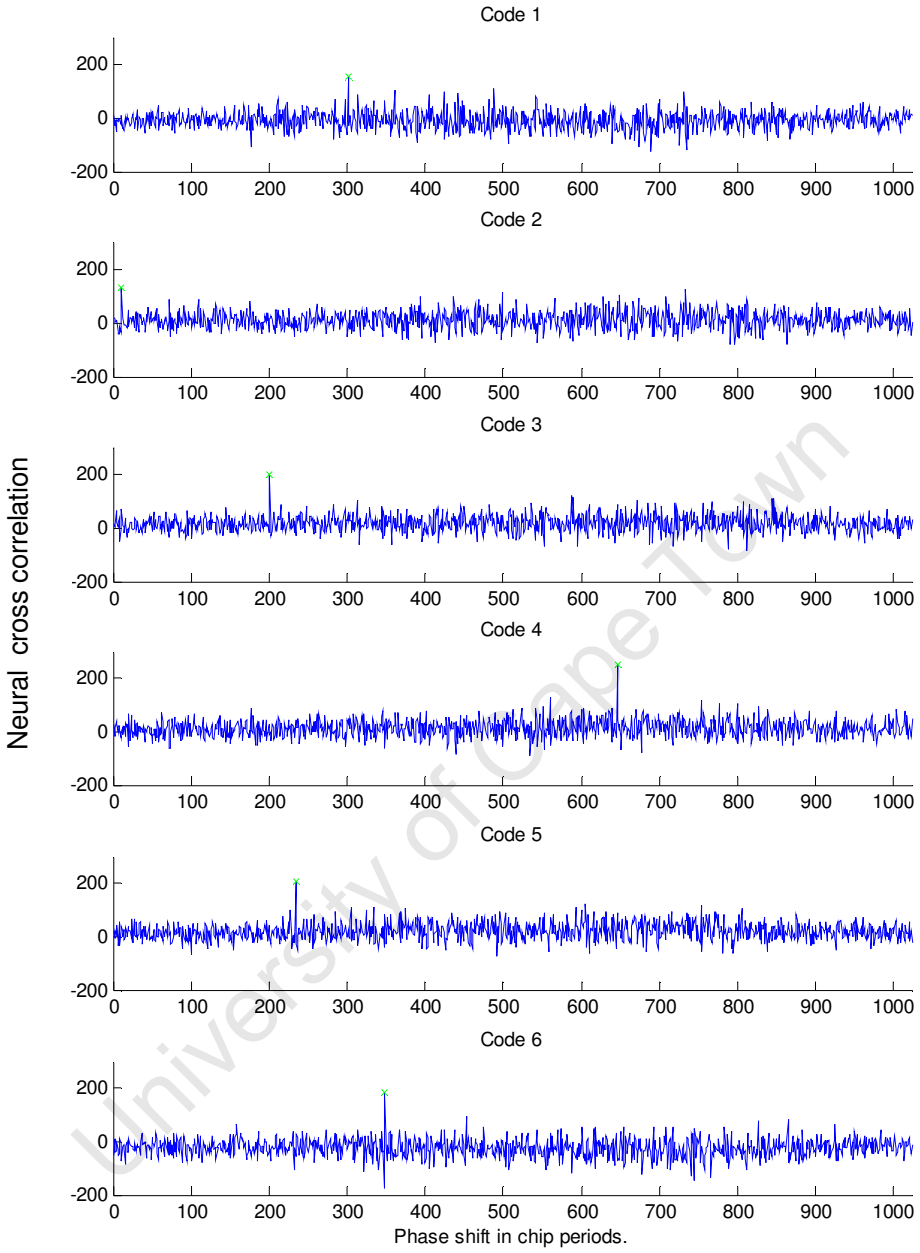
Table 4-1: Parameters for the MATLAB `neural_engine`.

Parameter	Description
numneurons	The number of neurons to run in parallel.
drift	The mean drift rate of each neuron.
mtolerance	This standard deviation of the drift term in each neuron. It can be used to model the variation in mean firing rate which may occur in a large population of physical neurons.
dnoise	The standard deviation of the noise in each neuron. It is generated as white Gaussian noise which is independent for each neuron and models diffusion noise in a neuron.
igain	The amount by which to scale the input stimuli to the neuron in order to generate an effective change in potential of the neuron.
refcodes	A matrix where each column contains a reference code to cross correlate with the input signal.

A software simulation using the `neural_engine` MATLAB class was performed. The results are shown in Figure 4-11. The code used for the simulation can be found in Appendix E and can be used to replicate the results shown here. In the simulation a population of 10000 homogenous IF neurons were used. For each neuron, the *drift* parameter was set to 1/1500 Vs^{-1}. The noise input parameter was independent for each neuron and had a standard deviation of 0.03 V. The gain of the input signal was 0.015 and its derivative was used as the input to the neuron. The input signal consisted of 6 different Gold codes labelled code 1 - 6. Each code had an arbitrarily selected phase shift in chip periods as follows:

- Code 1 – 300
- Code 2 – 10
- Code 3 – 200
- Code 4 – 645
- Code 5 – 233
- Code 6 – 347

University of Cape Town



University of Cape Town

Figure 4-11: Graphs of the output of the MNCCE for 6 reference codes which were each present in the input signal. In this simulation, 10000 neurons were run in parallel for 100 iterations of the input signal and the data from each neuron was combined to generate the graphs. The highest point in each graph is indicated with a green cross.

In the neural engine, the reference code is periodic and its frequency is constant. The resulting data array from each neural cross correlation generator can therefore be enveloped over one period of the reference code. Each graph in Figure 4-11 shows this neural cross correlation for each reference code configured to the neural engine. The peak of each graph is indicated with a green cross. In each ISIH, the peak corresponds with the offset of the respective Gold code. In a similar manner, simulations with 1000000 neurons were performed on only one epoch of the input signal. The peaks of the resulting plots also corresponded to the phase shifts of the respective Gold codes.

A considerable improvement in this neural engine over previous methods is that the neural cross correlation is not actually derived from an ISIH since the bins for each interval are both incremented and decremented. This results in the final graph having a mean of zero and a uniform trend unlike that shown in Figure 4-6. The peak of the graph can therefore easily be found at all phases without having to compensate for the uneven trend in the curve.

The results shown in Figure 4-11 indicate that the MNCCE is capable of simultaneously detecting the phase shift of 6 Gold codes in an unknown signal. Since the neural cross correlation relies on a stochastic process, the reliability of correctly detecting the phase shift is largely dependent on the number of neural spikes. In this simulation, a total of 667596 spikes were used. Although this is far more than required for the two-neuron cross correlation engine, it may prove to be more practical since multiple reference codes can be cross correlated.

In a wireless CDMA application such as an AER protocol (Folowosele *et al.*, 2007) which may involve hundreds to thousands of individually addressed nodes, the MNCCE may be a better choice of cross correlation engine. It may also be more practical in a GPS receiver where a typical problem is locking on to the satellite ranging signals. These signals which are transmitted as Gold codes are sent at a data rate which is synchronised with the satellite, but not with the receiver. The difference in the data rate results from the Doppler shift in the transmitted signal, and the clock error in the receiver since the clock is generally generated from a low-cost crystal. These problems are overcome by performing many cross correlations over a range of frequencies of each reference code. Figure 4-9 shows that each neural cross correlation can is performed independently of the others and the engine should therefore be capable of cross correlating multiple asynchronous digital codes simultaneously. In a GPS receiver, the MNCCE could therefore be used to cross correlate multiple Gold codes at multiple frequencies using the same population of neurons.

University of Cape Town

CHAPTER 5: DAAN ELECTRICAL CIRCUIT IMPLEMENTATION OF THE MNCCE

5.1 Overview

The electrical circuit presented in this chapter will demonstrate how the MNCCE can be used as a practical cross correlation engine. It acts as a proof of concept for a Very Large Scale Integration (VLSI) implementation. The foundation of the engine is an assembly of analogue neurons running in parallel. These neurons are aimed to be as simple as possible with low power consumption. Some applications may require that the cross correlation can be performed more quickly by using thousands of neurons. For this reason, the neurons should also be robust and simple with a small component count.

A high speed digital electronic interface will be needed to process the spikes from many neurons. A Field Programmable Gate Array (FPGA) is well suited for these requirements since it can be programmed to perform massively parallel operations. FPGA configuration and design techniques are also well suited for Application Specific Integrated Circuit design (ASIC). In fact, the same Hardware Description Language (HDL) code that is used to configure the FPGA can be used as part of a VLSI implementation. The FPGA platform chosen was the Altera Cyclone 1C12 Evaluation Kit with the EP1C12F324C8 FPGA. The evaluation board also has onboard SRAM and Flash memory so that a program can run on a soft-core processor programmed in to the FPGA. The soft-core processor provides a powerful platform for interfacing the PC to the neural cross correlation hardware which processes the neural spike-trains to generate the cross correlation.

The FPGA can also be configured to generate the Gold codes which are added together to form the "unknown signal" used to stimulate the neurons. This ensures that the clock for timing the neurons is synchronised with the base clock for generating the Gold codes. This is a task which otherwise requires a rigorous control and tracking system in order to cross correlate signals that may vary in frequency (Kaplan and Hegarty, 2006). Neural spike events were registered on the I/O pins of the FPGA. The spike train was processed and transmitted to a PC where the results could be plotted in MATLAB.

University of Cape Town

5.2 Analogue neuron design

An integrator can be made by charging a capacitor with a current. Adding an additional noise current creates an electrical circuit that approximates the IF-neuron. A circuit diagram is shown below which uses only one operational amplifier. The behaviour of this neuron differs slightly from the simple IF-neuron used in simulation in that it does not linearly integrate the drift current.

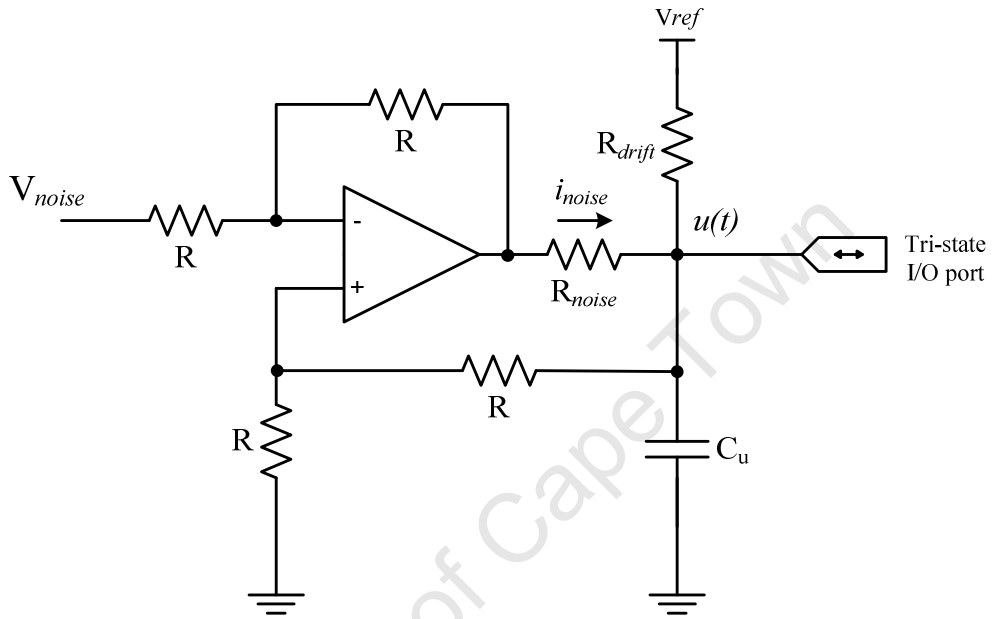


Figure 5-1: Diagram of an active analogue circuit to implement an IF-neuron using only one operational amplifier.

Figure 5-1 shows that the output of the circuit can be fed directly into a tri-state I/O port. The digital circuitry can thereby interface to the neuron using only one wire. This is significant when interfacing to a large population of neurons. The I/O pin defaults to a high impedance input mode during normal operation. When the voltage across C_1 becomes large enough for a logic high state, the I/O pin immediately latches a spike event so that it can be handled by the post processing circuitry. The I/O pin also changes to a logic low state which discharges the capacitor. It then returns to its input state after a controlled time period t_{ref}. This period is analogous to the refractory time which is characteristic of a biological neuron. In this circuit however, it ensures adequate time for the capacitor to discharge and is usually in the range of 5 to 30us depending on the type of components used in the neuron.

This circuit design was used because it shows how simply an electronic neuron can be created, the most complex and power-hungry component being the operational amplifier which drives the noise into the neuron. Ideally, for a low power VLSI

University of Cape Town

implementation only passive components for the neuron would be used. In a generation of ultra-small integrated circuits, one could use extremely small values for C_u and high values for R_{drift} and may then be able to rely on deep submicron noise inherently in the system. This noise traditionally poses a limitation in the production scale of digital electronics using submicron CMOS technology (Wang, 2003). For now however, this circuit is aimed at being a proof of concept. It can be analysed by summing the current inputs to the capacitor as follows:

$$C_u \frac{du}{dt} = \frac{v_{ref} - u(t)}{R_{drift}} + \frac{v_{noise}}{R_{noise}} - \frac{u(t)}{2R} \qquad 5\text{-}1$$

A rudimentary approximation can be made for the first passage times density for the electronic neuron by calculating the equivalent parameters in equation 2-8. The initial drift (μ) of the electronic neuron can be calculated using:

$$\mu = \frac{du}{dt}_{t=0} = \frac{v_{ref}}{C_u R_{drift}} \qquad 5\text{-}2$$

If v_{ref} = 3.3 V, then selecting C_u as 2.2nF and R_{drift} as 1MΩ, yields μ =1500 Vs^{-1} as the initial mean charge rate of the capacitor. To normalise from a firing threshold of ϑ=2V_{ref}/3= 2.2 V, the drift as used previously in the simulations would be:

$$\mu T_s = \frac{1500}{2.2} \times 10^{-6} = 681.8 V\mu s^{-1} \qquad 5\text{-}3$$

This applies since in the simulation the firing threshold was ϑ=1V and thus the result of 5-3 agrees with the *drift* parameter of 1/1500 Vs^{-1} used in the simulations. The calculation is useful only as a guide since in the electronic circuit, the capacitor charges exponentially. Furthermore, in practice, the variations of component values introduce an offset current in the noise input to the neuron. The mean firing rate consequently deviates considerably from neuron to neuron. To overcome these deviations, R_{drift} was varied slightly until the mean firing rate for the neuron was within the desired range.

A noise input voltage to the operation amplifier results in the equivalent voltage across R_{noise}= 1kΩ. For the purpose of this circuit, a signal generator was used to generate the noise. The gain at this noise source could be easily adjusted for optimal performance. In the simulation of the MNCCE, the noise was independent to each neuron when multiple neurons were run in parallel. Here however, the same noise source is used for each neuron. Fortunately, as mentioned earlier, the manufacturing tolerances in the electrical components are large enough to affect the mean firing rate of the neurons. A variation in mean firing rates is desired since it reduces the likelihood of neurons

University of Cape Town

synchronizing with each other – a phenomenon that would introduce redundant information into the network. For this small population of neurons, the problem of synchronization should be negligible. For a neural network that must perform the cross correlation over only a few iterations of the input signal, many more neurons would be needed and therefore many independent noise sources should be used.

Our neuron also needs to be stimulated by the input signal. This is achieved by driving the input signal into the neuron via another capacitor C_{in} and resistor R_{in} as shown in Figure 5-2. The input current to the neuron as a result of x is therefore:

$$i_{\text{in}}(t) = \text{C}_{\text{in}} \frac{d}{dt}\left[x(t) - i_{in}(t)\text{R}_{\text{in}} - u(t)\right] \qquad 5\text{-}4$$

This equation is the *membrane response model* since it models the amount of current injected into the neuron in response to a stimulus. C_{in} should be small relative to C_u. Both C_{in} and R_{in} are intuitively adjusted to find the best performance. As R_{in} is reduced, the input current signal tends towards the derivative of the voltage signal. The ratio of C_{in} to C_u defines the gain of the input signal.

Before the complete neural circuit can be interfaced to the post processing logic, the threshold crossing condition needs to be evaluated. A setback in this demonstration circuit is that the FPGA is also being used to generate the set of Gold codes which are later combined into analogue circuitry to form the input signal. If the FPGA pins are used as the tri-state I/O port of the neuron, then noise from generating the Gold codes for the input signal could couple to the reference voltage for the neuron's threshold. This would contaminate the experiment, so a separate power supply is used and the threshold crossing condition is evaluated first using a Schmitt trigger. Two I/O lines to the digital logic are required in this setup. The complete circuit diagram is shown below.

University of Cape Town

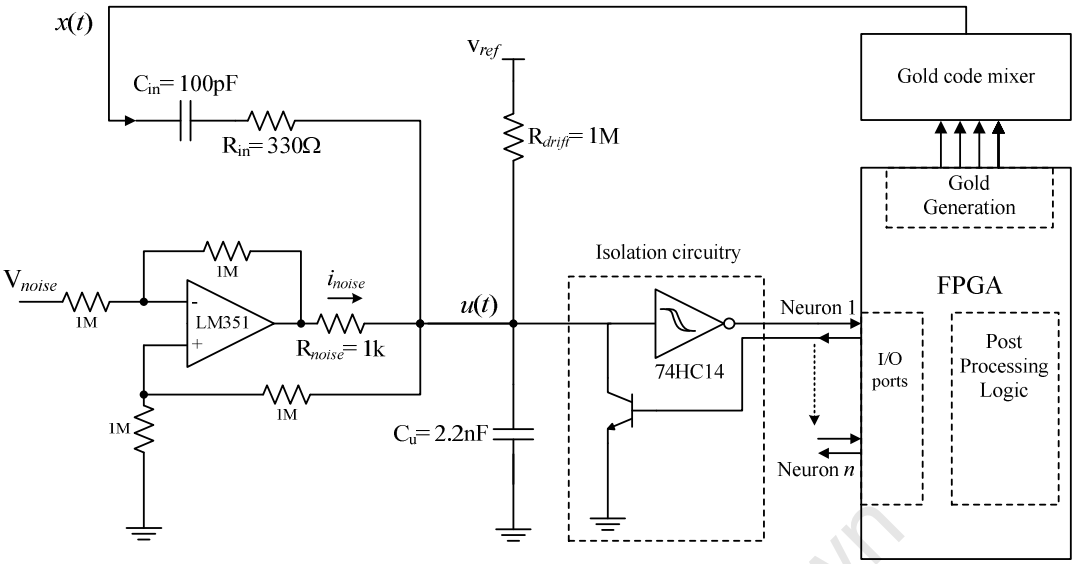


Figure 5-2: The MNCCE circuit showing how the analogue neuron interfaces to the FPGA

The input signal to the neuron is $x(t)$. It is the output of the Gold code mixer shown in Figure 5-3. The Gold code mixer sums all the Gold code input signals together. Note that this circuit produces the negative sum of its inputs. To ensure the correct output polarity of the mixer, the Gold codes are first inverted in the digital logic before being output on the I/O pins of the FPGA.

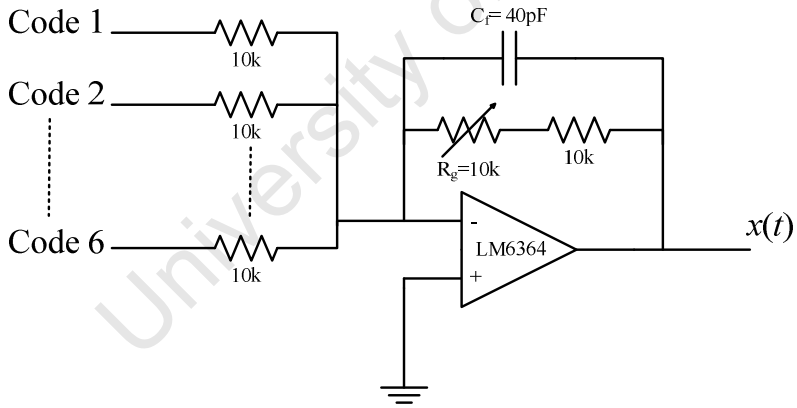


Figure 5-3: Diagram for a circuit which mixes up to 6 gold codes together to make the input signal for the neural cross correlation engine.

The combination of Gold codes which make up $x(t)$ are selected either in the software when uploading the codes, or by the user by physically configuring which codes are connected. The variable 10kΩ resistor allows the amplitude of $x(t)$ to be adjusted.

University of Cape Town

5.3 FPGA Configuration

The FPGA is the work-horse of the neural correlation. Its functional specification can be broken down as follows:

- Detect neural spikes
- Calculate spike intervals
- Generate reference Gold codes
- Update the data array with the neural cross correlation

In addition, the Gold codes that make up the input signal for the neurons need to be output on I/O ports. The FPGA must also support an interface to the PC so that the neural cross correlation can be viewed. This is managed by firmware which runs on the Nios II soft-core CPU. Altera Quartus II Web Edition and the Nios II EDS (Embedded Design Suite) were used to develop the code for the FPGA and the firmware for the Nios II. The project folder with all the FPGA design code, NIOS II firmware and Matlab functions can be found in Appendix F.

In Quartus II, a block diagram file (BDF) can be used to link signals between different modules and I/O ports on the chip. It is a good starting point for any new design. The separate modules in the BDF are defined as entities in VHDL code. VHDL stands for VHSIC Hardware Description Language, where VHSIC stands for Very-High-Speed Integrated Circuits. A simple and practical introduction to VHDL can be found in *Circuit Design with VHDL* by Volnei A. Petroni (Petroni, 2004). The simplest entity of the FPGA design is the `neuron_controller_a2.vhd` which handles interfacing to a neuron via a separate input and output I/O line. It is an adaptation of `neuron_controller_a.vhd` which can be used to interface directly to $u(t)$ with one wire, if the isolated threshold detection circuitry were not required. The purpose of `neuron_controller_a2.vhd` is to ensure that the neuron is correctly reset for at least t_{ref} after it spikes. It also ensures that only one positive edge is generated on its output signal whenever the neuron spikes.

A separate instance of the `neuron_controller_a2` entity is required for each neuron. The outputs from all the neuron controllers are combined to form a bus which connects to the cross correlation generator entity which is defined in `nx_generator.vhd`. This entity is a state machine which monitors each of the signals from the neuron controllers. It also has an interval timer derived from the system clock. It uses this timer as a reference for calculating the spike intervals for each neuron. When a positive edge is detected on one of the neuron controller signals, the spike interval for that neuron is calculated and the spike time for that neuron is saved so that it can be used to calculate the next time interval at the next spike. The spike interval is then used

University of Cape Town

as the address location in an array of on-chip 16 bit memory. The value at that location is read and either incremented or decremented depending on the state of the reference gold code at the neuron's previous spike. The new value is then written back to memory and the new state of reference gold code is also saved for the next spike.

The Gold codes used as the reference codes to perform the cross correlation and to make the input signal to the neurons can be generated using the reference design described in the Altera Application Note 295. Since only a small subset of the Gold codes is used in the demonstration circuit, it is pre-programmed into a separate memory in the FPGA configuration. This memory entity has an 8-bit word size with each bit of a word representing a chip for each Gold code. One byte is read from it every 1us and when the end of the ROM is reached, reading starts from the first word again. Bits 1-6 of the byte are output directly to I/O pins of the FPGA. Bits 7-8 are reserved for reference Gold codes and are connected to the `nx_generator`. The advantage of this method is that once the neural cross correlation engine and the Gold code generator have been programmed into the FPGA, they can easily be configured to use any set of 1023 bit codes. Note that the `nx_generator` entity has been written to support as many reference codes as space on the FPGA allows, but only two will be used for these simulations.

Perhaps the most important aspect of processing spike trains from a population of neurons is the speed at which a single neuron can be processed. The FPGA code used here requires 6 clock cycles to process each spike and any spikes that occur during this period are automatically queued. If a spike is queued then error is introduced into the timing of the spike. For this circuit however, the error is negligible since only 7 neurons are used and the source clock is 24MHz.

5.4 The soft-core processor

Last but not least, a soft-core processor was added to the FPGA configuration. This can be programmed to analyse the neural cross correlation, but for now it simply acts as a medium for interfacing the neural hardware to a PC via RS232. The Nios II soft-core processor was modified to support a peripheral 8-bit I/O port, a 32 bit input port and an external Avlon slave bus. The 8-bit I/O port is used to control the operation of the Gold code generator and the ISIH generator. This is implemented by connecting bit 7 and 8 to the reset lines of the Gold code and ISIH generator respectively. Bits 1-6 are reserved for future use. The 32-bit input port can be used to capture diagnostic information from the `nx_generator`. Currently this is only used to read the total number of spikes used to generate the neural cross correlation.

University of Cape Town

The Nios II processor needs to be capable of accessing the external memory used by the neural engine and Gold code generator. An Avlon slave bus is mapped over a free part of the address space of the Nios II processor. The external bus signal lines then interface to the external memory. The type of memory used is in fact dual port memory which supports two independent reading and writing operations. This configuration allows both the processor and the rest of the neural cross correlation hardware to access the ISIH independently. A program written for the Nios II processor relays data from the external memory to the UART module which is also part of the Nios II processor. This means that the neural cross correlations can easily be read, and that new Gold codes with which to perform the neural cross correlation can easily be uploaded onto the chip.

The software on the Nios II also supports resetting the other hardware on the FPGA and also reads diagnostic information about the neural cross correlation engine. Recall that the Nios II program with the Quartus II project for configuring the FPGA is provided in Appendix F. Also provided in Appendix F is a Quartus II project which programs the FPGA to compile a normal ISIH instead of the neural cross correlation produced by the MNCCE. It uses the `isih_generator` entity as a substitute for the `nx_generator` entity. The firmware for the Nios II is fully compatible with both projects. It can download the ISIH for each neuron in the two-neuron cross correlation engine and it can also combine the ISIHs from multiple neurons running in parallel.

5.5 The PC interface

The final tool necessary for running the neural engine is a collection of MATLAB functions which encapsulate interfacing with the hardware. These functions make up the Neural Hardware Interface (NHI) and can also be found in Appendix F. The first MATLAB function that must be called is `initdevice` which ensures that the COM port is safely opened and establishes a connection with the neural hardware. Next the progress of the neural correlation can be viewed by using the `download` function. This function downloads all the data from the entire address space of the neural hardware memory as an array of 8192 16 bit integers. If the ISIH generation configuration of the FPGA is being used, then each element of the array represents a 1ms bin of the ISIH. If the neural cross correlation configuration is being used, then elements 1-1023 and 1025-2047 contain the two neural cross correlations and elements 7169-8192 contain the data in the Gold code memory.

Alternatively, a specialized function `getnx` can be used to download only the neural cross correlation. The function `upload` can be used to upload the Gold codes. There are also support functions for clearing the memory and starting and stopping the

University of Cape Town

hardware. A description of these functions in the NHI can be found within the function files as per standard MATLAB documentation techniques.

5.6 Results

The ISIH generator configuration of the FPGA was first used to show that the neurons are behaving as expected. A population of 7 electrical neurons was used. They were tuned to have a mean firing interval of about 2ms and were inspected with an oscilloscope to ensure that there was no synchronisation in their firing times. A line graph which connects the peaks of a histogram is shown below. This histogram is the combination of the ISIHs generated for each neuron with stationary input.

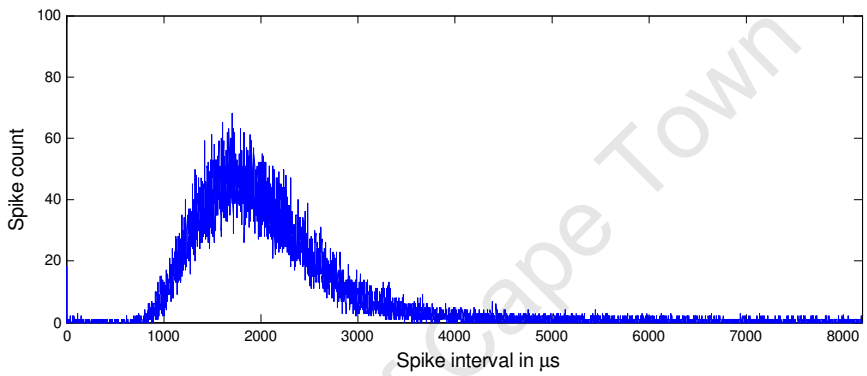


Figure 5-4: Combined ISIH from 7 electronic circuits of neurons with no input stimulus. The neurons were calibrated to have a mean firing rate of 2 ms.

The input signal $x(t)$ consisting of two Gold codes was then used to stimulate the neurons as shown in Figure 5-2. The first Gold code had a phase shift of 300 chip periods, and the second Gold code had a phase shift of 150 periods. Monte Carlo methods were used to find the optimum circuit parameters for generating the neural cross correlation. The amplitude of the noise source voltage was adjusted until the RMS of the noise current injected into the neuron was 70µV. The gain of $x(t)$ was also adjusted using R_g until the RMS input signal current into the neuron was 600µV. This is many times larger that the magnitude of the noise when compared with the simulation, but under the conditions of the test, it was necessary to achieve a suitable neural cross correlation.

University of Cape Town

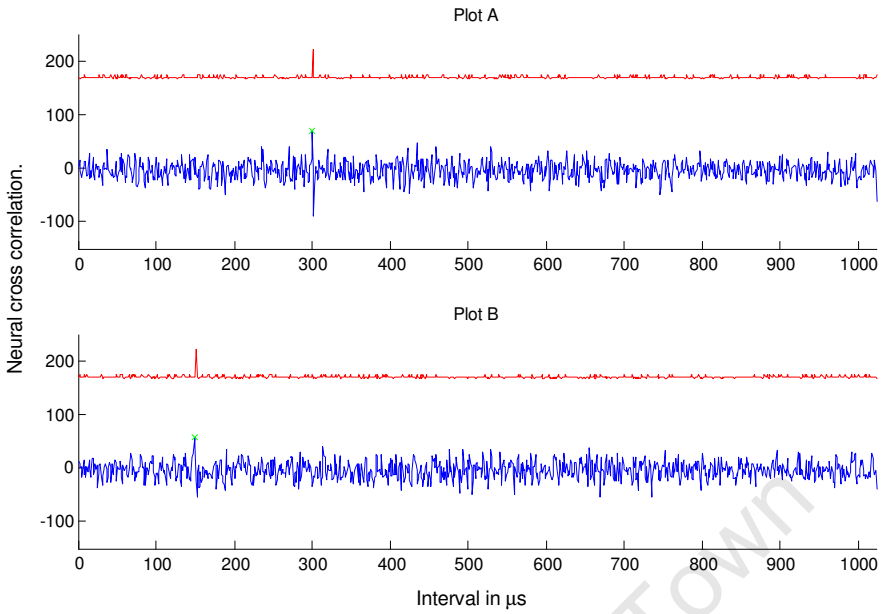


Figure 5-5: The blue curve in plot A shows the neural cross correlation for the first Gold code, compiled from the spike train of one neuron. A green cross indicates the peak which accurately reflects the correct phase shift of 300 chip periods. The similar curve in plot B indicates the 150 chip phase shift in the second Gold code. The curve of the mathematical cross correlation function is shown in red. It has been vertically offset for clarity.

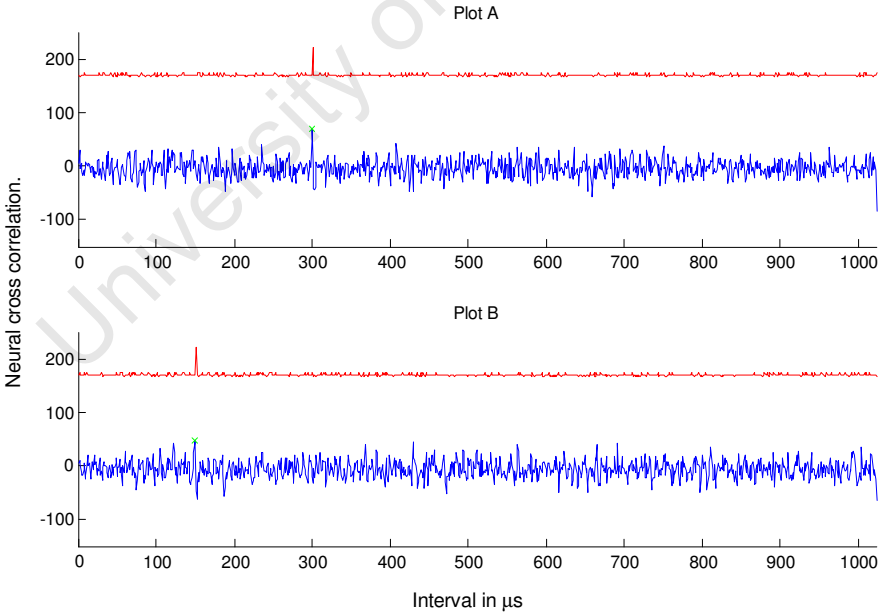


Figure 5-6: Shows similar results to the first figure, but in this case, the neural correlation was compiled from a population of 7 inhomogeneous neurons.

University of Cape Town

The neural cross correlation configuration of the FPGA that implements the MNCCE was used to control the neurons. The test was first performed with only one neuron. The resulting neural cross correlation for each of the two reference codes is shown in Figure 5-5. The theoretical cross correlation is also shown in red. The number of neuron spikes used to generate the neural cross correlation was counted since this more accurately reflects the computational requirements of the system. In this test, 114714 neuron spikes were used. The test was then repeated with the same configuration, but with all 7 neurons. The resulting neural cross correlation is shown in Figure 5-6. It was generated from 129152 spikes, but took less time to compile since multiple neurons were running in parallel.

The peaks in the neural correlation accurately indicated the phase shift of the Gold code in the input signal. The test was also performed for other combinations of Gold codes and at various phase differences. In each case, peaks in the neural combination also indicated the phase differences. Tests were also performed with three Gold codes added together to form the input signal to the neuron. The results for these tests have been omitted, but the findings were that under these conditions the phase shifts of the Gold codes could no longer be reliably detected.

5.7 Discussion

The Nios II Cyclone Evaluation Board and associated tools proved to be a dynamic and powerful system for prototyping a neural cross correlation engine. An ISIH could rapidly be generated from a population of neurons and the corresponding system for implementing the MNCCE could also be demonstrated.

The results in Figure 5-5 and Figure 5-6 showed that the electrical circuit for the MNCCE can generate a set of data in which the characteristic autocorrelation peak of a Gold code can be distinguished. Figure 5-6 also indicates that when two Gold codes are added together to form the input signal, then the neural engine can be used to calculate the phase of each code. Given that the input signal only consisted of two Gold codes, the results appear to be poorer than predicted by the simulation. The magnitude of the autocorrelation peak (in relation to the deviation of the rest of the neural correlation data) was less that that achieved in the simulation.

The poorer performance of the practical engine over the theoretical can be attributed to the dissimilarities between the simulated neuron and the actual electrical neuron, the most significant dissimilarity being in the way the input signal drives the neuron. The change in potential as a result of the input signal is considerably different from the ideal pulse assumed by the simulation. The form of the digital signal is altered by both the Gold code generator and R_{in} and C_{in} in the neuron.

University of Cape Town

Others dissimilarities are variations in component values which result in a non-homogenous population of neurons being used. In addition, the noise input to the neuron is band limited to about 4MHz by the LM3051. The simulation used a sample rate of 1MHz, which by the Nyquist criterion would be comparable to a noise bandwidth of only 500kHz.

Improvements to the system could involve increasing the resolution of timing the spike interval. A drawback of this is that a faster timer would be needed, and each interval calculation would require more circuitry which would inevitably require more power.

In Figure 5-2, the Schmitt trigger may be considered as a poor choice for a threshold detector. It was used in this case because it represents the simple input device which can be used to interface to the digital back-end of the circuit. Alternatively, a comparator could be used for a more accurate threshold detector. However, this would reduce the simplicity of the design.

A limitation of the circuit used to demonstrate neural cross correlation, is that the phase shifts of the Gold codes in the generated input stimulus can only be made in steps of the chip period of the Gold code. In order to test a more refined set of phase shifts, either an independent source must be used as the input signal, or the Gold code generator in the FPGA configuration needs to be modified to support this. Under these conditions, the resolution of the interval counter must be increased so that a histogram with a smaller bin size can be generated. This may enable a more accurate phase reading.

Further investigation into this circuit must aim to reduce the number of neural spikes required to perform the cross correlation and maximise the number of codes that can be mixed into the input signal. The investigation must involve testing with different pulse shapes by varying R_{in}, C_{in} and C_f. Different magnitudes of the noise input and drift may also influence the performance. Further investigation must also consider adding a noise to the input stimulus as would be the case when processing a received radio signal.

University of Cape Town

CHAPTER 6: CONCLUSIONS

This dissertation has covered a broad area of research involving a bio-inspired, analogue-digital hybrid cross correlation engine. This engine was proposed as a candidate for low-power design in CDMA communication systems. This dissertation investigated the behaviour of the cross correlation mechanism and proposed an improved circuit that would be more practical in a regime which requires multiple cross correlations. The circuit was dubbed the Multi-code Neural Cross Correlation Engine (MNCCE) and a practical implementation proved that it is feasible.

6.1 The PSTH needs to resemble the signal for cross correlation to occur

Perhaps the most significant weakness of previous neural cross correlation or autocorrelation methods was that the input signal was simply added to the potential of the neuron and the ISIH generated from the neural spike train was used as the neural cross correlation function. The result is that when a neural cross correlation is performed between a Gold code and a phase shifted version of itself, there is a global maxima that corresponds to the phase difference between the Gold codes, but also additional local maxima at other phase shifts. This disagrees with the ideal mathematical cross correlation which has a distinct narrow peak at the phase difference and a small pattern with uniform mean and deviation for the rest of the function.

Consequently, if there is an additional interference Gold code added to the initial Gold code signal, then the neural cross correlation with the reference Gold code can yield a false reading for the phase difference. This meant that the neural engine was not suited for cross correlating a signal which was subject to interference. Since wireless CDMA systems in general are subject to a large amount of interference, if a neural engine were to be used, then it would have to be able to perform a useful cross correlation even when interference is present.

When the probability density of the ISIH was derived as a function of the probability density of the PSTH and the probability density of the time-to-first-spike, it appeared that the neural cross correlation was strongly dependent on the from of the PSTH. Changing the way the input signal drove the neuron made the PSTH correlate more closely with the input signal. As a result the neural cross correlation resembled the ideal cross correlation more strongly and the neural engine could successfully be used to cross correlate a Gold code which had up to 5 other interference Gold codes. Based on equation 3-4 and the results of the investigation, the author therefore concludes that in

University of Cape Town

order to make the neural cross correlation correspond to the mathematical definition more strongly, the neuron's design should aim to make the PSTH reflect the signal to cross correlate as accurately as possible.

6.2 A leaky integrate-and-fire neuron will yield a more mathematically accurate cross correlation.

Leaky integrate-and-fire neurons were not discussed in this thesis. This model is however a more accurate representation of the electronic analogue neuron used in the demonstration circuit. A significant characteristic of the leaky IF-neuron is that the rate of change in the neuron's potential decreases as the potential reaches the firing threshold. Apart from its simple construction, the benefit of this model is that the time-to-first-spike density function should correlate more accurately with the magnitude of the stimulus to the neuron. This can be concluded based on the two important roles that noise plays in the neuron.

In order to generate a PSTH that correlates closely with the input signal, two properties are desired of the neuron. The first is that the model of the neuron must aim to have a uniform probability of firing over a particular range of intervals. This range of intervals will reflect the range of phases to detect. For an IF-neuron with stationary input, this distribution is dependent on the noise, drift and firing threshold. The first passage times density function proves that a uniform distribution is unrealisable for finite parameters, so instead the parameters in the neuron must be selected for a suitable compromise between the mean firing rate and the spread of the firing distribution. The second property of the neuron is that the probability of it firing must be strongly dependent on input stimuli at that instant.

The importance of noise for the two behavioural properties is discussed by Cecchi *et al.* (2002). They concluded that the noise in biological neurons accounts for the two behavioural properties because the noise affects the generation of an action potential in two ways: (i) it introduces randomness into the membrane potential; and (ii) it introduces randomness into the firing threshold of the neuron. Recall that the latter is known as escape noise because introduces uncertainty into whether a neuron will fire or not (i.e. it is responsible for the spike initiation variability). The neuron model used here uses only diffusion noise which indirectly causes spike initiation variability for a particular chip interval of the input to the neuron. It is this variability which ensures the direct relationship between the magnitude of the input signal and the probability of firing.

University of Cape Town

In order to increase the dependence of the neuron spiking on the input signal one must aim to maximise the spike initiation variability, but simultaneously reduce the uncertainty in the membrane potential. This can be achieved by minimising the rate of change in the neuron's potential when it is near its firing threshold as is the case with a leaky integrate-and-fire neuron. This is simply the converse of Cecchi *et al.* who argued that spike initiation variability can be reduced by increasing the rate of change in membrane potential when the membrane potential is near that of the firing threshold.

6.3 The Multi-code Neural Cross Correlation Engine

The MNCCE takes the design aim of the PSTH one step further by proposing a neural cross correlation engine in which the PSTH generated by the reference code is replaced by the reference code itself as shown in equation 4-2. In so doing, the reference signal which is inherently digital, no longer has to be used to drive the neuron which in turn is used to generate the PSTH histogram in the digital back end. When using a digital back-end for the neural engine, this method would be far more efficient since it can use the same spike train from a neuron driven by a signal to generate multiple neural cross correlations between that signal and other digital reference codes. In fact, all that is required for each additional cross correlation is that a bin in the neural cross correlation is either incremented or decremented depending on the state of the reference Gold code. The following conclusions can be drawn from the implementation of the MNCCE.

6.3.1 The MNCCE can perform multiple cross correlations on a signal.

A cross correlation engine for a GPS receiver would be required to perform 4 to 5 cross correlations with the received signal when in tracking mode and at least 24 cross correlations during acquisition mode. A wireless AER receiver would be required to cross correlate the received signal with a reference code for each node on the AER network. If the MNCCE were used under these conditions then the same set of spike intervals (calculated from one population of neurons driven by the received signal) can be used to generate all of the cross correlations.

Given that an adequately accurate cross correlation can be generated for each Gold code, fewer neural spikes and interval calculations would be needed than if a separate two-neuron type cross correlation engine were to be used for each Gold code

6.3.2 The circuit for the MNCCE works, but its performance is inadequate.

Tests on the MNCCE circuit shown in Figure 5-2, found that it could successfully be used to detect the phase shift of a Gold code in the input signal to the neuron. Cross

University of Cape Town

correlations for multiple codes can be performed at the same time and only when the Gold code is present, will the respective neural cross correlation show the phase difference between that code and the internally generated code.

The circuit shows conclusively that the neurons can be used in this manner as a cross correlation engine. Refining the circuit to match the conditions of the simulation more accurately may yield a neural correlation engine that is capable of processing an input signal that consists of up to 6 Gold codes. Chapter 5 only discusses one possible circuit. Other circuits may in fact yield better performance by tweaking the behaviour of the neuron such that the density of the time-to-first-spike represents the input signal as accurately as possible.

The circuit has shown that a full scale implementation of the neural cross correlation engine would require as many as one hundred thousand neurons. Therefore the ultimate goal for the neural correlation engine is a VSLI implementation. A design for such a system would require an in-depth analysis of the VSLI performance to ensure that the final circuit will operate as required.

6.3.3 There may be a better set of spreading codes for the MNCCE

The performance achieved by the neural cross correlation engine is inferior to the ideal mathematical cross correlation. Consequently, a higher signal to noise ratio is required for the input signal. However, the PRN codes used in the simulation were Gold codes, and these were designed for use in a cross correlation that is as close to the mathematical ideal as possible. Since the neural engine relies on a slightly different mechanism, there may be a better set of PRN codes for which it is better suited.

In an AER communication protocol where there are thousands of unique nodes, the neural cross correlation engine may be used because of its low power consumption. Under these circumstances there may be a more suitable set of spreading codes that would allow more efficient use of the communication bandwidth.

University of Cape Town

CHAPTER 7: RECOMMENDATIONS FOR FUTURE WORK

Only a small aspect of the neural cross correlation engine was covered in this dissertation. The circuit for the MNCCE showed that neurons can practically be used to cross correlate an unknown signal with a digital signal, and has opened up the doors for further research in this area. Based on the findings and the conclusions, the author recommends the following future work on a neural cross correlation engine.

7.1 Improve the performance of the MNCCE circuit

The first circuit proposed for the MNCCE was only capable of processing an input signal consisting of two Gold codes. If the accuracy of the neural cross correlation can be improved enough so that it is adequate for use in a GPS receiver then the engine will arguably be more efficient than the two-neuron cross correlation method. As motivated earlier, this is because it would require very little additional circuitry and processing for any additional cross correlations.

7.2 Analyse the power consumption of the neural cross correlation

As of yet, no formal argument or test results have been presented which show that a neural cross correlation engine would consume less power than its digital counterpart. If a circuit based on either the two-neuron cross correlation engine or the MNCCE can be found to be accurate enough for use in a GPS receiver, then its power consumption must be formally analysed.

The power consumption of the neural network would depend on the architecture of the VSLI implementation. The power consumption of the digital side of the MNCCE can be analysed using the "PowerPlay Power Analyzer" tool in Quartus II.

Once the proper analysis has been performed on the two neural engines, the results must be compared with those of existing cross correlation engines used in GPS receivers. A brief break down of the power consumption in a GPS solution using a selection of IC's from the **uNav** series is presented in Appendix E.

University of Cape Town

7.3 Demonstrate the MNCCE with the NAMARU V2 development board

The Namaru V2 board is a development and research platform that can be used for a GPS receiver. It has two RF front ends which produce the baseband signals. The most powerful feature of the NAMARU board is that all the digital processing required for the GPS receiver are programmable. The MNCCE can therefore be implemented as a replacement cross correlation engine for the pre-existing solution configured on the board.

The NAMARU board is also based on an Altera FPGA which is configured using the Quartus II Embedded Design Suite. Moreover, it uses the Nios II soft-core processor to run the firmware for the application. This is the same development platform and development architecture that was used to create the MNCCE which means that merging the two projects should be a simple process. More information regarding the NAMARU board can be found at http://www.dynamics.co.nz/gpsreceiver/.

7.4 Further research must follow before using the neural engine for AER

It was concluded that there may be a better set of spreading codes that would be more suitable for use with a neural cross correlation engine in a wireless networking regime in which there may be thousands of individual nodes. This proposal must be investigated before designing the wireless AER protocol. Since it is a novel protocol, the designer would be at liberty to redesign all aspects of the physical access layer and should therefore aim to ensure that the optimum set of spreading codes is used.

Another line of research must also be followed base on the initial demonstration of the neural correlation engine in 3.1. Figure 3-1 showed that the ISIH generated from the two-neuron cross correlation engine has a characteristic and replicable pattern that may be unique to the input signal. Instead of analysing the ISIH with a peak detector, one must consider using a more complex algorithm or even another neural network which can thereby be trained to recognise certain types of signals. This almost completely bio-inspired solution, together with a careful selection of unique codes assigned to each spike event in the AER protocol, may yield a wireless CDMA architecture far superior to any previously proposed system.

University of Cape Town

REFERENCES

Altera Corporation, 2003. Gold code Generator Reference Design. Application Note 295, v1.0.

Arbib, M. A., (2003). *The handbook of brain theory and neural networks*. Cambridge, Mass: MIT Press.

Bertsekas, D. P. and Tsitsiklis, J. N., (2002), *Introduction to Probability*. Belmont, Massachusetts: Athena Scientific.

Bharucha-Reid., A. T., (1960). *Elements of the Theory of Markov Processes and Their Applications*. London: McGraw-Hill Book Company, Inc.

Bakay, R. A., (2006). Limits of brain-computer interface. *Neurosurg Focus* 20, May 2006.

Bargas, J. and Galarraga, E., (1998). Ion channels: Key to neural specialization, *The handbook of brain theory and neural networks*, MIT Press, Cambridge, MA, USA, pp496-501.

Borg-Graham, L. J., Monier, C., and Frégnac, Y., (1998). Visual input evokes transient and strong shunting inhibition in visual cortical neurons. *Nature*, [Online], 393(6683). pp369-373. Abstract from PubMed database.

Available at: http://www.ncbi.nlm.nih.gov/sites/entrez?db=PubMed [accessed 29 October 2007]

Borst, A. and Egelhaaf, M., (1989). Principles of visual motion detection. Trends in *neuroscience 12(8),* pp297-305.

Burkitt, A. N., (2006). A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. Biological Cybernetics 85, pp1-19.

Cariani, P. A. and Delgutte, B., (1996). Neural Correlates of the Pitch of Complex Tones. I. Pitch and Pitch Salience. *Journal of Neurophysiology* 76(3). pp1698-1716.

Cecchi, G. A., Mariano, S., Alonso, J. Martinez, L., Chialvo, D.R., Magnasco, M.O., (2000). Noise in neurons is message dependent. *PNAS* 97(10), pp7776-5561.

Dayan, P. and Abbott, L.F., 2005, Theoretical Neuroscience, Available at: http://cns.bu.edu/~tren/antje/TOC.htm [accessed 22 April 2005]

Destexhe, A., Mainen, Z.F. and Sejnowski, T.J., (1995). Synaptic Currents, Neuromodulation, and Kinetic Models. *Handbook of Brain Theory and Neural Networks*. Bradford Books: MIT Press, pp956-959.

Dickinson, P. S., (1995). Neuromodulation in Invertebrate Nervous Systems. *Handbook of Brain Theory and Neural Networks*. Bradford Books: MIT Press, pp631-639.

Faisal A. A., White J.A., Laughlin S.B., (2005). Ion-channel noise places limits on the miniaturization of the brain's wiring. Curr Biol 15, pp1143–1149.

Faisal, A. A. and Laughlin, S.B., (2007). Stochastic simulations on the reliabitity of action potentials propagation in thin axons. *Computational Biology* 3(5), pp783-795.

Feng, J., (2004). *Computational Neuroscience: A Comprehensive Approach,* London: CRC Press UK.

University of Cape Town

Fernández, E., Pelayo, F., Ahnelt, P., Ammermüller, J. and Normann R.A., (2004). Cortical visual neuroprosthesis for the blind. *Restorative Neurology and Neuroscience*. In press (2004).

Folowosele, F., Tapson, J. and Etienne- Cummings, R., (2007). Wireless address event representation system for biological sensor networks. *Proceedings of SPIE*, vol. 6592, pp659-205.

Fofonoff, T., Martel, S., Wiseman, C., Dyer, R., Hunter, I., Hatsopoulos, N., Donoghue, J., (2002). A highly flexible manufacturing technique for microelectrode array fabrication. EMBS/BMES Conference, 2002. Proceedings of the Second Joint 2, pp2107-2108.

Gerstein, G. L. and Mandelbrot, B, (1964). Random walk models for the spike activity of a single neuron. Biophysical Journal 4, pp41-68.

Gerstner, W. and Kistler, W. M., (2002). *Spiking Neuron Models: Single neurons, populations and plasticity*. Online: Cambridge University Press. Available: http://icwww.epfl.ch/~gerstner/SPNM/SPNM.html, Accessed: 20 October 2007.

Glisic, S. and Vucetic, B., (1997). *Spread Spectrum CDMA Systems For Wireless Communications*. Boston, London: Artech House, Inc.

Harrison, R. R., Watkins, P.T., Kier, R., Lovejoy, R., Black, D., Greger, B. and Solzbacher, F., (2007). A Low-Power Integrated Circuit for a Wireless 100-Electrode Neural Recording System. IEEE Journal of Solid-State Electronics 42, pp123-133.

Hassenstein, B. and Reichardt, W.E., (1956). Functional structure of a mechanism of perception of optical movement. In Proceedings of the 1st Intemational Congress of Qbernetics in Namar, 797-801.

Herrmann, A. and Gerstner, W., (1999). Understanding the response to synaptic input. Artificial Neural Networks 7.

Hodgkin, A. L. and Huxley, A. F., (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiolo.* (117), pp500-44.

Kaplan, E. D. & Hegarty, C.J., (2006). *Understanding GPS: Principles and applications*, 2nd Ed, Norwood: Artech House.

Kenny, P. R. and Bakay, R. A., (1998). Restoration of neural output from a paralysed patient by a direct brain connection. *Neuroreport.* 9 (8), pp1707-1711.

Kramers, H. A.., (1940). Brownian motion in a field of force and the diffusion model of chemical reactions. *Physica* vol. 7, p284.

Licklider, J. C. R., (1951). A Duplex Theory of Pitch Perception. *The Journal of the Acoustical Society of America* 23 (1), p147.

Longtin, A., Bulsara, A. and Moss, F. (1991). Time-Inteval Sequences in Bistable Systems and the Noise-Induced Transmission of Information by Sensory Neurons. The American Physical Society 67 (5), pp656-659.

Ma, J. W., Beck, J.M., Latham, P.E. and Pouget, A., (2006). Bayesian inference with probabilistic population codes. *Nature Neuroscience* 9(11), pp1432-1438.

University of Cape Town

McCulloch, W. S. and Pitts, W. H., (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics* 5, pp115-133.

Mead, C. A.., (1989). *Analog VSLI and neural systems*. Addison – Wesley.

Mead, C. A.., (1990). Neuromorphic electronic systems. *Proceedings of the IEEE 78*,

Musallam, S., Bak, M., Troyk, P., Andersen, R., (2006). A floating metal microelectrode array for chronic implantation. *Journal of Neuroscience Methods* 160, pp122-127.

Newcomb, R. and Lohn, J., (1995). Analog VLSI for Neural Networks. *Handbook of Brain Theory and Neural Networks*. Bradford Books: MIT Press, pp96-90.

Norman, R., (2007). Technology Insight: future neuroprosthetic therapies for disorders of the nervous system. *Nature Clinical Practice Neurology* 3, pp444-452.

Pace, P. E., (2000). *Advanced Techniques for Digital Receivers*. Boston, London: Artech House.

Petroni, A. V., (2004). *Circuit Design With VHDL*. Cambridge, Massachusetts: MIT Press.

Plesser, H. E., (1999). Aspects of signal processing in noisy neurons, Georg-August-Universität, Göttingen, PhD thesis.

Price, R. and Green, P., (1958). A communication technique for multipath channels. Proceedings of the IRE.

Psiaki, M. L., (2001). Block Acquisition of Weak GPS Signals in a Software Receiver. *ION GPS*, 11-14 September 2001.

Rall, W., (1964). Theoretical significance of dentritic trees for neuronal input-output relations. *Neural Theory and Modeling*, Stanford, CA: Stanford University Press, p73-97.

Ross, S. M., (1983). *Stochastic processes*. United States of America: John Wiley & Sons, Inc.

Salinas, E., (2006). Noisy neurons can certainly compute. *Nature Neuroscience* 9(11), pp1349,1350.

Sarpeshkar, R., (1997). Efficient precise computation with noisy components: Extrapolating from an electronic cochlea to the brain. Unpublished doctoral dissertation, California Institute of Technology. Postscript copy available on http://www.pcmp.caltech.edu/anaprose/rahul/thesis/. Cited in Sharpeshkar, 1998.

Sarpeshkar, R., (1998). Analog Versus Digital: Extrapolating from Electronics to Neurobiology. *Neural Computation* 10, pp1601–1638.

Shadlen, M. N. and Newsome, W. T., (1998). The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. *Journal Neuroscience* 18, pp3870-3896.

Srinivasan, M. V. and Bernard, G. D., (1975). A Proposed Mechanism for Multiplication of Neural Signals. *Biol. Cybernetics* 21, pp227-236.

Softky, W. and Koch, C., (1993). The highly irregular firing pattern of cortical cells is inconsistent with temporal integration of random EPSPS. *Journal of Neuroscience* 13, pp334-350.

University of Cape Town

Sverre Lande, T. and Hjortland, H. A., (2005). Impulse Radio technology for Biomedical applications. Biomedical Circuits and Systems Conference. 27-30 Nov. 2007.

Tapson, J., (1998). Autocorrelation Properties of Single Neurons. IEEE Comsig '98, pp209-212.

Tapson, J. and Etienne-Cummings, R. (2007). A Simple Neural Cross-Correlation Engine. Proc. IEEE Int. Symp. Circuits and Systems (ISCAS 2007), New Orleans, pp1285-1288.

Tapson, J., Vismer, M. P., Jin, C., van Schaik, A., Folowosele, F. and Etienne-Cummings, R., (2008). A Two-Neuron Cross-Correlation Circuit with a Wide and Continuous Range of Time Delay. Accepted to *IEEE ISCAS 2008,* Seattle, Washington, May 2008.

Tuckwell, H. C., (1988). *Introduction to theoretical neurobiology: Vol. 2, Nonlinear and stochastic theories*. Cambridge University Press.

Petroni, V. A., 2004. *Circuit Design with VHDL*. United States of America: Massachusetts Institute of Technology.

Verdu, S., (2003). *Multiuser Detection*, Cambridge University Press. General reading.

University of Cape Town

APPENDIX A

```
function fptd= calc_fptd(insig, n, d)
%CALC_FPTD
%  Calculates the first passage time distribution of a perfect IF neuron. It models
%  the neuron as a random walk in steps (discrete-time) from an initial
%  state of 0 to an absorbing barrier at 1. An input adds to the step vector
%  with a positive input being in the direction of the barrier.  Note, this
%  is analogous to the input specifying the movement of the barrier towards
%  the state at each step.
%
% Assumes the neuron starts when sig starts
%   i.e potential=0 at insig(1).
%
% Parameters:
%  <insig> is the discrete input signal.
%  <d> drift of the neuron
%  <n> is the noise per Ts
%  Returns the probability of a spike occurring after <input(i)>
%
%                                                (C) Mark Vismer, 2007

% set up variables
nscale= n; %noise per sample
U= 1; %neuron firing threshold
range= length(insig); %number of samples for which to do the calculation
Tx= 0.002; %defines resolution of state space
x= -4*U:Tx:4*U;  %define the domain for our state space

%clear and allocate memory
PX= zeros(range,length(x)); %PX(i,:) is the pdf of the state of the neuron at time i
PXd= zeros(range,length(x));

mPf= zeros(range,1); % mPf(i) unconditional probability of firing at time i
mPS= zeros(range,1); % mPS(i) unconditional probability of not firing at time i


if (Tx>n)
    error('Resolution is too bad. . .');
end

%calculate for state 1
PX(1,:)= gauss_pdf(x,d+insig(1),nscale)*Tx; %calculate initial state space.
mPf(1)= sum(PX(1,x>=U)); % Gaussian cdf for probability of firing
mPSi= sum(PX(1,x<U)); % Gaussian cdf for conditional probability of not firing
mPS(1)= 1*mPSi; % Unconditional probability of firing
PXd(1,:)= PX(1,:).*(x<U)./mPSi; %State space on condition of surviving.

%calculate for the rest
for i= 2:range
    for j=1:length(x)
        %Use Chapman-Kolmogorov equation to calculate the new state space.
        PX(i,j)= sum(PXd(i-1,:).*gauss_pdf(x(j),x+d+insig(i), nscale)*Tx);
    end
    mPf(i)= mPS(i-1)*sum(PX(i,x>=U)); % Unconditional probability of firing.
    mPSi= sum(PX(i,x<U)); % Conditional probability of surviving.
    mPS(i)= mPS(i-1)*mPSi; % Unconditional probability of surviving.
    PXd(i,:)= PX(i,:).*(x<U)./mPSi; %State space on condition of surviving.

    if (mod(i,10)==0)
        fprintf(1,'%4.1f%%, ',i/range*100); %show progress
    end
end
disp(' ');

fptd= mPf;
return
```

University of Cape Town

APPENDIX B

```
function ttfs=nttfs(insig,g_noise,drift,n,maxt)
% ttfs=NTTFS(insig,g_noise, drift,n)  LAST UPDATED: 25/11/2007
% Complies a histogram of the time to first spike with a constrained input
% signal. (i.e same phase restart for each interval)
%  insig -> the input signal.
%  g_noise -> gain of the noise.
%  drift -> drift current input to the neuron.
%  n -> the number of SPIKES.
%  maxt -> the maximum length of the histogram
%
% Compiles a histogram of the time to first spike for a neuron stimulated
% by <insig>.
%
%                                                (C) Mark Vismer, 2007

if (isempty(whos('maxt')))
    maxt= 5/drift;         % max interval, expect around 5* mean firing rate.
end

thresh=1;           % threshold (scale g_noise, drift and insig p
ttfs=zeros(1,maxt); % ISIH output matrix

num=0;
codelength= length(insig);% 1023*bitlength;  % length of gold code in steps
while (num<n)     % main loop
    %pause(0.0001); %GUI friendly
    num= num+1;
    g= 0; % restart neuron
    int_count= 0;   % restart interval

    for j= 1 : codelength %inner loop
        g= g + randn*g_noise + drift + insig(j); % new potential
        int_count= int_count+1;
        if (int_count <= maxt)
            if (g>=thresh)            % if spike then...
                ttfs(1,int_count)=ttfs(1,int_count)+1;  % update ISIH
                break; %start again
            end
        else
            %out of range so ignore this one and start again.
            break; %start again
        end
    end
end
```

University of Cape Town

APPENDIX C

```
function isih=ncorr1(insig,g_noise, drift,n)
% Runs a neural autocorrelation. This is a function implementation of the
% neural autocorrelation by Prof. J. Tapson.
%  insig -> the input sig to the neuron. (x*gain_sig)
%  gain_noise -> gain of the noise over that interval.
%  drift -> drift of the neuron over each interval.
%  n -> number of iterations.
%  maxt -> range of the ISIH
% Returns the ISIH of the neuron.
%
%                                                (C) Jonathan Tapson, 2007


global pmf;  %stores spike density (this is p_j)

maxt= 5/drift;         % max interval, expect around 5* mean firing rate.

thresh=1;           % threshold (scale g_noise, drift and insig p
isih=zeros(1,maxt); % ISIH output matrix

pmf= zeros(length(insig),1);

g=0;  %neuron g correlates the input signal
int_count=0;
codelength= length(insig);% 1023*bitlength;  % length of gold code in steps
for its=1:n     % main loop
    %pause(0.0001); %GUI friendly
    for j= 1 : codelength %inner loop
        g= g + randn*gain_noise + drift + insig(j); % new potential
        int_count= int_count+1;
        if (g>thresh)             % if spike then...
            if (int_count < maxt-1)
                isih(1,int_count)=isih(1,int_count)+1;  % update ISIH
            end
            pmf(j)= pmf(j) +1;
            g= 0;
            int_count= 0;                 % and reset potentials
        end
    end
end
```

University of Cape Town

APPENDIX D

```
function isih= ncorr2(insig, g_noise, drift,n)
% isih= NCORR2(insig,g_noise, drift,n)
%
% Runs a neural cross correlation using two neurons.
% This is a function implementation of the cross - correlation method proposed
% by Tapson (2007).
%
%  insig -> two columns of input signal to cross correlate.
%  noise -> gain of the noise for each signal.
%  drift -> drift current input to the neuron for each signal.
%  n -> number of iterations.
%
% Returns two columns as the ISIH for each signal.
%
%                                                (C) Jonathan Tapson, 2007

global pmf;
pmf= zeros(length(insig),2);

siz= size(insig);
if (siz(2)~=2)
    error('Only two input signals is supported.');
end

ne= 0;   %the neuron. Used for both neuron 1 and 2. .since they do
        %not operate at the same time.

maxt= 5/min(drift);         % max interval, expect around 5*mean firing rate.

thresh=1;           % threshold (scale g_noise, drift and sigs appropriately)

isih=zeros(maxt,2); % ISIH output matrix (1 column for each ISIH)

ne_no=1; %idicates which neuron is running

int_count=0;
codelength= length(insig);% 1023*bitlength;  % length of gold code in steps
for its=1:n     % main loop
    for j= 1 : codelength %inner loop
        ne= ne + randn*g_noise(ne_no) + drift(ne_no) + insig(j,ne_no); % new
potiential
        int_count= int_count+1;
        if (ne>thresh)              % if spike then...
            if (int_count < maxt-1)
                isih(int_count,ne_no)=isih(int_count,ne_no)+1;  % update ISIH
            end
            ne= 0;
            pmf(j,ne_no)= pmf(j,ne_no) +1;
            int_count= 0;                % and reset potentials
            ne_no= 3-ne_no;  %toggle to a different neuron.
        end
    end
end
```

University of Cape Town

APPENDIX E

The uNav series is an all CMOS semiconductor family that permits a small, low-power and low-cost GPS architecture. uNav have several different GPS receivers (down converters) and baseband processors which calculate coordinates from the GPS signals and data. They also have some single chip solutions in which the radio receiver and processor are implemented on the same silicon die.

The uN1510 GPS down converter

The uN1510 is a complete GPS RF down converter that integrates all of the RF circuitry required to implement a high sensitivity GPS L1 RF front-end. It interfaces directly to a passive RF antenna and converts the radio signal to the GPS baseband. The module provides a 2-bit digital I/Q output that connects directly to a GPS baseband processor such as the uN8130 or uN2110. Each I and Q output includes a sign bit (ISIGN and QSIGN) and a magnitude bit (IMAGN and QMAGN) and are clocked synchronously with the MCLK output.

Power consumption: Active mode @ -150dBm: < 45mW (typ: 2.9mA @ 1.8V)

The uN8130 GPS Baseband Processor

The uN2110 is a GPS baseband processor that is compatible with the uN1510. It has a 16-bit DSP core to perform cross correlations. There are twelve parallel hardware correlation channels. Their power consumption is shown in the table below:

Table 7-1: Power consumption of the correlation channels in the uN8130

Condition	**Typ (mA)**	**Max (mA)**	**Typ Power @ 1.8V (mW)**
1 Channel	3.7	8.0	6.7
8 Channels	5.3	10	9.5
12 Channels	6.2	11	11.2

The uN3010 GPS Single Chip Solution

The uN3010 is a low cost single-chip GPS solution which has the digital and RF functions both implemented in a single monolithic die.

Power consumption:

60 mW Acquisition mode
50 mW Tracking mode @ 1Hz
13.5 mW Sleep mode

University of Cape Town

APPENDIX F

Please see the attached CDROM for the following:

\matlab\

This folder contains collections of MATLAB code that can be used to replicate all the simulations presented in this thesis.

\neural_correlator_a\

The Quartus II project folder for the FPGA configuration to perform an autocorrelation by generating an ISIH from a population of neurons.

\neural_correlator_nx\

The Quartus II project folder for the FPGA configuration for the MNCCE.

\eclipse_workspace\

The Eclipse workspace settings for the Nios II firmware projects.

\niosii_firmware\

Contains the C programs and libraries for the Nios II firmware.

\mylib\

A library of VHDL entities created for the "neural_correlator_a" and the "neural_correlator_nx" projects.

\mylib_test\

The Quartus II project for testing and simulating the VHDL entities before they are used in the neural engine.

\NHI\

The Neural Hardware Interface layer which is a collection a MATLAB functions created to encapsulate interfacing to the neural hardware.

University of Cape Town