Neural Cross-Correlation for Radio Astronomy

written by

Chipo Nancy Ngongoni
NGNCHI002
Supervised by Professor Jonathan Tapson

This thesis is submitted in partial fulfillment of the academic requirements for the degree of

Master of Science in Electrical Engineering

in the Faculty of Engineering and the Built Environment

University of Cape Town

February 2010

Declaration

I declare that this thesis is my own work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or materials are indicated in the acknowledgements or are explicitly stated with references as appropriate.

This work is being submitted for the Master of Science in Electrical Engineering at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

Author's Signature

Chipo Nancy Ngongoni Date: **12 February 2010**

To my dear departed mother, Liniah Ngongoni, who taught me that I only need to believe in myself and not succumb defeat from obstacles that cannot speak back…this is for you,

RIP

Abstract

Correlation engines are essential elements of most signal processing systems. Areas of applicability include image processing, speech synthesis and analysis, high energy physics, wireless and mobile communication systems, spread spectrum communication systems and even prosthetics [4, 50]. Finding cost effective and computationally less intensive engines is the thrust of most research. Neural networks have also been used as aids in making complex tasks relatively easy to process. Thus in this light recent research by Professor Jonathan Tapson and colleagues have proposed a relatively simple but efficient biologically inspired analog correlation engine [4, 45, 46]. The engine has been tested on digital spreading codes using pseudorandom Gold codes aimed at applications in Global Positioning Systems [52].

This thesis sets out to investigate the applicability of such a neural engine in a Radio astronomy environment. It also describes the research work, design and the implementation of a spiking neuron on a FPGA (Field Programmable Gate Array) migrating from an analog to a digital platform. The architecture of the proposed analog neuron is then adapted to be on a digital reconfigurable platform so that it can perform auto and cross correlations. Radio astronomy has high computational needs. These demands essentially stem from the ever-increasing analogue bandwidths that astronomers wish to observe. This results in large data volumes that need to be processed in real time, thus formulation of energy conserving systems is a necessity despite a system accuracy trade-off. Simulations of the analog and digital spiking neuron indicate that its accuracy may be sufficient for detecting various types of signals and performing the required correlation. This is comparable to the requirements of a cross correlation engine for a radio astronomy system. The benefits that FPGAs deliver to an embedded system design are also looked at.

Furthermore, the concept is analysed through various test runs with signals in the radio frequency range. The neuron is implemented entirely from configurable blocks on an FPGA with input signals supplied from an external source. Traditionally, noise is considered to be a nuisance in electronics and is often a limiting factor for most systems. The noise addition is for the simulation of a realistic neuron model and also to demonstrate how the architecture thrives in stochastic resonance. The stored FPGA output is further used to process the neural spike train and generate the cross correlation functions.

Lastly, this research lays the foundation for further exploration of FPGA technology as applied to neural networks and Radio astronomy, a range of possible project extensions are suggested.

Acknowledgements

To my supervisor, Professor Jonathan Tapson, I owe great gratitude for his continued assistance and guidance in the development of this thesis.

I also want to thank my husband, Rangarirai Matavire, for his occasional distractions and unwavering support. To my brother, Simbarashe G. Ngongoni, and sister, Kudzai Natasha Ngongoni, for always believing in me.

Above all, thank you Lord for seeing me through this period of my life, may greater things come from the fruit of my hands.

Table of Contents

List of Figures

List of Tables

Glossary and Acronyms

ADC: Analog to Digital Converter

ALMA: Atacama Large Millimeter Array

ASIC: Application Specific Integrated Circuit

ATA: Allen Telescope Array

ATNF: Australia Telescope National Facility

BIMA: Berkeley–Illinois–Maryland Association Array

CARMA: Combined Array for Research in Millimeter-wave Astronomy

DSN: Digital Spiking Neuron

EoR: Epoch of Reionization array.

EVN: European VLBI Network

FFT: Fast Fourier Transform

FPAA: Field Programmable Analog Array

FPGA: Field Programmable Gate Array

ISIH: Interspike Interval Histogram

LUT: Look Up Table

PSD: Power Spectral Density

SKA: Square Kilometre Array

SRM: Spike Response Model

VHDL: VHSIC Hardware Description Language

VHSIC: Very High Speed Integrated Circuit

VLA: Very Large Array

VLBA: Very Long Baseline Array

VLBI: Very Long Baseline Interferometry

TEM: Time Encoding Mechanism

Chapter 1: Introduction

The human nervous system can be equated to an information processing system. It continually receives stimuli and processes the stimuli accordingly. It has often been equated to a computer with a parallel algorithm which is highly complex and nonlinear [1]. In the centre of this system is the brain, which upon receiving information perceives it and makes appropriate decisions. The pioneering work of Cajal in 1911[1] introduced neurons as the basic constituents of brain structure. The human brain on average contains between 70 and 80 billion highly interconnected neurons which transmit information in the form of action potentials [10]. The efficiency of the brain has led to; the modelling of these entities on various platforms [3], the implementation of artificial neural models which are comparable to the biological equivalents, as well as use in diverse range of application areas. Neural networks have also been used in astronomy for amongst other things object identification and astronomy cluster identification [2, 5]. While past work on neural applications in the field of astronomy has centred on the application of neural networks as pattern identifiers for clustering algorithms or real time series prediction, this thesis concentrates on an essential part of astronomy instrumentation, the correlator, and how neurons can relatively ease the computational load.

1.1 Motivation and purpose

Analysis and cross reference of signals in communication systems is a vital area which requires a great deal of inquiry. Cross-correlation plays a fundamental role in several signal processing systems. Application systems range from Global Positioning Systems, Spread Spectrum Telecommunications, Radio and Optical astronomy [4]. In general, a received unknown signal that contains information is transmitted in a system, stored and later correlated with known reference signals. The level of correlation depicts the efficiency of the system and in other cases, like astronomy, the type of objects detected [5]. For such extensive signal processing such as Radio astronomy, an efficient algorithm and processor are required. Currently analog, digital, mixed signal and software correlators are being used in the different branches of astronomy. Each has its particular advantages and disadvantages mainly

[1] Santiago Ramón y Cajal was the one who pioneered investigation of the microscopic structure of the brain. He is known as the greatest neuroscientist of all time. In 1906 he won a Nobel prize in Physiology or Medicine.
http://nobelprize.org/nobel_prizes/medicine/laureates/index.html

stemming from the particular astronomy observations to be undertaken, efficiency and also power consumption considerations [6].

Spiking neurons are now widely used in neural modelling. Their strength is demonstrated in the high computational requirements in all biological systems, but using minimal power consumption. After all, the most novel engineering inventions are based on nature [7]. The base of the investigation is in the behaviour of synthetic neurons and their comparison with biological models. It has been widely agreed that in memory dependent tasks, network solutions that incorporate the spiking dynamics of neurons can become less complex and easier to evolve. Spiking neurons have proven to be efficient performing important arithmetic functions in signal processing systems. They have been studied from various viewpoints like mathematical analysis, modelling of biological systems, and engineering applications. The range of neural model circuits has ranged from purely analog, purely digital and mixed signal. Digital structures that have been introduced in literature include Hiroyuki's model [48] which comprises of two shift register generators coupled as a master and a slave, with the Digital Spiking Neuron DSN operating in discrete time. The model presented in the thesis works with the same theorem, building emphasis on a more simplistic implementation so as to use fewer resources in the FPGA.

This is particularly relevant to the Radio Astronomy environment which, because of the wide network of satellites, requires powerful, fast and efficient computational systems to process the large volumes of raw data. Thus in this respect the research investigates the applicability of the spiking neuron in processing signals in the radio frequency range, and efficiency of such a system as compared to other current correlation methods.

1.2 Hypothesis and objective

The primary purpose of this thesis is to implement and test a digital equivalent of the analog neuron that has been presented in past work [4, 44, 46]. The implementation is aimed at coming up with a reconfigurable unit that can be used in highly computational intensive processes. The platform used in the thesis is the Field Programmable Gate Array (FPGA). Preference would have been made for a Field Programmable Analog Array (FPAA), as the internal configuration blocks would have modelled the neuron membrane potential and synapses more accurately since the configuration blocks are already analog modules. Due to

the unavailability of an FPAA, a FPGA had to be used instead. The analog model of the integrate and fire neuron presented in this thesis has previously been used to perform arithmetic operations particularly auto and cross correlations. The accuracy of the digital model will be tested on this basis.

1.3 Thesis Outline

This thesis is organised into six chapters and progresses as outlined. The first chapter is the introduction which gave an overview of the thesis goals and objective. Chapter 2 outlines a brief background to neural modeling. It starts with a description of biological neurons and neural models and delves into the theory of stochastic resonance. Various applications of neural networks are also discussed. Chapter 3 concentrates on how neurons process information with an in-depth description of spike coding and the interspike interval histogram (ISIH). Two major types of spiking neurons are described, the integrate and fire neuron and the spike response model. Some of the simulation platforms used for spiking neurons are described. Chapter 4 then focuses on Radio astronomy and the various correlator algorithms that are currently in use. A brief overview of the advantages and disadvantages of each of the correlator platforms is done. The chapter concludes with the justification of why the neural correlation engine is being applied in a Radio Astronomy environment. Chapter 5 describes the circuit implementation of the integrate and fire neuron. It gives an analysis of the analog and digital designs as well as the thesis architecture of the neuron. Finally, chapter 6 concludes this thesis with a summary of the work done in this thesis and highlights the contributions made. Results are presented of a sine input signal with a range of different frequencies and the relative cross correlation functions from the interspike interval density histogram acquired from the digitally designed neuron. A discussion of the results is given with the relative recommendations. The Reference and Appendix sections conclude the thesis.

Chapter 2: Neural Modelling Background

2.1 Introduction to Spiking Neural networks

Biologically inspired computing has evolved and thrived on the concept of modelling systems according to how living things perceive information and process it. Computers are used to model nature; at the same time nature is studied to improve the functionality of computers [8]. This has led to an extensive range of beneficial research. Notable fields that have thrived from this type of research are genetic algorithms, emergent systems, neural networks, communication networks and protocols, sensor networks and robotics with a major spin off in prosthetic design. The major difference between biologically inspired systems and traditional artificial intelligence (AI) is the way that biologically inspired systems evolve on their own whereas in AI, the designer is the one who sets the limits and boundaries within which a system works and derives its intelligence from [8]. Focussing on neural networks, the effectiveness of the system is dependent upon the accuracy of modelling of the basic unit, the neuron. To be able to model the artificial neurons effectively there is a need to understand the functionality of biological neurons since the notation and representation of artificial neurons is taken from understanding biological neurons.

In this chapter, neural networks are introduced along with the way they are modelled. In section 2.1 the structure of a biological neuron is introduced. This leads to section 2.2 which describes some types of artificial neurons which are applicable to thesis development. The theory of stochastic resonance is assessed in Section 2.3 to gain understanding on how the neuron functions even under a noisy environment. This makes a neuron an ideal signal processing unit for synthesis in such systems as noise is always an inherent problem. Section 2.4 highlights some application areas of neural networks.

2.2 Biological neurons

Santiago Ramón y Cajal discovered that the brain constituted of discrete units which he called neurons which is a Greek equivalent of the word nerves. These neurons were described as cells which are polarised that receive information through dendrites and transmit the information via extensions called axons [9].Though the brain is diverse in the shape and type

of neurons it consists of, the processing of information is more or less the same. The neuron has a cell body (the soma) which has a lot of branched out fibres called dendrites and axons that extend from the cell to other neurons and branches as illustrated in Figure 1 below.

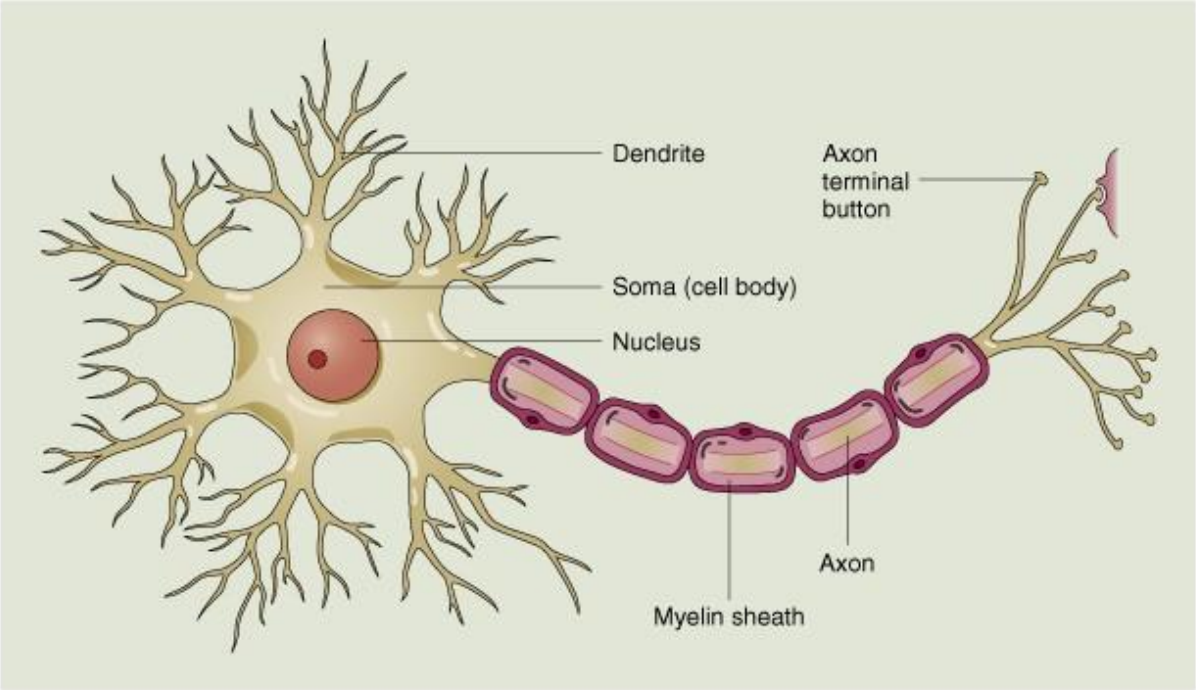


Figure 1: Basic structure of a biological neuron.[2]

The transmission of information between two neurons occurs between a pre-synaptic neuron's axons and post-synaptic neuron's dendrites [56]. On arrival of stimuli at the axon terminal, the neuron releases chemical neurotransmitters from the synaptic vesicle which open the ion channels in the cell membrane of the post-synaptic neuron, regulating the inter-neuron communication at the synapses. These neurotransmitters are bound to receptors which are in the membrane of the neuron ready to receive the signals. The receptors either excite or inhibit the neuron. The result of all these excitations and inhibitions is seen as the potential difference of the external membrane. Unexcited the membrane has been measured to have a negative resting potential of about 70 millivolts. Depending on the type of synapse the change on the neural membrane can be either positive or negative. An excitory synapse causes the potential to rise and when the potential exceeds a threshold limit, typically -55millivolts, the neuron fires or spikes. An inhibitory synapse lowers the potential making it more difficult for the neuron to fire. The type of synapse, whether it is inhibitory or excitory is

[2] Image available at http://www.thomasjwestmusic.com/neurologyapplied.htm

never altered but what changes is the intensity in potential change caused by the synapse to the membrane. The effect of the disruption in potential is only brief as the neuron will always try to stay at its resting potential state. Figure 2 illustrates a neurotransmitter and how the signals are received between two neurons.

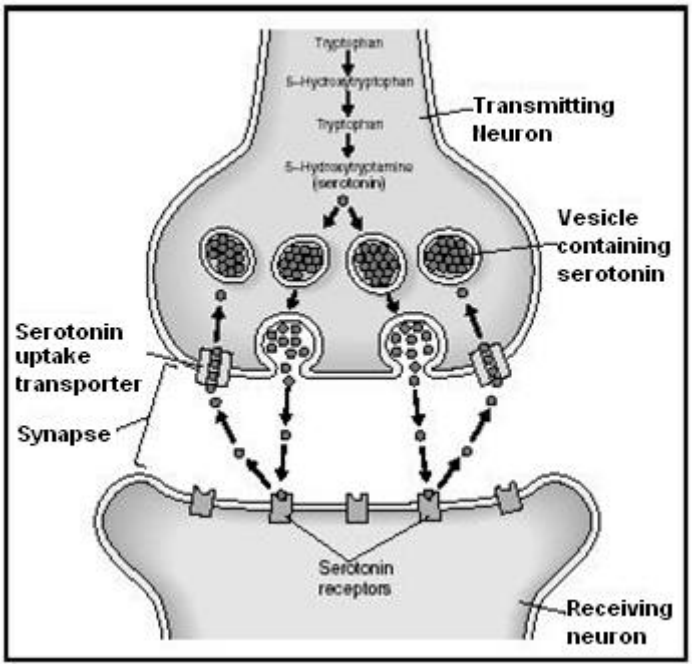


Figure 2: Illustration of the exchange in information between two neurons[3]

Information is also transmitted in the form of electrical impulses to other neurons. The average value of these action potentials is about 100 millivolts and at a frequency of 1 kilo-hertz [51]. The neuron's processing capabilities has led to engineers and scientists developing adaptive systems with tremendous learning capabilities. Of course the neurons that have been modelled thus far have been crude models owing from the fact that neurons have great complexity related to their structure. The following section describes some of the major neural computational models that have been effectively developed so far.

[3] Image available at http://www.chemistryexplained.com/Ne-Nu/Neurotransmitters.html

2.3 Neuron models

The major categories for artificial models that have been extensively studied are input-correlation neurons, weightless neurons and spiking neurons. Below are basic descriptions of artificial neural models that have been utilised successfully in neural research.

2.3.1 Input correlation neuron

From theory, artificial neurons generally classify or compute a particular function for a system given a set of inputs and outputs. To map an output to its appropriate input the neuron performs a type of correlation between its inputs, whilst adjusting internal interconnection weights through a learning algorithm. This correlation is usually a computed scalar product which depicts the measure of similarity. The correlation measure is analysed in various ways.
Firstly, as an analogy, if x_i and w_i are both the ith components of the input and the weight vectors respectively, then the scalar product is the sum of the weighted inputs given by:

$$\sum_i w_i x_i \qquad \textbf{(2.1)}$$

The correlation value in an artificial neuron is referred to as the potential of the neuron. The output of the neuron is a non linear function of the potential:

$$y = f(\sum_i w_i x_i) \qquad \textbf{(2.2)}$$

where f is typically a sigmoidal or hyperbolic tangent function used to map the input x to the output y.
Another method for measuring the correlation value is by the use of distance computations. In these algorithms distance and similarity are inversely related, where the more alike the points, the shorter the distance value and vice versa. An example of such an algorithm is the use of the Euclidean distance described by equation 2.3 which models similarity between two sets of signals, w and x.

$$\sqrt{\sum_i (x_i - w_i)^2} \qquad \textbf{(2.3)}$$

2.3.2 Weightless neurons

Weightless neurons are also known as logical or ram based neurons and do not use weights in computing their algorithms. The computation is based on the neuron learning a combinatorial mapping between a binary input n-tuple[4] and a single binary output [11]. They are implemented using random access memory (RAM). The hamming distance, which looks at the number of bits by which a pair of bit strings differs, is used as the measure of similarity to correlate the inputs to the neuron and a stored value [12]. An example of a type of neuron under this group is the generalizing random-access memory.

2.3.3 Spiking neurons

Biological neural networks use timing of a spike to encode information. A spike occurs when the neuron's membrane exceeds a threshold. The action potential of the neuron does not carry any information, since all spikes of a given neuron look alike. Therefore it is the timing and number of spikes which are critical and the action potential is the signal transmission unit. Therefore, information encoding is not dependant on the shape or size of the spike but on the time the spike occurs [13]. A mathematical model of the spiking neuron has been developed by Gerstner et al., [14] which has been used widely in neural model investigations. Networks of spiking neurons have been widely used and actually proven to be as computationally efficient as sigmoidal perceptrons. Circuitry used in testing spiking neurons has ranged from analog, digital to mixed analog systems. Spiking neurons have been noted as the third generation of neural models as a following to the McCulloch- Pitts neuron and those that have a weighted sum of inputs and activation functions [15].

The neuron used in this thesis is under this category, specifically the integrate and fire neuron where the architecture will be elaborated more in the next chapters. The structures of the neuron are based on previous work that has been proposed and carried out by Tapson et al., [45, 46]. In the model, the input can consist of a spike train (cortical neurons) or a continuous signal (sensory neurons). The membrane voltage $u(t)$ is obtained by integrating the input. A capacitance C together with an ohmic resistance R across the membrane leads to an

[4] In a relational database a tuple is one row.

exponential decay of the membrane voltage. When the membrane voltage reaches a critical value (the threshold), the neuron fires, then is reset.

2.4 Theory of stochastic resonance

Noise is viewed as being a hindrance in detecting and transferring information. However for some systems to function optimally, noise is a necessary component. Stochastic resonance is the phenomenon that occurs when a system that is nonlinear and bistable is driven by a weak periodic signal and addition of noise improves the response of the system to the input signal [16]. The theory asserts the notion that addition of noise to a system improves the system's sensitivity to detecting weak information carrying signals [57, 58, and 59].

The theory of stochastic resonance originated about 30 years ago when two groups of researchers from Rome and Brussels came up with the an idea to explain how, amongst other things, a minute change in the Earth's orbit around the Sun can result in a dramatic climate changes. Their basic idea was ignored due to lack of data. It later took two experimental demonstrations, one on a bistable electronic circuit and another on a bidirectional ring laser, for the theory to be noticed [59].

Further investigation led to discovery of the presence of noise in the interspike interval distributions of the trains of action potentials generated by a neuronal assembly. The standalone signal would be too small to cause any transition from one state to another in the system, thus only by addition of noise will the system overcome the barrier; nevertheless if the noise level is too high the system will become uncorrelated to the input signal [16]. The mechanism can best explained by a simple analogy given by Peter Hänggi in [58] of a marble in an egg carton.

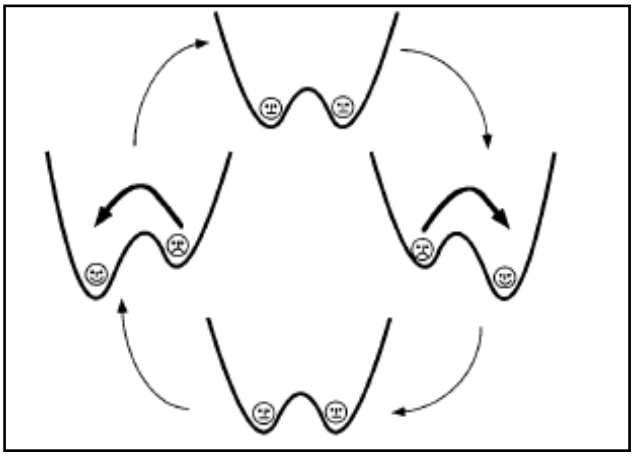


Figure 3: Demonstration of stochastic resonance acquired from [58]

Model to explain stochastic resonance: the figure depicts a ball that is sitting in one of two wells. It shows four possible states of the system. Under normal conditions, application of a small force (signal) results in no visible change in the system. However, if a suitable amount of noise is added to the system will induce hopping events between the two stable states of the system.

Consider the scenario: when a gentle force is applied to the system, it causes the marble to rock back and forth. When such a weak force is applied, the ball just rolls around in the well. If detection of the ball movement is only done when the marble jumps into a neighbouring well, this weak signal will go unnoticed, resulting in a loss of information. This can be equated to a weak information carrying signal acting on a nonlinear system. However, when noise is added to the system by tilting the egg carton randomly, under suitable conditions this will result in the ball occasionally exiting into the neighbouring well and thus being clearly detected. The theory of stochastic resonance states that these exit events do not occur completely at random, but become correlated with the weak signal. Therefore the jump events reveal a great deal of information about the time dependence of the information carrying input signal [58].

Applying it to the integrate and fire model, the stochastic resonance effect has been characterised in the Leaky Integrate and Fire model with noise on the input signal by Plesser and Geisel, Shimokawa., *et al* and Barbi., *et al* [16,17,18]. The fact that the neuron actually incorporates noise into the system to fully optimize the functional response makes it quite a useful processing tool in signal processing systems that are usually distorted by noise.

2.5 Applications using neural networks

A large amount of research has been done on Spiking Neural Networks (SNN). The research of SNNs is comparable with that of sigmoidal neural networks and thus can be divided into the same categories:

Auto association: the neural network is trained with a set of patterns which it stores by tuning its free parameters. The free parameters include weights or interspike intervals. When presented with unknown data, computation is done through association.

Pattern association: the neural network is fed a training data set consisting of pairs of input and output patterns. The network is trained with pairs of patterns and the network learns through generalized mapping between the input and output patterns. Testing is done when the network is presented with a new input-pattern, where it should produce an output consistent with the mapping.

Classification: the neural network is trained with predefined categories. Therefore, for each set of unknown data it is mapped onto the category to which it belongs.

Clustering: the neural network discovers salient features which it is able to divide into different classes for the training data. The difference is the classification patterns are not known *a priori*.

Some of the application areas which stem from the above mentioned research areas can be listed as:

- Financial: financial modeling, stock market prediction and fraud detection.
- Industrial and Manufacturing environment: process control, quality control.
- Data mining: time series analysis, classification and prediction.
- Signal processing and risk management [19].
- Medical: diagnosis of diseases, treatment cost estimation.
- Operational analysis: cash flow forecasting, scheduling optimization.
- Energy analysis: electrical load forecasting and power control systems[5].

[5] Available at: http://www.alyuda.com/products/forecaster/neural-network-applications.htm

Chapter 3: Neural network information processing

There are different methods for encoding an analog signal into spike trains. Computation is usually done on analog data streams but Spiking Neural Networks (SNNs) use spike trains as their input and output. To use SNNs effectively the analog data has to be encoded into spike trains. Research has been done to figure out the kind of coding that biological neurons require in order to appropriately represent analog data [22]. Spiking neuron models have been used as time encoding machines (TEMs) which use encoding and decoding mechanisms to represent analog information. In a TEM, the analog sensory input stimulus is represented using a sequence of action potentials [61, 62]. For an integrate and fire neuron the average rate at which the spike trains are produced is proportional to the bandwidth of the input signal [63]. Action potentials are also referred to as spikes.

According to Lazar and Toth a time encoding mechanism should satisfy the following two requirements: the encoding should be implemented as a real time asynchronous circuit and the encoding mechanism should be invertible meaning that the amplitude information can be recovered from the time sequence with accuracy [54, 61, 62, 63]. For an integrate and fire neuron the band limited stimuli can be extracted from the neural spike train at the output. The input stimulus consists of the input signal and a bias. The success of the decoding and recovery of the encoded signal is based on the knowledge of the times at which the spikes are triggered. A main application of this algorithm has been proposed as an alternative means of achieving analog to digital signal conversion [54].

The focus in this study was not aimed at digital signal reconstruction of the input stimulus to the neuron. It was rather aimed at acquiring the correlation of the input stimulus by using the information encoded by interspike interval times between the action potentials that occur from thresholding.

The coding algorithms can be divided into two main categories which are rate coding and pulse coding. This chapter presents an analysis of how the spiking neuron processes information through action potentials. Firstly an analogy of how spike coding is interpreted is presented, leading to how the accumulated spikes are analysed in the form of an Interspike

Interval Histogram. The different platforms that have been used for modelling the neural models will be discussed as well as the simulation tools that have been looked at for spiking neurons [20]. Inputs and outputs of neuronal computations are continuous signals, currents, voltages, chemical concentrations. Axons are the wires that connect the nervous system electrically, encoded as spikes amenable to active restoration and transmission over long distances [49]. However, spike encoding and decoding can distort neural signals. Encoding schemes have been analysed like integrate-and-fire encoding and Poisson encoding. Other decoding schemes are spike count decoding, where the decoded signal is proportional to number of spikes occurring in a sliding window, as well as interspike interval decoding where the decoded signal is proportional to the instantaneous spike rate as computed from the inverse of the time between successive spikes.

The integrate and fire mechanism was first proposed by Knight in 1972 [60] using a small signal approximation to analyze the encoding of analog information into a spike train. Stein *et al*., in 1972 analysed the Interspike Interval (ISI) distribution and the spike train Power Spectral Density (PSD).

3.1 Spike coding

The first two generations of neurons do not deploy pulses though their output lies between 0 and 1. The output can be seen as a normalized rate of the output of the neuron over a range of time (rate coding), where a higher rate correlates with a higher output signal. The definition of rate has been widely and successfully used in many areas particularly in sensory and motor experiments. One of the common examples is one by Adrian in 1926 where the number of spikes emitted by the receptor neuron was directly proportional to the force applied to the muscle [22]. Analysis of the spikes in rate coding is done either as a spike density, which is an average over several runs, or as a population activity which is an average over several neurons [22]. Information is encoded in the mean firing rate of the neuron. This information can be expressed in a Peri-Stimulus Time Histogram (PSTH). The histogram reports on the neural response of a neuron or neurons at a time t. For the histogram to hold valid information, the experiment has to be repeated many times under identical conditions so that the number of spikes occurring for every δt can be accumulated.

However another type of coding can be done using pulses. This utilises timing in the coding strategies [22]. The three basic strategies are:

Time -to-first-spike coding which is when conversion of an analog value to spike code is done using only one spike with the firing-time proportional to the analog value. It has been used in many studies [22, 23]. The major disadvantage of this technique is that since every neuron fires only once, this limits the capability.

Phase coding is when an analog value is sequentially encoded in one spike-time during a certain time-period, and these independent time periods are combined one after the other [22, 24]. The disadvantages are the expressive power of the spike-train is limited because there is only one spike in a specified period.

Population coding is when a stream of analog values is encoded through thresholding more than one neuron. This is when a stream of bits is seen as a spike train [25]. Data loss due to the sampling rate is a major disadvantage through thresholding as every analog value is reduced to a bit. Nonetheless the dimension-reduction results in compact spike trains which are easier to analyse. An illustration is in [22] when an analog value is transformed into a group of spike-times.

Coding method selection is dependent upon the purpose and application of the neural network. If the input consists of several data streams, it is better to use thresholding, whereas if only a few values should be encoded and the values have a high information density then a method where the information is distributed over a pool of neurons would be a good fit.

3.2 Inter-Spike Interval Histogram [ISIH]

The Interspike Interval Histogram (ISIH) is a histogram that is compiled from the spike times of the neuron or neurons. Recording is done of the spike times with respect to the time of the previous spike. The histogram can have various degrees of order. A first order ISIH is the time between the current spike and the previous spike only. In this thesis, a first order ISIH is compiled for a given period of neural firings through measurement of the time intervals between successive spikes. The time measurements are then sorted into separate bins making

the histogram. Typically in this work the ISIH is constructed from around 100000 intervals. The Probability Mass Function for the time-to-first-spike distribution is identical to an interval of the ISIH when the neuron is reset immediately after it fires [26]. It is performed with the assumption that the neural behaviour is the same throughout the test period and the input is constant or zero. Each spike interval is independent of all other spike intervals but analogous to the time the neuron fires. Thus if a neuron is stimulated by a signal dependant on time then each interval is also time dependent.

The usefulness of the interspike interval in analysing neural information and brain activity has been a subject of debate [26]. The main argument is centred on the fact that reaction times in behavioural experiments are too short to permit analysis over long periods of time which are required for the generation of the ISIH. The importance was pointed out by Adrian in 1926 and 1928 when he analysed motor control by the brain [64, 65]. Cariani and Delgutte in 1996 later went on to show how the central auditory processing mechanism appears to use the ISIH to determine pitch.

3.3 Neuron behaviour modelling

> *"Modelling is the process by which a complex phenomenon or concrete object is replaced by another entity called "the model" whose environment and operational characteristics are defined by prescribed rules, and whose behavior is taken to represent that of the concrete object or phenomenon (Regnier, 1966)."* [68]

The primary purpose of modelling then becomes that of describing reality in terms of models that are inherently coherent, and thereby order and predict the behavior of the real entity based on that of the model. In order to achieve this aim, the interaction between the real entity and its model is quantified by experiment.

Neuronal modelling is when a biological neuron is represented in a mathematical structure which fully describes its biophysical and geometrical properties. The importance of such a representation is to define the information processing and computational properties of the neuron [27]. Accurate modelling is essential in understanding the behaviour and thus predicting the required response from the neural network.

Practical modelling of the neuron has mainly been achieved on both analog and digital platforms. Concise analog models have been critically analysed over the year as they closely relate to the structure and response of a neuron [22, 55, and 56]. However, digital neural models have also gained momentum though as in any system, a trade off between accuracy and the computational requirements of the neuron have to be taken into consideration. A general description of the spiking neuron will be given, followed by an explanation of two particularly important spiking neuron models which are the Spike Response Model and the Integrate and Fire model.

3.4 Spiking Neurons

Spiking neurons are an abstraction of biological neurons which use spikes to encode information. This allows incorporating spatio-temporal information in communication and computation as real neurons do [21]. There are various schemes of how to use the spike timing information in neural computation. The thesis concentrates mainly on the integrate and fire model, but for the sake of comparison the spike response model will also be analysed. These models are under the category of threshold and fire models. The integrate and fire model is relatively simple and since it approximates the Hodgkin-Huxley model in detail, it also captures the generic properties of neural activity [22].

3.4.1 Spike Response Neuron Model

The Spike Response Model (SRM) describes how inbound spike-trains are processed to produce a new spike-train leaving the neuron. A short description of the model is given below.

A mathematical analogy of a neuron i with firing time $t_i^{(f)}$ where f is the number of the spike is given cumulatively by:

$$F_i = \left\{ t_i^{(1)}, \dots, t_i^{(n)} \right\} \quad \textbf{(3.1)}$$

where $t_i^{(n)}$is the most recent spike.

In this thesis, when a neuron emits an action potential that is the firing time of the neuron. The variable u_i refers to the membrane potential of a neuron i. When the neuron crosses the threshold value v the neuron generates a spike.

Adding the time the event occurs to F_i results in:

$$F_i = \{t | u_i(t) = v \wedge u_i'(t) > 0\} \quad \textbf{(3.2)}$$

When the neuron generates an action potential, the membrane's potential suddenly increases. After a spike is triggered, ion channels open and close; some of the ions flow through and others out of the cell membrane. The result of these ionic processes is the action potential, a sharp peak of the voltage followed by a sustained negative after-potential. The forms of the spike and after-potential are roughly the same. The membrane is lowered or reset, and mathematically this is done by adding a negative contribution to the state variable u_i, followed by a period of refractoriness. The period of refractoriness is when it is impossible for the neuron to generate another spike, as illustrated in figure 4.

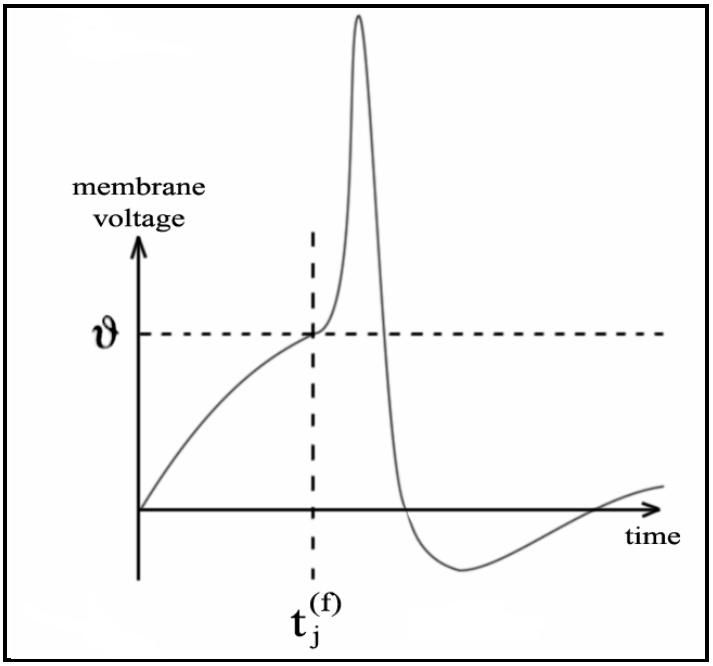


Figure 4: Action potential illustration from [20].

The absolute and negative value of this period can be modelled using kernel η:

$$\eta(s) = -n_o \exp\left(-\frac{s-\partial^{abs}}{t}\right) H(s-\partial^{abs}) - KH(s)H(\partial^{abs}-s)$$
$$H(s) = \begin{cases} 1 \; if \; s > 0 \\ 0 \; if \; s \leq 0 \end{cases} \quad \textbf{(3.3)}$$

The period of absolute refractoriness is set by ∂^{abs} where K ensures that the membrane potential is above the threshold value [20]. The constant n_o scales the negative after-potential duration. The transmission delay is defined by equation 3.4, $0 < \tau_s < \tau_m$ are time constants which define the duration of the effect of the postsynaptic potential. The effect of an excitatory postsynaptic is described by ε. The axon transmission delay is defined by Δ^{ij}.

$$\varepsilon_{ij}(s) = \left[\exp\left(-\frac{s-\Delta^{ij}}{\tau_m}\right) - \exp\left(-\frac{s-\Delta^{ij}}{\tau_s}\right)\right] H(s-\Delta^{ij}) \quad \textbf{(3.4)}$$

The variable w_{ij} is used to model the synaptic weight, which can also model inhibitory connections by using values lower than zero. This is in line with the fact that real synapses are either excitory or inhibitory. Synapses connect the output of one neuron (pre-synaptic) to the input of another neuron (post synaptic). The synapse between neuron i and the input of post-synaptic neuron j has an efficacy (weight) w_{ij} denoting the strength of the connections between the neurons. When w_{ij} = 0 then there is no connection between neurons i and j. Since spiking neurons use time in their computation, for effective computation, analog input information thus has to be converted into spikes or the membrane potential has to be altered directly [20]. Altering the membrane directly can be achieved by using an extra function h_{ext}, to describe the effect of an external influence on the membrane potential. This results in:

$$h(t) = h_{ext}(t) + \sum\sum_{j\in\Gamma_i \, t_j^{(f)}\in F_j} w_{ij}\varepsilon_{ij}\left(t - t_j^{(f)}\right) \quad \textbf{(3.5)}$$

For solutions that just require simulation, it is much handier to convert analog signals into spikes that can be fed directly into the network.

The current excitation of a neuron which combines the refractory state, incoming postsynaptic potentials and external events is illustrated by:

$$u_i(t) = \sum_{t_i^{(f)} \varepsilon F_i} n_i \left(t - t_i^{(f)}\right) + h(t) \qquad \textbf{(3.6)}$$

Equations 3.2 and 3.6 effectively describe the Spike Response Model.

3.4.2. Integrate and Fire Neuron Model

The Integrate and Fire neuron is one of the most widely used and modelled spiking neuron[20, 22]. The model is based on an electronic principle. Figure 5 shows schematic diagrams of both a real and an integrate and fire neuron. This type of neuron integrates all inputs until the neuron's potential reaches a threshold.

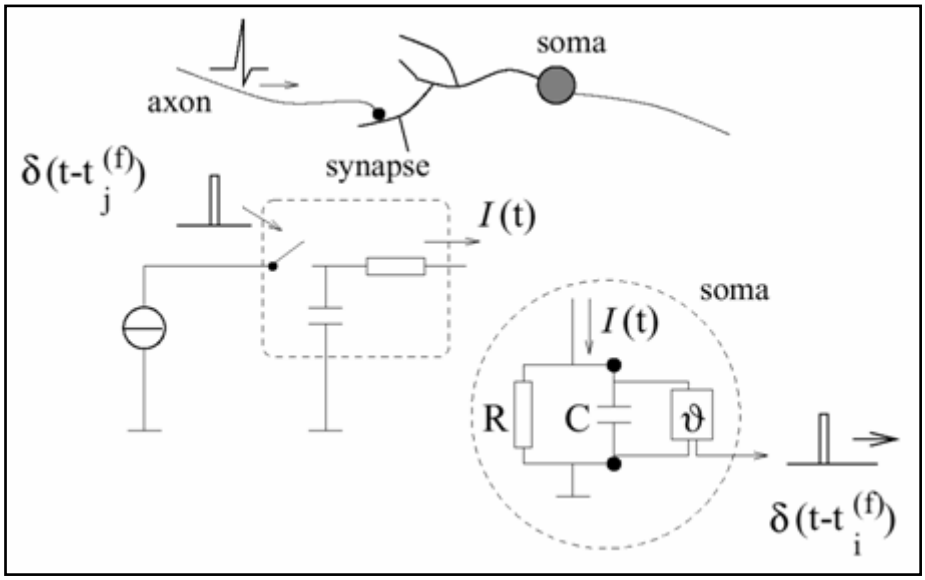


Figure 5: Schematic drawing of integrate and fire neuron.

The low pass filter on the left side transforms a spike to current pulse *I(t)* that charges the capacitor. On the right, is the analog electrical equivalent of the soma which generates a spike whenever the voltage *u* across capacitor exceeds a set threshold [20].

A brief description relates that a spike travels down the axon and is transformed by a low pass filter, which converts the short pulse into a current pulse that charges the capacitor comprising the integrate and fire circuit. This results in a voltage increase that can be seen as a postsynaptic potential. Once the voltage over the capacitor goes above threshold value v the neuron sends out a pulse.

A mathematical representation is given by equation 3.7 to describe the effects on membrane potential u over time,

$$\tau_m \frac{\partial u}{\partial m} = -u(t) + RI(t) \quad \textbf{(3.7)}$$

with τ_m being the membrane time constant in which voltage leaks away. As with the spike response model the neuron model fires once u crosses threshold v and a short pulse δ is generated. To force a refractory period after firing we set u to $k > 0$ for a period of δ^{abs}.

$$I_i(t) = \sum_{j \epsilon \Gamma_i} c_{ij} \sum_{t_j^{(f)} \epsilon F_j} \delta(t - t_j^{(f)}) \quad \textbf{(3.8)}$$

The input current I for neuron i will often be zero as incoming pulses ideally have an infinitely short length. Once a spike arrives, it is multiplied by synaptic efficacy factor c_{ij} forming the postsynaptic potential that charges the capacitor. This model is computationally simple and can easily be implemented in hardware. It is loosely linked to the more general spike response model and can be used like it by rewriting it into the correct kernels η and ε.

The digital model version of the integrate and fire neuron used in this thesis bases most of its assumptions on that used in Tapson (1998). However, the major criticism that this model faces is the fact that the behavior of the neuron is basic since it is a non-leaky integrate and fire neuron. The major advantage of the model is it is the only diffusion model that can be solved exactly for all parameter values, and amply describes synaptic integration [28]. In a digital form, this is ideal as this relates to a simple model which is easier to work with and thus offers fewer constraints. Realistic models of neurons incorporate the decay of the membrane potential between the synaptic input. The approximating process is the Wiener process which belongs to the class of Markhov diffusion processes. The Wiener process describes the perfect integrator with a positive drift value. The firing of the neuron can be viewed a first passage time of the membrane potential through a defined threshold. The spike is the point event for the neuron by which the membrane is reset. Therefore the Wiener model can be used to describe the membrane potential of a neuron whose continuous renewal process consists of the interspike interval times which follow the probability density function defined by equation 3.9. This equation is modeled in the absence of an input signal.

$$\rho(\tau) = \frac{\theta}{\sqrt{2\pi\sigma^2\tau^3}} exp\left(-\frac{(\theta - m\tau)^2}{2\sigma^2\tau}\right)$$

(3.9)

where σ^2 represents the variance of a noise parameter , τ is the interspike interval and m is the drift value.

3.5 Simulation of neural behavior

The experimental evidence that spike timing is a crucial aid to explaining neural computations motivated the use of spiking neuron models rather, than rate-based models. This has resulted in a growing number of simulation tools which allow the simulation of spiking neural networks. The tools offer the user an option of being able to obtain precise simulations of a computational paradigm in a relatively short time. The range of computational problems related to spiking neurons is very large. In some cases it requires the use of detailed biophysical representations for example the conductance based Hodgkin and Huxley model but in other cases it is not necessary to realistically capture the spike generating mechanisms thus the integrate and fire model (IF) can be used. IF models are relatively fast to simulate therefore they offer attractive properties to be used as the basic unit for large-scale network simulations.

For the simulation of neural networks there are two major algorithms; synchronous or 'clock-driven' algorithms or 'event driven' algorithms. In synchronous algorithms, the neurons are updated simultaneously at every tick of the clock. Event-driven algorithms are when the neurons are updated only when they receive or emit a spike. Hybrid algorithms also exist which incorporate both methods for different parts of the system.

Whilst synchronous algorithms can be easily coded and apply to any model, precision of the simulation can be an issue since the spike times are bound to a discrete time grid. Asynchronous algorithms are more easily applied to exact simulation for simple models.

Due to the abundant interconnections between neurons in the brain, neural dynamics are too complex to solve analytically. In computational neuroscience, prediction of the response of a neuron(s) to an input signal is very important. The model response can be compared with *in vivo* experiments to investigate information processing and representation of the neural network [26].

Running a discrete simulation and approximating the integration of the neuron is one simple alternative for determining a neuron's behaviour. For an ideal integrate and fire model, discrete behaviour of the neuron's potential is described by

$$u_{n+1} = u_n + \mu T_s + \eta_n \sqrt{\sigma^2 T_s}\,,\ u \in N_o \qquad \textbf{(3.1.1)}$$

where Ts is the time in seconds between each interval. The variable η_n is a randomly generated variable with a Gaussian distribution. In the discrete model when u_n reaches the threshold value, a spike is generated and the potential (u) gets reset to $u_0 = 0$.

A brief description of the most commonly used simulation tools will be listed below with the major advantages and disadvantages.

3.5.1 Neuron (NEURON Simulation Environment)

NEURON is an environment for simulation used for creating empirically-based models of biological neurons and neural networks. The computational engine exploits descriptive equations of neural properties. Its functions are designed for ease in controlling simulations and presenting results of familiar neuroscience concepts. The main goal is for modelers to address high-level research goals without emphasis on low-level mathematical or computational issues. So instead of users modeling in terms of equations, they can model in terms of the biophysical membrane properties, cytoplasm, synaptic communication and branched architecture of the neurons.

This is achieved by allowing users to use the 'natural syntax' to specify model properties using familiar idioms in specifying model properties. Additionally NEURON has several user-selectable numerical integration methods, namely, Euler, Crank-Nicholson method and adaptive integration, which allows for adjustment of integration order and time step as necessary. Users can switch between integration methods without the need to rewrite the whole model specification since NEURON avoids computation specific representations of biological properties [29].

NEURON programming is done with Hoc (High Order Calculator). Hoc was developed by Brian Kernighan and Rob Pike and has syntax like C. The language has been extended to address functions that are specific to domain of modeling neurons with a graphical user

interface (GUI) and object oriented programming. The default GUI can be used to create models with publication-quality results without writing code, though a powerful and flexible strategy would be to combine the GUI and hoc programming. NEURON is open source and runs under Windows, UNIX, Linux and parallel clusters like IBM Blue Gene and the Cray XT3 [29].

3.5.2 GENESIS, (the GEneral NEural SImulation System)

GENESIS was designed by Dr. James M. Bower at Caltech as a general software platform that realistically models biological neural systems ranging from sub cellular components, complex models and large networks. Based on the assumption that in order to gain a deeper understanding of neural networks, users need to have the ability to base computer models of the nervous system on its actual anatomy and physiology. The object oriented and high level simulation programming approach allows users to modify and reuse model components as required and achieve this goal [31].

Some of GENESIS's other major aims are for the model to rely less on a particular operating system, include a GUI that supports a wide range of computer expertise, use the software for educational courses and commit resources to user support and online tutorials and handbooks. It also can easily be adapted for use in parallel computing. GENESIS was written in C but it is being re-implemented in C++. Operating system environments in which it can run are Linux, Macintosh with OS/X and Microsoft Windows with Cygwin environment. GENESIS is open source.

3.5.3 Neural Simulation Tool (NEST)

NEST was founded in 2001 to collaborate development of new simulation methods for biologically realistic spiking neural networks best suited for large network simulations. NESTmakes it possible to combine different types of neural models to build one network. It was aimed at further development of analysis and visualization tools. Another goal was to enhance collection of information and resources related to neural simulations and share results through publications and releases of simulation tools in the scientific community [31, 32].

The major difference of NEST from GENESIS or NEURON is that NEST represents the neuron in a detailed complex neuronal geometry way. This is because the main focus of GENESIS is on the dynamics and behaviour of large neural networks. NEST is also a research tool which users can add network nodes, write libraries of programs, routines and extend the instruction set. It is a command-line application. At the end of each simulation the results are stored as a set of data files which can be post-processed using Matlab, IDL or Mathematica.

The major difference is that NEST has no GUI since the origins were in 1993 when GUIs were still volatile and platform dependent. NEST runs on various operating systems with an ISO 98 conforming C++ compiler adapting to multiple processors. The accuracy of NEST has been enhanced by not having a global equation solver for the network thus each neural node updates its state from its algorithms. For a large network, NEST uses Exact Integration for pulse coupled neuronal systems. The integration step is chosen independent of the size of the simulation step, thus making it possible to solve equations with an adaptive step size. NEST is also open source.

These are all very important simulation tools which have been widely used in spiking neuron research. Some of the publications with results obtained from the engines will be referred to in this thesis.

Chapter 4: Basics of Radio Astronomy

4.1 Introduction

Astronomy is an intensive study of the universe traditionally based on observations at optical wavelengths of electromagnetic radiation which range from 400 nanometers to 700 nanometers. However, bodies emit radiation that is not only limited to the optical range. In this way Astronomy can be classified according to the media that is used in the analysis and the area being studied. Radio astronomy is the study of the universe using radio frequency signals emitted from cosmic objects as a result of physical processes. Radio astronomy is in the wavelength range of approximately 1 millimeter (300 GHz) to 10 meters (30 MHz). Karl Jansky pioneered radio astronomy in 1932 when he discovered that the galaxy was emitting signals at 20.5 MHz. From this discovery Grote Reber went on to build the first steerable radio telescope and receivers to observe cosmic signals. Detection and mapping of the radio waves detected from radio sources in the universe is done using radio receivers [33].

This chapter takes a look at what a basic radio astronomy signal processing system comprises of and, offers an overview of the current correlator structures under construction and use. This chapter serves to justify why there is need for continual research on correlator algorithms which are both computationally accurate and yet use fewer resources especially for systems like the SKA[6].

4.2 A simple radio astronomy signal processing system

Astronomy requires precise mapping and this is accomplished by a computationally efficient signal processing system. The basic components of the signal processing system are an antenna, receiver and the backend which is a data processing system. Collectively the components are referred to as a radio telescope. Figure 6 shows how the components are interconnected in the system.

[6] Square Kilometre Array (SKA) is an array of telescopes where South Africa has been shortlisted as one of the bidders to host the array. Financial aid of this thesis has been from the SKA. Further information is found at http://www.ska.ac.za/index.php

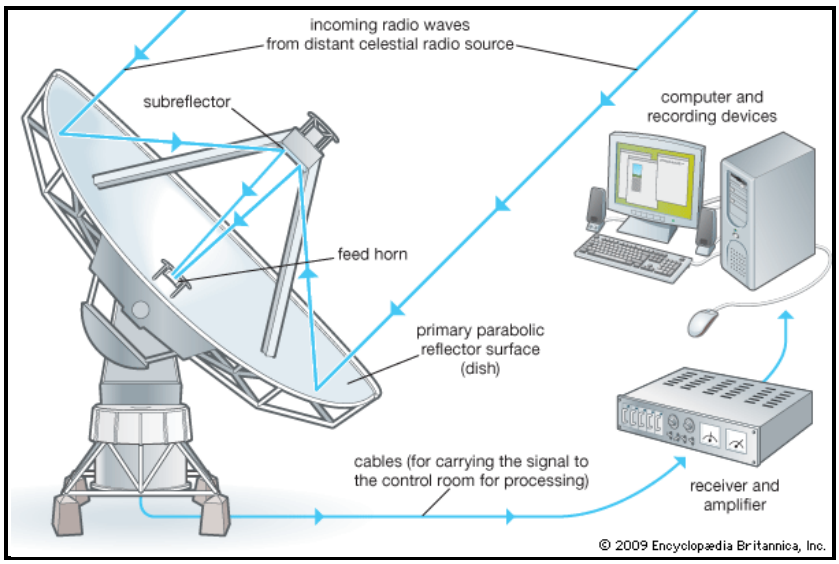


Figure 6: Telescope system[7]

Generally there are three main categories of radio telescope layouts [33].

Type 1: In a transit telescope, the antenna only moves in one axis, up or down along the meridian. The telescope points to the declination of the observed source and waits until the motion of the stars in the sky brings the source in front of the telescope. Unfortunately only one observation per day is possible of each source. This is a time wasting method and requires many hours of observation in order to correctly map an observed source. However, the telescope is relatively cheap to build.

Type 2: the antenna can point to any place in the sky, following the daily apparent path of a source in the sky for about 10 hours. The telescope can be operated manually or automatically. Fully steerable single dishes come in many different sizes. They range from small approximately 10 metre diameter dishes up to the large 100 metre diameter class dishes. The problem with this type of design just lies in resolution as analysis is being done by one instrument and in the event of a fault, observation is at a standstill.

Type 3: has two or more antennas separated by a known measured distance (base line) installed in an array. This technique, known as interferometry, was first developed at optical

[7] Available at: http://www.britannica.com/EBchecked/topic-art/488967/107380/Radio-telescope-system

wavelengths by Michelson in 1890. Interferometry was later developed for radio astronomy to allow multiple dishes to be synthesized into a single observing instrument. The antenna signals are combined taking into consideration the phase difference between the signals received by the antennas thus obtaining the performance of a large antenna, the size of the total area covered by all the antennas. Each of the telescopes in the array points to the radio-signal-emitting celestial source under observation. A time difference occurs between the signals that reach the telescopes in the array but compensation is made using a reference coordinate in the system. The major advantage of such a system is the attainment of better resolution and a narrower beam width, however acquiring free space to mount the antennas in radio quiet zones is quite a delicate and time consuming process. This configuration is the one on which the SKA is being built and based on.

Compensation of the phase difference is done in algorithms via the backend and software algorithms. The separate data from the different telescopes in the array must be combined and that is usually done by correlation of the signals from the antennas in pairs. This is where the correlator is essential in the system. The thesis is mainly centered on signal processing, thus an explanation of the backend will be further outlined.

4.2.1 Overview

The backend is mainly responsible for capturing and storing the data for later analysis. The backend has three main segments; receiver, detector and the data processing unit. Components of the each subsystem depend on the requirements of the whole system. Signals studied in radio astronomy are very weak where a strong source is approximately 1 Jansky[8]. Analog electronics is used in telescopes to amplify the signals so that they may be readily processed. The receiver and amplifier boost and condition a weak signal to a measurable level [33]. The data processing unit is usually a computer that post processes and records the signal results.

In previous astronomy data-gathering systems, after being filtered, down-converted, sampled and quantized, the data would have been recorded on magnetic tapes [34] as exemplified by the Mark I, II, III, IV and S2 systems. The Mark IV correlator is used to correlate data from the European VLBI Network (EVN); the Canadian S2 correlator is used primarily for analysis of

[8] Jansky is equivalent to 10^{-26} watts per square meter of receiving area per hertz (W/m^2Hz)

geodetic data and supporting space VLBI observations using the S2 recording format; and the S2 correlator of the Australia Telescope National Facility (ATNF) is used for processing data from its Long Baseline Array. The tapes from each telescope would be shipped to a correlator that is purposely built computer for digital signal processing.

The correlator is based on a parallel algorithm, since the correlation operation can be divided into time and/or frequency. When the SKA was proposed, a major technical obstacle to its feasibility was the cost and power consumption of the correlator. The number of iterations, multiplications and memory requirements necessitate a great deal of electronic circuitry and space, resulting in overly expensive systems. This led to different proposals and justifications for various correlator designs.

4.2.2 Mathematical Analysis

Correlation is a measure of similarity. Cross correlation is used to determine the time delay between a signal arriving at two spatially separate sensors, and plays a vital part for correct assembling of images in radio astronomy. The purpose of the correlator is to align the recorded data streams; correct effects caused by instruments and geometry; and coherently combine the data from different pairs of radio telescopes to produce sets of complex visibilities. Visibilities are a measure of how the radio source is distributed across the sky [35]. They are a function of the antenna pair, polarization, frequency and time. Once stored, visibilities are later used for imaging, spectroscopy and astrometry [36]. Quantisation from the analog to digital converter, spectral response and the delay errors cause imperfection in visibilities.
A simple illustration of a baseline between two antennas is shown below:

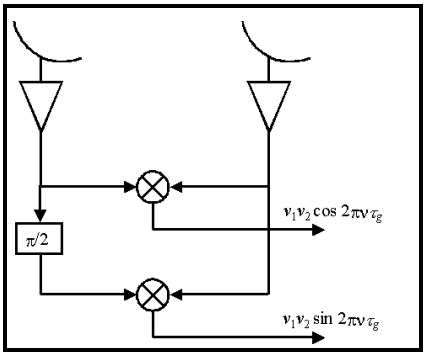


Figure 7: Baseline between two antennas

The cross correlation of the voltages from two spatially separated antennas can be represented by:

$$x_{ij} = \int V_i V_j^* \, dt \tag{4.1}$$

where the complex voltages from antennas i and j are V_i and V_j respectively.

Since the signal reaches the two antennas at different times this is compensated for by adding a delay to one of the signals. $V_j(t)$ is delayed by a time τ, and the correlation equation becomes equation 4.2, where the correlation function can also be used to find time delays in two separate waveforms .

$$x_{ij}(t) = \int V_i(t) V_j^*(t + \tau) \, dt \tag{4.2}$$

This result can also be referred to as a visibility. Further analysis which takes into consideration the oscillator frequency of the correlator leads to equation 4.3.

$$x_{ij}(t) = \int V_i(t) V_j^*\left(t + t_{ij}\right) dt = \int A_i(t) A_j^*\left(t + t_{ij}\right) \exp\left(i 2\pi n_o \tau_{ij}\right) \tag{4.3}$$

τ_{ij} is the geometric delay for baseline i - j, n_o is the local oscillator frequency and $A_i(t)$ is the slowly varying amplitude. The exponential term in equation 4.3 represents a time variation that can be calculated from geometrical considerations and removed in software.

4.3 Types of Astronomy Correlators

Astronomy correlators range from application specific analog, digital and software correlators. Even though the technology of the correlators differs, most of them are based on the same principles. Currently the two major classes of spectral line correlators utilized in radio astronomy are FX and XF (or lag correlator). Example systems that use these configurations are the VLBA and the VLA respectively.

Combination of the recorded signals requires two fundamental operations; a Fourier transform (F) and a cross multiplication (X). The order of the operations is interchangeable to obtain the same result, thus the notation of FX or XF correlator architectures [35]. The algorithms encompass taking the Fast Fourier Transform (FFT) of the data coming from each sensor, conjugating and multiplying the data point by point, and then inverse transforming. Both algorithms require great computation and information storage requirements. The degree of accuracy required in the system usually determines the computational needs and cost of the system.

Traditionally correlators were built using the XF architecture. Systems which demonstrate this architecture are namely the Very Large Array (VLA), Berkeley–Illinois–Maryland Association Array (BIMA) and the first version at Atacama Large Millimeter Array (ALMA). In 1989 D'Addario proposed the name since it directly measures the cross correlation function [6].

4.3.1 XF Correlator

In an XF correlator, the signal is delayed in unit steps and the correlations are done simultaneously (X part) and then the Fourier transform (F part) is taken on these correlations. This is to get the visibilities as a function of frequency. An example of the XF correlator is shown in figure 8 below. Each frequency channel requires one multiplier and accumulator for complex data. A delay nD is necessary to allow the measurement of a two sided cross correlation function.

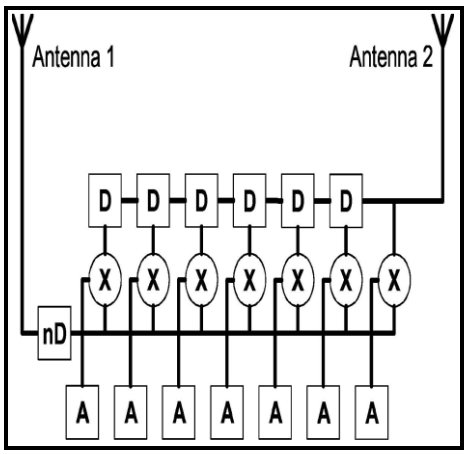


Figure 8: Basic XF correlator.

D is a delay, X is a multiplier and A is an accumulator.

Some advantages of the XF correlator encompass the fact that the correlator algorithm uses 2 bit sampling, meaning there is a degree of minimisation of the data transmission costs. This reduces the signal to noise ratio, however, with an increase in bandwidth the balance is offset. The algorithm also has a high resolution but marginally fewer computational advantages, when compared to the FX correlator.

4.3.2 FX Correlator

The FX correlator, as proposed by Chikada and D'Addario [6, 53], derives its name from the fact that the frequency transformation (F) precedes the cross multiply function (X). Incoming data from each antenna is transformed into the frequency domain then the cross power spectrum is derived from individual cross power values for each spectral channel. The FX correlator measures the cross power spectrum instead of measuring the cross correlation. This reduces the computational load as only one complex multiplier and adder is required per baseline regardless of the number of frequency channels. An illustration of the algorithm is given in figure 9.

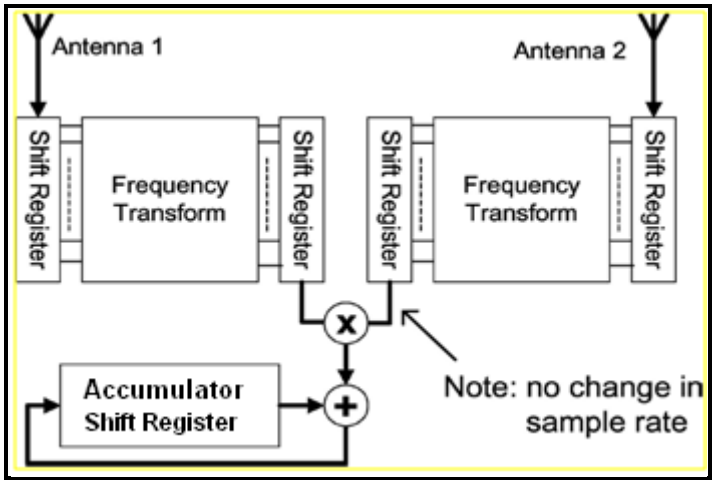


Figure 9: Illustration of FX correlator architecture [42].

In the past, the Fast Fourier Transform (FFT) has been used to perform the frequency transformation. The number of input samples is equal to the number of output samples since the FFT is an isomorphism. However since each frequency channel requires a register to store the correlations, the memory requirement is still the same as that required by the XF correlator but instead of registers the FX correlator can utilise RAM.

A major advantage of the FX correlator is that divisions of the spectrum into frequency bands causes a reduction in the number of cross multiply and accumulate banks required in the system. To validate this point consider a XF correlator with input data sampled at a rate of S with useable bandwidth S. If the frequency resolution is $\frac{S}{n}$, where n is the number of frequency bands, then for an XF correlator this totals to $k \cdot n$ lags to be correlated [42], where k[9] is an implementation dependent term that is approximately one for a rectangular window and 1.7 for a blackman window. For an FX correlator, with n equal frequency bands, each band is sampled at a rate of $\frac{S}{n}$, with the bands being wide. A single XMAC is required for each of the frequency bands, resulting in a reduction in the rate at which the XMAC operates by a factor of n **.** A comparison of the results is shown below [42]:

	Number frequency band	Data rate per frequency band	Total Data Rate	Lags per frequency band	XMAC ops/sec one baseline
XF correlator	1	S	S	k.n	k.n.S
FX correlator	n	S/n	S	1	S

Table 1: Comparison of XMAC processing of FX and XF correlators

FX correlators are unpopular due to the fact that they carry a high interconnection cost owed to bit and data rate increase, a high signal to noise ratio and high implementation costs. The implementation costs rise due to the need for frequency transformation of the input signal as the architecture requires custom chips for the filter banks. They are also more complex than XF correlators because of the need for real time filter banks [38].

Both algorithms have been implemented in a range of technology and on different platforms. Below is a description of the various categories of correlators that have been used and are currently in use in the Radio astronomy arena. Most of the correlator designs are centred around the XF or FX correlation schemes. Each category will be presented and a review of their general advantages and disadvantages will be given. Most of the platforms described are based on highly application specific purpose-built integrated circuits.

[9] In signal processing a window function is a function whose value is zero outside a chosen interval. It is also referred to as an apodization or tapering function.

4.4 Analog correlators

Analog correlators have been used in wideband correlation, cosmic background imaging, continuum correlation and electro-optic correlation. The operation principle can be coherent or incoherent. The coherent principle applies to complex cross correlation of two orthogonal polarizations, with baseline set at zero [37].

Since analog correlators use analog components they have the distinct advantage of offering low complexity and low power consumption due to inexpensive, compact and simple system configurations. The advantage of compact systems is that the resultant correlator circuit can easily be mounted on the telescope close to the receiver and spectrometer. Comparing analog correlators with acousto-optical spectrometers and digital correlators, analog correlators have wider bandwidths and vastly decrease the correlator's power consumption and complexity. Compared to filter-bank spectrometers, analog correlator circuits are printed or soldered on to circuit boards reducing the need for high-tolerance machining. Delays and multiplication functions in analog correlators do not degrade the signal to noise ratio with analog to digital conversion. Nevertheless some of the disadvantages of the sampling time delay are that the sample is not precise which results in each sample having its own frequency response. Furthermore, upgrades after design completion are typically costly and impractical. This does not significantly affect the spectrometer's performance but adds more work for the data reduction software to carry out.

4.5 Digital correlators

In digital correlators, FPGAs have been used in the last 20 years. The flexibility is greatly illustrated in the VLBA (Very Long Baseline Array). The structure offers more flexibility than application-specific integrated circuits (ASICs) and yet still highly efficient. Other research has been done with the implementation of a digital FX correlator on BEE2 hardware, i.e BEE2, iBOB, iBOB-ADC board [40, 41]. Spectral data for each antenna is computed using a polyphase filter bank design and each X engine is responsible for processing baseline data for a single frequency. The advantage of the dedicated frequency architecture is that it requires the same number of modules as antennas as opposed to dedicated baseline modules. Digital correlators are more flexible since they are reprogrammable, modular and easily upgradeable. This gives room for maximum scalability by sharing computational resources along both axes of the

correlator matrix calculations. This is because most digital correlation schemes are based on the FPGA [39]. This adds a further advantage of platform independent algorithms which have reusable libraries and ease of expansion. Digital correlators are not limited by the number of inputs as this would only require adding more hardware platforms. Significant examples which illustrate the advantages of digital correlators are one developed by University of California Berkeley's Radio Astronomy Lab, where the developed module has parameters for the computation and accumulation for FX correlation architecture. Another demonstrator project is an FX correlator for a dipole antenna array that incorporates 32 stations and gives full Stokes parameters developed by UC Berkeley's Don Backer and NRAO's Rich Bradley. The demonstrator focuses on crucial technologies and problem areas for correlator development. Digital correlator platforms include the Allen Telescope Array (ATA), Combined Array for Research in Millimeter-wave Astronomy (CARMA), Square Kilometer Array (SKA) and Epoch of Reionization (EoR) array.

4.6 Software Correlators

Software correlators are usually general purpose computers running special purpose software based on the XF or FX architecture. The circuits that make up hardware correlators are replaced with subroutines written in high-level programming languages such as C++.

This allows correlators to be built quickly, since an intimate knowledge of the underlying hardware is not required. Because of the ease of programming, it also allows more features and flexibility than is typically feasible for a hardware correlator. Precision is higher as there is no wiring involved in the circuitry, thus fewer compromises in accuracy. Software algorithms are extremely flexible as there is no limitation on spectral resolution, time resolution or even the number of inputs to the system. Flexibility is further reinforced as scalability is just a matter of upgrading the computer cluster on which the correlator is running. Furthermore debugging and special modes are a lot simpler to implement in software compared to hardware. This is a luxury not available to systems requiring significant (re-)development between architectures, such as graphics-processing units (GPUs) or FPGAs. The whole system is relatively cheaper to develop compared to hardware correlators. However compared to an equivalent hardware correlator, a software correlator is generally more power hungry and more expensive in terms of processing power.

The most widely used software correlator at present is the distributed-FX (DiFX) written in C++. It uses a user-specified vector-arithmetic library for the majority of computations. An example of some of the libraries is the Intel R Performance Primitives. Software correlation has also been widely used in Japanese systems e.g the Japanese VLBI arrays and another software correlator was under construction for the European VLBI network [40]. Other examples of software correlators include Astro Space Center (Russia) which is based on the recording system of the S2 correlator. Furthermore at the University of California, Berkeley a software correlator was developed by Harp in 2002 to cross-correlate data from the Rapid Prototyping Array (RPA). The RPA is the system that acts as a testing platform to improve and test technology and techniques that can be used on the Allen Telescope Array (ATA) [40].

Additionally, hybrid systems exist, a mixture of software correlation and hardware. This system is being suggested for monitoring the gradual descent of the Huygens probe into Titan's[10] atmosphere. The hardware section is the Jive correlator which will monitor the phase closure and descent of the VLBI network after the data is recorded on the Mark V. The Mark V is a disk based system.

4.7 Neural networks in astronomy

Correlation is basically undertaken in the same way across various algorithms, applications and fields. The modeling equations may require different variables for the systems, but the basic correlation function does not alter, thus this research looks at two major things. Firstly, modeling a spiking neuron in reconfigurable hardware and secondly, deriving a correlation function from the modeled neuron that is reusable in any environment like radio astronomy. A correlator architecture which models the correlation function has been identified in past literature using a neural model of the 'integrate and fire' neuron [46]. The correlation function should also show the basic waveform characteristics of the input signals into the system.

[10] Titan is Saturn's largest moon.

Chapter 5: Circuit implementation of the integrate and fire neuron

5.1 Introduction

In 1959, Licklider proposed that autocorrelation can be found in the functional neural response of spiking neurons [43]. This later produced research which yielded the observation that the Interspike Interval Histogram (ISIH)[11] exhibited the same profile as the autocorrelation function of the input signal. As cross and auto correlation are important mathematical operations in most signal processing systems, a relatively low power alternative that is not computationally intensive has been the main aim of most research [44, 46]. The major advantage of using neurons lies in the theory of stochastic resonance where even in noisy environments, single neurons are capable of performing the autocorrelation on an incoming signal as observed from the output of a single IF neuron [43].

This chapter looks at the practical analog circuit design including the mathematical analysis. The FPGA modules are fully explained including an illustration of the interconnections using a register transfer level schematic. The analog to digital conversion algorithm and the PC interface are described.

5.2 Derivation of the correlation function from the practical circuit

Different correlation engines have been proposed in literature. This is an explanation of the neural engines proposed by Tapson et al, [4, 44, 45, and 46].

$$R_{xy}(\tau) = \frac{1}{T}\int_0^T x(t+\tau)y(t)\,dt \qquad \textbf{(5.1)}$$

$R_{xy}(t)$ refers to the mathematical cross correlation function between two input signals x and y.

If x and y are two real periodic signals $x(t) + y(t)$ with period T then the autocorrelation function is found by computing the following set of equations:

[11] Explanation of the ISIH is given in Chapter 3 of this thesis.

$$R_{(x+y)(x+y)}(\tau) = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}}\left[\left(x(t)+y(t)\right)\cdot\left(x(t+\tau)+y(t+\tau)\right)\right]\,dt$$

$$= \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}}[x(t)x(t+\tau)+x(t)y(t+\tau)+y(t)x(t+\tau)+y(t)y(t+\tau)]\,dt$$

$$= R_{xx}(\tau)+R_{xy}(\tau)+R_{yx}(\tau)+R_{yy}(\tau)\,dt \qquad \textbf{(5.2)}$$

The autocorrelation and cross-correlation functions are expressed in the output, but for uncorrelated input signals the cross terms are zero.

With relation to the spiking neuron and how it brings out the correlation function it helps to first analyse the membrane potential and how it reacts to stimuli. The integrate and fire neuron can be modelled as having a potential v_1 which is cumulatively the sum of the input signal, drift and noise as shown by equation 5.3. For another input signal, $y(t)$, it is a matter of substituting $x(t)$ in the equation:

$$v_1(t) = \int_{t_0}^{t} m + x(t) + \zeta(t)\,dt \qquad \textbf{(5.3)}$$

where m is the constant drift parameter which represents the net difference between polarising and depolarising the input, and which determines the free running rate of neuron when noise is absent in the system. $\zeta(t)$ represents a noise process in the system. The noise added to the system can be additive white Gaussian noise with a standard deviation of σ and mean at zero. At rest the neuron is considered to have a potential of zero, firing and resetting once the threshold becomes or exceeds a value θ. For an explanation of how the correlation function is derived from the spiking intervals of the neuron, consider a non leaky integrate and fire neuron with two signals $x(t)$ and $y(t)$ which are to be correlated. When the circuit described in [46] is integrating the potential v associated with $x(t)$, the integration period is distributed in length with a probability density function ρ_{sx} which has some dependency on $x(t)$. Similarly the intervals described by the probability density ρ_{sy} are dependant on $y(t)$ and pertain to the potential v is associated with $y(t)$ Once the x interval ends it triggers the y interval and vice versa [4].

This results in an interval distribution of the spike interval being easily modelled as a probability density function. In the absence of a driving signal the neuron interspike interval

density can be expressed as a Wiener process with a probability density function as shown in 5.4.

$$\rho(\tau) = \frac{\theta}{\sqrt{2\pi\sigma^2\tau^3}} exp\left(-\frac{(\theta - m\tau)^2}{2\sigma^2\tau}\right) \quad \textbf{(5.4)}$$

where τ is the time since integration of the current signal has started.

When a signal is present at the neural input a simple linear approximation can be used instead, which has been analysed in [55]. The analysis works on the assumption that the firing times of neurons form a first-order non-homogenous Markov process where the invariance in time, illustrated by the Wiener process, is broken by addition of signals to the neural input. The firing events, modulated by the signal, are dependent on an absolute time t, since the signals are continuous. In equation 5.3 the effect of the input signal $x(t)$ is to modulate the membrane potential as it reaches the threshold thus the change in the membrane potential with time can be referred to as:

$$\frac{d_{v1}}{dt}\Big|v_{1}{}_{\rightarrow 0} = m + x(t) + \zeta(t) \quad \textbf{(5.5)}$$

Continuing with this assumption, a positive signal represents a higher probability of the membrane potential crossing the threshold as the slope is steep, whilst a negative slope prevents threshold crossings altogether [46]. Thus the conditional ISI density for a neuron, say Integrate and Fire Neuron 1 (IFN_1) which starts integrating at a time t_0 , driven by signal $x(t)$ gives

$$\rho_{c1}(\tau|t_0) = \rho(\tau)(1 + wx(\tau + t_0)) \quad \textbf{(5.6)}$$

with w being a constant weight whose value can optimally be found by using Monte-Carlo methods. Deriving a stationary spike density for the neural circuit based on the conditional interspike interval densities involves looking at any given spike train between the input signals , where the spike train t alternates between the two input signals $x(t)$ and $y(t)$ as given by:

$$t = [t_1(IFN_1), t_2(IFN_2), t_3(IFN_1) \ldots t_n \quad \textbf{(5.7)}$$

The timeline can be explained better by the timeline diagram below.

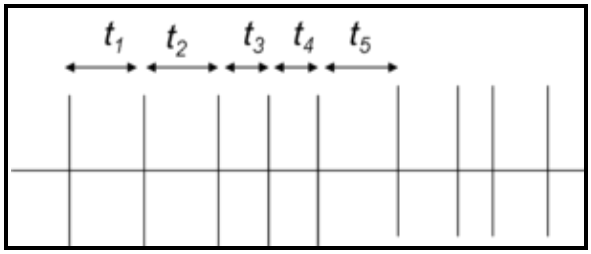


Figure 10: Timeline showing time of spike events to better illustrate how the interspike intervals are measured. *The figure illustrates how the different times are calculated from the neuron events.*

The spike train density obtained for say IFN_2 is given by

$$Y_n(t) = \rho_{C1}(t_1 - t_0|t_0)\rho_{c2}(t_2 - t_1|t_1)\rho_{c1}(t_3 - t_2|t_2) \dots$$
$$= \prod_{k=1}^{n} \rho_{cmod(k,2)}(t_k - t_{k-1}|t_{k-1}). \quad \textbf{(5.8)}$$

Since the spike interval density that is of interest is the first order spike interval density, $Z(t)$, taken from only the most recent spikes and marginalizing the values prior to the spike interval in equation 5.8, this results in:

$$Z(t) = q_{t2}(IFN_2) - t_1(IFN_1)(\tau) \quad \textbf{(5.9)}$$

Which refers to the probability that in the sequence described by 5.7, t_2 and t_1 are separated by τ.

Further analyzing the sequence results in

$$Z(t) = \int_0^\infty \rho_{c2}(t_1 + \tau|t_1)\rho_{c1}(t_1|t_0)\, dt_1$$
$$= \int_0^\infty \rho(\tau)\rho(t_1 - t_0)\big(1 + wy(t_1 + \tau)\big)\big(1 + wx(t_1)\big)dt_1$$
$$= \int_0^\infty \rho(\tau)\rho(t_1 - t_0)dt_1 + w\int_0^\infty \rho(\tau)\rho(t_1 - t_0)y(t_1 + \tau)dt_1$$

$$+w\int_0^\infty \rho(\tau)\rho(t_1-t_0)x(t_1)dt_1 + w^2\int_0^\infty \rho(\tau)\rho(t_1-t_0)y(t_1+\tau)\,x(t_1)dt_1$$

$$\approx \rho(\tau) + w^2\rho(\tau)\int_0^\infty y(t_1+\tau)x(t_1)dt_1$$

$$= \rho(\tau)\Big(1 + w^2 R_{xy}(\tau)\Big). \qquad \textbf{(5.1.1)}$$

As can be easily noticed from the derivation the cross correlation function $R_{xy}(\tau)$ is evident in equation 5.1.1 above can be extracted for the correlation engine. This correlation function denotes the input signal $x(t)$ relative to $y(t)$ If the derivation was repeated for $q_{t3}(IFN_1) - t_2(IFN_2)(\tau)$ then the cross correlation of the input signal $y(t)$ relative to $x(t)$, $R_{yx}(\tau)$ instead would have been found, resulting in the sign of the delay term being inverted. The probabilities $qt_2(IFN_2) - t_1(IFN_1)(\tau)$ and $q_3(IFN_1) - t_2(IFN_2)(\tau)$ represent the interspike interval densities for the neurons IFN_1 and IFN_2.

The resultant correlation function $R_{xy}(\tau)$, whether in a neural correlation engine or in the Radio astronomy signal, does not change, thus use of the neural correlation engine is theoretically feasible.

5.3 Integrate and fire neuron analogue circuit neuron design

The two neuron cross correlation engine, [44, 46], extracts the cross correlation function between two signals. The two neurons are cross coupled like a digital flip flop using half centre oscillators. However, the silicon implemented circuit algorithm is restricted to similar signals. When one neuron is integrating the other is held into an inhibited state until the other neuron reaches a threshold and emits a spike event. The signal that is sent from the spike event activates the other neuron to start integrating. The structure and resultant waveforms are shown in figure 11.

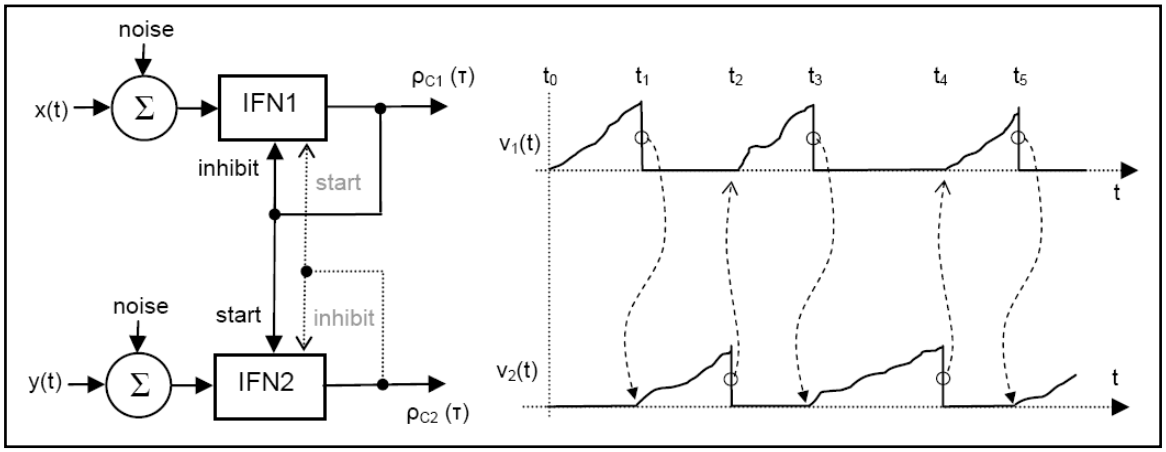


Figure 11: Circuit diagram of a two-neuron cross correlation engine.

This particular circuit design is meant to operate in a regime where the input signals, $x(t)$ and $y(t)$, are small as compared to the firing threshold limit and the noise variance must have an amplitude greater than the input signals. These conditions are necessary in order to guarantee that the neuron does not enter a state where it phase locks its firing with the input signal and so that the firing intervals span over a wide time range.

In this engine, the two neurons operate alternately such that when one fires, it inhibits itself and activates the other. An ISIH is compiled separately for each neuron, and the mathematical analysis has been done and described in the previous section.

The two neurons are connected to a processing unit like a microprocessor or FPGA which accumulates the interspike intervals. One neuron receives the reference signal and the other neuron receives the unknown signal. In the accumulated histogram, the reference signal's intervals are subtracted whilst the other input signal neuron's intervals are added to the accumulator. In order to prevent phase locking, noise is added into the system [44, 46]. The cross correlation function in the circuit is found by firstly compilation of a histogram of intervals between the spikes of the two neurons. Once the interspike interval histograms are acquired the correlation function is obtained by subtracting the two subsequent interspike interval distribution values. Another method in [46] suggests that the histogram is simulated from concurrent addition and subtraction of the bins accumulating the interspike interval count. Whilst one neuron, driven by an input signal, adds to the histogram the spike counts, the other neuron, driven by the other signal, subtracts from the accumulation bins. With regards to the layout of the data acquisition system, the first method was utilised for results acquired in this thesis and only one digital spiking neuron was used for both signals that were being cross correlated. Each of the neurons was simulated by its own input signal.

A further improvement to the circuit above is instead of using two different neurons, only one neuron is utilised for correlating two input signals. The neuron uses a switch to activate and switch between the two input signals. This means that only one neural base unit circuit is required [47]. The processing unit algorithm stores the resulting ISIHs from the neuron and processes them accordingly to acquire the required correlation function.

Figure 12 shows the analog circuit representation of the basic integrate and fire neuron unit in the correlation engine. The neuron sums the signal, noise and drift terms at its input. For an input signal $x(t)$, $n_x(t)$ and m_x are the additive noise and drift signals respectively. The resultant summed signal is integrated over a period of time. Once the threshold is exceeded, a comparator generates a reset signal. The comparator is represented by a Schmitt inverter. This signal resets the integration output to zero, switches the integrator input alternately to the other source, captures and resets the value of the counter, indicating the event to an accumulator.

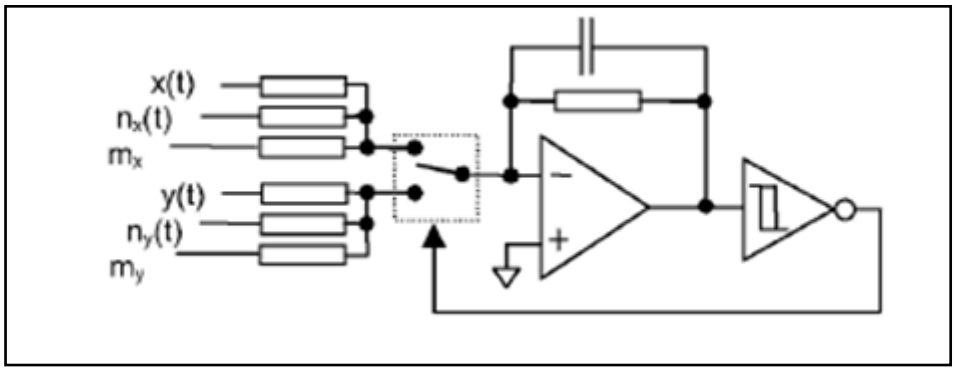


Figure 12: Analog structure of basic unit of integrate and fire neuron

This figure illustrates a single integrate and fire neuron unit. The neuron has a cumulative input which consists of the drift signal, noise and input signal. The integrator represents the membrane potential. When the potential exceeds a preset threshold, the Schmitt inverter generates a reset signal known as a spike. The reset signal is responsible for resetting the integration output to zero, capturing and resetting the value of the running counter and indicating the event to an accumulator. The accumulator and counter are run on a digital platform like a Microprocessor or FPGA. The counter is responsible for computing the interspike interval times. This particular design uses a single neural unit to alternate between two input signals using a switch. Once the membrane potential is reset for one input signal the switch crosses over to the other input signal and the process continues. The difference is in the values that are added to the interspike interval histogram, as the spike events of one signal add to the histogram whilst the spike events from the alternate signal subtracts values from the histogram resulting in the cross correlation function.

The only small discrepancy with the described algorithm is that it is optimally applicable to similar signals. This means the algorithm is actually ideal in a radio astronomy correlation systems as the difference in the input signals on each antenna is not the profile or shape but the time taken to reach the different antenna as measured from the reference point.

5.4 Digital Neuron design and architecture

The design was implemented using Altera's Quartus® II version 8 software using VHDL (VHSIC Hardware Description Language where VHSIC stands for Very-High-Speed Integrated Circuit) programming. The resultant code was downloaded onto an Altera Nios® II evaluation kit which comprises of a Cyclone® EP1C12F1208 FPGA with Nios II processor cores and peripherals.
The design has been separated into different sections to offer an in-depth explanation of the different system blocks.

The basic neuron module design was based on the analog circuit shown in figure 13, and the Leaky Integrate and Fire neuron structure proposed in [4]. It consists of different modules with relative equivalents in the neuron's physiology.

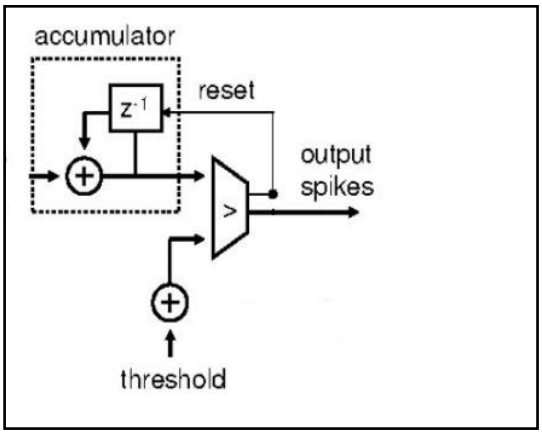


Figure 13: Digital interconnection of integrate and fire neuron blocks

The digital neuron had to incorporate the relevant physiology of the biological neuron in order to be modelled accurately. The block diagram in figure 13 shows the interconnection design shown by Cassidy [4]. The neuron architecture was done in such a way that the accumulator, comparator, counter were written as separate VHDL scripts and interconnected in the overall

design script shown in Appendix 2. The resultant register transfer level schematic of the design is shown below in figure 14.

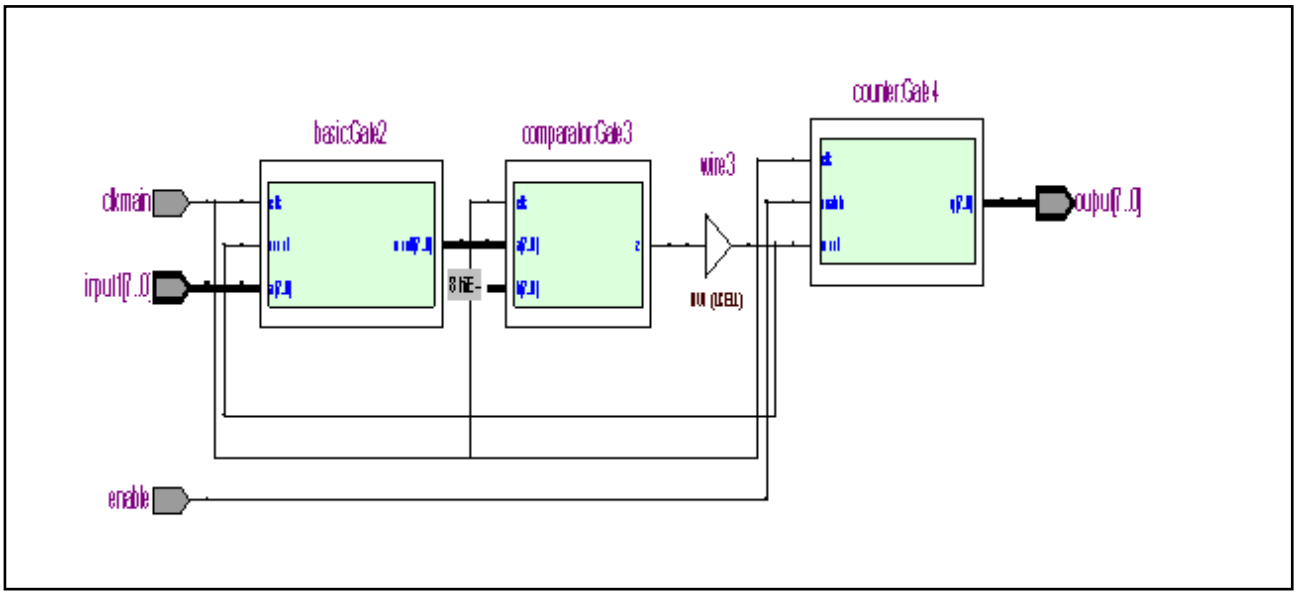


Figure 14: RTL schematic of basic neuron unit

This diagram is the Register Transfer Level schematic which represents a basic digital neuron circuit as a set of connected primitives i.e (adders, counter, comparators and registers)

The summed inputs to the neuron enter the neuron serially, and are accumulated in the accumulator. When the accumulation value reaches or exceeds a preset threshold value the comparator generates a spike event which in turn resets the accumulator value and starts a counter. The counter gives out the time between spikes as it is reset and triggered every time a spike event occurs, with everything being dependent on various enable signals. Since the model used is a non-leaky integrate and fire neuron, the exponential decay is not represented in this design.

In the digital algorithm,

1. The membrane potential is represented by an accumulator, which accumulates the incoming signal until a particular threshold limit is exceeded.
2. A comparator continuously compares the threshold limit to the value of the membrane potential and emits a spike once the neuron's potential exceeds that threshold.
3. A counter is triggered at every spike event to count the interspike interval in clock cycles which are based on the design clocking system.

This is an event driven algorithm where an occurrence in the system's previous module triggers an appropriate response from the interconnected module. Response of the module

receiving the input depends on the given output of the previous module. The aim of modelling a digital neuron is to assess how many of the logic elements it utilises on the FPGA platform and to acquire an easily reconfigurable system that is relatively easy to upgrade, repair and maintain. The FPGA architecture is ideal as the design techniques are suited for Application Specific Integrated Circuits (ASICs) as well as parallel architectures.

5.4 .1 Signal Conditioning: ADC0809

Before the analog signal[12] was supplied to the FPGA, relative signal conditioning had to be applied. The analog input signal was converted by the ADC0809 analog to digital converter which is microprocessor compatible. The key features of the Integrated Circuit (IC) are:

- a resolution of 8 bits
- low power rating of 15 milliwatts
- a total unadjusted error of ± 1/2 LSB and ±1 LSB
- a conversion time of 100 micro seconds

The ADC acquires control signals from the FPGA in order to function properly as well as give an accurate output conversion of the input data. The control signals were incorporated in the VHDL script with the block name *ADC0809*. The ADC input was digitised at a rate of approximately 8 clock cycles, where a clock period was supplied by the FPGA at a time interval of 10uS. Whenever the conversion of an analog input value was complete the output was only passed forward to the *Look up table (LUT)* module if the output enable signal was high. Once the output of the ADC0809 was passed onto the look up table for conversion, the VHDL script for the LUT (look up table) converted the analog values into a full range of positive and negative values. Some of the conversion values are shown in Table 2 and Figure 5.8.

The FPGA clock speed is 24 MHz with the clock enable of about 80 uS representing the 8 clock cycles required to convert an analog input signal into a digital value for the ADC0809. The FPGA sends control signals for the Start, Output Enable and Clock frequency to the ADC0809 chip. When the ADC0809 receives an analog signal and it converts it into a digital equivalent,

[12]The analog signal was supplied by the HP33120A, 15MHz Function, Arbitrary Waveform Generator. The analog input value had amplitude of 825mVpp and an offset of 1.65volts as the FPGA is has a reference voltage of 3.3 volts.

the ADC0809 integrated circuit output is sent to the FPGA ADC0809 module control block and assigns the input value as an address in the *Look up table* which is relayed as a digital output. Once the full conversion is completed in the FPGA the start signal is sent to the ADC0809 chip. In turn the FPGA receives an End of Conversion signal to signify that the FPGA can read the output of the ADC0809 IC. Figure 15 illustrates the input and output signals of the ADC0809 control block in the FPGA and Figure 5.7 illustrates the Look Up Table module that converts signals into a signed twos complement format to enable conversion of signals in both positive and negative ranges.

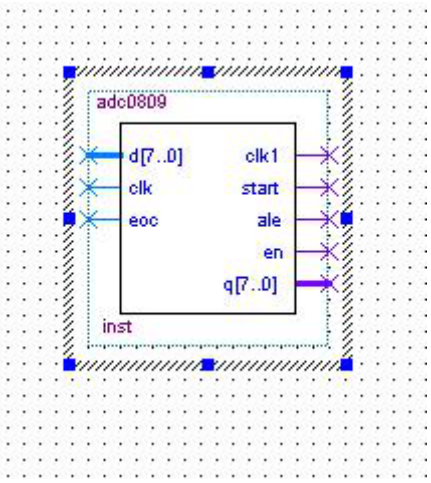


Figure 15: ADC0809 FPGA control module

The figure below shows the Look up Table module.

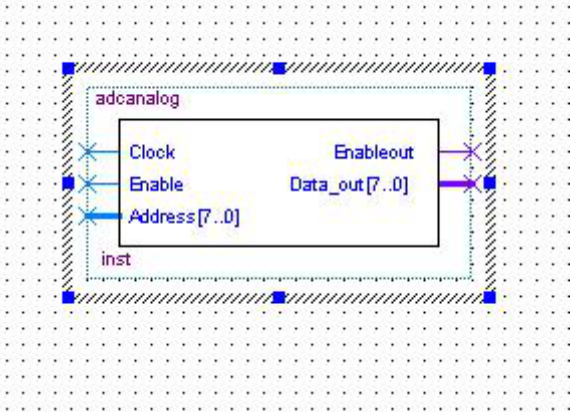


Figure 16: Look up Table module.

The module is responsible for converting AD0809 IC output values into positive and negative numbers.

As an illustration of the conversion procedure to the twos complement format, Table 2 shows some numbers and their equivalent values and Figure 17 shows the linear relationship between the values in graphical form covering all possible output values for the system.

ADC output	2s Complement
-127	11111111
-110	11101110
-50	10110010
0	10000000
50	00110010
110	01101110
127	01111111

Table 2: Analog Converted Equivalent Signals

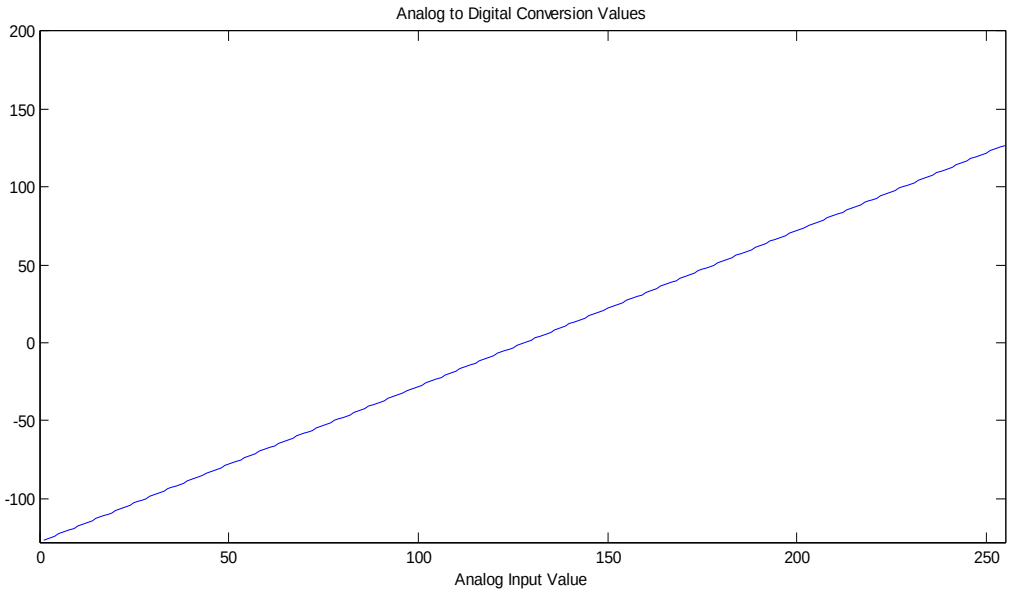


Figure 17 : Graph of Analog input values and their digital equivalent.

5.4.2 The PC interface

A VHDL script was written to carry out the function of a Universal Asynchronous Receiver/ Transmitter (UART) module. The UART is responsible for converting the parallel output of the

neuron module into a serial format. The output of the UART block, which is the count of the number of cycles between each spike event, is sent to the Max232 chip at a baud rate of 115200 bits per second. UART implements simple RS-232 asynchronous transmit and receive logic inside an Altera FPGA. The UART used in this thesis is a transmitter only thus the receiver side of the unit is redundant as exemplified by figure 18.

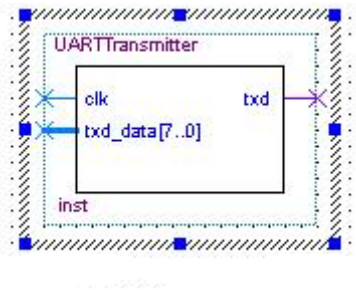


Figure 18: Diagram of UART Block on FPGA.

The MAX232 chip was configured for connection to the RS232 communication port of the computer. Once data arrived at the COM1 port, Windows HyperTerminal was utilised to poll the digits and store data in an appropriate text file. COM1 port is the name assigned to the first serial port on the computer. The resulting text file was then loaded, post-processed and plotted using Matlab®. The HyperTerminal settings were a baud rate of 115200, no flow control, 1 stop bit and an ASCII 8 bit number representation.

Chapter 6: Presentation and discussion of the results

This chapter discusses the results obtained in the thesis and recommendations that can improve the algorithm. There is an analysis of how the neuron responded to different frequency ranges and suggestions of the reasons that caused the observed discrepancy.

6.1 Digital Neuron Output and Graphical Representation

For proper analysis of the correlation function from the digital neuron, a range of frequency values were used to assess how the neuron model responds to varying frequencies. The test runs on the neuron were done using sine waves that used the following ranges: 1 Hz, 100 Hz, 100 KHz and 1 MHz. The analog signal was supplied by the HP33120A, 15MHz Function, Arbitrary Waveform Generator. The analog input value had amplitude of 825mVpp and an offset of 1.65volts as the FPGA has a reference voltage of 3.3 volts. In order to test that the analog to digital converter system implemented on the FPGA was fully functional the analog input signals were stored as various text files and plotted as graphs. The resultant plots of the input signal at I Hz and 100 KHz resulted in the output waveforms shown in figure 19. and 20.

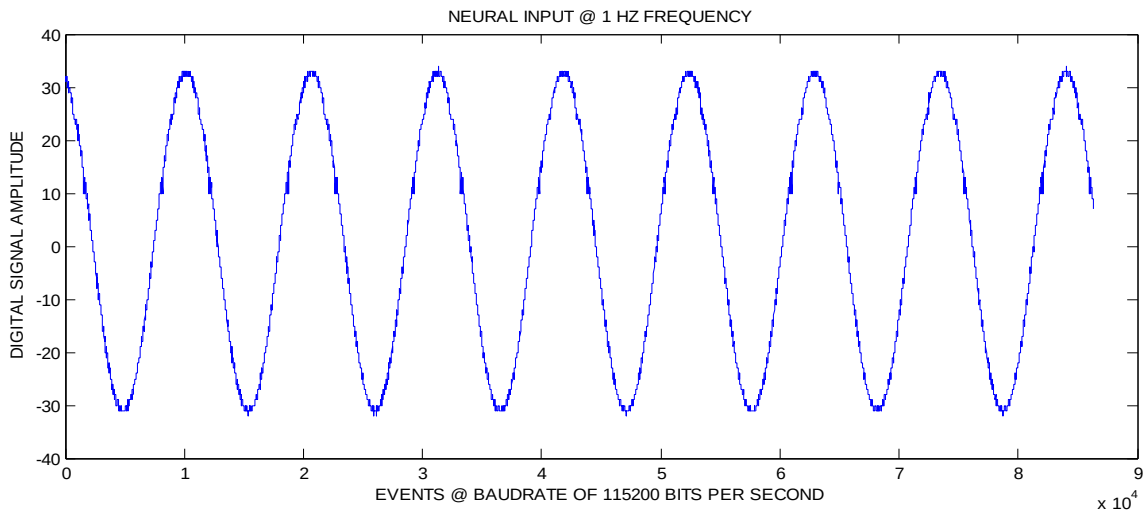


Figure 19: Neural input signal at 1 Hz.

Plot of the analog signal to the neuron once converted by UART to the COM1 RS232 port. The plots were done to test that the data conversion and acquisition system was functionally correct. This plot illustrates the waveform of the input signal at 1Hz.

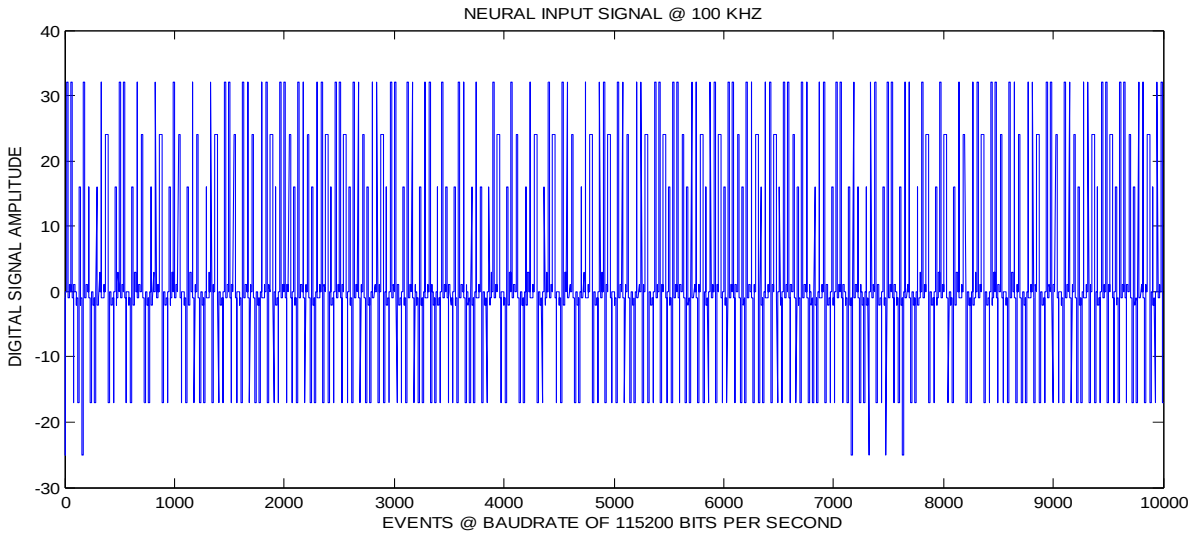


Figure 20: Neural input signal at 100 KHz.

Figure 20 is a graphical representation of the analog input at 100 KHz. Although the graph should show a smooth sine wave, due to the sampling rate which is modulated by the clock frequency of the ADC0809 which is currently 100 KHz for the system described in this thesis. However, even with distortion, the neuron can extract the required information.

The range of signals to be analysed by the neuron was limited to 5 MHz as the signal generator could only support signals that ended at that range. Moreover, the ADC0809 was running at a frequency of 100 KHz for the design in this thesis with its operational clock frequency at a maximum of 1.2 MHz. This means that the Nyquist limit for the system was for signals that had an input frequency of 50 KHz because the analog to digital converter runs at 100KHz. So for signals above that frequency the full proper conversion was distorted as full proper conversion of the signals was limited. Addition of a sample and hold circuit in the system would have been ideal although it was not utilised in this thesis as the major aim was not on the analog to digital conversion but on the modelling of a reconfigurable neuron that relays the information correctly as demonstrated by the correlation functions. This is explained further by looking at figure 20, which shows the analog input conversion at 100 KHz, the signal does not reach the full range peak to peak value of 30 to -30 bits in the digitally converted amplitude which is equivalent to the amplitude of the analog input signal. This affects the interspike interval count, as there were fewer points that were acquired for the interspike interval to construct the ISIH of input signals in higher frequency ranges from 100 KHz as compared to the correlation histogram for 1Hz. However, as can be seen in figures 22, 23, 24 and 25, the correlator is still functional, although emphasis would be placed on the

design and analysis of an efficient analog to digital conversion system that is efficient for a wide frequency range.

For the analog sine input signal from the ADC0809, the ideal Matlab correlation output for the analog input signal at all the used ranges of correlation frequencies is shown in figure 21. The vertical axis relates to the digital signal amplitude equivalent and the horizontal axis is the output measured on a time scale of 115200 bits per second. This is roughly the expected correlation function required from the input signal. This correlation function is obtained from the reference signal and another that is phase shifted by 90 degrees.

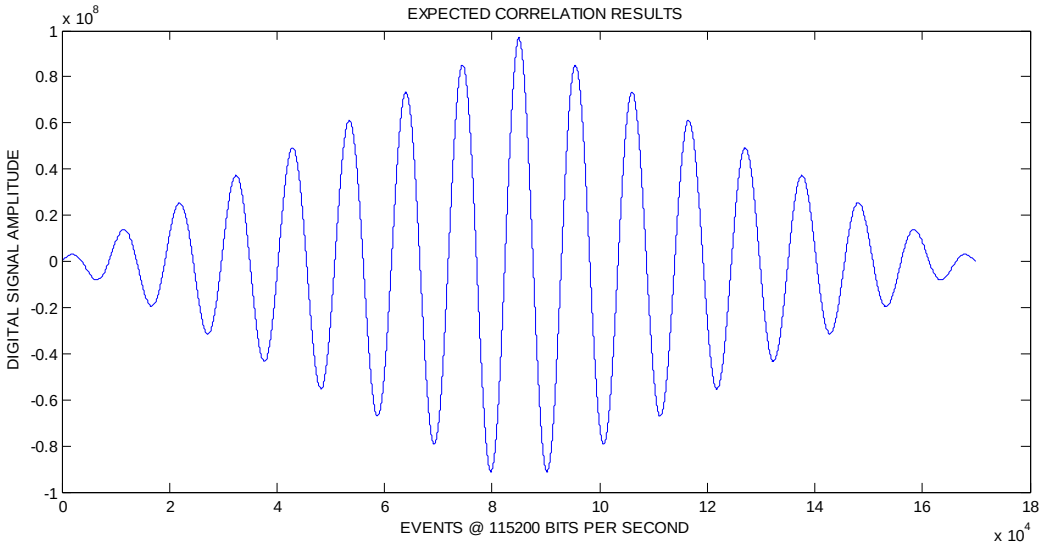


Figure 21: Ideal computed correlation of analog input signal

This is a graph to illustrate the correlation of two sine waves with one of the signals at a phase difference of 90 degrees.

The neural correlation waveforms acquired at the different frequencies are plotted below; however the plots of the neural correlation function are based on the interspike interval counts from the neurons. This results in a different timing scheme as related to the timing applied by Matlab to calculate the correlation function. Therefore, the horizontal axis is based on the distribution of the interspike interval bins, with the width being the interspike interval count of the number of clock cycles that elapse between spikes referenced to the system's internal FPGA clock for the designed system. Figures 22, 23, 24 and 25 illustrate the correlation function acquired from subtracting the first order interval counts of the spiking neuron for the analog sine input waveform at various frequencies.

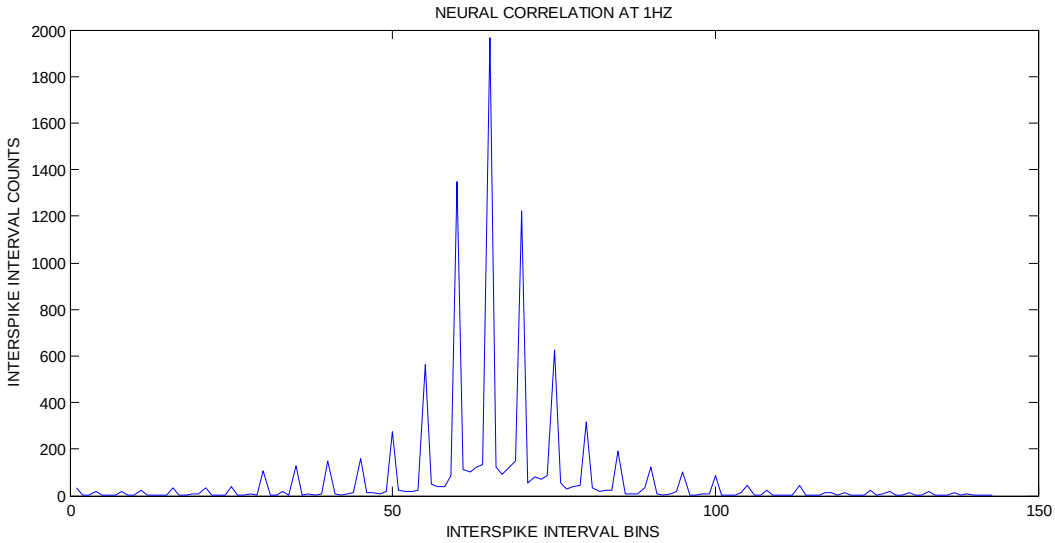


Figure 22: Neural cross correlation at 1 Hz

The graph is plotted by joining the peaks of the interspike histogram Bins. The bins consist of the count value accumulated between the spike events of the neuron. The count is based on the clock cycles of the FPGA system.

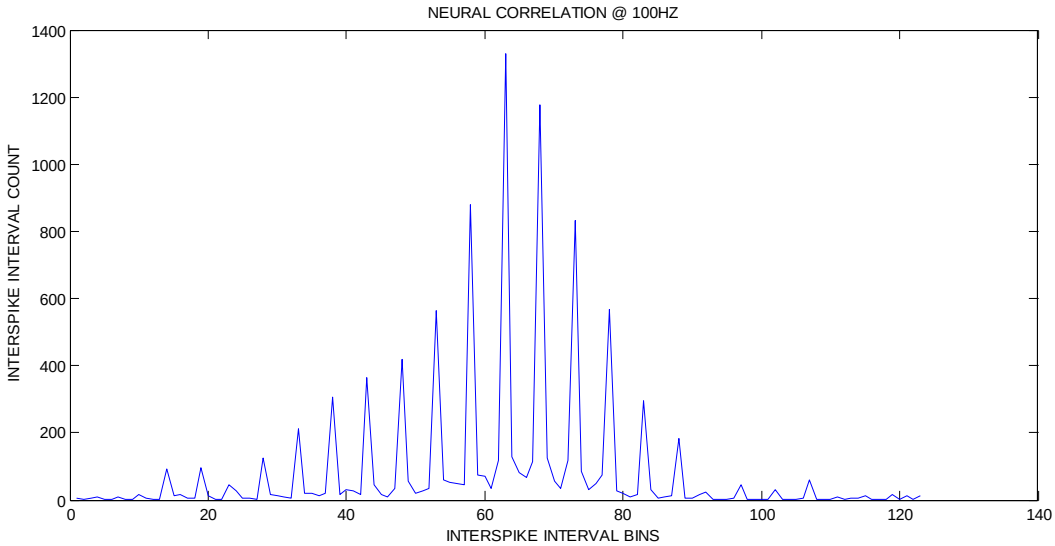


Figure 23: Neural cross correlation at 100 Hz.

The slight distortion of the correlation curve for the figure can be as a result of noise in the system.

The following graphs further illustrate the correlation functions at other frequencies.

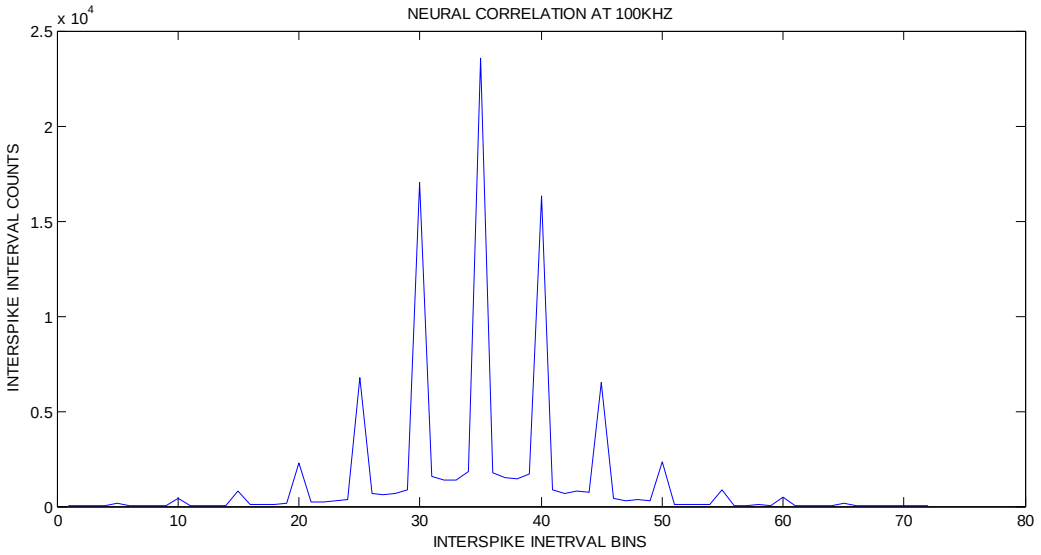


Figure 24: Neural cross correlation at 100 KHz

The difference in the density of the spike count is due to the conversion rate of the analog input signal.

The correlation is evident in the graphical representation. For an increase in the density of the interspike interval count in higher frequency ranges, shown in figures 24 and 25, is possible with an efficient analog to digital conversion system with a higher resolution and sampling rate.

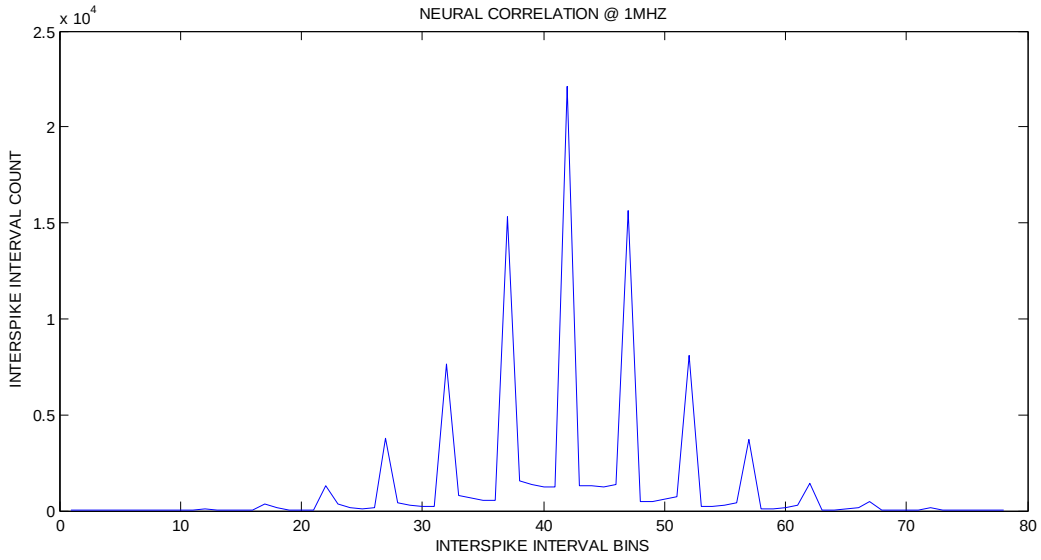


Figure 25: Neural cross correlation at 1 MHz

6.2 Scalability of the system

The major advantage of this design as noted before is the scalability as each neuron can be used to correlate two signals. The neuron only utilises 114 logic elements out of a possible 12060 which is less than 1% of the FPGA. That means roughly about 85 neurons can be implemented on the Altera® Cyclone 1C12 Evaluation Kit with the EP1C12F324C8 FPGA. The neural design inclusive of the UART module, ADC controller and Look Up Table has only 15 input/output pins out of a possible 249 for each neuron. In the current design no memory bits were even utilized resulting in the FPGA's 239 616 bits being unutilized. However, utilisation of the onboard memory could have aided in an easier timing scheme for the neural output, as the data could have been stored on the board.

6.3 Conclusions

As a point of revision, the thesis aimed at;

1. Converting an analog integrate and fire neuron into a digital reconfigurable equivalent using the relative parameters.
2. Analysing the model on the basis of the interspike interval histogram of the counts between the spike emitted by the neuron to extract the mathematically correlation function of the system's input signals as suggested by Tapson in 1998 and compare results obtained by the designed module to the ideal correlation function.
3. Analysing the range of the digital neuron's stability and assessing how it would aid the computational constraints faced in neural correlation in signal processing systems with emphasis on radio astronomy signals where resources used need to be kept at a minimum and at the same time acquiring accurate results.

This thesis investigated how the correlation function can be extracted from the response of a biologically inspired electrical circuit. Analysis was done using the interspike interval histograms of neural spike responses to an input signal. The fact that sample waveforms that hold a likeness to the correlation function were extracted from the neuron show that modelling of the neuron was done using the appropriate parameters. The resultant architecture was proposed for use in intensive signal processing environments like the radio astronomy platform as a means of an alternative algorithm that uses less power. Moreover, all the goals mentioned were met in the course of the development of the thesis with a discussion

of the acquired results at various segments of the thesis. The next section offers recommendations that can aid in making the digital neural model a more realistic approximation of the proposed analog model. This is to point in the direction of further research and application areas.

6.4 Recommendations and suggestions of Future work

Following on the conclusions drawn, recommendations can be made on how the neural engine can be improved in order to have a more biologically plausible digital equivalent of the integrate and fire neuron. Additional changes that can be made to the system to make the computational processing in correlator signals easier are also suggested. Future work and recommendations include:

- Modelling of the integrate and fire neuron, or any model of the spiking neuron, on an analog based architectural platforms. One example of such a platform would be a Field Programmable Analog Array (FPAA). A FPAA is an analog equivalent of an FPGA but the configuration blocks are analog based, consisting of operational amplifiers, and connected to passive components. Another practical alternative is usage of VHDL language libraries based on analog circuit components like capacitors and resistors. This is what VHDL-AMS (VHDL- Analog Mixed Signal) is based on. VHDL-AMS contains libraries with analog and mixed-signal extensions that define the behaviour of analog and mixed-signal systems in line with the IEEE standard IEEE 1076.1-1999. This is a bonus for the design as the analog based platforms add a more realistic feel to the design, as modelling a digital platform from an originally analog concept can lead to discarding some important parameters of the neuron.

- The neural model is based on a non-leaky integrate and fire neuron. A leaky integrate and fire neuron would be a basis of a more accurate neural model. The leaky integrate and fire neuron incorporates a leakage term that is added to the membrane potential which is a measure depicting the rate of diffusion of ions that occurs through the membrane when there is a lack of equilibrium in the cell.

References

1. Haykin, S., (2005). *Neural networks – A comprehensive foundation*. Prentice Hall.

2. Valděs, J.J., and Bonham-Carter, G., (2006). *Time dependent neural network models for detecting changes of state in complex processes: Applications in earth sciences and astronomy.* Neural Networks, vol. 19, pp. 196-207.

3. Brette, R., Rudolph, M., and Carnevale, T., (2007). *Simulation of networks of spiking neurons: A review of tools and strategies.* Journal of Computational Neuroscience, vol. 23, pp. 349–398.

4. Tapson, J., Jin, C., and van Schaik, A., (2007). *A Scalable Architecture for Event-Based Cross-Correlation.* Biomedical Circuits and Systems Conference, BIOCAS 2007. IEEE, pp. 83-86.

5. Bringer, M., and Boër, M., (2000). *An Automatic Astronomical Classifier Based on Topological Neural Networks*. Astronomical Society of the Pacific Series, vol. 216, pp. 640.

6. D'Addario, L., (1989). *Cross correlators*, *Lecture 4 in Synthesis Imaging in Radio Astronomy*. Astronomical Society of the Pacific Series, vol. 6, pp. 59.

7. Tips and Hints for the HP 33120. Available at: http://tritium.fis.unb.br/Fis3Exp/www.tmo.hp.com/tmo/pia/BasicInstrument/TUTnBRIEF/English/BI-B-33120A-005.html [accessed 14 March 2009].

8. Biologically inspired computing. Available at: http://en.wikipedia.org/wiki/Biologically_inspired_computing [accessed 21 November 2009].

9. Fischback, G., (1992). *Mind and Brain*. Scientific American, vol. 267, pp. 24-33.

10. Schutz, A., (1995). *Neuroanatomy in a computational perspective.* Handbook of Brain Theory and Neural Networks, pp. 622-626. MIT Press.

11. Aleksander, I., and Morton, H., (1995). *An Introduction to Neural Computing*. 2nd edition International Thomson Computer Press.

12. Andre, D., Friedman, N., and R. Parr., (1998) *Generalized prioritized sweeping* in *Advances in Neural Information Processing Systems* in Proceedings of the 1997 conference on Advances in neural information processing systems, vol. 10, pp. 1001-1007.

13. Santiago, A., McNames, J., Burchiel, K., and Lendaris, G.G., (2003). *Developments in understanding neuronal spike trains and functional specializations in brain regions.* Neural Networks, vol 16, pp. 601–607.

14. Gerstner, W and van Hemmen, J.L., (1994). *How to describe neural activity: spikes, rates, or assemblies?* Advances in Neural Information Processing Systems, vol. 6, pp. 463-470.

15. Maass, W., (1996). *Networks of Spiking neurons: The third generation of neural network models.* In Proceedings of the 7th Australian Conference on Neural Networks, pp. 1-10.

16. Barbi, M., Chillemi, S., and Di Garbo, A., (2000). *The leaky integrate-and fire with noise: a useful tool to investigate SR*. Chaos, Solitons and Fractals, vol. 11, pp. 1849–1853.

17. Shimokawa, T., Pakdaman, K., and Sato, S., (1999). *Time-scale matching in the response of a leaky integrate-and-fire neuron model to periodic stimulus with additive noise*. Physical Review, vol.59, pp. 3427–3443.

18. Plesser, H.E., and Geisel, T., (1999). *Markov analysis of stochastic resonance in a periodically driven integrate-and-fire neuron*. Physical Review, vol. 59, pp. 7008–7017.

19. Stergiou, C., and Siganos, D. *Neural Networks* .Available at: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Applications%20of%20neural%20networks [accessed 13 December 2008].

20. Vreeken, J., Spiking Neural Networks, an introduction. Available at http:\\ai-lab.cs.uu.nl/pubs/SNN_Vreeken_Introduction.pdf [accessed 15 March 2009].

21. Ferster, D., and Spruston, N., (1995). *Cracking the neuronal code.* Science, vol. 270, pp. 756-757.

22. Gerstner, W., *Spiking Neurons* in Maass, W., and Bishop, C.M., (1999) in *Pulsed Neural Networks*, MIT Press Cambridge.

23. Bohte, S.M., (2003) *Spiking Neural Networks*. PhD thesis, Universiteit Leiden. Available at http://homepages.cwi.nl/˜sbohte/publications2.htm.[accessed 12 May 2009].

24. Hopfield, J.J., (1995). *Pattern recognition computation using action potential timing for stimulus representation*. Nature, vol. 376, pp. 33–36.

25. Rieke, F., Warland, D., de Ruyter van Steveninck, R., and Bialek W., (1997) *Spike: Exploring the Neural Code*. Computational Neurosciences. MIT Press, Cambridge.

26. Vismer, M.P., (2008). *A Neural Correlation Engine for A Wireless AER Network.* Master of Science Thesis. University of Cape Town.

27. Lindsay, K. A., Ogden, J. M., Halliday, D. M., and Rosenberg, J. R. (1999). *An Introduction To The Principles Of Neuronal Modelling* in Modern techniques in neuroscience research U. Windhorst & H. Johannson eds. Berlin: Springer-Verlag pp. 213-306.

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

29. Available from: http://www.neuron.yale.edu/neuron/about [accessed 28 December 2009].

30. Available from: http://www.ncrg.aston.ac.uk/netlab/index.php [accessed 30 December 2009].

31. Available from: http://www.scholarpedia.org/article/GENESIS [accessed 30 December 2009].

32. Available from: http://www.nest-initiative.org [accessed 30 December 2009].

33. Valdez, M.E., *Radio astronomy for Beginners*.

34. Deller, A.T., Tingay, S.J., Bailes, M., and West, C., (2007). *DiFX: A Software Correlator for Very Long Baseline Interferometry Using Multiprocessor Computing Environments.* Astronomical Society of the Pacific, vol. 119, pp. 318–336.

35. Thompson, A. R., Moran, J. M., and Swenson, G. W., (1994). *Interferometry and Synthesis in Radio Astronomy.* (Malabar: Kreiger) Second Edition, John Wiley and Sons, New York.

36. Brisken, W., (2006). *Cross correlators.* Synthesis Imaging Summer School UNM, June13-20. Available at: http://www.aoc.nrao.edu/events/synthesis/2004/presentations/BriskenCrossCorrelators.pdf

37. Koistinen, O., Lahtinen, J., and Hallikainen, M.T., (2002). *Comparison of Analog Continuum Correlators for Remote Sensing and Radio Astronomy .* IEEE Transactions On Instrumentation And Measurement, vol. 51, pp. 227–234.

38. Bunton, J.D., (2004). *SKA Correlator Advances.* Experimental Astronomy, vol. 17, pp. 251-259**.**

39. Parsons, A., Backer, D., Chang, C., Chapman, D., Chen, H., Droz, P., de Jesus, C., MacMahon, D., Siemion, A., Wawrzynek, J., Werthimer, D., and Wright, M., (2006). *A New Approach to Radio Astronomy Signal Processing: Packet Switched, FPGA-based, Upgradeable, Modular Hardware and Reusable, Platform-Independent Signal Processing Libraries.* In: Proceedings of the XXXth General Assembly of the International Union of Radio Science, Boulder, Colorado.

40. West, C.J., (2004). *Development of disk-based baseband recorders and software correlators for radio astronomy.* Dissertation Masters of Applied Science, Swinburne University of Technology.

41. Parsons, A., (2006). *Correlator Development plans*. Available at: http://astro.berkeley.edu/~dbacker/eor/p005.aparsons.pdf .

42. Bunton, J.D., (2000). *An Improved FX correlator*. Alma Memo 342. Available at: http://www.alma.nrao.edu/memos/html-memos/abstracts/abs342.html.

43. Licklider, J.C.R., (1959). *Three auditory theories in Psychologv: a study of a science, vol. I* (ed) Koch., S., McGraw-Hill, New York, pp.41-144.

44. Folowosele, F., Tenore, F., Russell, A., Orchard, G., Vismer, M., Tapson, J., and Etienne-Cummings R., (2008). *Implementing a Neuromorphic Cross-Correlation Engine with Silicon Neurons*. IEEE Proc. ISCAS, pp. 2162-2165.

45. Tapson, J., (1998). *Autocorrelation Properties of Single Neurons.* Proceedings of IEEE Comsig '98, pp. 209-212.

46. Tapson, J., Vismer, M. P., Jin, C., van Schaik, A., Folowosele, F. O., and Etienne-Cummings, R., (2008). *A two-neuron cross-correlation circuit with a wide and continuous range of time delay.* Circuits and Systems, ISCAS 2008. IEEE International Symposium. pp. 420-423.

47. Cassidy, A., Denham, S., Kanold, P., and Andreou, A., (2007). *FPGA based silicon spiking neural array.* Proceedings of the Biomedical Circuits and Systems, pp. 75–78.

48. Hiroyuki, T., (2008). *Basic spike-train properties of a digital spiking neuron. D*iscrete and Continuous Dynamical Systems Series B, vol. 9, pp. 183-198.

49. Goldberg, D.H., and Andreou, A. G., (2007). *Distortion of Neural Signals by Spike Coding.* Neural Computation, vol. 19, pp. 2797-2839.

50. Upegui, A., Peña-Reyes, C.A., and Sanchez. E., *A functional spiking neuron hardware oriented model.* Swiss Federal Institute of Technology at Lausanne, Logic Systems Laboratory. Available at: http://lslwww.epfl.ch/~upegui/Welcome%20to%20the%20Logic%20Systems%20Laboratory%20of%20the%20EPFL_files/iwann_03.pdf

51. Perez-Uribe, A., (1999). *Structure-adaptable digital neural networks*. PhD thesis. Ecole Polytechnique Fédérale de Lausanne**,** EPFL. Available at: http://lslwww.epfl.ch/pages/publications/rcnt_theses/perez/PerezU_thesis.pdf

52. Tapson, J., and Etienne-Cummings, R., (2007). *A Simple Neural Cross-Correlation Engine.* Circuits and Systems, ISCAS 2007. IEEE International Symposium, pp. 1285-1288.

53. Chikada, Y., Ishiguro, M., Hirabayashi, H., Morimoto, M., Morita, K. I., Miyazawa, K., Nagane, K., Murata, K., Tojo, A., Inoue, S., Kanzawa, T., and Iwashita, H., (1984). *Digital FFT Spectro-Correlator for Radio astronomy* in J. A. Roberts (ed.), *Indirect Imaging*, Cambridge University Press, Cambridge, U.K, pp. 387–404.

54. Wei, D., and Harris, J.G., (2004). *Signal Reconstruction from Spiking Neuron Models.* IEEE International Symposium on Circuits and Systems, vol. 5, pp. 353-356.

55. Tapson, J., Jin, C., van Schaik, A., and Etienne-Cummings, R., (2009). *First-Order Non-Homogeneous Markov Model for Integrate-and-Fire Neurons Stimulated by Small Phase-Continuous Signals.* Neural Computation, vol. 21, pp. 1554-1588.

56. Tapson, J., (2009). *Emergence of Cross-Correlation Functions in Neural Spike Interval Distributions*.The 57th Session of the ISI, August 16-22, 2009, Durban, South Africa. Available from: http://www.statssa.gov.za/isi2009/ScientificProgramme/IPMS/0232.pdf [accessed 14 December 2009]

57. Wiesenfield, K., and Moss, F., (1995). *Stochastic resonance and the benefits of noise: from ice ages to crayfish and SQUIDs*. Nature, vol. 373, pp. 33-36.

58. Gammaitoni, L., Hänggi P., Jung P., and Marchesoni F., (1998). *Stochastic resonance*. Review of Modern Physics, vol. 70, pp. 223-287.

59. Gammaitoni, L., Hänggi, P., Jung, P., and Marchesoni, F., (2009). *Stochastic Resonance: A remarkable idea that changed our perception of noise*. The European Physical Journal B, vol. 69, pp. 1-3.

60. Knight, B.W., (1972). *Dynamics of encoding in a population of neurons*. Journal of General Physiology, vol. 59, pp. 734–766.

61. Lazar, L.L., (2004). *Time encoding with an integrate-and-fire neuron with a refractory period.* Neurocomputing, vol. 58-60, pp. 53-58.

62. Lazar, A.A., and Toth, L.T., (2003). *Time encoding and perfect recovery of bandlimited signals*. Proceedings of the ICASSP'03, vol. 6, pp. 709–712.

63. Lazar, A.A., (2003). *Time Encoding Using Filter Banks and Integrate-and-Fire Neurons*. BNET Technical Report, no.2-03, Department of Electrical Engineering, Columbia University, New York.

64. Adrian, E.D., and Zotterman, Y., (1926). *The impulse produced by sensory nerve endings. Part 2: The response of a single end-organ*. The Journal of Physiology, vol. 61, pp. 151–171.

65. Adrian, E.D., (1928). *The Basis of Sensation: The Action of the Sense Organs.* W. W. Norton, New York.

66. Cariani, P.A., Delgutte, B., (1996). *Neural correlates of the pitch of complex tones. I. Pitch and pitch salience.* Journal of Neurophysiology, vol.76, pp. 1698-1716.

67. Cariani, P.A., and Delgutte, B., (1996). *Neural correlates of the pitch of complex tones, II. Pitch shift, pitch ambiguity, phase-invariance, pitch circularity, rate pitch, and the dominance region of pitch.* Journal of Neurophysiology, vol.76, pp. 1717-1734.

68. Lindsay, K.A., Ogden, J.M., Halliday, and D.M, Rosenberg, J.R., (1999). *An Introduction to Principles of Neuronal Modelling in* Windhorst, U., Johansson, H. (Eds.), *Modern Techniques in Neuroscience*. Springer Verlag, Berlin. pp. 705-755.

Appendix

A.1 Matlab script for loading text file from Hyperterminal.

Plots of the two different spike count are found through the variable w. The interspike interval density is computed by the tabulate function and the resultant interspike interval values are subtracted from each other to aquire the cross correlation function between the two incoming signals.

```
file=fopen('C:\7.txt','r');
w=fread(file,1000000);
table=tabulate(w); %figure(1);
%plot(w)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
file=fopen('C:\8.txt','r');
w1=fread(file,1000000);
table1=tabulate(w1); %figure(2);
```

A.2 VHDL script for basic neural unit

This is the VHDL script for the basic neuron. It does not show the UART or the ADC0809 controller

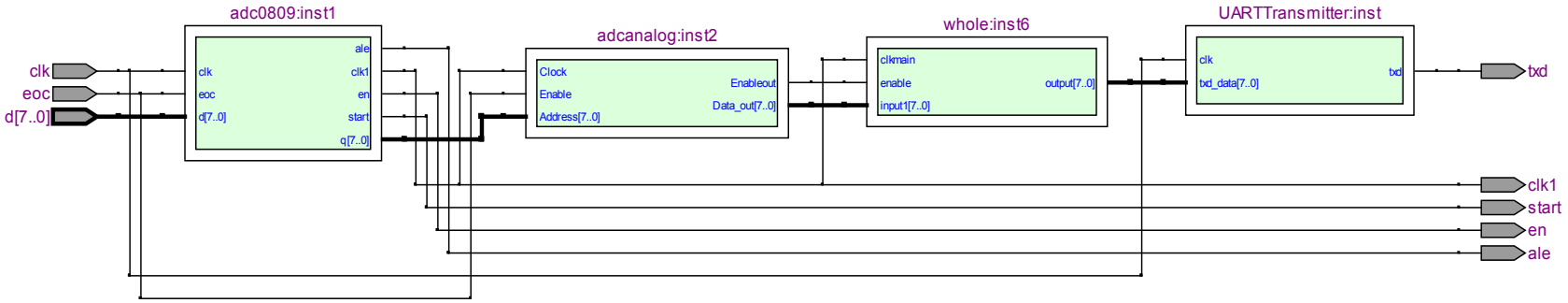
```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.numeric_std.all;  --for logic conversions
--use ieee.std_logic_signed.all;
use ieee.std_logic_arith.all;

entity  comparator is port (
        a,b:  in  std_logic_vector (7 DOWNTO 0);  -- defines a vector of 4 values a3 to a0, b3 to b0
        clk: in std_logic;
        z:        out std_logic) ;--reset,
end comparator;

architecture comparator of comparator is      --  the architecture is being described by its behavior
signal b1:  signed (7 DOWNTO 0);
signal a1:  signed (7 DOWNTO 0);
begin

comp: process (a,clk)         --  comp is an optional label identifying the process

        begin
        IF rising_edge (clk) THEN


                a1<= signed(a);
                b1<= signed(b);
                --a1(b1'left)<= not a1(a1'left);
                --b1(b1'left)<= not b1(b1'left);

                if b1 <= a1 then
                z <= '1';
                else z <= '0';
                end if;
                end if;
end process comp;
end comparator;



-- Quartus II VHDL Template
-- Signed Adder

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity basic is

        generic
        (
```

```
                DATA_WIDTH : natural := 8
        );

        port
        (
                a           : in signed       ((DATA_WIDTH-1) downto 0);
                reset,clk : in std_logic;

                result : out std_logic_vector((DATA_WIDTH-1) downto 0)
        );

end entity;

architecture rtl of basic is
signal b_reg:  signed (7 DOWNTO 0);
signal a_reg:  signed (7 DOWNTO 0);

begin

PROCESS (a,clk, a_reg,b_reg, reset)
  begin
if reset ='1' then
                a_reg <= "00000000";
                b_reg <= "00000000";
elsiF (clk'event and clk = '1') THEN
a_reg <= a;
        b_reg <= a_reg + b_reg;
        end if;
        end process;
result <= std_logic_vector(b_reg);
end rtl;


-- Quartus II VHDL Template
-- Binary Counter

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
entity counter is


        port
        (
                clk                 : in std_logic;
                reset     : in std_logic;
                enable    : in std_logic;
                q                   : out std_logic_vector (7 DOWNTO 0)
        );

end entity;

architecture rtl of counter is
```

```
                signal  cnt             : std_logic_vector (7 DOWNTO 0);
                signal  d               : std_logic_vector (7 DOWNTO 0);
begin

        process (enable,reset,cnt,clk)

        begin
                --if (rising_edge(clk)) then

                        if reset = '1' then
                                -- Reset the counter to 0
                                cnt <= "00000000";

                        elsif (clk'event and clk = '1') THEN
                        if enable = '1' then
                                -- Increment the counter if counting is enabled
                                cnt <= cnt + 1;
                                d <= cnt;
                        end if;
                        end if;
                --end if;

                -- Output the current count
                q <= cnt;
        end process;

end rtl;




 ----- File my_components.vhd: ---------------
 LIBRARY ieee;
 USE ieee.std_logic_1164.all;
  ------------------------
 PACKAGE  my_components   IS

   ------ basic: ---
  COMPONENT    basic IS
    PORT (
                a:              IN STD_LOGIC_VECTOR(7 downto 0);
                clk:            IN STD_LOGIC;
                reset:          in STD_LOGIC;
                result:         OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
        ) ;
        END  COMPONENT   ;

        ------ counter: ---
   COMPONENT    counter IS
  port (clk             : in std_logic;
                reset     : in std_logic;
                enable    : in std_logic;
                q               : out std_logic_vector(7 downto 0));

         end COMPONENT   ;
```

```
        ------ comparator: ---
   COMPONENT comparator IS
port (
        a,b:  in  STD_LOGIC_VECTOR (7 DOWNTO 0);  -- defines a vector of 4 values a3 to a0, b3 to b0
        clk: in std_logic;
        z:      out std_logic) ;--reset,

 end COMPONENT    ;
END  my_components;

 ---------------------------------------------
  ----- File neuron.vhd: ----------------------
  LIBRARY ieee;
  USE ieee.std_logic_1164.all;
  USE work.my_components.   all;


entity whole is
port(    input1: IN STD_LOGIC_VECTOR(7 downto 0) ;
                 clkmain,enable:                 IN STD_LOGIC;
                 output: out std_logic_vector(7 downto 0));
end whole;


architecture  structural of whole is

signal wire2 :std_logic_vector (7 DOWNTO 0);
 attribute preserve: boolean;
attribute preserve of wire2: signal is true;
constant threshold: STD_LOGIC_VECTOR ( 7 DOWNTO 0) := "01111110" ;

signal wire3 : STD_LOGIC;--_VECTOR (7 DOWNTO 0);
attribute keep: boolean;
attribute keep of wire3: signal is true;



begin


                 Gate2: basic            port map  (a=>input1, clk => clkmain,reset=>wire3,result =>
wire2);
                 Gate3: comparator       port map  (a=>wire2, clk => clkmain,z=>wire3,b=>threshold);
                 Gate4: counter          port map  (reset=>wire3, clk=>
clkmain,q=>output,enable=>enable);



end structural;
```

A.3 Register Transfer Logic Schematic for system



This schematic diagram show all components used to design the neural model on an FPGA platform in this thesis.

- *ADC0809* is the control module for the ADC0809.
- *Adcnanalog* is the look up table for conversion to a twos complement platform.
- *Whole* is the neuron.
- *UARTTransmitter* is the UART control block.