DEVELOPMENT OF THE ICAANN; A LOW-POWERED, ANALOG, NEURAL SIGNAL PROCESSOR

A Thesis

Submitted to the Faculty

of

Purdue University

by

Michael J. Bertram

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Biomedical Engineering

December 2017

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Ken Yoshida, Chair

Department of Biomedical Engineering

Dr. Edward Berbari

Department of Biomedical Engineering

Dr. Paul Salama

Department of Electrical Engineering

Approved by:

Dr. Julie Ji

Head of the Graduate Program

For Dr. Johnathan Mills

ACKNOWLEDGMENTS

I would like to acknowledge, first and foremost, Dr. Ken Yoshida, who saw the potential in me and gave me my first real job, then the opportunity to further my education by supporting me in getting my masters degree. He has always had faith in me and made sure the work I have done is the best work I can do.

Also of note is Gregory Mattes and Bryce Himebaugh, who worked tirelessly and motivated me to do the same. Both gave me opportunity to show my ability and I thank them for the confidence they gave me.

I would also like to acknowledge my colleagues in the lab; Nhan Do, Caleb Comoglio, Chandrama Ahmed, Hunter Cox, and Ryne Horn. Without them, my days in the lab would have been filled with only work and would have been quite boring. They were always ready to have an idea bounced off of them or to listen to me complain about something not working. They truly helped me get through this program easier.

I would like to acknowledge my family, who were always there to keep me going on the right path and keep my head on straight even during the worst times.

Finally, this work has been dedicated to the late Johnathan Mills, without whom, this work would have never been possible.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
ANOVA	Analysis of Variance
AP	Action Potential
BP	Back-propagation
BR	Bayesian Regulated
CNS	Central Nervous System
D1503	Integrated Circuit Name for ICAANN
DSP	Digital Signal Processor
EAC	Extended Analog Computer
FIR	Finite Impulse Response
GDX	Gradient Descent with Adaptive Learning Rate
HLN	Hidden Layer Node
IC	Integrated Circuit
ICAANN	Integrated Configurable Analog Artificial Neural Network
IIR	Infinite Impulse Response
IMES	Implantable Myoelectric Sensor
LDA	Linear Discrimination Analysis
LDF	Linear Discrimination Function
LIFE	Longitudinal Intrafasicular Electrode
LM	Levenberg-Marquardt
LP	Linear Perceptron
MF	Match Filter
MLP	Multiple Layer Perceptron
PCA	Principal Component Analysis

PNS	Peripheral Nervous System
RF BION	Radio Frequency BIOnic Neuron
SFAP	Single Fiber Action Potential
S/H	Sample and Hold
SNR	Signal to Noise Ratio
TIME	Transversely Intrafasicular Multichannel Electrode

ABSTRACT

Bertram, Michael J. M.S.B.M.E., Purdue University, December 2017. Development of the ICAANN; A Low-Powered, Analog, Neural Signal Processor. Major Professor: Ken Yoshida.

In the world of neuroscience and neural engineering, there is a desire to understand the language of the nerves. This language is spoken in action potentials. But this requires the ability to discriminate and track single fiber action potentials. Electrodes today are able to discriminate single fibers firing, but a single action potential doesn't say much. What is needed is a way to decipher the patterns and modulations of these action potentials and develop them into something meaningful. This is where the process of spike sorting comes into play. However, current mobile digital signal processors cannot process the amount of information coming from the electrode arrays in real-time. This greatly limits the mobility and growth into markets such as bioeletric medicines or neuroprosthesis. However, a solution exists using analog computing technologies. This work sets out to optimize a configurable integrated chip through using machine learning techniques. Initially, this resolution was realized through a conductive foam sheet and the delta rule to identify single fiber action potentials. Eventually, using a machine learning technique known as back-propagation, the work extended to artificial neural networks to identify and classify single fiber action potentials. Once it was demonstrated that artificial neural networks were well suited for the classification tasks, design parameters were found to develop an integrated circuit chip, known as the D1503 or the Integrated Configurable Analog Artificial Neural Network (ICAANN). Finally, once this chip was realized *in-silico*, it was tested and shown to behave similarly to the simulations on conduction foam.

1. INTRODUCTION

For decades, interfacing with the human nervous system has been a dream that has been realized more and more with each advance of technology. Research from industry and academia in the field of neural engineering has yielded the ability to sense and communicate with the nervous system through simulation and recording techniques. However, there is still much to be uncovered in this realm; specifically, the language of the nerves - also know as decoding. This is where the field of neuromodulation finds its place. Current technology can interface with the nervous system and its end organs with relative ease. There is a large variety of electrodes that can interact with the central and peripheral nervous system, as discussed in Section 1.2. These electrodes record the extracelluar environment of the targeted cell, which includes noise; causing problems in recording. Due to the signal's small amplitude, tens of microvolts at best for nerves, extracting the signal is arduous and a very difficult processing problem develops. Extensive processing is needed to identify the signal and separate them into the different firing fibers. It's important to separate the different fiber signals; larger nerves, such as the vagus, can have multiple end organs under its control.

This process is termed spike sorting and is a major tool for many neuroscientists and neural engineers. With this tool, researcher are able to understand the nerve's intent for the end organ and in turn the result that should occur. However, due to the amount of computation needed, this tool is currently only a *post-hoc* analysis or performed on a desktop computer with high-powered computing hardware. This is not very useful clinically or with closed-loop control of embedded bioelectric medical devices.

There is a need in this market for a real-time spike classifier that has the ability to both identify and sort neural spikes as they are sensed by the interface. This would allow for clinicians to have instant access to data that could help with neural disease

diagnosis. This would cut the cord connecting patients to computers for high degree of freedom movement in upper-limb prostheses. This would allow for further control of devices for those suffering from paralysis. Ultimately, the first steps of tackling this problem starts with understanding how the nervous system communicates.

1.1 The Nervous System's Communication Mechanism

The nervous system is one of the most complex systems found in the human body. It allows for humans to sense, to respond, and to interact with the outside world. Everything that is human consciousness arises from the nervous system relaying sensory information from the peripheral nervous system (PNS) to the central nervous system (CNS) and back again, occurring millions of times a day. These phenomena are mediated via nerve bundles that run from the end organs or muscle actuators, also known as motor units, to the CNS in the afferent pathway and from the CNS to the end organs or motor units in the efferent pathway. Within the nerve bundle, there are individual nerve fibers that carry information through either the efferent pathway or the afferent pathway. This same phenomena occur within the brain; millions of synapses fire generating patterns that arise as behavior, thought, and consciousness.

Nervous information is carried by a bioelectrical phenomenon known as the action potential (AP) that is the result of opening and closing of ion channels on a particular patch of the nerve fiber's membrane, known as the axonal hillock. When the channels open, ions flow across the membrane resulting in a current. Once this current reaches a certain threshold value, the membrane will depolarize causing the channels to open en mass. This depolarization causes a chain reaction down the length of the membrane, causing other ion channels to open and resulting in a propagating AP. These APs can travel short distances – like between neurons in the brain – or can go as far as your spinal cord to your foot. This all occurs in milliseconds, for what seems like flawless motion to a normal person.

These APs do not hold any information at all for the end target – be it a muscle, a gland, or another neuron – but rather they are a single pulse in a larger signal. Put simply, it is an all or nothing type of signal. Therefore, information sent through nerves can be thought of as a binary digital signal being sent over a cable. This signal can be interpreted at the target either by the number of APs received by a single fiber (rate code) or by the number of nerve fibers that are firing onto the target (population code). Being able to tap into individual, even multiple nerve fibers, allows for interpretation of the nerves firing patterns. This is the language of the nerves and within lies the intention of the brain. However, before the intention can be interpreted, there must first be an interface to communicate with the nerves.

1.2 Human-Computer Interfaces

To access the bioelectric potentials generated by the body, there must be an electrical connection. Usually this is facilitated by a type of metal-metal ion chemistry that allows for the passage of current. These interfaces can be simple as an insulated wire with a sharp de-insulated tip to more exquisite structures that consist of bio-compatible materials with multiple minute contacts. This section covers a few of the more advanced interfaces, such as intramuscular electrodes, intrafasicular neural electrodes, and cortical electrodes. Each one of these interfaces generate spike activity that leads to the need for spike sorting algorithms, discussed in Section 1.3.

1.2.1 Intramuscular Electrodes

Intramuscular electrodes can be very crude – a simple de-insulated tip of an insulated needle is all that is needed for percutaneous recording or stimulation of muscle activity. The electrode can be placed with a percutaneous guide, leaving the tip in the muscle and the wire through the skin. Naturally this is quite damaging to the muscle, but does not show any loss in muscle force [1]. Kilgore et al., has also demonstrated that these electrodes have very rare instances of infection or rejection over a 3 year

period [2]. However, these electrodes are not ideal; mostly due to their crudeness – they are too invasive and are not conducive for recording due to their size. A more sophisticated solution to this problem is the Radio Frequency BIOnic Neuron (RF BION) and the implantable myoelectric sensor (IMES) [3, 4]. Their focus for their uses may be different, but they are very similar to delivery. Both are implanted percutaneously and are powered, commanded, and addressed using telemetry from an external coil [5]. In the case of the IMES, there are small bipolar electrodes that send raw or integrated data back through the coil which can be housed in the device. The IMES has shown great potential in clinical use, because it has eliminated the issues with percutaneous wires. It has shown it's durability by remaining fully operational for 9 months in a human [4]. They have even shown the ability to house a linear discriminator analysis (LDA) classifier to predict finger movement [5] advancing the mobility of the neuroprosthetic

1.2.2 Intrafasicular Electrodes

Electrodes that penetrate the nerve trunk are considered intrafasicular electrodes. They lie in the endoneural space between axons within the nerve fascicle. These are, by far, the most invasive electrodes, but they allow for the most selectivity. The direction of implantation not only generates different signals, but it also differentiates the two major types of intrafasicular electrodes. Longitudinal Intrafasicular Electrodes (LIFE) are electrodes where the contact sites are aligned parallel to the nerve fibers. Transverse Intrafasicular Multichannel Electrodes (TIME) are electrode structures where the active sites lie transversely to the nerve fibers. There is also a third class of intrafasicular electrodes that is quite more complex, the Self-Opening Intraneural Peripheral Interface (SELINE). This electrode was developed in an attempt to solve the mechanical stability issue seen in both LIFE and TIME electrodes [6].

TIME structures were first implanted into muscle and were able to record electromyograph (EMG) activity, but no nerve activity, was present [7]. The use of

multiple contact sites is a key advantage for this type of intrafasicular electrode. TIMEs are designed to maximize the probability of finding a unit, overcoming the disadvantage of high selectivity electrodes by extending the reach of the overall electrode structure. These simultaneous recordings allow for spatial averaging, a technique originally applied in cortical and deep brain structures for unit separation [8,9]. Furthermore, Harreby et al. demonstrated chronic implantation feasibility by examining the encapsulation of the nerve in larger human-sized nerve in the Gottingen Minipig [10].

LIFE structures differ from TIME electrodes not only by orientation, but because the active site is not the tip of the wire electrode. Rather, the structure is pulled through via an insertion needle and anchored at entry and exit points usually by some adhesive. This ensures the active site is secured within the nerve. This concept was tested and confirmed by Bowman et al. in both a chronic rabbit and cat model [11] . They saw both stimulation stability and no evidence of demyelination or denervation. LIFE structures have changed over the years from thick stainless steel to thin 25 micron Teflon insulated, platinum-iridium wires which are de-insulated for the active site [11]. From here platinum black is electro-deposited onto the active site to reduce interfacial impedance and thermal noise [12]. This technique has been extended to fully awake cats [13], and recently in humans [14], where it was demonstrated that large populations from motor units could be sampled selectively.

1.2.3 Cortical Electrodes

There have been multiple attempts to access the activity of brain's neural cell bodies in the hopes of establishing some computer-brain interface. However, recording from the brain can be quite difficult being that there are thousands of cells firing at once in a small area. Dropping a sharpened needle into the brain can isolate 1, perhaps 2, individual neurons. Whereas a simple tetrode can record 10-20 simultaneously [15]. These electrodes are rather crude, dating back to the 1940s, but still have given a

wealth of knowledge to the community [16]. These microwire electrodes are composed of insulated conductors with the tip exposed to the extracellular conditions. They have evolved from more corrosive metals, such as Ag/AgCl, to more stable alloys, like Pt/Ir alloys. The substrate the electrode sits in has also evolved to polymers, such as parylene-c, Al_2O_3 or epoxylite [16]. Naturally, these electrodes are not FDA cleared for the human use and are not fit for computer-brain interfaces.

The microwire electrodes are elementary compared to the complex microelectrode arrays (MEAs), of which there are two types – the planar Michigan-style array (MI) and the Utah electrode array (UEA). With the MI array, diffusion-based etch stops were used by Wise and his colleagues [16] to developed a planar array of micro-electrodes. The main advantage of these planar structures is their ability to access larger volumes of neural tissues. Although, no MI-style array has reached human clinical trials, they are still used in research [16]. The UEA was developed around the same time using thin-film processing, by Normann and colleagues [16]. Specifically, the UEA are generated by a series of glass reflow, dicing, chemical etchings and metallization processes. These devices come in various electrode spacings and shaft architectures. They have had much more success in clinical trials, including six in the U.S. [16].

1.3 Spike Sorting

Whether recording from a nerve cell in the brain or from muscle fibers in the patient's arm, the investigator is still recording from the extracellular environment. Within this extracellular environment there are a multitude of factors that act as a spatial filter on the recorded signal, which can distort the signal and make it difficult to record. When these APs are fired from a neuron or from a muscle cell, they have a very distinct shape. The morphology of the AP changes based on the cell type, naturally, but even within cell types the shape can change based on ionic channel distribution or relative distance from the electrode [17]. As mentioned in Section

1.1, the AP holds no real information – other than an action potential occurred. Therefore, it is important to recognize the patterns in which the APs are fired. This leads to a process known as spike sorting (SS) – where each morphology is clustered into groups that are associated with a nerve or muscle cell [17]. With these clusters it is possible to pin point what cells are firing and hopefully decipher their intention based on the patterning. The basic steps of SS algorithms are as follows:

1. Filtering
2. Detection
3. Feature Extraction
4. Clustering

Current SS algorithms are housed in software applications, such as Plexon's Offline Sorter and NeuroExplorer. These are state-of-the-art applications, and can run SS algorithms in real-time – not a simple feat as discussed in Section 1.4. However, the problem arises where this SS occurs only on a computer and not a mobile standalone platform that can be used on a neuroprosthetic or in the clinic for bioelectric medicines. The reasoning behind this lack of mobility is two-fold – lack of dependability due to low computational power on high levels of signal processing and the amount of energy required to perform the computations is too high [17] inhibiting movement to more mobile platforms.

1.4 The Signal Processing Problem

With all of these different types of electrode structures there is one common theme, a constant stream of information. As can be imagined, the amount of information being transmitted from a single electrode is massive. Not only that, but the window of time in which these bioelectrical events occur is so minuscule, capturing the signal properly requires a high sampling rate between 30 to 50 kHz [13] for neural signals.

This gives a window for calculation between 20 and 33.3 microseconds – not a whole lot of time. Compounding the issue, these are the parameters for only one electrode. When considering current neural interfaces, there could be anywhere from 8 to 16 active electrodes making the combined needed sampling rate closer to 1 MHz [18].

The problem does not stop with acquisition of the signal, but the additional operations needed to be performed for denoising, detection, and classification of the nerve signal. Take for example, a 32-point fully-connected matched filter (MF) programmed to detect a single fiber action potential (SFAP) in the peripheral nervous system (PNS). In a multiple unit recording, it would take approximately 130 operations to perform the matched filtered calculations at each contact [18]. Furthermore, using principal component analysis (PCA) to classify the multiple units with just two principal components with largest variance along Euclidean distances for clustering would require at least 500 operations per electrode. Altogether, this would require 100,000 operations per sampling cycle, which could be as short as 20 microseconds for 50 kHz [18]. Not to mention this all should be done with float-point precision. Currently, this must be done either on an embedded digital signal processor (DSP) to give the patient more mobility, or on a desktop computer to give the user more real-time use of the device. However, it has been shown that a state-of-the-art embedded DSP, such as the TI TMS320C6$^{\text{TM}}$, can only perform little more than half the needed operations in the allotted sampling time frame [19]. Even with more current technology, Mastinu et al. still found time delays that prevented the user from perfectly smooth usage [20].

What this means is that a single embedded DSP chip could not do the operations needed in real-time and that multiple embedded DSP chips would be needed in a parallel configuration. The combination of many DSP chips would not only take up more space on the devices, but it would also consume a lot of energy and limit the battery life of the device needed to run more important aspects of the device, such as movement in prosthetics, or stimulation for closed-loop systems. The amount of energy spent to process the signal must be minimal in these embedded systems.

Section 1.5 dives deeper into the energy needs of an embedded system. This lends itself to the development of a new kind of technology that would bring the best of both worlds - performance and low power mobility. A piece of equipment that not only can perform the hundreds of thousands of operations in real-time, but also can be put in parallel architectures that do not require such high energy loads or space as does a large bank of DSPs.

Unfortunately, the end result of the work done by Soliman [18] and most recent state of the technology was a machine that took seconds to propagate a signal and get a result. Although, this machine (described in Section 1.7) could perform thousands, if not millions, of analog calculations not only instantaneously, but simultaneously; the propagation of the signal via a spatio-temporal mapping (discussed in Section 2.3.4) impedes the system's bandwidth. This issue however, can be remedied by using a more advanced IC to propagate the signal - for example the AD781; a sample and hold circuit. This circuit allows for the propagation of signal in approximately 70 nanoseconds. Doubling this value, to eliminate jitter, would result in a total bandwidth of 7.1 MHz. While this is not impressive for a digital computer, it is important to remember that digital computers function in a serial manner. Soliman's machine and previous instances of them (see Section 1.7 function in a parallel manner. So even though the serial bandwidth is very slow, this machine can perform thousands, if not millions of analog calculations. This would be analogous to running a digital super-computer with millions upon millions of cores, but at a fraction of the power consumption. Therefore, it is apparent that this machine has great potential as a signal processor.

1.5 The Power Consumption Problem

In a fully embedded system, it is crucial the device stays powered for the intended usage time. A realistic example of this is the embedded cardiac pacemaker. They are placed once and left to run for 5-10 years on a battery without charging. This

is because the engineers who built the pacemaker had a power budget – they knew exactly how many hours they could use the microcontroller, how many times they could stimulate, how long they could sense the heart, etc. However, as can be seen in section 1.4, spike sorting devices need to perform a few more calculations than simple threshold detection. The computational load on the processor requires amounts of energy to complete that can't be serviced by batteries in embedded systems. This low power qualification automatically excludes any kind of PC solution for CPU, GPU, or DSP processing, because they require power from the wall socket. Regardless, even for a TI C55 DSP chip to complete a benchmark test, it requires about 140 mW [21]. This is almost a thousand times greater than the power ability to run an embedded device, which can only service tens of μWs.

Power consumption becomes a bit better with a hybrid DSP and microcontroller (μC), where the DSP can service a problem and be put into low-power mode, rather than being always on. For example, an STM32F303 μC takes only 6 μA when in low-power mode; as opposed to 35 mA when running. However, when the signal is being sampled at higher and higher frequencies, these hybrids turn on more and more – the longer the processor is on the more energy it would take. Take the example from Section 1.4, at 50 kHz, the μC would be turning on every 20 microseconds, just to sample the point. Now, it may only take a few microseconds or less to sample a point, but if signal is acquired just for a minute, that's nine seconds of on-time, just acquiring. This doesn't account for the time it would have to be left on for the additional calculation. From the benchmark, a 32-point finite impulse response (FIR) filter, similar to a MF, would require an additional 13 microseconds [21]. This would be another 39 seconds of on-time. With just two operations, the μC would already be on for 80% of the recording. Further, this is for only one channel and these structures can be up to 16 electrodes. The consequences of this in an embedded system would be that any battery put into the system would be drained, and would have to be replaced every couple of days – something undesirable. Section 1.4 and the current

Section carve out a very specific niche for the kind of signal processing needed. This will be further discussed in Section 1.6.

1.6 Filling the Niche

From Sections 1.2–1.3, a gap has developed between the information stream coming from the electrode interface and how it's processed. Sections 1.4 and 1.5 has shown that processing quickly and efficiently is no simple task. This particular task requires a very specific niche processor. As shown in Figure 1.1, the list of processors are greatly along a trade-off line, where better performance requires more energy, and less energy yields lower performance. The theoretical processor needs to fill the niche in the area above μCs and to the left of DSPs; the niche of high performance, but low power. This work aims to show that this need can be filled by analog computing, which will be discussed further in Section 1.7.

1.7 History of Analog Computing and the Extended Analog Computer (EAC)

Analog computers were a method of computing developed before the current age of digital computers. They were originally designed to solve ordinary differential equations, specifically for ballistic applications, in the 1940s and 1950s [22]. This technology was developed into the General Purpose Analog Computer (GPAC) by Shannon in 1941 [23], refined by Pour-El [24], and later advanced by Lipshitz and Rubel [25]. The GPAC was first built with mechanical components, but later versions were electronic. It had to be first described as a mathematical model, which then resulted in an analog device with finite analog components, such as resistors or capacitors. The GPAC was considered a simple device that could do linear computations such as summation, multiplication and simple integration, all in real-time. However, Rubel did not believe the GPAC was able to solve problems such as decision making processes due to the linear nature of the device - it simply could not handle non-

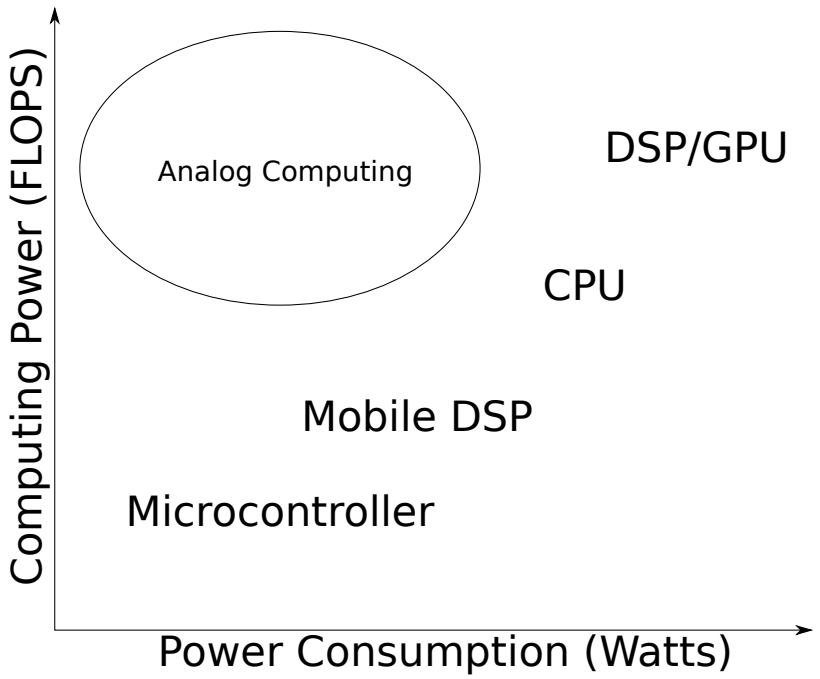

Fig. 1.1. List of Processors that could theoretically complete the spike sorting problem and where they would sit on a scale of performance (x-axis) vs. power consumption (y-axis).

linearities [26]. Therefore, L.A. Rubel developed an implementation of the GPAC, known as the extended analog computer (EAC) in 1993 [27].

The EAC took advantage of the intrinsic computational power of distributed potentials in some conductive space. This allows for calculations to be performed as quickly as the potential can form – essentially at the speed it takes an electron to move. Mills et al., manufactured an EAC consisting of a conductive sheet and a matrix of contacts, bounded by insulated boundaries [28]. The EAC was described as a uniform volumed conductor with Neumann boundaries and a regular array of input/output contacts. Each contact could be configured as a current source or current sink - sources were inputs and sinks outputs. This architecture, developed by Dr. Mills and colleagues at Indiana University Bloomington in 1995, was based on sam-

pling electric current or voltage distribution in a conductive foam sheet. This EAC was simple and proved to be able to compute solutions to complex problems [29].

1.8 Problem Statement

In his work, Soliman took the work by Mills et al. and extended it to the more specific application of neural signal processing [18]. He codified the fuction of the EAC and defined an algorithm to map any Finite Impulse Response (FIR), Infinite Impulse Response (IIR), FIR-IIR filter onto the EAC. He also demonstrated that the EAC was capable of implementing not only a Bessel Filter, but more importantly, a MF capable of identifying SFAPs to a very low, even fractional, signal-to-noise ratios.

However, both of these filters were realized by calculating the coefficients of the filter by hand. This is not desirable for a portable device or neuroprosthesis, being that new filter coefficients would have to be calculated every time there was a slight change in electrode placement. This issue of ever changing parameters lends itself to a filter with the ability to calculate these coefficients automatically. This is the first aim of the present work; to implement automatic machine learning/training methods to automatically configure the weight values of the EAC.

Other issues that arose in Soliman's work was that the MF was great at identifying the SFAPs in high levels of noise, but it could not classify them easily. This comes from the linear nature of the MF and the non-linear divisions of the SFAPs in principal component space. The solution to this problem is to combine multiple EACs into a network of EACs, in order to overcome these non-linearities. Physically, a network of EACs is rather bulky - a proper network would require some 40 sheets and over 1000 wires for connections. This is neither conducive or appropriate for packaging into an embedded device. Therefore, the second aim of this work is to optimize the geometry of the EAC, in order to minimize size of the EAC networks.

Even with the optimization of the geometry of the EAC, these networks are still rather bulky. An integrated chip is needed that contains some implementation of

the EAC networks. The results found in the first two aims will drive the final aim, which is to implement a physical instance of an EAC network that can realistically be utilized in an embedded device. This instance may not specifically be the same structure as a network of EACs, but should still be fully analog and behave similar to the EAC network.

1.9 Organization of Thesis

This thesis roughly follows the work done in a chronological order to help the reader understand the process that lead to the final outcomes.

Chapter 2 covers the initial attempts to use machine learning algorithms, specifically the Delta rule, to train a simple linear perceptron (LP) to behave as an optimal MF. This chapter reviews the mathematical basis of the delta rule, LP, and the MF. It will also introduce the methods of implementing the delta rule as a means to train a LP and then translating the nodes and weights of the LP to use with the EAC; first as a virtual simulation using COMSOL Multiphysics®. and then as a physical instance of the EAC.

Chapter 3 considers a very important shift in concept. It was discovered that instead of using a two-dimensional media, such as the conductive foam sheets, the EAC could be realized as a resistor ladder. With this change in media, the optimization of geometry to reduce the size became massively easier. Knowing that resistor ladders can be made incredibly small, the focus was not on geometries of sheets, but more on the structure and training methods of the network. This is the main focus of Chapter 3 - the methods used to find the optimal back-propagation training algorithm and structure of the networks of resistor ladders. This was accomplished by considering these resistor networks as artificial neural networks (ANNs). The application considered when optimizing the networks was neural spike detection of SFAPs.

Chapter 4 translates the results of Chapter 3 into the physical instance that was desired in the third aim of this work. Chapter 4 details the design parameters that

were found in the work completed in Chapter 3 and how they were used to design the resulting integrated circuit chip.

Chapter 5 begins looking at the testing of the developed IC to determine the accuracy and possible calibrations of the system. Chapter 5 also demonstrates that the chip, known as the ICAANN, operates similarly to the simulations of the LP in Chapter 2.

Finally, Chapter 6 discusses future directions of this work and summarizes the findings.

2. TRAINING LINEAR PERCEPTRONS AS A MATCHED FILTER USING THE DELTA RULE

The Extended Analog Computer (EAC) is unique in that it can be configured into a filter whose coefficients are determined by the spatial distance between an input pin and output pin. This allows for an extremely low-power, fully analog implementation of many filters. This is realized by simply injecting current at specific points in a conductive sheet. Soliman discovered in [18] that the voltage generated by injecting current through a cylindrical source drops off radially, more specifically, as a 0^{th} order Bessel Function of the second kind. This radially-based voltage profile can be utilized to represent an analog filter's coefficients; as such that every voltage value needed for the filter's coefficients can be found along some ray of a circle. What is more interesting is that these analog filters can also be realized as perceptrons and artificial neural networks, which means these filters can be trained. The following chapter takes this concept and extends it to the EAC, which results in a LP that is optimized to a MF that can identify a SFAP in real-time.

2.1 The Linear Perceptron

The linear perceptron (LP), first envisioned in the late 1950s by Rosenblatt [30], was an attempt at mimicking the neural receptors in the retina. A perceptron is a layer of N inputs and a bias term each connected to a single output by a weighting factor. The output node acts as both a summation node and a threshold function. This is illustrated in Figure 2.1.

Figure 2.1 is an N^{th} input perceptron and analyzing it can be very complex. It is more desirable to view the perceptron in a simpler form. The simplest example of this perceptron is a case of a network with two inputs and a bias term. The following

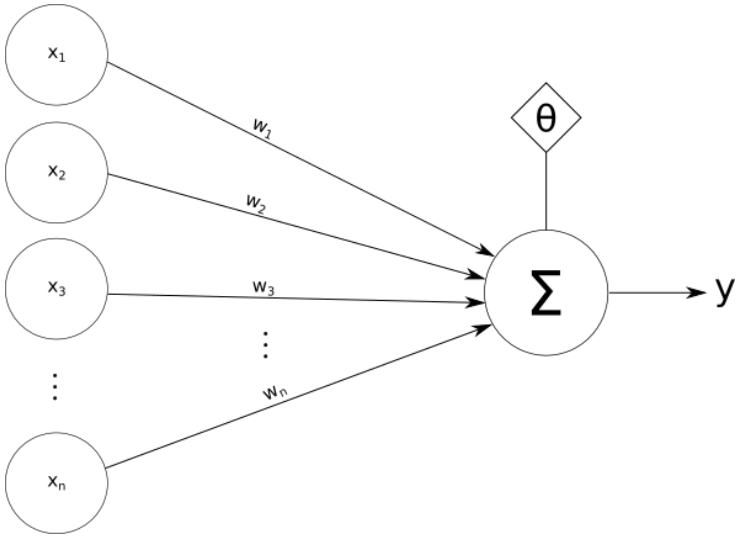

Fig. 2.1. Simple Illustration of a Linear Perceptron. $x_1...x_n$ are inputs to the LP, $w_1...w_n$ are the weights connecting the inputs to the summation node in the next layer, θ is the bias term, and y is the output of the LP.

derivation can be seen in greater detail in the work of Kröse [31]. The output junction acts as a summation node generating the output function as in Equation 2.1.

$$y = \sum_{i}^{N} w_i x_i + \theta \tag{2.1}$$

Where x_i is the i^{th} input to the system, w_i is the i^{th} connection to the output y, known as the weighting function. If a Heaviside function, as described in Equation 2.2, is utilized as a thresholding function the perceptron can be used in a classification task. The Heaviside function constrains the output to either -1 or $+1$ dependent on the input allowing for discrimination between the two classes.

$$\Phi(threshold) = \begin{cases} 1, \ threshold > 0 \\ -1, \ otherwise \end{cases} \tag{2.2}$$

When considering only two classes, equation 2.1 can be simplified to equation 2.3. These two classes can be separated by a straight line, a linear discriminant function, described as Equation 2.3.

$$w_1 x_1 + w_2 x_2 + \theta = 0 \tag{2.3}$$

Solving for x_2, we can generate the linear discriminant function (Equation 2.4) that will separate the two classes. All points above the line are one class, all those points below that line are a different class.

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{\theta}{w_2} \tag{2.4}$$

Here the ratio of weights determine the slope of the line, and the bias sets the offset. Initially, the weights and biases are set randomly, and as can be seen by the yellow line in Figure 2.2, the initial linear discriminant function can be quite inaccurate. This result is due to the randomization of the initial weights and biases. To remedy this issue, training of the weights and biases must take place.

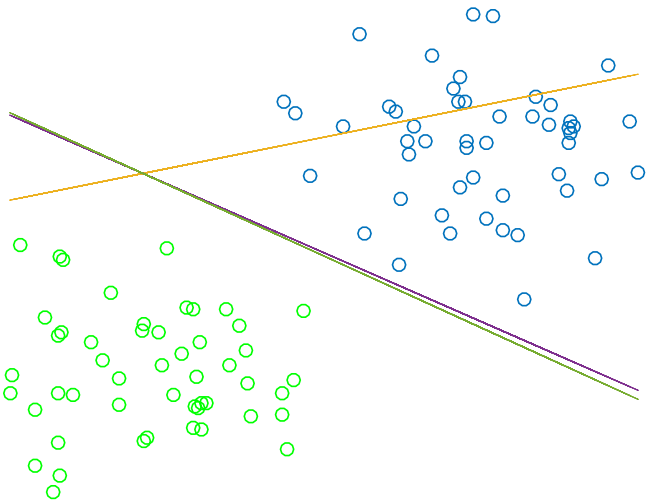

Fig. 2.2. Demonstration of simple linear discriminate function (LDF). The yellow line is the initial random guess made by the LDF, and the purple and green lines represent the following two iterations of training.

The solution utilized in this work is known as the delta rule [30,31]. It uses error between a desired output and the actual output of the LP, the inputs to the LP, and a learning rate. The delta rule is further described in Equation 2.5,

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} \tag{2.5}$$

where Δw_j is the change in weight, η is the learning rate, w_j is the j$^{\text{th}}$ weight, and E is the least mean squared error term described as in Equation 2.6,

$$E = \frac{1}{2} \sum_{n=1}^{N} (d_n - y_n)^2 \tag{2.6}$$

d_n is the desired output for the n^{th} output and y_n is the actual output of the perceptron for the the n^{th}. For a linear perceptron N = 1, therefore the summation is not needed. The partial derivative of error with respect to the j^{th} weight can be broken down further via the chain rule, as in Equation 2.7.

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial w_j} \tag{2.7}$$

By observation of the definition of y from Equation 2.1 and E from Equation 2.6, the equation can be further reduced to Equation 2.8 and 2.9, respectively,

$$\frac{\partial y}{\partial w_j} = x_j \tag{2.8}$$

where x_j is the j^{th} input to the LP, and

$$\frac{\partial E}{\partial y} = -(d - y) = \delta \tag{2.9}$$

where d is the desired output, and y is the output of the LP, resulting in the following final equation, Equation 2.10

$$\Delta w_j = \eta \delta x_j \tag{2.10}$$

Δw_j is then used to update the weights, as in Equation 2.11

$$w_j = w_{j-1} + \Delta w_j \tag{2.11}$$

Using the delta rule, the purple and green line in Figure 2.2 were found, and it can be qualitatively determined that there is an improvement in the separation of classes.

The delta rule is a very basic form of a machine learning algorithm known as back-propagation that can be used to train networks with more layers [31], which will be discussed in Chapter 3.

2.2 The Matched Filter

The MF is a type of FIR filter that acts as a basic correlator detecting the presence or absence of a known template within a noisy channel. The MF operates by cross-correlating the known template with the unknown signal and grading the response. The underlying equation that dictates the MF response is as below.

$$y(n) = \sum_{i=0}^{M} b_i x(n-i) \tag{2.12}$$

Where *y(n)* is the filtered signal at time point n. This output is normalized to the largest amplitude and ranges from -1 to 1; -1 indicates inverse correlation, 0 indicates no correlations, 1 indicates positive correlation. Here b is a set of M weights that define the MF that make up an M order filter. *x(n-i)* represents the present time signal for each discrete time increment in the past.

The output of an optimized MF should be the auto-correlation function of the template when the template is centered, i.e. there will be a peak. Since the result of the MF is the auto-correlation, it would also make sense that the coefficients of the filter would be a time reversed image of the input. Both of these traits are used later to determine the efficacy of training.

2.3 Methods

This chapter focuses on adapting the EAC to a LP which is then optimized to a MF. To do this a few things had to be considered. This section describes how the EAC was simulated, how pin placement was determined, and how the spacing issue of the pins was handled. The type of signal used to test the MF, how that signal was "digitized", and injected into the conductive media are also expounded upon here.

2.3.1 Simulating the Conductive Sheet

The first step in adapting the EAC to a LP was to find a proper way to model the EAC's electrical properties, so the proper voltage profile could be utilized as weight values. This could be done with similar methods as Muller [18] using the Finite Element Modeling (FEM) software COMSOL Multiphysics®(COMSOL Multiphysics, v5.2, www.COMSOL.com COMSOL AB, Stockholm, Sweden). The EAC model that was utilized is the same as described in [18], where the conductive media was two types of boundary conditions - an insulated boundary, which reflects the potential and a grounded boundary where the potential is held at zero. These boundary conditions were modeled as a Neumann and Dirchlet boundary, respectively in COMSOL. Current (1 A/m^3) was injected into the media via a cylindrical source at the corner of the two insulated boundaries. This resulted in a quarter circle voltage profile.

2.3.2 Using the Delta Rule to Place Current Injecting Pins

Placement of the pins in the EAC determine the coefficients for the MF, as well as the weight values for the LP. Keeping this in mind, it is very important to place the pins precisely in the EAC, in order to get the best results. To do this, a specific electrical property, known as reciprocity theory, can be used.

The concept of reciprocity can be explained as a linear relationship between a current sink and source, such that the voltage from a source observed at a sink in

a conductive medium would be the same if the sink and source were flipped [32]. Using this idea, all the values needed to define the MF could be generated by simply injecting current into the media at a point source at the corner of the media. Since the values of voltage drop off continually in a radial fashion [18], all coefficients needed for the filter can be found on one of the insulated boundaries, which happened to be a ray of the circle. However, the simulation of the model had only a finite number of solutions based on the fineness of the mesh.

As stated in Section 2.2, the output of a MF optimally will be the auto-correlation function of the template. With this in mind, a LP was trained via the delta rule described in Section 2.1. The training algorithm used a known template as input and its auto-correlation function as a targeted output. Once the weights of the LP were found, each weight value was then found along the ray of voltage values via a binary search. When these values were found, their corresponding position in centimeters, from the source, along the EAC were placed.

2.3.3 Solving the Spacing Issue

With a fine enough mesh, all the coefficients were found along the ray of the EAC. However, there was a problem that arose due to the low amplitude of the signal. The values of the coefficients were so close together that the values found along the ray were bunched together, often only micrometers away. This was not conducive to placing relatively large pins that were on the order of millimeters. A relatively easy method to solve this issue was to scale the values of the weights.

Even by spreading the points out by scaling, some of the points were so close together they were still micrometers apart. By using a rotation matrix this issue could be remedied as can be seen in Equation 2.13.

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{2.13}$$

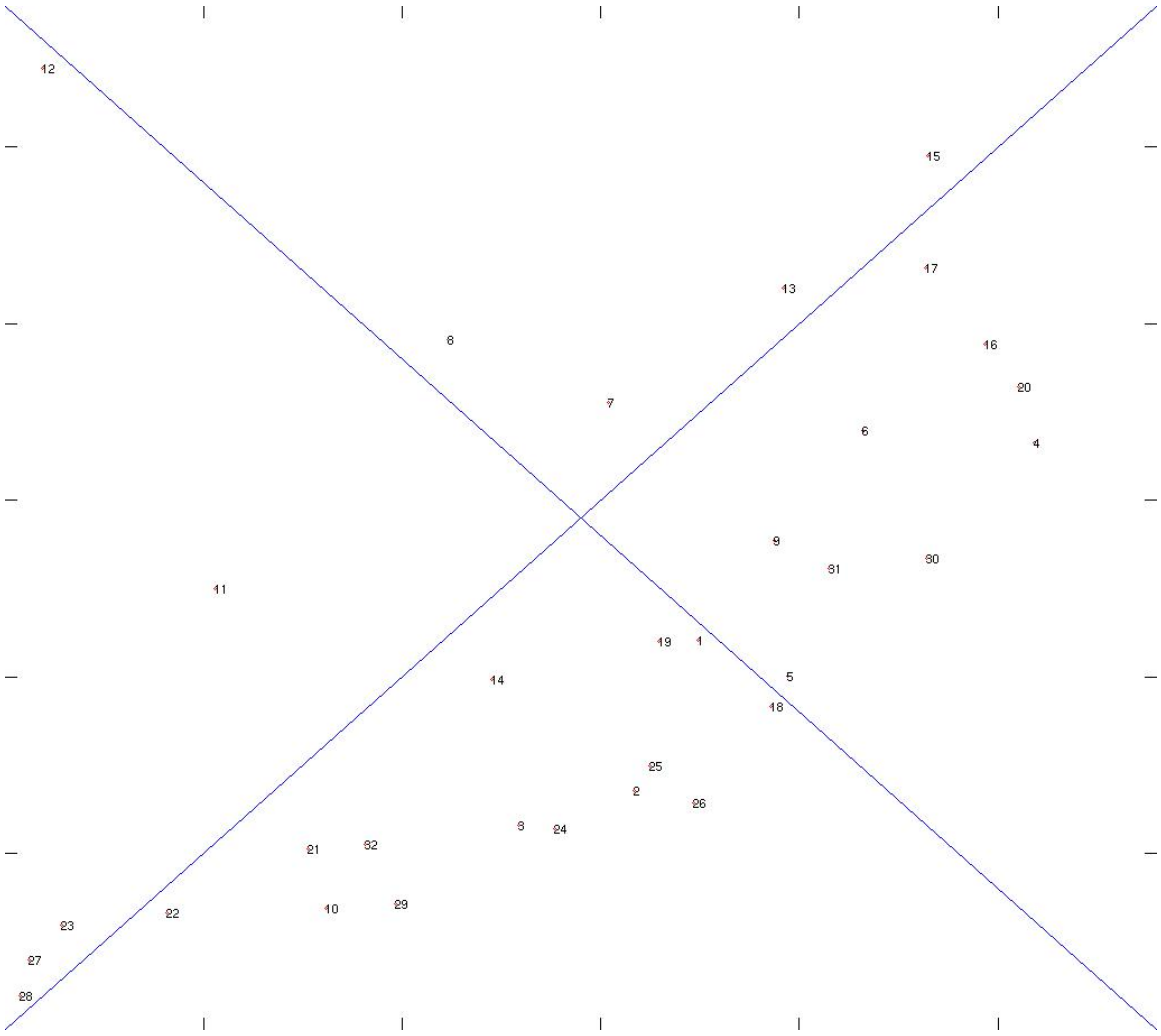

Fig. 2.3. Programmed sheet of the first SFAP. Each point is numbered so that pins can be placed properly for the filter.

where x_0 and y_0 are the original xy coordinates, and the x_r and y_r are the rotated coordinates. An angle θ was selected at random (to ensure spread of points), between 0 and 90 degrees and each point was rotated to that (x_r,y_r) coordinate. This resulted in the final programmable sheet shown in Figure 2.3, which was used in the physical demonstration of the LP as a MF.

2.3.4 Sample and Hold Circuitry

The EAC acts as a spatial filter; when a current is injected into a certain point in the medium, the voltage output drops exponentially because it is being spatially filtered. However the MF and other FIR and IIR filters are temporal filters. When injecting the current into the medium, the signal had to be sampled in order for the filter to function properly. There needs to be a spatio-temporal mapping occurring, therefore an analog sample and hold S/H circuit was utilized. This circuit allows for the values of the EAC to be mapped from space to time in a fashion that allows for the voltage values to be leveraged as filter values.

A S/H circuit is an analog circuit that will capture a voltage value and keep that value for a specific time. This acts as a stabilizer for many analog-to-digital converters; keeping the input signal from varying to much, thereby corrupting the conversion. Typical S/H circuits consist of at least one operational amplifier (op amp), a switch, and a capacitor. An example of a simple S/H circuit is shown in Figure 2.4.

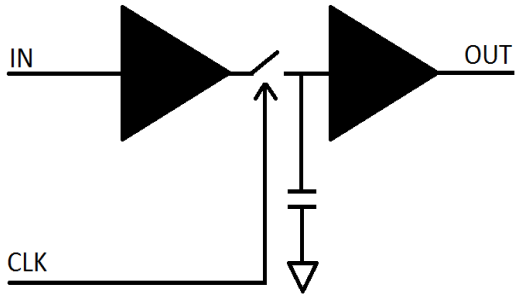

Fig. 2.4. A simple S/H circuit.

For this example, when the switch is closed, the signal is being sampled continuously, as the voltage is being passed through the circuit. Then, if the switch is open, the charge is held on the capacitor and this is where the signal can be read or transferred.

However, for the purpose of this project, a more complex system needed to be utilized - a system that actually simulated memory. To do this, a pair of S/H circuits were used to form a transmission gate, where the first S/H would transmit the signal to another S/H circuit, acting as a memory buffer. A simple block diagram of this "double-clutch" circuit is shown in Figure 2.5. The result of this circuit is that the value held in Tap 1-B could be transmitted while the Tap 1-A sampled. This circuit was constructed into a 32-input daisy chained "bucket brigade", using 64 total AD781 SH amplifiers (Analog Devices, www.AnalogDevices.com, Norwood, MA, USA), that would pass signal down its taps and generate a discrete signal on which the MF could operate.

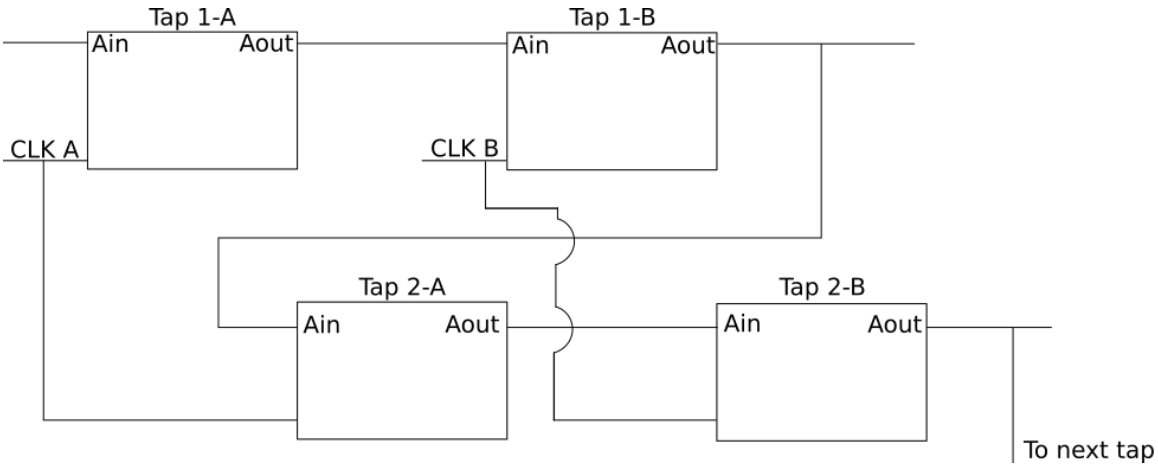

Fig. 2.5. SH double clutch circuitry used in present work.

2.3.5 Single Fiber Action Potential Templates

To test if the MF could identify different signals, SFAP signals that were recorded from a rabbit's sciatic nerve as described by Qiao in [33] were used. There was a total of seven unique SFAP units from the data. Each template was sampled at 96 kHz for 2.67 ms, resulting in 256 sample points. Each template was then decimated using MATLAB's (MATLAB 2015a, Mathworks, Inc, Natick, MA, USA) *decimate*

function, resulting in a 32 point waveform. The resulting waveforms can be seen in Figure 2.6.

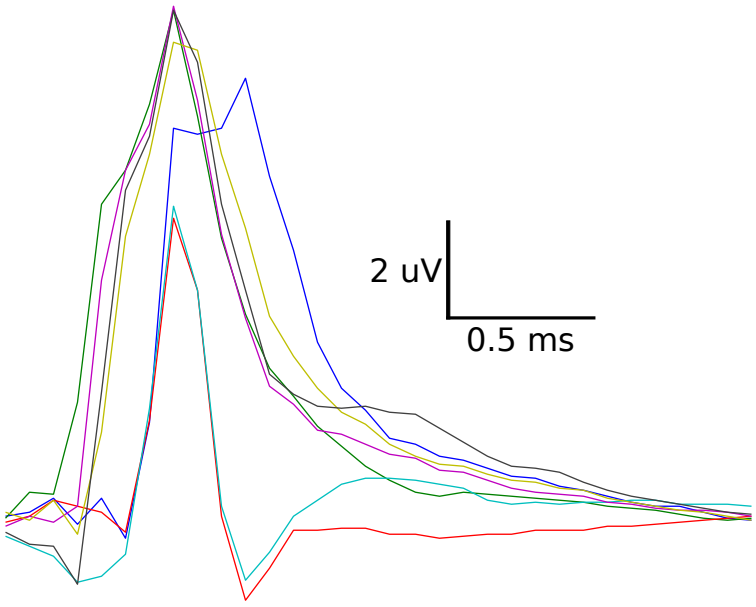

Fig. 2.6. Templates used for training the LP as a MF. These Templates were recorded using TIME structures from the sciatic nerve of a white New Zealand rabbit. The signals were evoked by moving the ankle joint of the rabbit.

After decimation, Gaussian white noise, peak to peak with signal to noise ratio (SNR), from 1:1 to 33:1, was added on top of the data to make training more robust. It was also necessary, due to the signals small amplitude (microvolt scale), to auto-scale the data by dividing by the standard deviation of the signal. This protocol of generating training and testing signals is consistent for Sections 2.5 and 3.8.

2.4 Simulations of Linear Perceptron in MATLAB

Before the physical demonstration, it had to be verified that the simulation in Section 2.3.1 properly trained the LP as a matched filter. To do this, two checks were ran; the impulse response and the SFAP response.

The impulse response can be seen in Figure 2.7. From Section 2.2, it's known that the impulse response of the MF is a time reversed image of the given template. In Figure 2.7, it can be verified that this is the case. The output of the network in response to an impulse, shown in blue, is a time reversed image of the SFAP template, shown in green. This is the first verification check.

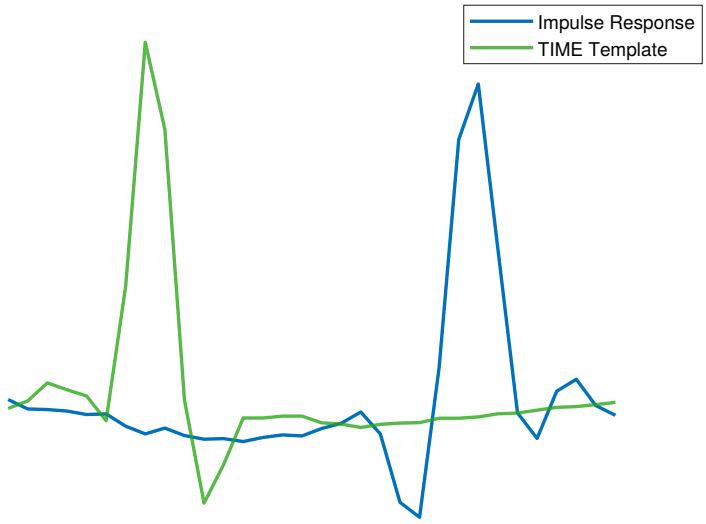

Fig. 2.7. The impulse response of the LP (blue) compared with the SFAP template (green) demonstrating that the LP has been properly trained as a MF.

The second check is the LP's response to a SFAP. Again, in Section 2.2, the output of the MF when the trained signal is present is the autocorrelation of the trained signal. However, in Figure 2.8, it can be seen that the output of this particular LP is not a perfect match to the autocorrelation (in green). This is indicative that the network was not properly trained. This could be because the network was not given enough time to converge properly. Other LPs, trained for other SFAPs, have a better fit, but their impulse response do not look like an exact match. Again, this could be due to training time, but could also be a problem with the simple training algorithm used.

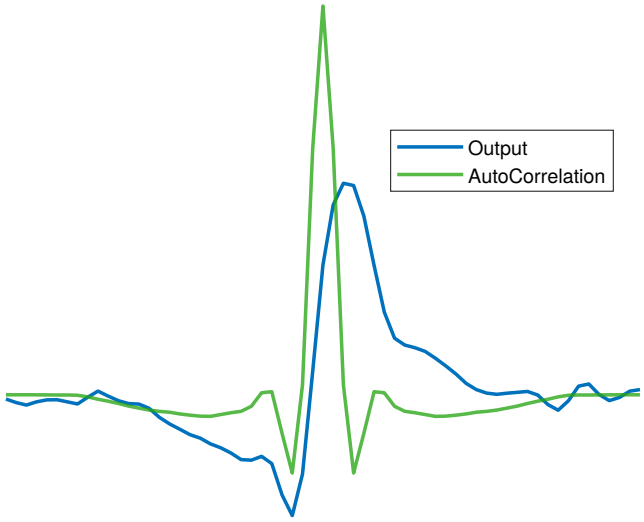

Fig. 2.8. The output of the LP when given a SFAP template (blue) compared with the autocorrelation of the SFAP (green) to demonstrate that the MF is trained properly.

2.5 EAC as a Linear Perceptron Running Real-Time Matched Filter

The final step to adapt the EAC as a MF trained as a LP was to run the MF in a physical implementation, rather than in simulation. To do this, a 12x12 inch square of conductive foam was cut. The sheet had an approximate conductivity of 0.3 S/m. Two adjacent sides of the foam were covered with copper electrical tape (3M, www.3M.com, Maplewood, Minnesota, USA) and then spray painted with a Super Shield^{TM} liquid nickel paint (MG Chemicals, www.mgchemicals.com, Surrey, B.C., Canada) to form a reliable electrical connection with the foam. This generated the grounded boundary as seen in simulation. The S/H circuit and foam sheet can be seen in Figure 2.9. From here, wires were soldered to 23G hypodermic needles (BD, www.bd.com, Franklin Lakes, NJ, USA), which were used to inject the current into the foam. A programmed sheet, like the one shown in Figure 2.3, was printed and placed in the corner of the adjacent insulated boundaries.

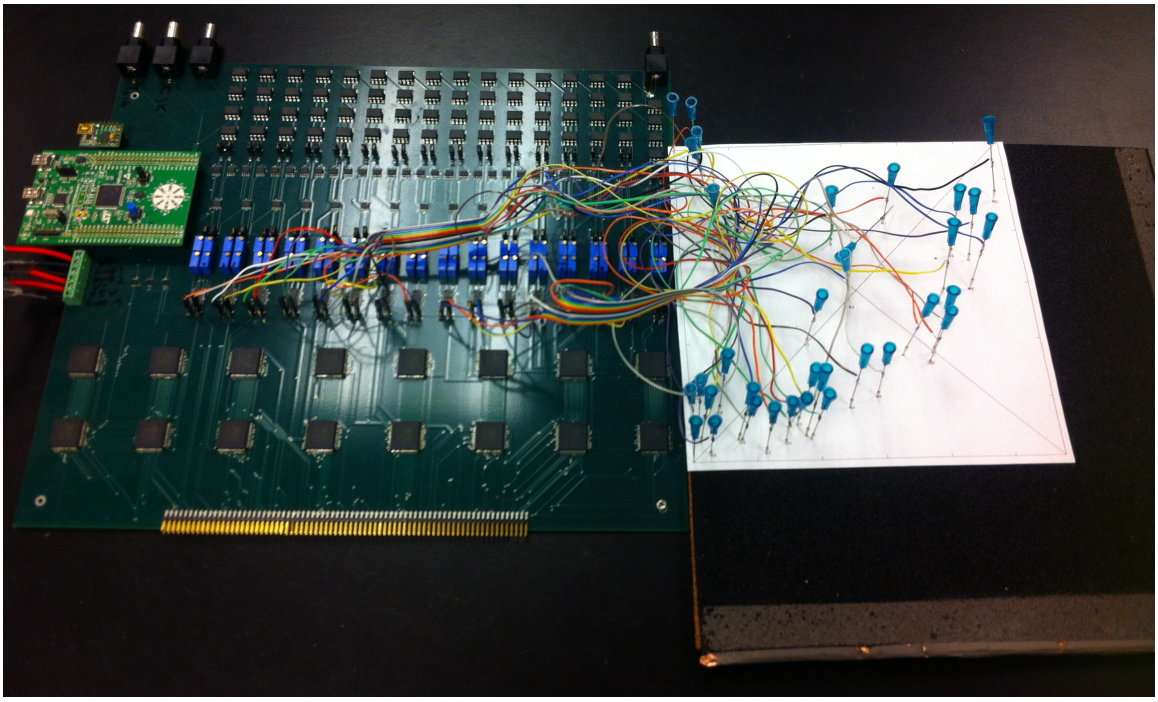

Fig. 2.9. Physical implementation of the LP. To the left is a S/H circuit that will sample the signal and inject it into the conductive sheet to the right.

Thereafter, analog signals were generated by a Rigol DG5072 arbitrary waveform generator (Rigol, www.rigolna.com, Beaverton, OR, USA) and fed through the S/H circuitry. This sampled the signal at 32 samples per clock cycle, so they could be pushed through an array of voltage to current converters and then injected into the conductive foam board. A single output pin was placed in the corner of the insulated sides.

Once the signal was injected into the foam sheet, an output formed, as can be seen in Figure 2.10. Here, the yellow waveform is the first SFAP template being injected into the conductive foam and the blue is the resulting output from the EAC. It should be noted, that some tuning took place to get this desired output. The SH circuit had to be clocking the input at a specific frequency for this output. Anything other than that frequency, the output would have been diminished. Regardless, this

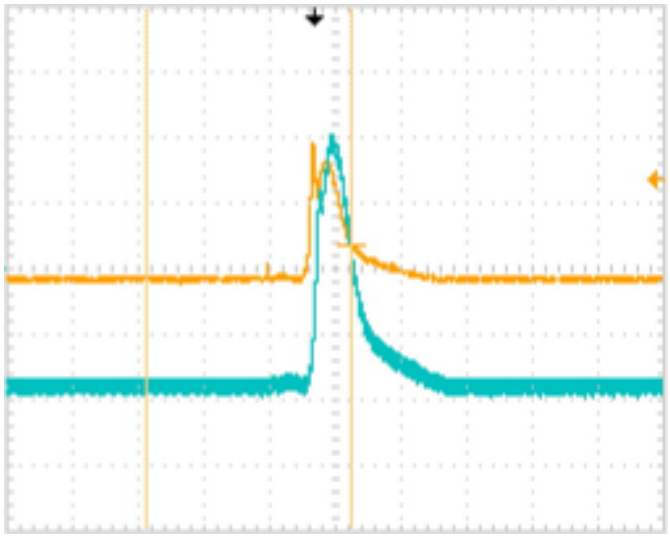

Fig. 2.10. Oscilloscope shot from a SFAP (yellow) going through the conductive foam and resulting in a MF response (blue).

output demonstrates that the EAC could be implemented as a MF which could be trained as a LP.

3. SIMULATING ARTIFICIAL NEURAL NETWORKS FOR NEURAL SPIKE DETECTION

With the physical demonstration of the EAC's power as a MF, the concept of connecting multiple EACs together into a network became highly likely. However, with the increase in network complexity, this would entail more complex training methods. With impending increased intricacy, came a realization that allowed for a shift in media, concept, and focus of the project. This lead to work that is the focus of this Chapter, simulation of artificial neural networks (ANNs).

3.1 Shift in Concept - from Conductive Sheets to Resistor Ladders

As stated in section 1.7, the EAC had many applications that were found useful. However, soon after the initial physical demonstration of the EACs power as a MF, it was realized that the EAC could be reduced from a two dimensional sheet to a one dimensional array of resistors. The idea was further supported by Karplus [34], which states that analog resistive networks of one, two, and three dimensions all can be useful solutions to both Laplace and Poisson's equations. In Soliman's work, it has been demonstrated extensively that the EAC operates as two-dimensional resistive network [18]. Therefore, it is logical that the EAC's ability to be trained as a LP to be optimized to a MF can be realized in the one-dimensional space.

This completely changed the approaches taken moving forward. Not only did it allow for miniaturization orders of magnitude smaller than could have been realized with EAC sheets, but it also freed time that would have been earmarked for geometric considerations to shrink the EAC. This time could now be spent on exploring the optimal network to solve specific applications.

3.2 R-2R networks

An R-2R network is an electrical circuit made of a repeating pattern of resistors in a ladder-like fashion, hence its other name, a resistor ladder. It is a combination between an analog system and a digital system, where each input ($a_{n-1}...a_0$) can be switched between $V = 0$ (low) and V_{ref} (high) and the digital bits are weighted based on their contribution to V_{out}. Depending on the which bits are high and which are low, V_{out} will have some weighted value between 0 and V_{ref} minus the value of the minimal weight value (where all bits are zero). Figure 3.1 is an example of an R-2R network structure.

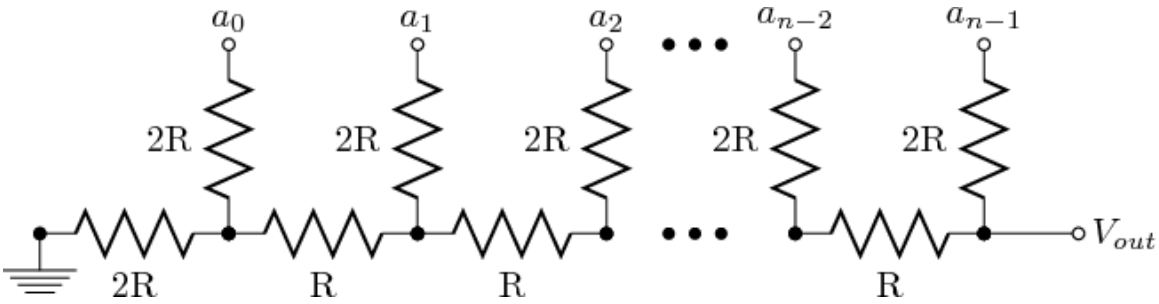

Fig. 3.1. A diagram of a typical R-2R network. The most significant bit (MSB), a_{n-1}, through least significant bit (LSB), a_0 can be gated high or low to measure the contribution of each part of the resistor chain.

Using this technology, it was possible to develop a system of weights for a LP that could be programmed via digital means. Not only that, but as mentioned in Section 3.1 this structure could be miniaturized very easily and effectively. However, there are some parameters that need to be considered. The number of resistors in the structure are very important – they set the number of bits in the system. Theoretically, there could be an infinite number of bits in a resistor ladder, but this would take up a ridiculous amount of space, even when size is reduced. This would also require an infinitely large address space to address all the segments of the R-2R ladder; something that is not very viable. Large number of bits is not always

needed; often lower resolution is sufficient enough. The resolution needed is actually analyzed in Section 4.1.3. Unfortunately, smaller resistor networks also have an issue, besides just limiting the resolution of the system. The amount of current needed to make a measurable difference increases the smaller the network gets. Therefore, it is important to balance between resolution need and size of the network.

3.3 Artificial Neural Networks

Artificial neural networks were first devised in the 1940s by Warren McCulloch and Walter Pitts. Their work [35] was based in mathematics and threshold logic, but eventually split into two methods of research - biological processes in the brain and applications to artificial intelligence.

From the biological processes side, Donald Hebb - famous for his saying "Those who fire together wire together" - generated his hypothesis of Hebbian learning, which he considered as a mechanism for neural plasticity [36]. This closely resembled the way that the weighted values of a neural network would become more weighted if they were more "important" and less weighted if not.

As already discussed in section 2.1, the first take at an artificial neural network was the perceptron by Rosenblatt [30]. However, it was found by Minsky and Papert [37] that linear perceptrons could not process a simple exclusive-or circuit, due to the non-linear nature of the problem. This lead to the idea of extending the linear preceptron to a multiple layer perceptron (MLP). This would combined many linear perceptrons into one network, as can be seen in Figure 3.2.

Minsky and Papert later determined that computers were not yet powerful enough to handle large neural networks. It wasn't until Paul Werbos developed back-propagation in the 1970s were researchers able to develop neural networks that could solve the exclusive-or problem [38]. Back-propagation will be discussed in further detail in section 3.4.

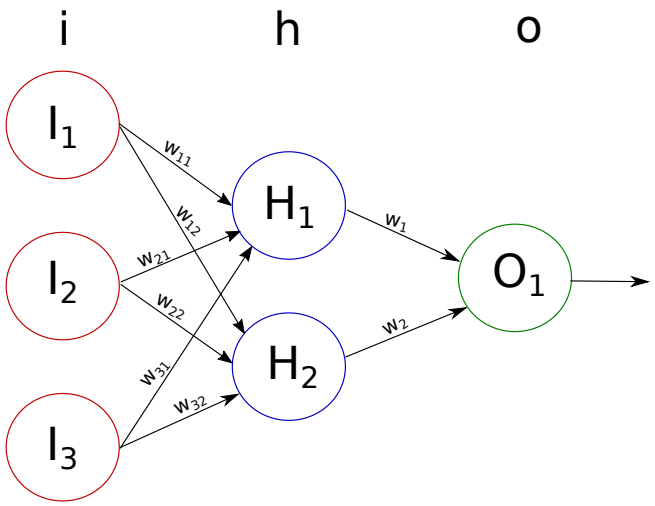

Fig. 3.2. Example of an artificial neural network.

The neural network is really an elegant computing machine; all it consists of is a group of nodes formed into layers. These layers are connected together by weights. The simplest neural network has three layers; an input layer, a hidden layer, and an output layer. An elementary example of this can be shown in Figure 3.2. These nodes contain within them, a summing junction and an activation function.

When a signal is fed through the network, the inputs are multiplied by the weights, forming the output of the input layer.

$$O_i = I_i w_{ih} \tag{3.1}$$

At each node, the weighted inputs are summated as in Equation 3.2,

$$H_h = \sum_{i=1}^{N} I_i w_{ih} = \sum_{i=1}^{N} O_i \tag{3.2}$$

where H_h is the weighted sum of the product of the i^{th} input times the ij^{th} weight. From here, the weighted sum is passed through the activation function, which could be a variety of functions, as long as they are differentiable. In the case of this work a saturated linear function was used, which operated as in Equation 3.3

$$\Phi(H_h) = O_h = \begin{cases} 0, & H_h \leq 0 \\ H_h, & 0 < H_h < 1 \\ 1, & H_h \geq 1 \end{cases} \tag{3.3}$$

The output of the node, $\Phi(H_h)$, is passed to the next layer, the output layer, where the same process is repeated. The weighted sum, which is the input to the output layer, would be as in Equation 3.4

$$I_o = \sum_{h=1}^{N} \Phi(H_h) w_{ho} = \sum_{h=1}^{N} O_h \tag{3.4}$$

The output would then be

$$O_o = \Phi(I_o) = \begin{cases} 0, & I_o \leq 0 \\ I_o, & 0 < I_o < 1 \\ 1, & I_o \geq 1 \end{cases} \tag{3.5}$$

This output can then be taken and used in back-propagation, discussed in section 3.4.

3.4 Back-Propagation

Already discussed in section 2.1, the delta rule can be used to train a simple two input linear perceptron, The delta rule, as demonstrated in section 2.3.2, can be used to train a multiple input linear perceptron, as well. When adding additional layers, however, a more complex algorithm is needed. For training a multiple layer perceptron (MLP), a training algorithm known as back-propagation (BP) is used. The basic design of BP is as follows:

1. Run input forward through network to get output.

2. Determine error between output and target value.

3. Calculate error gradient with respect to weights.

4. Adjust weights based on error gradient and learning rate.

5. Repeat steps 1-4 until error reaches desired value.

Here the function to calculate the error is the mean squared error function as seen in equation 3.6,

$$E = \frac{1}{2}\sum(O_o - t_o)^2 \tag{3.6}$$

where O_o is the output of the o^{th} output node and t_o is the target value of the o^{th} output. For a MLP, the error needs to be propagated backward with respect to the weights through each layer. How the error is propagated changes based on whether the layer is an output layer or a "hidden" layer.

Generally, the delta rule is written as Equation 3.7,

$$\Delta w_{xy} = -\eta \frac{\partial E}{\partial w_{xy}} \tag{3.7}$$

where Δw_{xy} is the calculated change in the weight, w_{xy}, which is the weight connecting the x^{th} node of the previous layer to the y^{th} node of the current layer.

Using the chain rule twice, it can be found in 3.8 that,

$$\frac{\partial E}{\partial w_{xy}} = \frac{\partial E}{\partial O_k}\frac{\partial O_k}{\partial I_k}\frac{\partial I_k}{\partial w_{xy}} \tag{3.8}$$

where O_k is the output of the current layer, k, and I_k is the input to the current layer, k. From here, the individual parts of Equation 3.8 need to be derived. For an output node, the k^{th} layer becomes the output layer or the o^{th}. For the first term of Equation 3.8 yields Equation 3.9.

$$\frac{\partial E}{\partial O_o} = \frac{\partial}{\partial O_o}\frac{1}{2}(O_o - t_o)^2 = (O_o - t_o) \tag{3.9}$$

For the second term as seen in Equation 3.10, considering that we have a saturated linear transfer function.

$$\frac{\partial O_o}{\partial I_o} = \frac{\partial}{\partial I_o}\Phi(I_o) = \begin{cases} 0 & I_o \leq 0 \\ 1 & 0 < I_o < 1 \\ 0 & I_o \geq 1 \end{cases} \tag{3.10}$$

For the third term shown in 3.11,

$$\frac{\partial I_o}{\partial w_{ho}} = \frac{\partial}{\partial w_{ho}}\Phi(H_h)w_{ho} = \Phi(H_h) = \begin{cases} 0 & H_h \leq 0 \\ H_h & 0 < H_h < 1 \\ 1 & H_h \geq 1 \end{cases} \tag{3.11}$$

Putting together Equations 3.9, 3.10, 3.11, yields the equation for an output node described in Equation 3.12,

$$\frac{\partial E}{\partial w_{ho}} = (O_o - t_o) \begin{cases} 0 & I_o \leq 0, I_o \geq 0, H_h \leq 0 \\ H_h & 0 < H_h < 1, 0 < I_o < 1 \\ 1 & H_h \geq 1 \end{cases} \tag{3.12}$$

resulting in the final equation for Δw_{ho}, as seen in 3.13

$$\Delta w_{ho} = \eta(O_o - t_o) \begin{cases} 0 & I_o \leq 0, I_o \geq 0, H_h \leq 0 \\ H_h & 0 < H_h < 1, 0 < I_o < 1 \\ 1 & H_h \geq 1 \end{cases} \tag{3.13}$$

Repeating this process with a hidden layer, each node's contribution to the error is not known. Fortunately, the error can be written as a function of the network's inputs from the hidden to the output layer, as such

$$\frac{\partial E}{\partial I_h} = \sum_{o=1}^{N_o} \frac{\partial E}{\partial I_o}\frac{\partial I_o}{\partial O_h} = \sum_{o=1}^{N_o} \frac{\partial E}{\partial I_o}\frac{\partial}{\partial O_h}\sum_{h=1}^{N_h} w_{ho}O_h = \sum_{o=1}^{N_o} \frac{\partial E}{\partial I_o}w_{ho} \tag{3.14}$$

From equations 3.10 and 3.9, we know that

$$\frac{\partial E}{\partial I_o} = (O_o - t_o) \begin{cases} 0 & I_o \leq 0 \\ 1 & 0 < I_o < 1 \\ 0 & I_o \geq 1 \end{cases} \tag{3.15}$$

for simplicity sake the result of 3.15 will be considered as δ_o, resulting in Equation 3.16

$$\frac{\partial E}{\partial I_h} = -\sum_{o=1}^{N_o} \delta_o w_{ho} \tag{3.16}$$

Which, when added together with the remaining term, yields δ_h,

$$\delta_h = \sum_{o=1}^{N_o} \delta_o w_{ho} \frac{\partial O_h}{\partial I_h} = \sum_{o=1}^{N_o} \delta_o w_{ho} \begin{cases} 0 & H_h \leq 0 \\ 1 & 0 < H_h < 1 \\ 0 & H_h \geq 1 \end{cases} \tag{3.17}$$

Combining these terms, as done previously, the result for Δw_{ih} is shown in Equation 3.18

$$\Delta w_{ih} = \delta_h I_i \tag{3.18}$$

Where I_i is the input to the system.

Taken together, these equations seem quite cumbersome and downright confusion. However, simply put, the process explained above is taking the gradient of the error space and pointing the weights towards the minimum of the space, much like a ball rolling down a valley.

3.5 The Neural Network Toolbox in MATLAB

As can be seen, the back-propagation algorithm can be quite complex and difficult to handle. Unfortunately, this algorithm alone wasn't significant enough to solve the tasks at hand. This is one of the many reasons why in this thesis, after many attempts

at writing an algorithm from scratch, it was deemed necessary to use pre-established algorithms found in MATLAB's Neural Network Toolbox^{TM}. The toolbox provides users with more advanced algorithms to create, train, visualize, and simulate artificial neural networks. It allows for regression, clustering, dimensionality reduction, and most importantly for this work's purpose, classification.

The algorithms used in this work were Gradient Descent with momentum (GDX), Levenberg-Marquardt back-propagation (LM), and Bayesian Regulated back-propagation (BR).These three algorithms added either an adaptive learning rate, cost function regulations, or changes in the calculation of the error gradient to allow for better convergence to the answer.

The most similar to the general back-propagation as discussed in 3.4 is GDX. However, the calculation of of Δw is slightly altered to included a momentum parameter m and an additional term that alters the learning rate, η.

$$\Delta w_j = m\Delta w_{j-1} + \eta m\frac{\partial E}{\partial w} \tag{3.19}$$

Where alterations to the learning rate are determined based on the performance of E. If E decreases towards the goal, η increases by some predetermined $\Delta\eta$. If E increases by more than some max increment in performance, set by the user, the result is thrown out and η is decreased by $\Delta\eta$.

LM changes the way both the error gradient and the weight perturbation by using the Jacobian and an approximated Hessian matrix, rather than a partial derivative. The LM gradient is calculated as found in Hagan and Menhaj [39].

$$J_e = J_w E \tag{3.20}$$

Where J_w is the Jacobian of partial derivatives of error with respect to weights as calculated in Marquardt [40], where Δw is then calculated as in 3.21

$$\Delta w = -(H - \mu I)^{-1} J_e \tag{3.21}$$

Where I is the identity matrix, μ is the adaptive constant dependent on performance of E, such that if $E_{new} > E_{old}$, μ increases by some $\Delta\mu$ and Δw is recalculated until $E_{old} > E_{new}$. In this case, μ is decreased by $\Delta\mu$ and weights are updated by Δw. The Hessian matrix is approximated as in Equation 3.22

$$H = J_w^T J_w \tag{3.22}$$

BR is an extension of the LM process in that it alters the performance function by considering the mean error or the network and the sum of squared error [41, 42] resulting in an error calculation as shown in Equation 3.23

$$E = \beta E_D + \alpha E_w \tag{3.23}$$

where E_D is the mean squared error, E_w is the sum of squared weights. α and β are regulation parameters determined by

$$\beta = \frac{(N - \gamma)}{2E_D} \tag{3.24}$$

$$\alpha = \frac{\gamma}{2E_w} \tag{3.25}$$

$$\gamma = W - (tr(H^{-1})) \tag{3.26}$$

here W is the number of parameters (weights and biases), N is the number of entries in the training set, and $tr(H^{-1})$ is the trace of the inverted Hessian.

Together these three algorithms were tested for their ability to solve different applications which will be discussed further in Section 3.6.

3.6 Applications of Interest

As described in 1.7, the EAC can tackle many applications with some ease. However, it would not be possible to go through every single application the EAC could

complete. Therefore, this thesis focused on an application specific to spike sorting – processing SFAP spikes from TIME structures.

3.6.1 Neural Spike Detection and Classification

SFAPs are the way neurons communicate between the CNS and the PNS, as explained in Section 1.1. Therefore, it is very advantageous for spike sorting to be able to record these transmissions. However, these SFAPs are rather difficult to record consistently. They are often low in amplitude (microvolt scale), and only 0.5-1 millisecond in time. In section 1.2, various interfaces have been utilized in order to tap into these conversations - one of which is the TIME structure, an intrafasicular electrode. With this electrode type, it is much more likely to record from a single fiber, as stated in section 1.2.2. Often, this electrode type can record multiple fibers in the same fasicle, leading to differences in shape, amplitude, and even time base; adding another layer of complexity to processing these signals' intent. This leads to the need of not only a spike detector, but a classifier, as well. It has already been established that the MF can be applied as a detector, but fails when trying to classify [18].

The work done by Dr. Qiao [33], found a set of SFAPs that were used in this thesis as templates for this application. The SFAPs were treated the same as in section 2.3.5.

3.7 Methods

In an ANN, there are many parameters that need to be tuned, in order to optimize the network for an application. These parameters such as: the number of nodes at each layer, or the training algorithm used can change the outcome of the network drastically. In this chapter, these parameters were tested in order to optimize the ANN for the application stated in section 3.6. Prior to showing their ability in this application, however, some steps had to be taken to properly train the network. Once

the network was trained, a method of testing the network had to be developed, as well. Thereafter, a process of determining the failure/success of the network had to be built.

3.7.1 Training the Network

The mathematics for training the network - such as the approximation of the error gradient and altering of the weights - were handled by the Neural Network Toolbox™. However, the development of the training sets was left to the user. In the SFAP application, the neural network converged to what is known as a Bayesian Classifier (BC), where the output of the network is some probability that the signal presented is the desired signal. Therefore, instead of using the autocorrelation function, as in section 2.3.2, the target output would be 1 when the signal is "centered" and 0 when it is not.

To develop training sets used for the SFAP application, a template 96 points long was made. This 96-point template was constructed from a decimated 32-point TIME template (see section 2.3.5), and two 32-point "noise" pads on either side of the TIME template. A 32-point windowing function was developed as to break the 96-point template into 64 snapshots of the signal. Along with each snapshot, the signal was given either a 0 or 1, depending on if it was a snapshot of a "centered" signal (1) or not (0). This process was repeated multiple times, adding Gaussian white noise on top of the template, for each of the seven TIME templates, resulting in over 8000 examples of the signal. This was done, in order to generalize the network so that it could determine classifications of signals when they were contaminated with noise. The relatively large size of the training data was also to offset the Neural Network Toolbox's™ built-in function that would divide the total set into training, testing, and validation data.

3.7.2 Testing the Network

As stated in section 3.7.1, the total set of data generated is split into training, testing, and validation data by the Neural Network Toolbox™. This allows for testing of the network during the training period to assess if the network is trending in the right direction. It also can be used to grade the network, as well. However, the built-in testing does not account for time. Therefore, this testing method was not consistent with a real-time operation that was desired.

This issue was solved by generating a test set that mimicked real-time data and feeding it into the network 32 samples at a time. At first, this synthetic data set was generated by concatenating a decimated TIME template (see section 2.3.5) selected at random, adding a 32 point pad of noise to either side, and adding Gaussian white noise on top of the now 96 point template. This process was repeated 100 times and the templates were concatenated for a 9600x1 array. Unfortunately, it was later found that this method did not allow for even representation of each signal. It was then determined that each template should get their own test set; meaning that each template had a 9600x1 array of test data that consisted of 100 concatenated templates of the same type with Gaussian white noise contaminating the signal.

Each of these test sets were ran through the network 32 points at a time, and the output was taken for grading - as discussed in Section 3.7.3.

3.7.3 Grading the Network

The grading of the network is greatly determined by a few parameters: network accuracy, inclusion and exclusion error, and confusion. Network accuracy is described as the number of correct identified and classified SFAPs out of total SFAP presented. For example, if 20 SFAPs are presented and only 10 are identified and classified properly, the network accuracy would be 50%. Inclusion error is where an output is identified and classified when no SFAP was present at the input. Exclusion error represents when an SFAP is present, but does not get identified and classified. Con-

fusion is where the presented SFAP is identified, but classified into the wrong class (i.e., class 1 presented, but classified as class 2). The output of the network also had a set threshold of 50% to be considered signal; meaning the that the network had to show a probability of higher than 50% to be counted as a spike. This was done to reduce some of the noise that is inherent in the network.

All this information can be constructed into what is known as a confusion matrix, where the network accuracy can be found along the diagonal of the matrix, the inclusion error along the far right column, and the exclusion error can be found along the bottom row. The off-diagonal components of the matrix represent the confusion. For simplicity in analysis, the network accuracy, inclusion, and exclusion errors were averaged across classes.

3.7.4 Neural Spike Application

For the SFAP detection and classification application, the main focus was finding the optimal training algorithm and network structure. For the training algorithms, the three methods discussed in section 3.5 were tested against one another for network accuracy, as well as, inclusion and exclusion error.

The structure of the network was also tested for these same parameters. However, it became apparent that the parameter that had the most impact was the number of output nodes. Therefore, this parameter was heavily considered here. The structures of interest were the multiple unit classifier, where it was trained to distinguish between all seven templates plus noise, and the single unit classifier, which was only trained to recognize one template and reject everything else as noise. Each level of consideration (i.e. training algorithm, network structure) was replicated 25 times to decrease variance.

3.8 ANN Classifying SFAPs in Simulation

The first demonstration of the Neural Network Toolbox's^{TM} power will be to show that they can accurately identify and properly classify SFAP signals from TIME structures. This can be shown in Figure 3.3. Each hash mark represents a point in time where the SFAP would be "centered" in the window, and therefore would be present. As can be seen in Figure 3.3, it is apparent that this particular classifier, which was trained to identify class one, captured every signal that was present. It

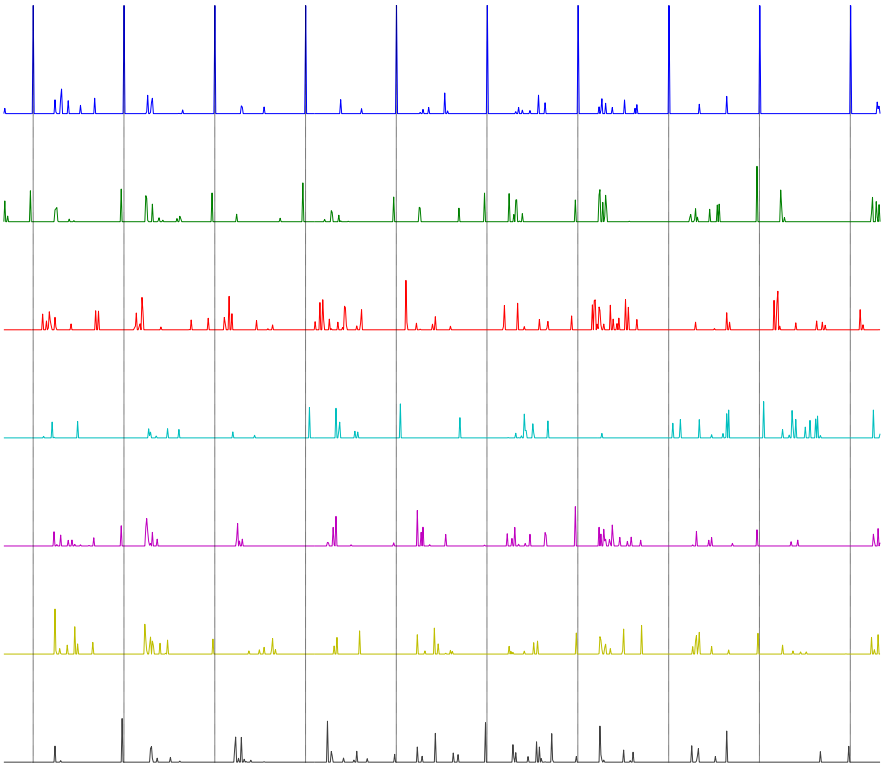

Fig. 3.3. Output of the ANN that is trained for class one. Every has hash mark represents where a class one template was present. It can be seen that this classifier captures every signal presented.

may also be noted, that there is some "noise" associated with the other channels. This is due to the nature of the classifier; in that it is giving an output of how likely the signal presented to the network is the signal it was trained to identify. Shown

in Figure 2.6, these SFAPs are distinct, but similar. Therefore, sometimes a portion of a different signal may confuse the network, causing a small peak. However, the thresholding used prevented anything smaller than 50% to be rejected as noise.

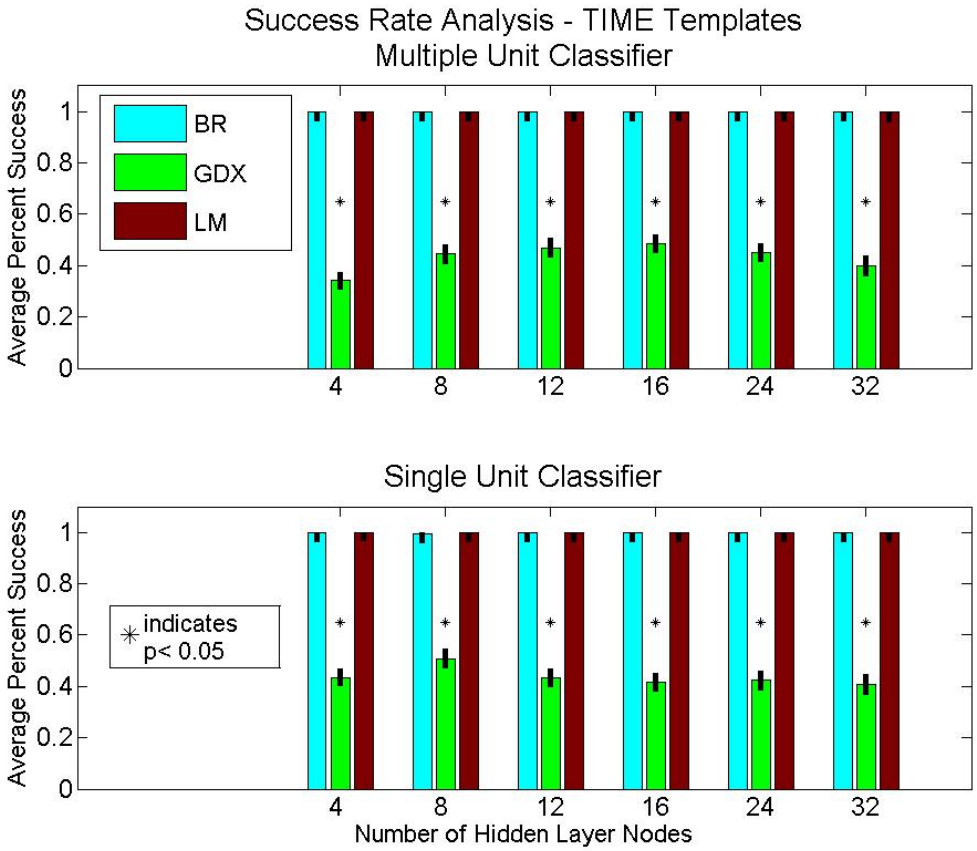

Fig. 3.4. The network accuracy of each training algorithm for both the multiple and single output neural networks. The star indicates a significant difference between variables.

The network outputs in Figure 3.3 - which came from a single unit classifier using the BR training algorithm - was not always mimicked by other networks trained using different algorithms. When testing for the network accuracy, not all training algorithms tested well. It can be seen, in Figure 3.4, that GDX does not perform above 50% overall network accuracy on either single or multiple unit classifiers regardless of hidden layer nodes. This would exclude GDX from consideration of further

study. However, it would appear that both LM and BR training algorithms performed equally well, showing 98% or better network accuracy across all network architectures.

However, an observation was made when examining the outputs of the networks - the inclusion error was almost double in multiple unit classifiers than in single unit classifiers. This spurred an investigation into the inclusion error of each network structure. The results found in Figure 3.5 shows that, in fact, the average inclusion error for single unit classifiers is about half of the inclusion error of a multiple unit classifier. This made the choice of a single unit classifier as the optimal network structure.

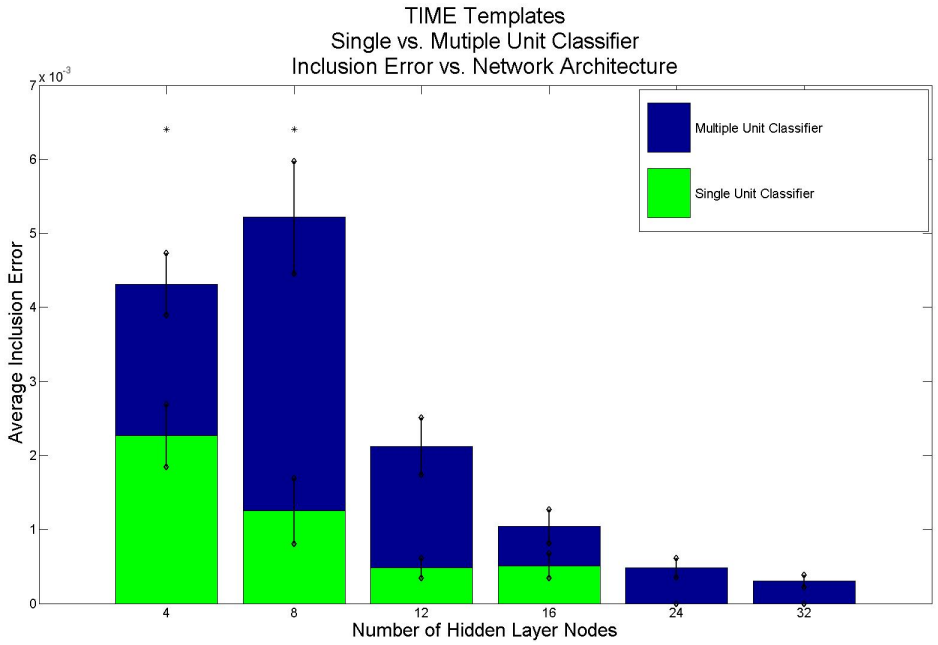

Fig. 3.5. The average inclusion error for both multiple and single output neural networks. The star represents a statistical difference of alpha = 0.05.

There is a trade off to this choice. With the multiple unit classifier, there is only need for a single network. Whereas, with the single unit classifier, a network is needed for each class. This increases size at the expense of accuracy is a design constraint that will be discussed further in Chapter 4.

4. DESIGN OF THE INTEGRATED CONFIGURABLE ANALOG ARTIFICAL NEURAL NETWORK (ICAANN) CHIP

With the shift in concept, discussed in Section 3.1, it became apparent that these networks can be realized in an integrated circuit (IC). However, this work is not the first attempt at the realization of an ANNs in an IC. There have been multiple attempts at this end explained by Jabri et al., Misra et al. and Draghici et al. [43–45]. Therefore, this chapter concerns itself with the development of chip parameters and the search to fit those parameters with current technology.

4.1 Design Constraints

Taking a step back, it must be remembered that the end goal for this work is to take an ANN and drop it into use on an embedded system that will hopefully better the life of the individual using it. With this in mind, a few design constraints must be considered. It should be said that the solution to these design constraints were optimized for the neural spike application. All methods in this chapter are the same as in Section 3.7.4, but the repetition had been decreased to eight rather than 25. All training, testing, and grading was conducted as in Section 3.7.1-3.7.3.

4.1.1 Network Accuracy

Network accuracy is of utmost importance when considering the constraints for this IC. If the network cannot accurately identify and classify then it would be worthless to the user. It would cause frustration and eventually lead to disuse of the prod-

uct. Throughout every level of consideration for constraints, network accuracy was the variable of measure. Accuracy was measured as detailed in Section 3.7.3.

4.1.2 Network Size

The size of the chip is a parameter of great importance. If it cannot fit into an embedded device, then there is simply no difference than the user being attached to a desktop computer. It was this consideration that brought the design constraint of size to the forefront. The size of the chip is directly related to how many nodes are in the network - the more nodes the larger the network would have to be. It has already been determined in Section 3.8 that the optimal number of output nodes is one and from Mirfakhraei et al. [46], the optimal number of input nodes is 32 for SFAPs. Therefore, the number of hidden layer nodes (HLN) was the only layer considered. The size of the chip is also related to the resolution of the weight space, which is discussed in Section 4.1.3.

4.1.3 Weight Space Resolution

The weight space in a neural network is one of the most important aspects. It must be an adjustable value that will be multiplied by the input. In simulation, both the weight space and the input had 64 bits to describe them, essentially an analog value. In an IC, this would not be the case. The input will be an analog value that will be sampled by a SH circuit as discussed in Section 2.3.4. The weight space, due to the space constraint, cannot be represented at the same 64 bits as in simulation. From Soliman, it was found that a conductive media, such as a resistor network, could be utilized as a binary tree and represent bits [18].

This leads to a trade off between weight space resolution and the size of resistor network and therefore the size of the network. This was an important trade off that had to be balanced.

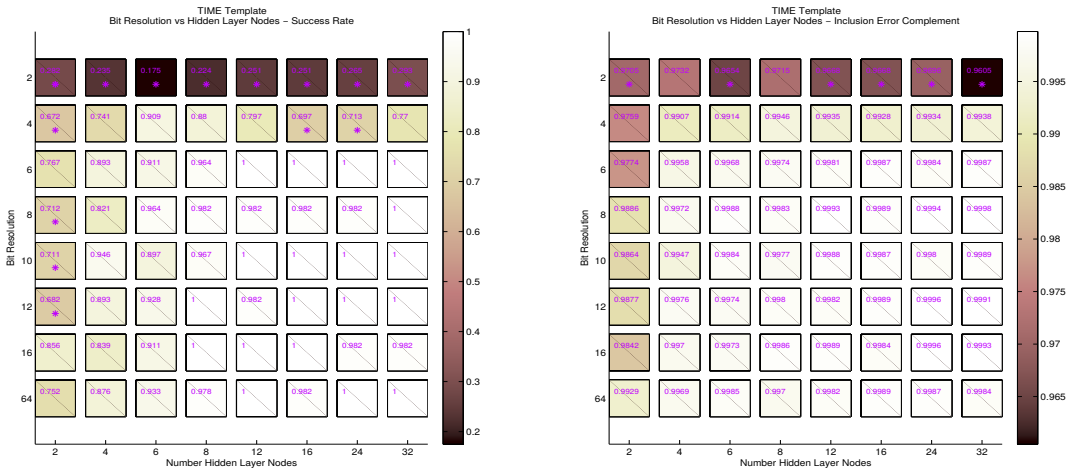

Fig. 4.1. The results of the 2-level analysis of HLN versus bit resolution. On the left is the network accuracy from 0-1, and on the right is the inclusion error's complement (1-inclusion error), also 0-1. For this figure, white is closer to 1 and brown is closer to 0, as indicated by the color bar. A 2-way ANOVA test with Bonferroni Post-Hoc test was used to determine differences in network accuracy, $\alpha = 0.05$.

In order to find this balance, a series of simulations were conducted. ANNs with 2, 4, 6, 8, 12, 16, 24, or a control of 32 HLNs were simulated with a weight space resolution of 2, 4, 6, 8, 10, 12, 16, and a control of 64 bits. Again, as in Section 2.3.5, Gaussian white noise was added on top of each signal with a SNR of 9, peak to peak. To find the most optimal network size and bit resolution, Figure 4.1 can be examined. All networks were compared against the control group (32 HLN, 64-bit resolution). From Figure 4.1, it is apparent that 2 bits of resolution does not perform very well at all, barely exceeding chance (less than 20% accuracy). In the same respect, two HLN does not perform well, hovering between 65-85% accuracy. However, as would be expected, as both HLNs and bit resolutions increase, so does network accuracy. When looking at the inclusion error, there does not seem to be much of a difference

between the ANNs. However, it is still important to have this parameter minimized. The difficult part of this analysis is to find where the "sweet spot" is. This will be addressed further in Section 4.2.

4.1.4 Signal to Noise Ratio

In neural spikes, SNR is a characteristic of a system that causes major issues. Normally, for electroneurograms (ENGs) the SNR would sit around two peak to peak. So it is very important that the ANN be able to identify and classify signals with very low SNR. This was the motivation behind examining SNR as a design constraint.

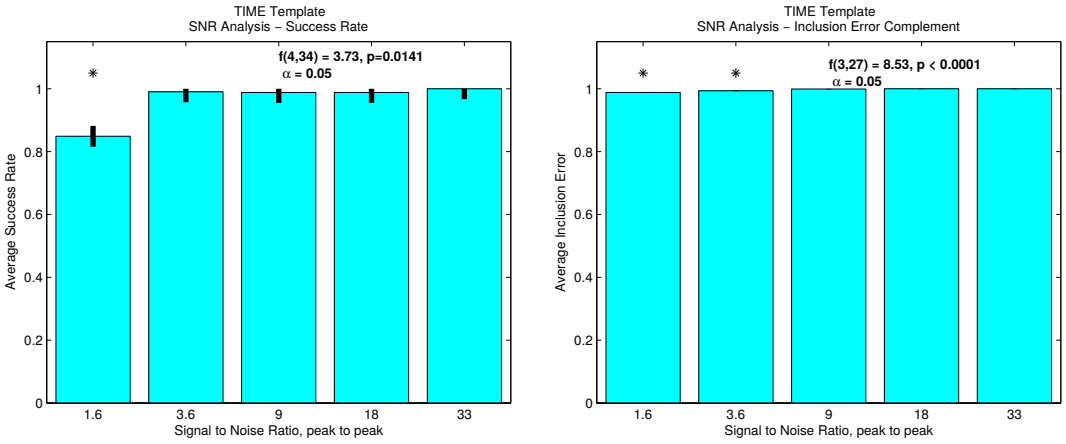

Fig. 4.2. The results of the signal to noise ratio analysis. On the left is the network accuracy from 0-1, and on the right is the inclusion error's complement (1-inclusion error), also 0-1. A 2-way ANOVA test with Bonferroni Post-Hoc test was used to determine differences in network accuracy, $\alpha = 0.05$.

To test the effect of SNR on network accuracy, ANNs with 16 HLN and a weight space of 10 bits of resolution were simulated. The signals that were used to trained and test the network had varying Gaussian white noise of 1.6:1, 3.6:1, 9:1, 18:1, and 33:1 peak to peak.

From Figure 4.2, that SNR does have an effect on network accuracy. However, the only instance in which the network does not perform is when the signal is approximately the same amplitude as the noise (1.6:1 peak to peak). Amazingly even at this level the network still performs above 80% accuracy. Once again, there is an effect of SNR on network inclusion error, but the difference is at SNR values of 1.6:1 and 3.6:1. Even then the difference is fractional.

4.1.5 Weight Accuracy

The way the weight space was chosen to be represented (see section 4.1.3), the value of the weight will not be exact due to noise. This is expected and acceptable to some point. But again, it must be optimized.

To optimize this parameter is very similar to the previous experiments. Networks with 16 HLNs and a weight space resolution of 10 bits were simulated. The network was trained and tested with signals that had a SNR of 9 peak to peak. What was changed, however, was that the weight space had addition Gaussian white noise added on top. This was measured as a 1%, 2%, 5%, 10%, or 25% noise added, meaning the weight could be 1% to 25% off the desired value. All networks were compared to the 1% noise networks, as a control.

From Figure 4.3, there is an effect of percent noise on network accuracy. The performance of the network starts to degrade after 2% weight error, where at 5% it drops to approximately 50% accuracy. It continues to drop to around chance (20% accuracy) with 10% and 25%. Once again, with inclusion error, there is an effect from percent noise, but it is only significant at 25% noise where the inclusion error complement is around 80%. This shows that the accuracy of the weight space is a very important parameter to control.

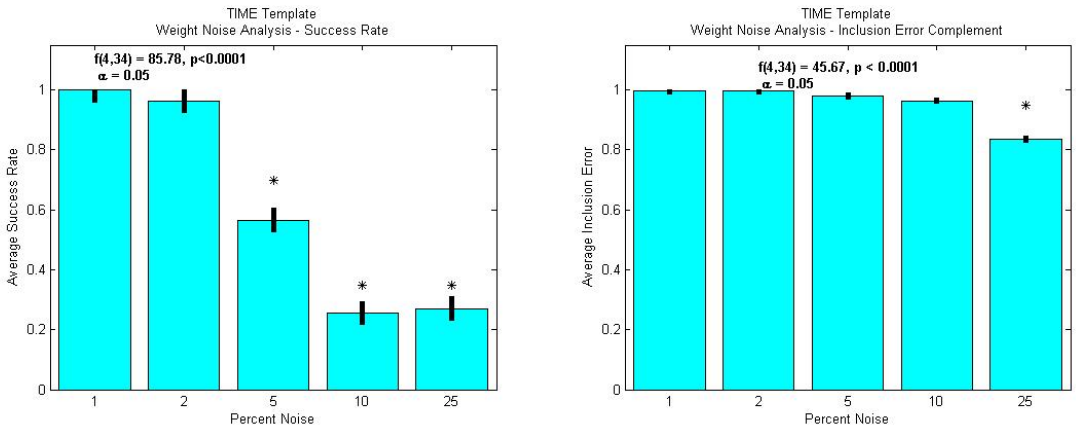

Fig. 4.3. The results of the weight accuracy analysis. On the left is the network accuracy from 0-1, and on the right is the inclusion error's complement (1-inclusion error), also 0-1. A 2-way ANOVA test with Bonferroni Post-Hoc test was used to determine differences in network accuracy, $\alpha = 0.05$.

4.1.6 Transfer Function

As part of each node of the ANN is the transfer function. This transfer function can be any function that has a derivation, as detailed in Section 3.4. However, this work is somewhat restricted due to the use of MATLAB's Neural Network Toolbox™. The four transfers functions considered here were *satlin*, *satlins*, *logsig*, *tansig*. Both *satlin* and *satlins* are linear, but *satlin* ranges between 0 and 1 and is asymmetrical, and *satlins* ranges from -1 to 1, and is symmetric. *Logsig* and *tansig* are both non-linear transfer functions, but *logsig* ranges between 0 and 1 and is asymmetrical, and *tansig* ranges from -1 to 1, and is symmetric. This is an important constraint to consider, because it will determine what kind of circuitry is needed for the IC, and could cause some issue with the size of the chip.

These transfer functions were tested very much the same way as previous experiments. Once again, networks with 16 HLNs with a weight space resolution of 10 bits were simulated. A SNR of 9:1 peak-to-peak was used to add noise on top of the

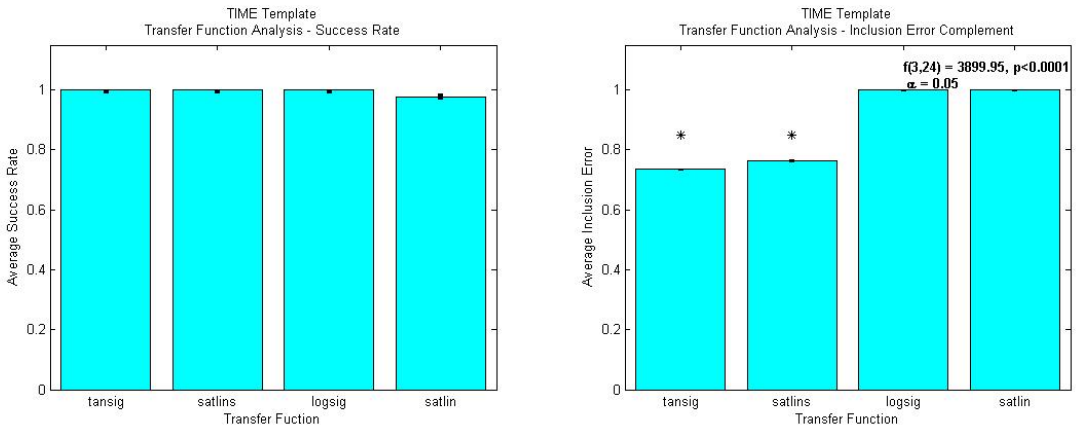

Fig. 4.4. The result from the transfer function analysis. On the left is the network accuracy from 0-1, and on the right is the inclusion error's complement (1-inclusion error), also 0-1. A 2-way ANOVA test with Bonferroni Post-Hoc test was used to determine differences in network accuracy, $\alpha = 0.05$.

signals. And again, a 2-way ANOVA was used to see if there was a difference between *satlin* and the other three.

From Figure 4.4, it can be seen that there is no significant effect of transfer function on the network accuracy. This would indicate that any selection of transfer function would have the same effect. However, for inclusion error, there is a significant effect due to choice of transfer function. The difference is between the asymmetric and the symmetrical transfer functions, with the asymmetrical transfer functions outperforming the symmetrical by almost 30%. This would designate that either symmetrical transfer function would be adequate for realization in an IC.

4.2 Chip Design Parameters Found

From Sections 4.1.1-4.1.6, the design criteria found in Table 4.1 can be determined. For the transfer function, an asymmetrical saturated linear function constituted the most optimal solution. This is due to the simplicity in which a piecewise linear function - such as the saturated linear function - could be implemented in the IC. A

non-linear function, like the tansig function, would be very difficult to implement and could take up valuable space on the chip for not much difference in performance.

The cutoff for number of HLNs was determined to be 8 HLNs or more. This can be determine by looking at Figure 4.1, where the performance begins to drop below about 97% around the 6 HLN area. For bit resolution, the same type of visual analysis was conducted. For this, the same 97% threshold was used in order to find the boundary. From Figure 4.1, it can be seen that this threshold is met at 8 bits of resolution. Therefore, two more parameters were found - number of HLNs (eight HLNs or more) and weight space resolution (eight bits or more).

Table 4.1. Design recommendations found.

Group	Features
Transfer Function	Asymmetric Saturated Linear
Number of HLNs	8 HLNs or more
Weight Space Resolution	8 bits or more
SNR	greater than 5 peak to peak
Weight Space Error	less than 5%

The next parameter that was found was the operating SNR. From Figure 4.1.4, it can be seen that there is not a lot of difference between 3.6:1 to 9:1 peak to peak, but there is a significant drop in performance at 1.6:1 peak to peak. This was a clear boundary point, but for design considerations, it was determined that an operating value of 5:1 peak to peak would be desirable.

The final design consideration that was made was for weight space error. This was to simulate how robust the network's weight space needed to be. From Figure 4.3, it can be seen that there is a clear drop in performance from 2% to 5% error. This is another clear boundary and is the specification that was determined - 5% or less.

5. TESTING THE ICAANN

From the design constraints found in Chapter 4, an IC chip was developed. It was designed by Desert Microtechnology Associates, Inc and was given the name D1503. The D1503 is a prototype design that implements a fully integrated ANN. It was processed using the XFAB XH035 with added high resistance, isolated 5V and MIM capacitor modules. It measures 4.2x5.1 mm, for a relatively small package, as desired. A microcontroller (μC) is used as communication between the IC and user. The μC loads all of the configurations and synapse weights through the serial peripheral interface (SPI) and can be read through the same port. These configurations and weights will be written to registers which can be updated continuously for in-the-loop training which would occur between the μC and the multiplexer of the D1503. The protoboard that is used for the D1503 can be seen in Figure 5.1.

Although the chip functions, it is not without its faults. These faults ranged from improper clocking to memory registers being swapped. Many of these issues were simple to solve, while other required a more sophisticated work-around. Further discussion of the faults of the chip will be discussed in Chapter 6. This chapter will present not only the successes of this IC, but also the work-arounds that allowed those successes.

5.1 Determination of Weight Accuracy and Calibration

The first test that needs to be made is to check the accuracy of and to calibrate the weight space of the chip. To test this accuracy, the voltage from a single output of the chip was measured as the number of weights contributing to that output were incrementally increased. Each weight started with a contribution of zero and one by one they were increased to a gain of 20. This value was chosen because it allowed for

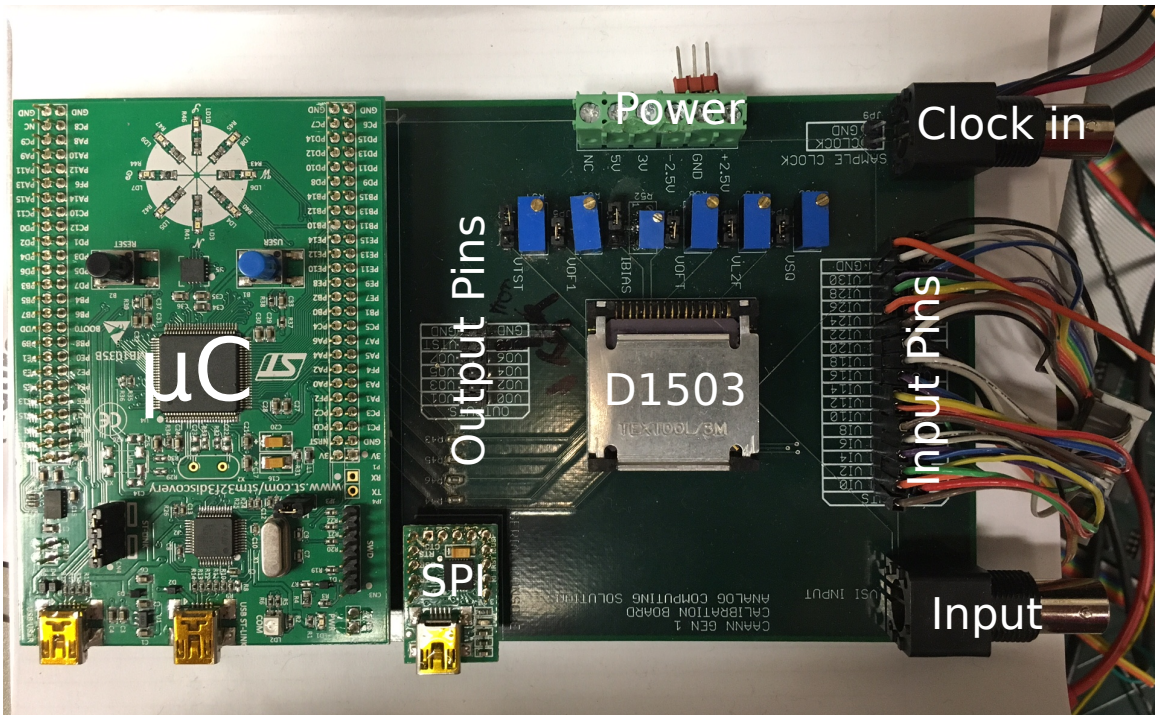

Fig. 5.1. The protoboard that housed the D1503. The board holds a μC that communicates with the D1503 through the SPI. The signal can be input either through the BNC labeled "Input" or through the "Input Pins". "Clock in" is the clock used for the built in SH circuit. Output of the chip can be measured at "Output Pins".

the value to be seen on the measurement device, but did not cause the output of the chip to rail. All measurements were taken with a Rohde & Schwarz HMO1002 series oscilloscope (Rohde & Scwarz USA, Inc., rohde-schwarz.com, Columbia, MD, USA).

The number of weights was increased from 0 to 31 and the corresponding voltage was recorded. From here, the data was input into MATLAB, where a built-in linear regression function was used to find the best fit line through the data. It was expected that the more weights that were added, the higher the voltage would be; resulting in a linear monotonically increasing function. From Figure 5.2, it can be seen that the values collected can be fit to a line (in green) with an $R^2 = 0.9969$, which is a very

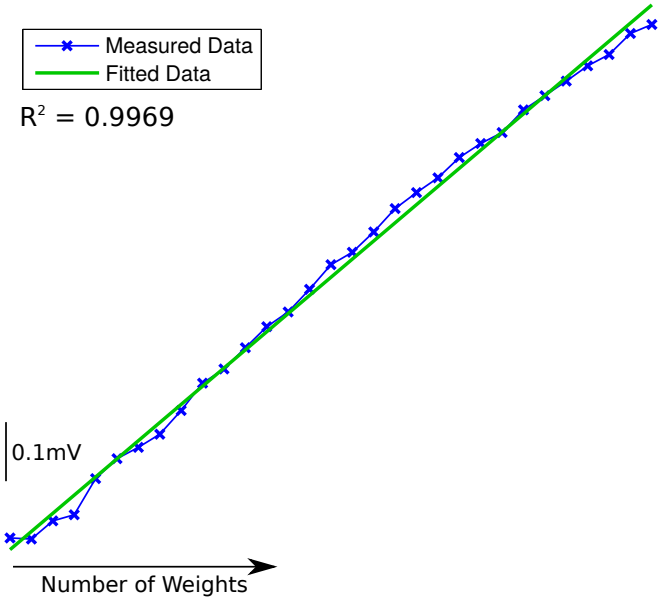

Fig. 5.2. Results of monitoring a single output and increase the number of weights attached to that output. The weight value going into the output node is 20. The result should be linear. This is verified by the fitted line in green. The R^2 value is 0.9969.

good fit. This suggests that the D1503's weight space spatially summates for a single output, and it does so in strongly linear fashion.

The next problem to consider was the behavior of the weights - are they linear or not? A similar method of collection was done for the collection of these weights. Again, a single output was considered, but instead of increasing the number of weights contributing to that output, a single weight was varied over positive range of the weights. Once again, the oscilloscope was used to capture the data and transferred into MATLAB for analysis.

From Figure 5.3, it does not seem that these weights are anything close to linear. There seems to be major irregularities that makes the weight space highly non-linear. However, when observing the individual weight changes on the oscilloscope, the weights would load out of order. This was indicative of an issue with the memory

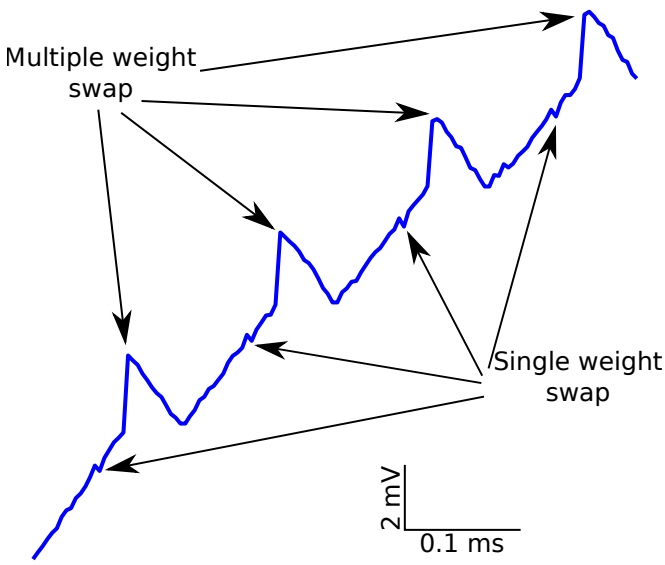

Fig. 5.3. One of the faults of the chip was a memory register swap. This can be seen in the figure above. The weight value increases by a value of one every increment (0-127). The "glitches" that are seen are where the registers are swapped.

registers - a fault in hardware that would require a re-run of the IC. This issue was easily solved by changing the order the weights were presented in software. This resulted in a somewhat linear function, but still has some glitches in the monotonicity, as can be seen in Figure 5.4.

These two considerations taken together suggest that the weight space of the D1503, after some work-arounds, functions as desired and will be suited for the work ahead.

5.2 The Sample and Hold Work-Around

As discussed in Section 2.3.4, how the system "sees" the signal is very important. Therefore, the S/H circuit that was built into the D1503 had to be tested to assure that the shape of the SPAF to be input was being sampled properly. That being said, unfortunately, there were some errors that were found in the chip's S/H circuit. The

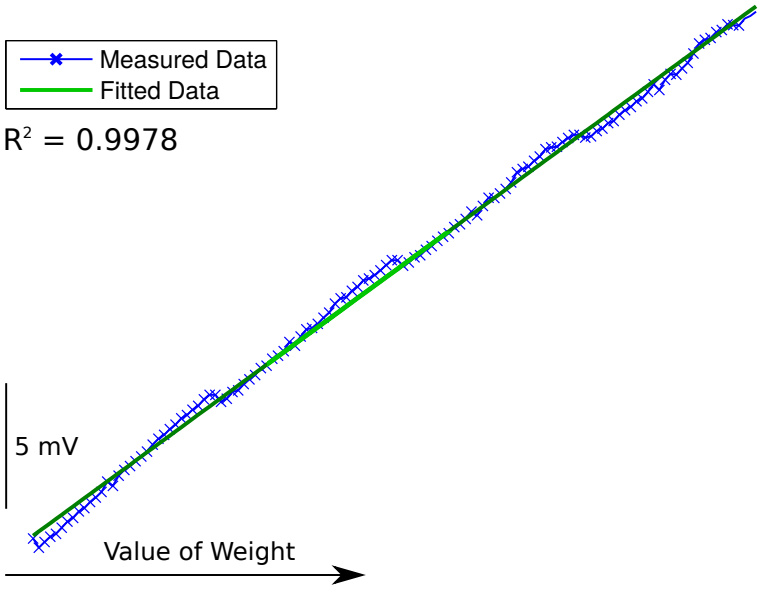

Fig. 5.4. Corrected weight values fitted to a linear function. This was done using MATLABs linear regression tool. The blue x's mark the value measured, and the green line is the linear model based on the x's. There are still some errors in monotonicity.

chip is equipped with the ability to "look-out" certain parts of the output through what will now be known as the Test MUX. This allows the user to take a look at the S/H's effect on the signal as it propagates down the taps of the S/H.

In order to test the S/H circuit, a SFAP was passed through the input of the chip and the S/H circuit. Then the signal was viewed out of the four available points of the Test MUX (0, 15, 16, and 31). If the S/H circuit is functioning properly, then the signal should be delayed by the number of clock cycles of the point the user is looking out (i.e. if the user is looking out 15, the signal should be delayed 15 clock cycles).

Found in Figure 5.5, the clock used for timing the system is in blue and the input signal in green. It can be seen that, in fact, the signal is delayed by the appropriate numbers of clock cycles. However, there is a discrepancy in the timing. Instead of

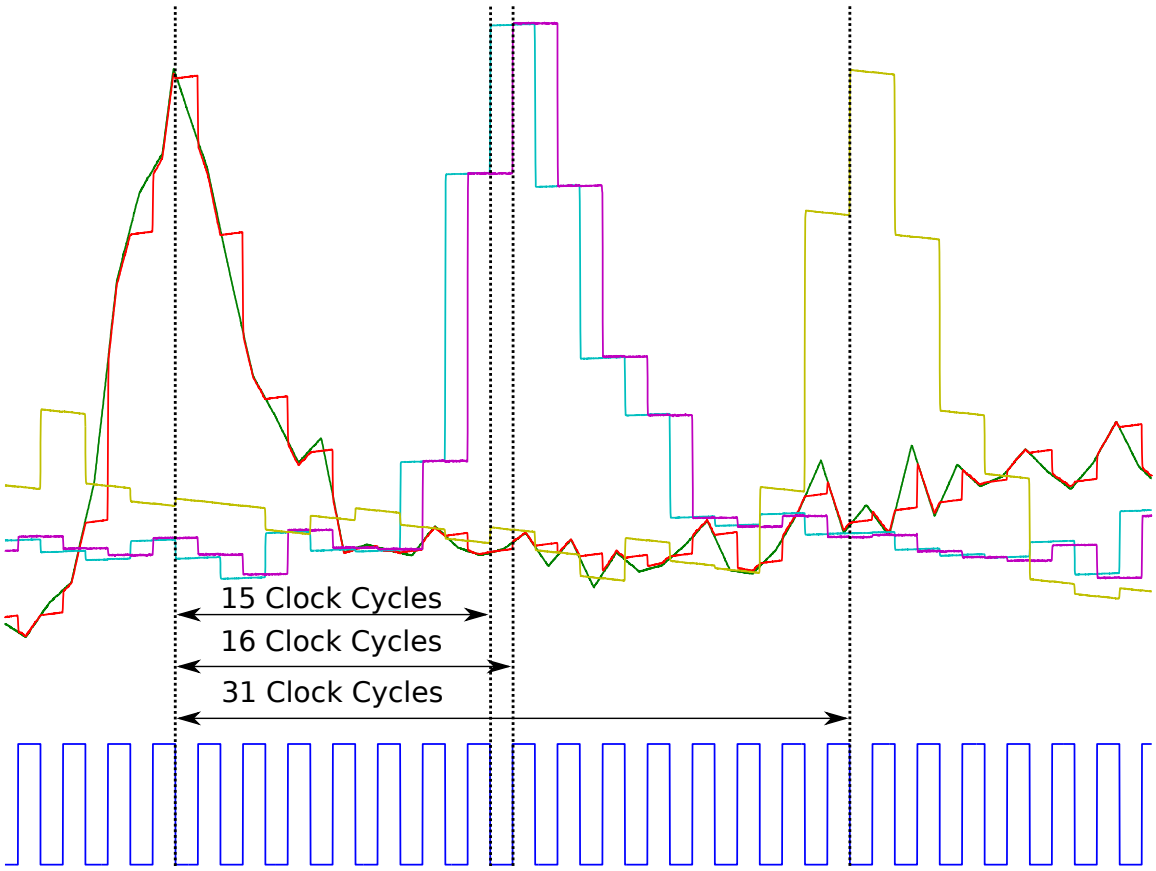

Fig. 5.5. Outputs of the SH circuit build into the D1503. The circuit moves the point down on half clock cycles, but is sampled every clock cycle. This results in a loss of resolution that is shown here. This issue lead to the use of the prototype S/H board from previous experiments (See Section 2.5).

clocking the delays on a full clock cycle, it clocks them on a half clock cycle. However, the sample is held for a full clock cycle. This means that the signal propagates twice as fast as it is sampled. This leads to the slurring of the signal that is seen in Figure 5.5. This is not desirable because the chip wasn't seeing the signal properly and the dysfunction did not allow use of the on-board S/H. To overcomer this dysfunction, the S/H circuit from Section 2.3.4 was used instead. Clocked properly the chip's LP could be test appropriately and connected directly into the 32 input pins on the

D1503, seen in Figure 5.1. This was a rather crude, but a necessary work around that allowed for the proper use of the D1503.

5.3 Impulse response of ICAANN

In the two previous sections, it was shown that the D1503 functions somewhat as hoped. However, these were just preliminary findings, but there needs to be more evidence to show that this IC can function as a LP. Not only that, but the output of the D1503 should be similar to what is seen from the simulations from Chapter 2. In order to demonstrate the D1503's power, the chip had to be programmed properly. The weights for each SFAP were found from the simulations in Chapter 2. These weights were then transformed from the 64-bit float point number to a signed 8-bit number that could be read into the μC via LabView (National Instruments, www.ni.com/labview, Austin, TX, USA) and loaded on the chip via the SPI. The chip was configured using Hyperterminal (Hilgraeve, www.hilgraeve.com/hyperterminal, Monroe, MI,USA)

Once the weights were loaded onto the chip, a pulse template that lasted for two clock cycles was manufactured on an arbitrary function generator (Rigol DG5072 LXI, Rigol, www.rigolna.com, Beaverton, OR, USA) and ran through the S/H circuit and then put into the D1503. The output measured was from the 0^{th} output of the first layer through the test MUX. The resulting waveform was captured by a Rohde & Schwarz HMO 1002 series oscilloscope (Rohde & Schwarz USA, Inc., www.rohde-scwarz.com, Columbia, MD, USA), where the signal was averaged using 1024 sample sweeps. Then the data was collected using HMO Explorer from Rohde & Schwarz into a .csv file, then loaded into MATLAB.

It was observed that the data had a lot of artifact from the S/H circuit, in that the sample of interest was held for a full clock cycle. This could not be easily compared with the simulation strictly because it would be comparing thousands of points to just 32. This issue was solved via MATLAB. Each sample length of the signal had to be

compressed to a signal point. This was done by using the *diff* function in MATLAB to find where the value changed. Once these regions were identified, their averages were taken and made into a single point in a new array. This process was conducted for all seven SFAP template weights.

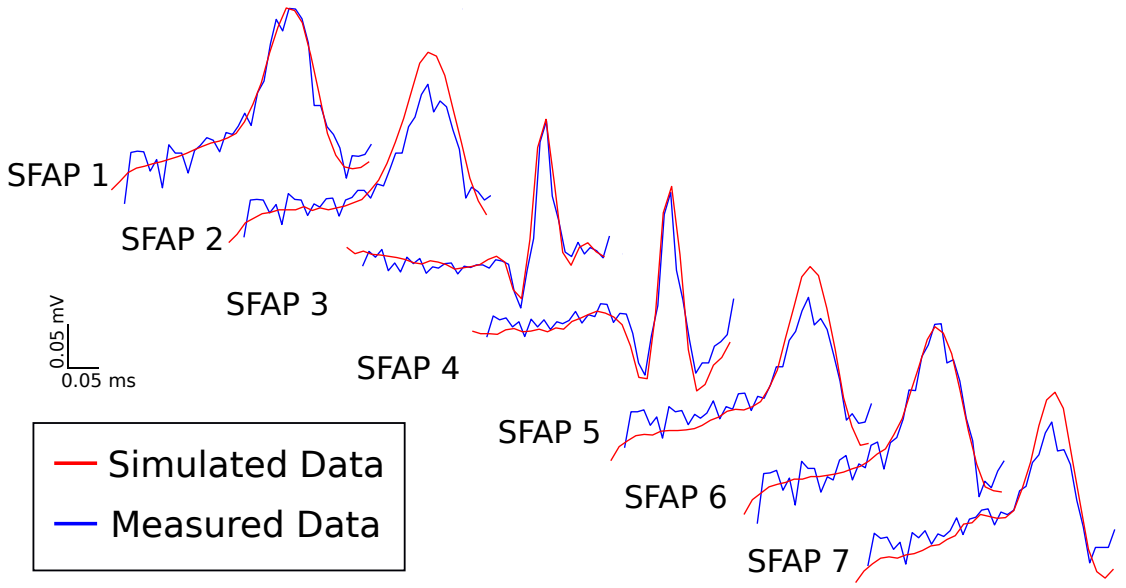

Fig. 5.6. The results of running an impulse response of the ICAANN. There are seven SFAP presented to the ICAANN for both simulated (red), and measured data (blue).

The results of this can be seen in blue in Figure 5.6. The simulated data from Section 2.4 was then superimposed onto the measured data, as shown in Figure 5.6. It can be seen that each SFAP impulse response simulation seems to match quite well with the measured data from the D1503. This is a suggestion that the D1503 performs as a MF.

5.4 Comparison of ICAANN to Matched Filter Simulations

To further support the idea that the D1503 can be programmed like an LP as a MF, the same progression had to be taken as in Chapter 2. This means that each

SFAP template had to be run through the D1503 when it is trained for that SFAP template. The same procedures to program and configure the chip and acquire and process the data were the same as in Section 5.3. The only difference between this Section and Section 5.3 is that the input to the chip is not a pulse, but rather a SFAP template. Each sample of the SFAP template has a length of one clock cycle. This SFAP was pushed through the D1503 and the 0^{th} output of the first layer of the test MUX. Each SFAP was tested for both the weights trained for the SFAP and the weights for other SFAPs.

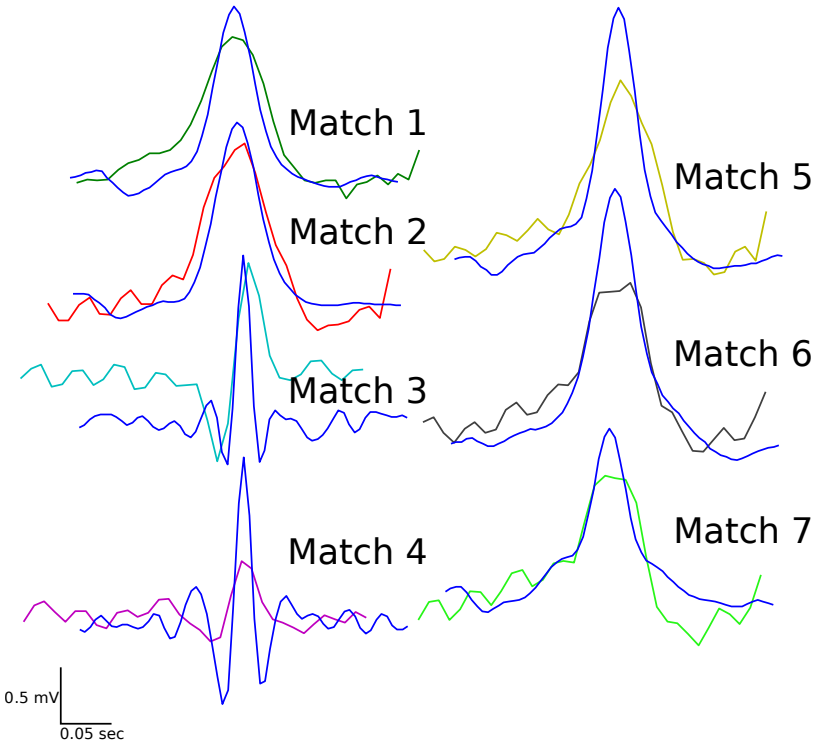

Fig. 5.7. This is a comparison between the simulated data (blue) and the measured data of the MF (multicolored). Each spike is the result of a centered SPAF on the taps.

Figure 5.7 is the result of the each weight set trained for a specific SFAP having their respective SFAP presented. As mentioned in Section 2.2, the output of the MF should be the autocorrelation function of the signal. In Figure 5.7, it can be seen that

the simulation (blue) from Section 2.4 matches each measured data, but not perfectly. This could be due to a multitude of things ranging from the S/H to the value of the weights not being precise to issues with training. This will be further discussed in Chapter 6.

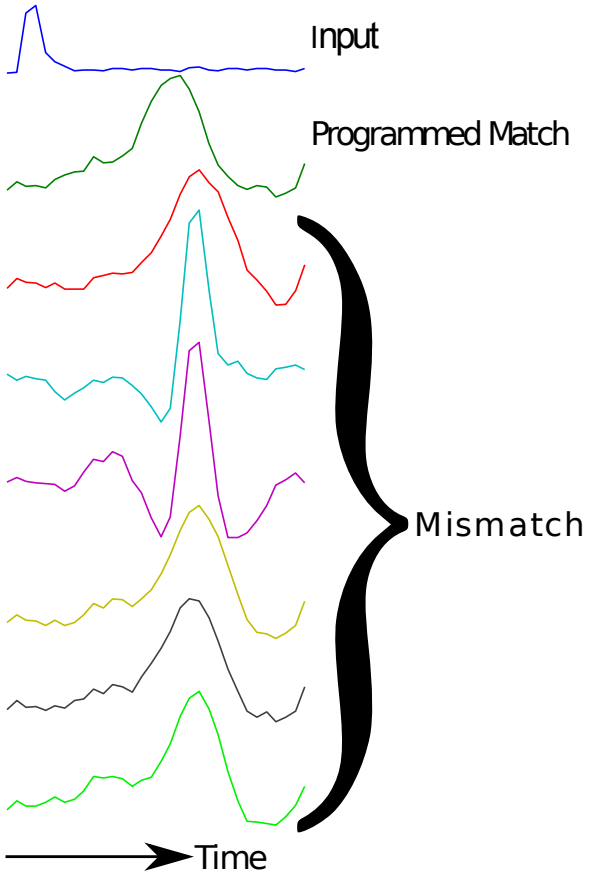

Fig. 5.8. Example of MF for one set of trained weights. It can be seen that there is some issue with there being a match signaled when the wrong SFAP is in the window.

There is also another issue that needs to be addressed. This issue is illustrated in Figure 5.8. This figure illustrates the input presented in blue and the resulting output of the ICAANN when the chip is trained for each of the seven SFAPs. As can be seen, it is quite clear that the same SFAP elicits a response for each weight set.

This is quite troublesome because it means that training a weight set for a specific SFAP won't result in a MF that will selectively identify only that SFAP. This is an inherit problem with the MF, they often get confused with like signals. Most of the SFAPs presented in this work are very similar to one another as can be seen in Figure 2.6. This is something that was realized earlier in the work and pushed the work into the ANNs. Unfortunately, due to the flaws found in the chip, work into the second layer could not be explored further. This is discussed in greater detail in Chapter 6.

5.5 4-channel Real-Time SFAP Detection

The final display of the D1503's power was to demonstrate the parallelizability. One of the reasons a fully analog chip was considered for an ANN is due to the ability to place them in parallel. Essentially giving the user as many networks as outputs; in this case 8 total ANNs could be used. However, due to limitations of the chip, only the first layer could be used for this demonstration, and it would have to be done at the output of the test MUX.

The work done in this section was done by loading weights for different SFAP onto the weight space for outputs 0, 15, 16, and 31. This allowed for the use of all four outputs. The data was collected one by one, not simultaneously. Once the weights were loaded, the SFAP was run through the chip going through each weight set. The results of this experiment can be seen in Figure 5.9. The signal was averaged with 1024 samples, but was not averaged as described in 5.4. From Figure 5.9, it is shown that each channel reacts to the SFAP coming down the taps, however, the reaction is much smaller than shown previously in Figure 5.8. This is very confusing would require further analysis of the chip. Regardless of the reason, the point is still demonstrated that the D1503 can run in parallel.

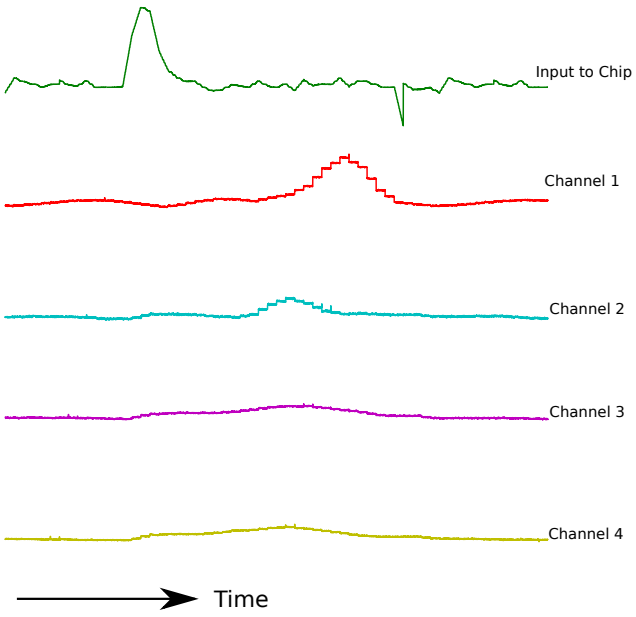

Fig. 5.9. This is the output of the D1503 where each of the available 4 channels were trained for a different SFAP. Each output arises from having a SFAP put through the taps but directed to a different output. Expectation is that there should be a larger response the more "match" there is.

6. DISCUSSION

This thesis focuses on the ability to train, automatically, either a LP or an ANN to an end that will be used to identify patterns from ENG signals. These signals are used in spike sorting, where development of low-powered processing technology is a strong need. This work aimed at extending this technology into the machine learning field, where this processing could be done in real-time and eventually be trained in real-time. This is in contrast to contemporary technology where the user is forced to do all of the work to learn how to use the device. This final chapter will review the work done in this thesis and will outline the work still to be done.

6.1 Summary

Chapter 1 introduced the reader to the world of spike sorting. Explanations of the desired human-computer interface that could restore full use of a limb back to the user or aide in the diagnosis of diseases were discussed and how neural interfaces were used in this end were introduced. Also discussed was the history of analog computing and how it might be suited for use in these applications. Further, the statement of work was developed and delivered in this section.

In Chapter 2, the concept of the LP and the MF was described in detail, in order to introduce the reader into the concepts utilized in the first steps of the work. The first training algorithm used, the delta rule, was also detailed here. Chapter 2, then, went into details of how the conductive sheet was simulated using MATLAB and COMSOL. With the sheet simulated, strategies of placement of the input pins using the delta rule was described. After placing the pins, an issue of space arose and was solved by scaling the rotating the points throughout the conductive sheet. Digitization of the signal had to be accomplished, which was done by a sample and

hold circuit, also outlined in more detail. Then the description of the signals used (SFAPs) rounded out the methods section of Chapter 2. The end results of the chapter was the simulation and physical demonstration of the LP running as a MF in real-time identifying a SFAP.

Further in Chapter 3, a shift in concept arose, where the idea of using a conductive sheet became obsolete allowing for greater miniaturization to an IC. Coupling this with the need to move to more sophisticated networks from the LP, shifted focus from geometry of sheets to structure of networks. From here, the concepts of ANNs and the training algorithm used to train them, BP, were introduced. This lead to the introduction of the application that was considered in this thesis, ENG spikes. Thereafter, the process of training, testing, and grading the networks were discussed and further details about how the application was accomplished was detailed. With the methods laid out, the results were shown that the task of sorting spikes can be achieved through use of the ANNs.

Continuing on with Chapter 4, these parameters of these networks were tested, in order to develop the design constraints for an IC chip, known as the integrated configurable analog artificial neural network, or the ICAANN. These design constraints included network accuracy and size, how much resolution was needed for the weight space, and what level of SNR the simulations could handle. The resulting parameters were then used to develop parameters for an IC that was then passed onto a microfabrication company for design and processing.

Concluding with the work in Chapter 5, once the ICAANN was developed and realized in the D1503 chip, it had to be tested to demonstrate its functionality and power. This was done by determining the weight space accuracy and then calibrating it to function for the same processes as in Chapter 2. Then, issues with the built-in S/H circuit on the D1503 arose, that were fixed with a simple work around that involved the use of the S/H circuit from Chapter 2. From here, the chip was tested for both the impulse response and the MF case. The results from these test were compared against the simulations found in Chapter 2. Finally, the D1503's ability to

run in a parallel nature was tested. This was done through testing all four available outputs on the first layer through the test MUX of the D1503. All these results points towards the chips ability to run as MF when trained as an LP, just as simulated.

6.2 Future Directions

Although this work took great strides in developing a low-powered signal processing paradigm with analog ANNs, there is still work that needs to be done to get this work to a more marketable place. The desired end result of this project was to develop a small IC chip that would be able to fit into an embedded system that could process signals coming from complex electrode arrays in real time. The D1503 was a step towards this end, however, there were flaws in the chip that occurred in the manufacturing and design phase.

Some of these design restrictions (i.e. S/H circuit problems) are mentioned in Chapter 5, but there were others that restricted the work further. When developing the bias for the system, there were errors made which restricted the use of half the weights. Halfway through the initial work with the D1503, nothing could be done because of this bias issue, which was exposed supposedly when a load resistor blew on the protoboard. This issue was eventually solved by the addition of a current source (Keithley 2400 Sourcemeter,TRS RenTelco, www.trsrentelco.com, DFW Airport, TX, USA). This, of course, would not be desirable to have to deal with in the embedded system. It is obvious that the D1503 is a far cry from being product ready. However, this was not the scope of the work. Moving forward, however, it would be desirable to make the next generation of this chip to further its power from the single layer to the second layer, which was not available for use due to previously mentioned biasing issues.

This would be the first step toward making the chip more suitable for the task of running an ANN, however, what steps would needed to be taken in order to get this product into a fully embedded system? Well, the bugs previously mentioned must

be dealt with, which requires a new run of the chip. However, this would be quite expensive, exceeding $80,000. Barring another round of funding, this would almost be impossible. Despite this looming road block, there is some lower hanging fruit that can be done with the first layer to show the chip is a viable product. This includes demonstrating that the D1503 can be trained "in-loop", where weights are updated via the SPI communication in response to some change in the signal. This was the main focus of a Phase IB funding that never saw fruition due to hardware issues with the chip. Once again, this would only be viable for the first layer, but it would still demonstrate that the chip could be utilized as a stand alone entity.

Taking a step back from the chip itself, it would also be beneficial to demonstrate the ANNs ability to tackle other applications besides those mentioned here. There is a wide array of applications that are suited for ANNs. This is demonstrated in the work done in [47–49]. This work shows that the ANN can be used in a similar manner to classify arrhythmias, identify when there is a fault in bearings, and sense different hazardous gases These applications are but a small sample of what could be done with a fully functioning D1503 chip.

6.3 Conclusions

To conclude this work, this work was a marginal success with an eye towards the future. It was demonstrated that machine learning techniques can be utilized to automatically train a LP to act as a MF. This was demonstrated in simulation, the conductive foam, and the D1503 that was developed. Each of these configurations demonstrated that they could identify SFAPs. From these developments, the work was pushed into determining the optimal structures for ANNs. These ANNs were then optimized for SFAP identification and classification. This work showed that these ANNs could, in fact, perform these applications at a high proficiency. From these findings, design criteria for an IC were determined which lead to the fabrication of the D1503.

This work was funded initially by a NSF STTR grant and then further by the Department of Biomedical Engineering at IUPUI. Work done in this thesis will, hopefully, culminate into a paper or two that will be pointed toward the Journal of Unconventional Computing.

LIST OF REFERENCES

LIST OF REFERENCES

[1] X. Navarro, T. B. Krueger, N. Lago, S. Micera, T. Stieglitz, and P. Dario, "A critical review of interfaces with the peripheral nervous system for the control of neuroprostheses and hybrid bionic systems," *Journal of the Peripheral Nervous System*, vol. 10, no. 3, pp. 229–258, 2005.

[2] K. L. Kilgore, P. H. Peckham, M. W. Keith, F. W. Montague, and others, "Durability of implanted electrodes and leads in an upper-limb neuroprosthesis," *Journal of rehabilitation research and development*, vol. 40, no. 6, p. 457, 2003.

[3] G. E. Loeb, R. A. Peck, W. H. Moore, and K. Hood, "BION system for distributed neural prosthetic interfaces," *Medical engineering & physics*, vol. 23, no. 1, pp. 9–18, 2001.

[4] R. F. Weir, P. R. Troyk, G. A. DeMichele, D. A. Kerns, J. F. Schorsch, and H. Maas, "Implantable myoelectric sensors (IMESs) for intramuscular electromyogram recording," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 1, pp. 159–171, 2009.

[5] B. Ortiz-Catalan, Max Håkansson, R. Brånemark, and J. Delbeke, "On the viability of implantable electrodes for the natural control of artificial limbs: Review and discussion," *BioMedical Engineering OnLine*, vol. 11, no. 1, pp. 33–56, Jan. 2012.

[6] A. Cutrone, J. D. Valle, D. Santos, J. Badia, C. Filippeschi, S. Micera, X. Navarro, and S. Bossi, "A three-dimensional self-opening intraneural peripheral interface (SELINE)," *Journal of Neural Engineering*, vol. 12, no. 1, p. 016016, Feb. 2015.

[7] C. Gonzlez and M. Rodrguez, "A flexible perforated microelectrode array probe for action potential recording in nerve and muscle tissues," *Journal of Neuroscience Methods*, vol. 72, no. 2, pp. 189–195, Apr. 1997.

[8] C. M. Gray, P. E. Maldonado, M. Wilson, and B. McNaughton, "Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex," *Journal of Neuroscience Methods*, vol. 63, no. 12, pp. 43–54, Dec. 1995.

[9] B. L. McNaughton, J. O'Keefe, and C. A. Barnes, "The stereotrode: A new technique for simultaneous isolation of several single units in the central nervous system from multiple unit records," *Journal of Neuroscience Methods*, vol. 8, no. 4, pp. 391–397, Aug. 1983.

[10] K. R. Harreby, A. Kundu, K. Yoshida, T. Boretius, T. Stieglitz, and W. Jensen, "Subchronic stimulation performance of transverse intrafascicular multichannel electrodes in the median nerve of the Gttingen minipig," *Artificial Organs*, vol. 39, no. 2, pp. E36–48, Feb. 2015.

[11] B. R. Bowman and R. C. Erickson, "Acute and chronic implantation of coiled wire intraneural electrodes during cyclical electrical stimulation," *Annals of Biomedical Engineering*, vol. 13, no. 1, pp. 75–93, Jan. 1985.

[12] M. S. Malagodi, K. W. Horch, and A. A. Schoenberg, "An intrafascicular electrode for recording of action potentials in peripheral nerves," *Annals of Biomedical Engineering*, vol. 17, no. 4, pp. 397–410, Jul. 1989.

[13] K. Yoshida and R. B. Stein, "Characterization of signals and noise rejection with bipolar longitudinal intrafascicular electrodes," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 2, pp. 226–234, Feb. 1999.

[14] S. Muceli, "Sampling large populations of motor units in humans with multichannel thin-film electrodes," *40th Annual Meeting of the Society for Neuroscience, Neuroscience 2010, 13-17 November 2010, San Diego, Usa*, 2010.

[15] C. Rossant, S. N. Kadir, D. F. M. Goodman, J. Schulman, M. L. D. Hunter, A. B. Saleem, A. Grosmark, M. Belluscio, G. H. Denfield, A. S. Ecker, A. S. Tolias, S. Solomon, G. Buzski, M. Carandini, and K. D. Harris, "Spike sorting for large, dense electrode arrays," *Nature Neuroscience*, vol. 19, no. 4, pp. 634–641, Apr. 2016.

[16] K. Horch and D. Kipke, *Neuroprosthetics: theory and practice*. World Scientific, 2017, vol. 8.

[17] H. G. Rey, C. Pedreira, and R. Quian Quiroga, "Past, present and future of spike sorting techniques," *Brain Research Bulletin*, vol. 119, Part B, pp. 106–117, Oct. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0361923015000684

[18] M. M. Soliman, "Developing a neural signal processor using the extended analog computer," Thesis, Purdue University, Aug. 2013.

[19] "TMS320c6746 | C674x DSP | C6000 DSP | Technical documents."

[20] E. Mastinu, P. Doguet, Y. Botquin, B. Håkansson, and M. Ortiz-Catalan, "Embedded System for Prosthetic Control Using Implanted Neuromuscular Interfaces Accessed Via an Osseointegrated Implant," *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, no. 99, pp. 1–11, 2017.

[21] "C55x Core Benchmarks | Processors | TI.com."

[22] D. Silva Graa and J. Flix Costa, "Analog computers and recursive functions over the reals," *Journal of Complexity*, vol. 19, no. 5, pp. 644–664, Oct. 2003.

[23] C. E. Shannon, "Mathematical Theory of the Differential Analyzer," *Journal of Mathematics and Physics*, vol. 20, no. 1-4, pp. 337–354, Apr. 1941.

[24] M. B. Pour-el, "Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers)," *Transactions of the American Mathematical Society*, vol. 199, pp. 1–28, 1974.

[25] L. Lipshitz and L. A. Rubel, "A differentially algebraic replacement theorem, and analog computability," *Proceedings of the American Mathematical Society*, vol. 99, no. 2, pp. 367–372, 1987.

[26] L. A. Rubel, "The brain as an analog computer," *Journal of theoretical neurobiology*, vol. 4, no. 2, pp. 73–81, 1985.

[27] L. Rubel, "The extended analog computer," *Advances in Applied Mathematics*, vol. 14, no. 1, pp. 39–50, 1993.

[28] J. W. Mills, "The nature of the extended analog computer," *Physica D: Nonlinear Phenomena*, vol. 237, no. 9, pp. 1235–1256, 2008.

[29] J. W. Mills, M. Parker, B. Himebaugh, C. Shue, B. Kopecky, and C. Weilemann, "Empty space computes: The evolution of an unconventional supercomputer," in *Proceedings of the 3rd conference on Computing frontiers*. ACM, 2006, pp. 115–126.

[30] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," DTIC Document, Tech. Rep., 1961.

[31] B. Kröse and P. van der Smagt, "An introduction to neural networks," 1993.

[32] H. von Helmholtz, "Ueber einige Gesetze der Vertheilung elektrischer Strme in krperlichen Leitern mit Anwendung auf die thierisch-elektrischen Versuche," *Ann. Phys. Chem*, vol. 89, pp. 211–233, 1853.

[33] S. Qiao, "Bioelectric nerve fiber to electrode coupling for unit identification and tracking," *Theses and Dissertations Available from ProQuest*, pp. 1–152, Jan. 2014.

[34] W. J. Karplus, *Analog simulation: solution of field problems*. McGraw-Hill, 1958.

[35] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[36] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[37] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry, Expanded Edition*, expanded edition edition ed. Cambridge, Mass: The MIT Press, Dec. 1987.

[38] P. J. Werbos, "Beyond regression: new tools for prediction and analysis in the behavioral science," *Ph. D. Thesis, Harvard University*, 1974.

[39] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[40] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[41] D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[42] F. D. Foresee and M. T. Hagan, "Gauss-Newton approximation to Bayesian regularization, paper presented at IEEE 1997 International Joint Conference on Neural Networks (IJCNN97)," *IEEE, Houston, Tex*, pp. 9–12, 1997.

[43] M. Jabri, R. J. Coggins, and B. G. Flower, *Adaptive analog VLSI neural systems*. Springer Science & Business Media, 2012.

[44] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1, pp. 239–255, 2010.

[45] S. Draghici, "Neural networks in analog hardwareDesign and implementation issues," *International journal of neural systems*, vol. 10, no. 01, pp. 19–42, 2000.

[46] K. Mirfakhraei and K. Horch, "Classification of action potentials in multi-unit intrafascicular recordings using neural network pattern-recognition techniques," *IEEE Transactions on Biomedical Engineering*, vol. 41, no. 1, pp. 89–91, 1994.

[47] M. Bertram, "White Paper towards Utilization of Artificial Neural Networks for Induction Motor Bearing Fault Classification."

[48] M. J. Bertram, "White Paper towards the Utilization of Artificial Neural Networks for Classification of Multiple EKG Morphologies."

[49] M. Bertram, "White Paper towards Utilization of Artificial Neural Networks for Classification of Hazardous Gases in Electronic Noses."