A REAL-TIME NEURAL SIGNAL PROCESSING SYSTEM FOR DRAGONFLIES

by

Thuy T. Pham

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2 0 1 1

FINAL EXAMINING COMMITTEE APPROVAL FORM

As members of the Final Examination Committee, we certify that we have read the thesis prepared by Thuy T. Pham
entitled A REAL-TIME NEURAL SIGNAL PROCESSING SYSTEM FOR DRAGONFLIES
and recommend that it be accepted as fulfilling the thesis requirement for the Degree of Master of Science.

_____ Date: .../.../2011
Charles Higgins

_____ Date: .../.../2011
Jeffrey Rodriguez

_____ Date: .../.../2011
Ali Bilgin

_____ Date: .../.../2011

_____ Date: .../.../2011

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to the Graduate College.

I hereby certify that I have read this thesis prepared under my direction and recommend that it be accepted as fulfilling the thesis requirement.

_____ Date: .../.../2011
Dissertation Director: Charles Higgins

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____

Thuy T. Pham

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

_____ _____

Charles Higgins, Associate Professor of Neuroscience Date

ACKNOWLEDGMENTS

I would like to express my thankfulness to my advisor Professor Charles Higgins for all the treasurable advices and the encouragements. This thesis would have been impossible without his guidance.

I would like to thank Professor ..... and Professor ..... for agreeing to be members of my thesis defense committee and all the valuable suggestions that make this thesis better.

I highly appreciate Professor Marwan Krunz, Professor Jeffrey J Rodriguez and Ms.Tami Whelan for financial supports and advising during my study at the ECE Department.

I wish give thanks to Ms.Tracey R. Purcell and Department of Neuroscience for her help and the respected research grant I received in this project. I would also thank Jerrie Fairbanks for his collaboration at Higgins' laboratory.

The encouragement from my parents, my husband, and my little daughter gave me the strength and self-confidence to pursue my goals of life.

DEDICATION

To my beloved family.

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF FIGURES – *Continued*

LIST OF TABLES

ABSTRACT

This thesis focuses on hybrid bio-robotics (robots incorporating living animals as sensors) and visual electrophysiology in insects. The motivation of this study is solving a current problem of perception in neuromorphic systems. When imitating biological sensors, we have not completely understood the early processing of the input to reproduce artificially. Building hybrid systems with both artificial and real biological components is a promising solution. In hybrid bio-robots using a dragonfly as a living sensor, the early processing of visual information is performed fully in the brain of the dragonfly. The only significant remaining tasks are recording neural signals and processing, along with interpreting neural information in software and/or hardware for a robot platform. Based on existing works which focused on recording neural signals, this thesis adds a software application of neural information processing to make a visual processing module for dragonfly hybrid bio-robots. After a neural signal is recorded in real-time, spikes of this signal can be detected either promptly by a hardware module using a simple threshold-based detection method or more accurately by a software module using an energy-based detection algorithm. Features of spikes are then extracted using a wavelet decomposition method. Finally, the system matches spikes with templates to find relevant neurons. The output of the whole visual processing module will be used to control other parts of a dragonfly hybrid bio-robot.

CHAPTER 1

INTRODUCTION

1.1 Motivation and Recent Research

In the field of engineering, biologically inspired systems have been designed with inspiration from biology and principles underlying the neural control in animals. For example, using models of insect eyes, Jeong et al. [1] have created artificial compound eyes with thousands of tiny lenses packed side by side. These eyes are expected to be used in camcorders for omnidirectional surveillance imaging. However, in simple applications of robotics (e.g., target tracking or collision avoidance), such artificial eyes may not be suitable for low-cost mobile neuromorphic systems. Moreover, to reproduce biological visual sensors better, we still need more efforts to make these eyes more comparable to those found in nature [2]. Additionally, the early processing of the visual information has not been understood completely.

From a neuroscience viewpoint, one of the contemporary research topics is identifying how sensory systems of insects operate under closed-loop control, i.e. where an animal can interact with its sensory inputs. Because performing closed-loop control experiments is limited by the fact that conventional electrophysiology equipment does not support a freely behaving animal, a promising solution is attaching the animal on a mobile robot platform in a configuration of a hybrid bio-robot. In this model, the early processing of the input is done by the brain of a living insect and other duties are performed by a robot platform [3]. For instance, in 2011 introducing with moths, Melano [4] proposed an insect-machine interfacing design with a robotic electrophysiology instrument whose velocity is determined by bioelectrical signals from the animal.

This hybrid bio-robotics approach not only provides a research tool to better understand the neurobiological processes underlying behavior, but also is a promising

solution for the perception of neuromorphic systems. For example, visual response signals from a dragonfly brain can serve as the visual sensor inputs of a robot. The dragonfly has attracted our attention for this purpose due to its excellent vision. Dragonflies have the largest compound eyes of any insect; each containing up to $30,000$ facets, and the eyes cover most of the head. Because of their large, multi-faceted eyes, the adult dragonfly can see in almost all directions at the same time.

Inspired by the brains of flying insects, the Higgins laboratory (University of Arizona) has been interfacing the living brains of moths to mobile robots. Ortiz [5] designed two prototype electrophysiological recording printed circuit boards (PCBs): an intracellular recording PCB and an extracellular recording PCB. Based on the extracellular recording board in [5], Routh et al. [6] designed an additional PCB which is capable of running in real time a software module for signal processing with four neural signal recording channels. In 2011, Melano used Ortiz's design to interface directionally-sensitive visual neurons and pleurodorsal steering muscles of the mesothorax with a robot. In his work, Melano used the rate of spikes to control its rotation, thus emulating the classical optomotor response known from studies of the fly visual system [4].

With a robotic approach for analysis of sensory signals in insects, this thesis focuses on dragonflies' visual motion detection neurons. Sensory signals of insects are transferred from the brain to thoracic ganglia through neurons called *descending interneurons*. In 1984, Kien et al. [7] suggested that behaviors of insects were probably directed by many descending interneurons acting in concert. In the dragonfly, there are eight large individual descending neurons in the ventral nerve cord (VNC) that are visual target-selective [8]. Their visual receptive field properties were analyzed well by Frye et al. [9] in 1995. From these properties, the thesis proposes a system to characterize dragonflies' visual motion detection neurons and to work as a visual processing module for dragonfly hybrid bio-robots.

To characterize dragonfly visual motion detection neurons, action potentials ("spikes") in recorded electrophysiological signals should be detected and classified. Then, from the dragonfly visual receptive field properties, spike templates for

visual motion detection neurons are pre-defined. These templates will be used to recognize which descending neurons transmit which spikes. In this study, this process is called *spike sorting*. Recently, several spike-detection and spike-classification algorithms have been introduced.

There are three groups of spike-detection algorithms: threshold-based methods, energy-based methods, and template-based ones. A simple example of the first group, which is often found in spike detector circuits, is using a threshold level to detect a spike whenever the amplitude value of the input signal goes over this level [10]. Other examples of this group apply the threshold level to the absolute value of the input with a static technique [11] or an adaptive technique [12]. Threshold-based algorithms execute relatively simple computation. Thus, they are suitable for real-time implementations. However, they are sensitive to noise and require a step of setting threshold levels [13]. The second group of spike detection algorithms involves a nonlinear energy operator (NEO) to estimate the square of the instantaneous product of amplitude and frequency of a sufficiently sampled signal [14]. The third group uses filters to detect spikes whenever a signal sample matches the filter. Examples for this algorithm group are average-filter matching [15] and wavelet-based matching [16]. Because this group involves multiple convolutions, it should not be a candidate to detect spikes in real-time for multichannel systems. From the review by Obeid et al. [13], we chose the NEO algorithm to detect spikes.

Spike classification processes include two main steps: extracting spike features, and then classifying spikes by these spike features. Common spike feature extraction algorithms are based on principal component analysis (PCA, [17]) used in [18] [19], the discrete wavelet transform (DWT, [20]) applied in [21][22], independent component analysis [23] applied in [24][25], or discrete derivatives [26]. Other existing algorithms use waveform derivatives [27], the integral transform [28], inter-spike intervals [29], or Laplacian eigenmaps [30]. For the second step of the classification processes, algorithms sort similar spikes by k-means clustering [23], mean shift clustering [27], Bayesian classification [31], template matching [32], neural network based [33], super paramagnetic clustering (SPC) [34], or density grid contour clustering

[35].

Spike feature extraction algorithms have been reviewed recently [36] while classification algorithms have been in need of a newer review since 1998. From the review by Gibson et al. [36] and other criteria of the system design (e.g. high accurate spike-feature extraction), we choose the wavelet transform approach along with the SPC method to launch a spike-sorting module for the proposed system (more details in Sections 2.4 and 2.5).

1.2 Thesis Overview

With the motivation discussed above, this thesis proposes a real-time neural signal processing system for dragonflies. A part of this work is based on some pre-existing designs by Routh et al. [6] and Melano [4]. Instead of recording from moths as before in the Higgins laboratory, this project studies the vision system of a dragonfly, and fabricates an upgraded version of previous designs to process more channels and digital signal flow faster. We also implement a new spike-sorting application module for this upgraded version, which enables characterizing visual motion detection neurons and employing these neurons as biosensors for robots.

The thesis is organized in seven chapters as follows. Chapter 2 provides a background on the neurological workings of the dragonfly ventral nerve cord, measuring neural activity, and spike-sorting algorithms as well as neural signal processing system models. Chapter 3 describes an analog processing module of the design. Chapter 4 presents a digital processing module. The spike-sorting application is described in Chapter 5. Chapter 6 shows experimental results for testing hardware devices and the software module. The final chapter concludes with a summary and future work for this report.

CHAPTER 2

BACKGROUND

In this chapter, the visual receptive field properties of dragonfly descending neurons, electrophysiological recordings, and the measuring of neural activity are discussed in the first three sections. Then spike-sorting algorithms and a model of a neural signal processing system are presented.

2.1 Target Selectivity Properties of Dragonfly Descending Neurons

Descending interneurons in insects are neurons that transfer signals from the brain to the thoracic ganglia. In the dragonfly, there are eight large individual descending neurons in the ventral nerve cord (VNC) that are visual target-selective [8]. The locations of these eight interneurons in a cross section of the prothoracic ganglion are indicated in Figure 2.1. These neurons have some similar characteristics (Table 2.1) [8]. All of them descend from the brain and have very big axons in the VNC. They all respond selectively to movements of small targets; therefore, they are called target-selective descending neurons (TSDNs). According to the longitudinal tract through which these neurons pass in the prothoracic ganglion, Frye et al. [9] named these eight neurons as follows: DIT1, DIT2, and DIT3 for the dorsal intermediate tract; and MDT1, MDT2, MDT3, MDT4, and MDT5 for the median dorsal tract.

Five of the eight TSDNs have strong directional preferences: MDT1, MDT2, MDT4, MDT3, and DIT1. Frye et al. [9] identified the visual receptive field properties of these TSDNs by recording responses when moving a target in each of four orthogonal directions on a 90^o x 90^o screen. The location and direction of the greatest response to target movements are shown in Figure 2.2. MDT4 and DIT1 are small-size selective while all others have a wide range of size preference. In these experiments, the observed receptive field centers were consistent from animal to

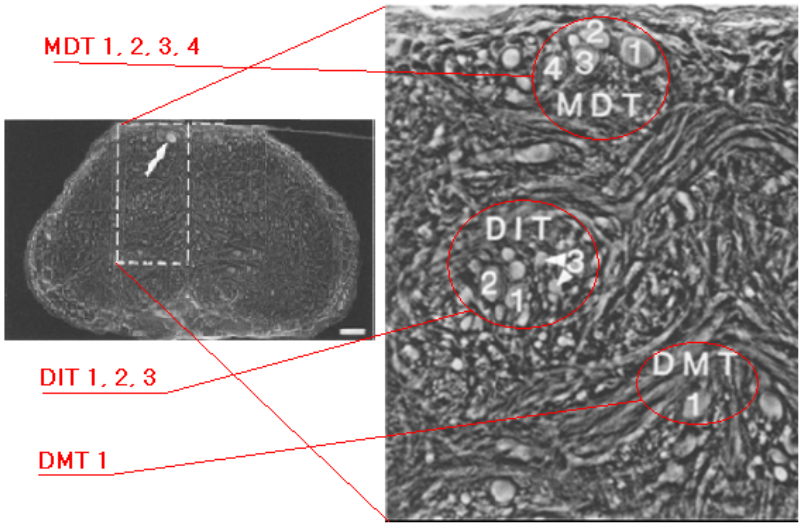

Figure 2.1: Locations of the eight TSDNs in the cross section of the prothoracic ganglion. Left: The eight axons with a background detail. Right: a magnified view of the outlined area in the left [8]. The white arrow show the MDT group.

Table 2.1: Some properties of the eight TSDNs. Abbreviation: Left (L), right (R), top (T), dorsal (D), ventral (V), up (U), down (D), no directional (ND), small (S) 2^o or 4^o, large (Lg) 16^o or 32^o, all sizes(A) [8].

Cell	MDT1	MDT2	MDT3	MDT4	DIT1	DIT2	DIT3	DMT1
Soma location in brain.	LT	RV	LT	RV	LV	RV	LD	LD
Directional preference.	U	L	ND	D	R	L	ND	-
Preferred target size.	A	S	Lg	S	S	A	A	-

animal but the borders were not. Across the preparations, the location and the directional preference of the receptive fields remained constant, but the strength of the response (vector length) and the size of the receptive field (number of vectors) varied.

In summary, the strong selectivity responses of the eight dragonfly TSDNs to the target sizes and the movement directions indicate that these neurons control oriented flight responses to the image of a target in the dragonfly retina. This behavioral function guides the dragonfly during the prey-tracking. Researchers really want to get more insight into how these neurons respond when a freely behaving dragonfly receives images of a real prey item or a fake target.

2.2 Electrophysiological Recordings

Electrophysiology is the study of the electrical properties of biological cells and tissues. It involves measurements of the electrical activity of neurons, particularly action potential activities [37]. Electrophysiological recordings can be classified into intracellular and extracellular recordings. In the intracellular recordings, the tip of a microelectrode is inserted inside the cell to measure the membrane potential. By contrast, in the extracellular recordings, signals are recorded from outside of the cell. Thus, in comparison to intracellular signals, the extracellular ones are much smaller (only about μV, not mV) , and the polarity is reversed as recording from across the membrane. The magnitude and the shape of the extracellular signals vary with the distance between the electrode tip and the cell. Extracellular recording results can present the activity of a single cell ("single-unit") or several cells ("multi-unit") depending on the size and placement of the electrode tip [38].

The microelectrode resistance and the capacitance in an electrophysiological recording are two main sources of recording errors. The resistance depends on several factors including the size of the electrode tip and the thickness of the cell walls. Typical resistance values are from 10 $M\Omega$ to 50 $M\Omega$ with a 500 nm glass micropipette tip or from 10 $M\Omega$ to 200$M\Omega$ with a 50 nm to 120 nm tip [39]. The

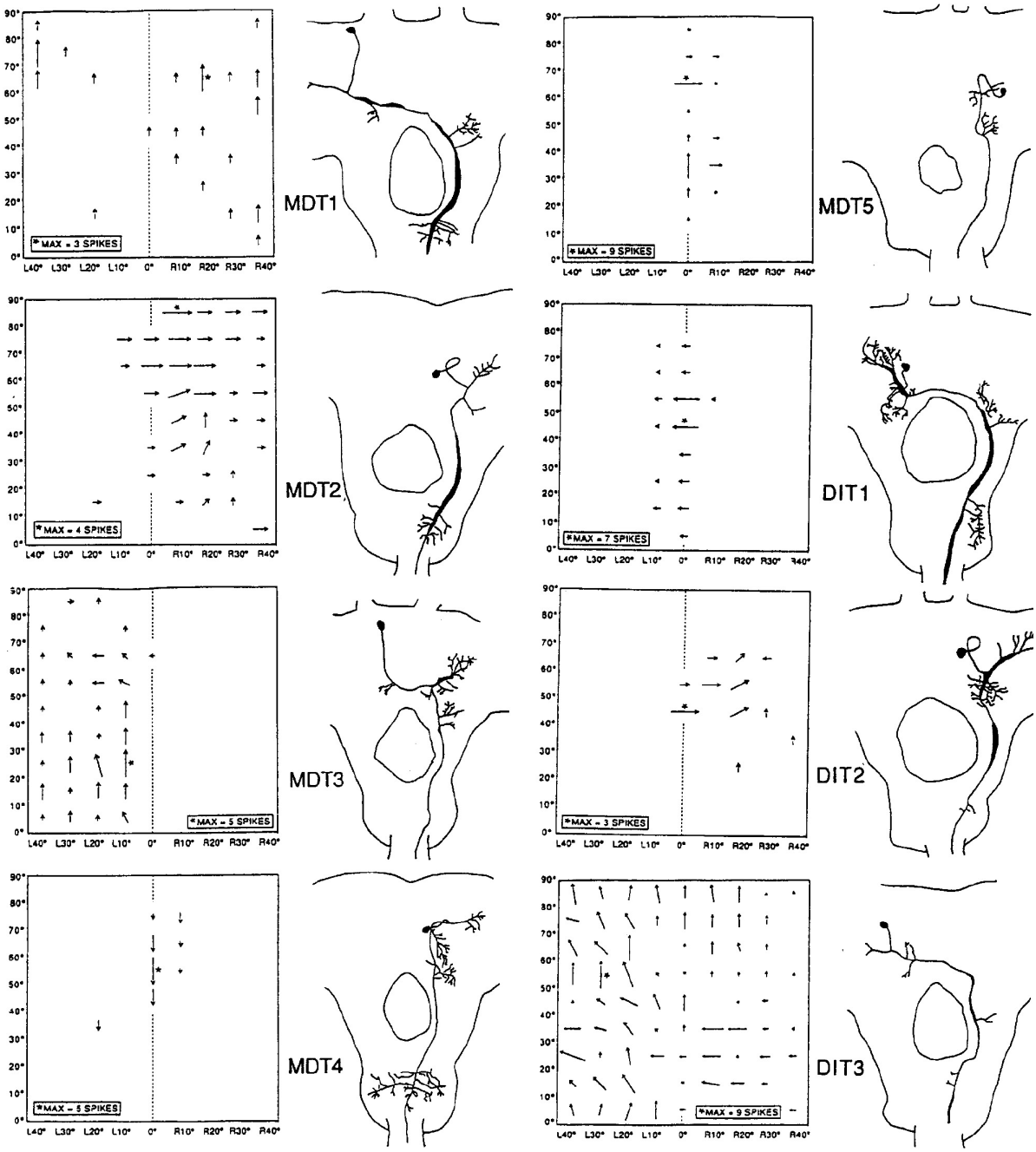

Figure 2.2: The visual receptive field properties of the eight TSDNs [9].

resistance creates a voltage drop across the electrode which can produce biological effects in the cell. To avoid this voltage drop, the instrument should have a small input current (less than 1 nA). On the other hand, the capacitance in combination with the resistance create a low-pass filter that prevents the propagation of the high frequency components of the signal and makes the signal distort [5]. The capacitance includes the transmural capacitance (C_t) between the solution inside the pipette and the biological tissue and the stray capacitance (C_s) between the electrode's stem and the lead joining to the preamplifier [40]. While the value of C_t cannot be eliminated, the value of C_s depends on the length of the joining lead. Normally, C_s is a few picofarads and can be reduced by placing the first amplifier as close as possible to the electrode.

In electrophysiological recordings, the quality of collected data depends significantly on the presence of noise sources. Some sources of noise can be suppressed (e.g., 50 Hz or 60 Hz hums from power supplies and mechanical interference from surroundings) while others cannot be blocked (e.g., thermal voltage noise).

2.3 Measuring Neural Activity

An electrophysiological signal represents the potential between its reference potential and the tip of the electrode. According to Lewicki [41], the largest component of the current flow due to the potential changes is generated by the cellular action potential. Other sources of signals which may come from axonal fiber bundles (also called fibers of passage), or come from the field potential, introduce less prominent components. These components are often much more localized and smaller or occur at low frequencies, which can be easily filtered out from the action potential.

In an extracellular recording, there are several problems when measuring action potentials. From an example segment of an extracellular recording signal (Figure 2.3), numerous different shapes of action potentials appeared. They may belong to different relevant neurons. Next, a considerable amount of background noise should be filtered out. Finally, spike overlaps appeared sometimes. When an over-

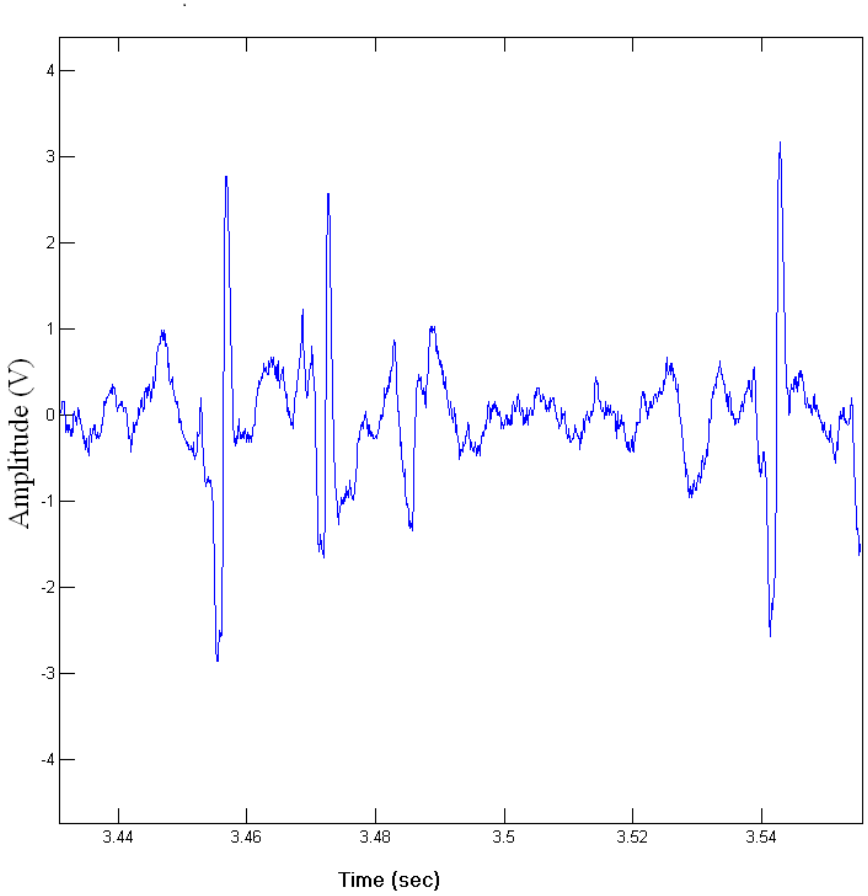

Figure 2.3: Example of an extracellular signal segment recorded from a dragonfly by Higgins, 2010. The amplitude is amplified by the commercial instrument in the Higgins laboratory. The signal included the background noise and spike information.

lap occurs, the height of a spike can be greatly distorted. If an action potential has a trough that lines up with a peak of an action potential which is generated by a background neuron, the spike will be missed. Then again, in threshold-based spike detection methods, when some background spikes combine together to cross over the threshold level, a false positive error may occur.

2.4 Spike Detection and Classification Algorithms

According to several recent reviews for spike detection and classification, this section presents some efficient algorithms for real-time spike detection and classification.

2.4.1 Real-Time Spike Detection Algorithms

Based on recent evaluations [13] [36] [42], NEO and the simple threshold method are the two most efficient algorithms to detect spikes in real time. Obeid et al. [13] proposed a technique to evaluate the performance of spike-detectors by defining cost-function scores. They concluded that the simple threshold method is just as effective as applying more elaborate energy-based detectors. On the other hand, Gibson et al. [36] used the probability of a right detection and the probability of a false detection to evaluate spike detection algorithms. They showed that the stationary wavelet transform method is the most accurate one, followed by NEO and the simple threshold method. However, according to Gibson et al. [36], the NEO-based method is less sensitive to the choice of a threshold level than the simple threshold method. Hence, a proposed design for a neural signal processing system will be better if it can support these two most efficient algorithms. While the threshold method is very computationally simple, the NEO-based method is more complicated.

At first, NEO was introduced to highlight the spike peaks by computing the energy difference between the current power of a signal and its power in adjacent time intervals [16]. Then, in an interpretation of NEO, when the Wigner distribution (WD) [43] is taken as the instantaneous spectrum, the result is considered the instantaneous energy of the high pass filtered version of a signal. This feature makes

NEO an ideal detector of transients [44].

Let $x(t)$ be a real process which is band limited with a spectrum $S_{xx}(w) = 0$ for $|w| > B$ and $x(nT)$ be the uniformly sampled version of $x(t)$ with a sampling interval $T < \frac{\pi}{B}$. For a discrete time signal, the non-linear energy of $x(n)$, $\psi[x(n)]$, is defined by Kaiser [14]:

$$\psi[x(n)] = x^2(n) - x(n+1)\,x(n-1). \tag{2.1}$$

Let $x(n)$ be a combination of two uncorrelated signals $x_1(n)$ and $x_2(n)$. Let $x_1(n)$ and $x_2(n)$ represent a background signal and a spike train, respectively. When interpreting NEO, Mukhopadhyay et al. [44] proved the following relationship:

$$E\{\psi[x(n)]\} = R_x(n,n) - R_x(n+1,n-1) \tag{2.2}$$

where $E\{.\}$ is an expectation operator, and $R_x(n,n)$ is an autocorrelation of $x(n)$. Let $W(n,w)$ denote the Fourier transform of the autocorrelation $R_x(n,n)$. Mukhopadhyay et al. [44] also showed that:

$$E\{\psi[x(n)]\} = \int_{-B}^{B} W(n,w)\,(1-\cos(2wT))dw. \tag{2.3}$$

Because the term $(1-\cos(2wT))$ acts as a high-pass filter [44], $W(n,w)\,(1-\cos(2wT))$ is high-pass filtered version of $W(n,w)$. Let $K_x(n)$ be the ratio of the energy in the high-frequency band to the total signal energy at an instant nT:

$$K_x(n) = \frac{\int_{-B}^{B} W(n,w)\,(1-\cos(2wT))dw}{\int_{-B}^{B} W(n,w)\,dw}. \tag{2.4}$$

According to Mukhopadhyay et al. [44], another representation for equation 2.3 is:

$$E\{\psi\left[x\left(n\right)\right]\} = K_{x_1}\left(n\right) R_{x_1}\left(n,n\right) + K_{x_2}\left(n\right) R_{x_2}\left(n,n\right). \tag{2.5}$$

Since a spike is characterized by localized high frequencies and an increase in instantaneous energy, the second term in the right hand side of equation 2.5 does not dominate until a spike appears. In other words, the expectation value will be much greater than a threshold level whenever a spike appears. The expectation operator always returns a positive value. Therefore, $E\{\psi\left[x\left(n\right)\right]\}$ can do as an indicator to detect spikes.

Applying the above interpretation, most NEO-based spike detection algorithms ([44], [45], [13], [36], [42]) have been introduced with an general adjacent time δ rather than 1. Hence, equation 2.1 is rewritten as:

$$\psi\left[x\left(n\right)\right] = x^2\left(n\right) - x\left(n+\delta\right)x\left(n-\delta\right). \tag{2.6}$$

A threshold level Θ is given by:

$$\Theta = C\frac{1}{N}\sum_{n=1}^{N}\psi[x(n)] \tag{2.7}$$

where C is a constant which is used to tune Θ, and N is the number of samples.

2.4.2 Real-Time Spike Classification Algorithms

Among spike feature extraction algorithms, Sarna et al. [46] showed that the DWT method outperformed the PCA method and the reduced feature set method (RFS, [47]).The main reason is the fact that DWT can separate complex spikes that could not be done by PCA and RFS methods due to the similarity of their temporal profiles and the masking actions of the noise.

DWT relies on the wavelet analysis technique to extract features of a spike. Basically, to find differences among spikes, DWT is based on the quantification of

energy found in specific frequency bands at specific time locations. The following mathematical equation describes briefly a wavelet transform (WT) (more details in [20]). Let $s(t)$ be a spike waveform which can be presented by WT coefficients $C(a, b)$.

$$C(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} s(t)\psi_{a,b}(t)dt \tag{2.8}$$

where $\psi_{a,b}(t) = \psi\left(\frac{t-b}{a}\right)$ is an expanded or contracted and shifted version of a unique wavelet function $\psi(t)$ with a and b are the scale and the time localization, respectively.

For algorithmic implementations, equation 2.8 is usually defined at discrete scales and discrete times by choosing the set of parameters $\{a_j = 2^{-j}; b_{j,k} = 2^{-j}k\}$ for integers j and k [22]. Contracted versions of the wavelet function match the high-frequency components, while dilated versions match the low-frequency components.

Among spike clustering algorithms, in a very early review, Lewicki [41] concluded that the reviewed methods (Bayesian clustering, template-based, and classification to the commercial package Brainwaves, which relied on the user to define the two-dimensional cluster boundaries by hand) gave similar results if clusters were well separated. However, for weakly separated clusters, the Bayesian clustering method was the most accurate one. However, this review was published a long time ago; thereby it could not cover recent algorithms. Furthermore, up to now, classification algorithms have been still in need of a newer review. In the literature, the SPC method has been suggested as a proficient tool of classification in several recent published papers such as [22][34]. In this thesis, we also choose SPC to classify spikes when building templates.

The SPC method is based on interactions between a data point (a spike) and its K-nearest neighbors [34]. If the interactions are strong, spikes are more similar. This method is implemented as a Monte Carlo iteration of a Potts model [48] which suggests the behavior of ferromagnets and certain other phenomena of solid-state physics.

In the SPC method, the term 'temperature' is used to interpret the probability at which the states of a number of neighboring data points change simultaneously [22]. At a relatively high temperature, all the points are switching randomly, regardless of their interactions (paramagnetic phase). At a low temperature, all the points change their states together (ferromagnetic phase). But, at a medium temperature (super paramagnetic phase) only points in the same group change their states concurrently. In a clustering application, the ferromagnetic phase, the paramagnetic phase, and the super paramagnetic phase can be considered a classifying result of one single cluster, several tiny clusters, and a number of medium-size clusters, respectively.

The SPC method first represents m features of a spike i by a point x_i in an m-dimensional space. Then it finds the interaction strengths between the point x_i and k nearest neighboring points. The interaction strength J_{ij} between x_i and one of its neighbors, named x_j, is given by [22]:

$$J_{ij} = \begin{cases} \frac{1}{K} \exp(-\frac{\|x_i - x_j\|^2}{2a^2}) & \text{if } x_i \text{ is one of K nearest neighbors of } x_j \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

where a is the average distance from x_i to its K nearest neighbors.

From equation 2.9, J_{ij} reduces exponentially when the Euclidean distance $\|x_i - x_j\|^2$ increases. A smaller distance results in a stronger similarity between two spikes.

In the second step, SPC assigns each point x_i to a random state s in a set of q states. Then, N Monte Carlo iterations are run for different temperatures using the Swendnsen-Wang algorithm [49] or the Wolff algorithm [50]. Blatt et al. [49] recommended a setting of $q = 20$ states, $K = 11$ nearest neighbors, and $N = 500$ iterations for clustering. With this setting, the clustering process would mainly depend on the temperature parameter and is robust to small changes of other parameters.

2.5 Proposed Neural Signal Processing System

In electrophysiological recordings, signal amplitudes at the tip of the electrode range from 50 μV to 500 μV(in extracellular recordings) or few millivolts (in intracellular recordings). This signal is then amplified and filtered out from the noise. After that, spikes are detected and processed further for specific applications.

We propose a real-time neural signal processing system with a combination of a PCB-based hardware structure (Figure 4.1) and FPGA-based software modules (Figure 2.5). The PCB-based hardware structure provides the analog signal recording and digital signal processing functions. To avoid interference between the two different formats of signal processing (analog and digital) as well as per the small-size requirement for the system to be attached on a mobile robot platform, we construct two separate PCBs: an analog PCB and a digital PCB. The analog PCB is an upgraded version of the design by Melano [4] while the digital PCB is an upgraded version of the design by Routh et al. [6] in our laboratory. The analog PCB is employed with configurable cut-off frequencies to configure a suitable setting for a particular recording condition. The digital PCB is equipped with an FPGA-based microprocessor which can execute software modules for data processing tasks of the proposed system.

Spikes of the input signal can be detected by a circuit block in the analog PCB or by a software module in the digital PCB. While the hardware-based detector uses the simple threshold method, the software-based one uses the NEO method.

To extract spike features, we use the DWT algorithm with a four-level decomposition Haar wavelets setting. Then, we gather information to build templates by clustering these spike features with a SPC software package provided by Quiroga et al. [22]. In the SPC process, we use a temperature range from 0 to 20 in increments of 0.01 and record the super paramagnetic phase at a minimum cluster size of 60 points. After that, based on the visual receptive field properties defined by Frye et al. [9], we select a suitable cluster and take the average as a respective template. Finally, we measure the correlation between a detected spike in real time with every

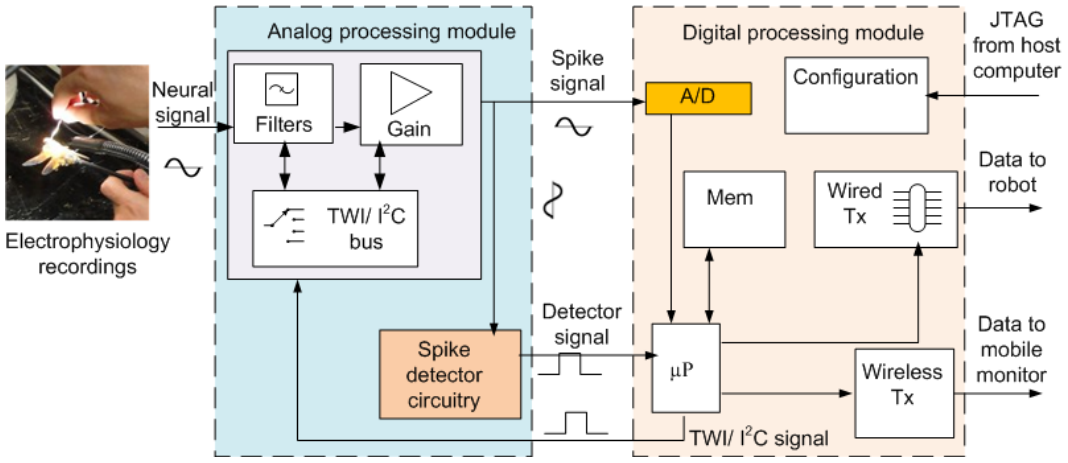

Figure 2.4: Hardware structure design for the proposed system.

template (from the template set) to find a histogram of TSDNs.

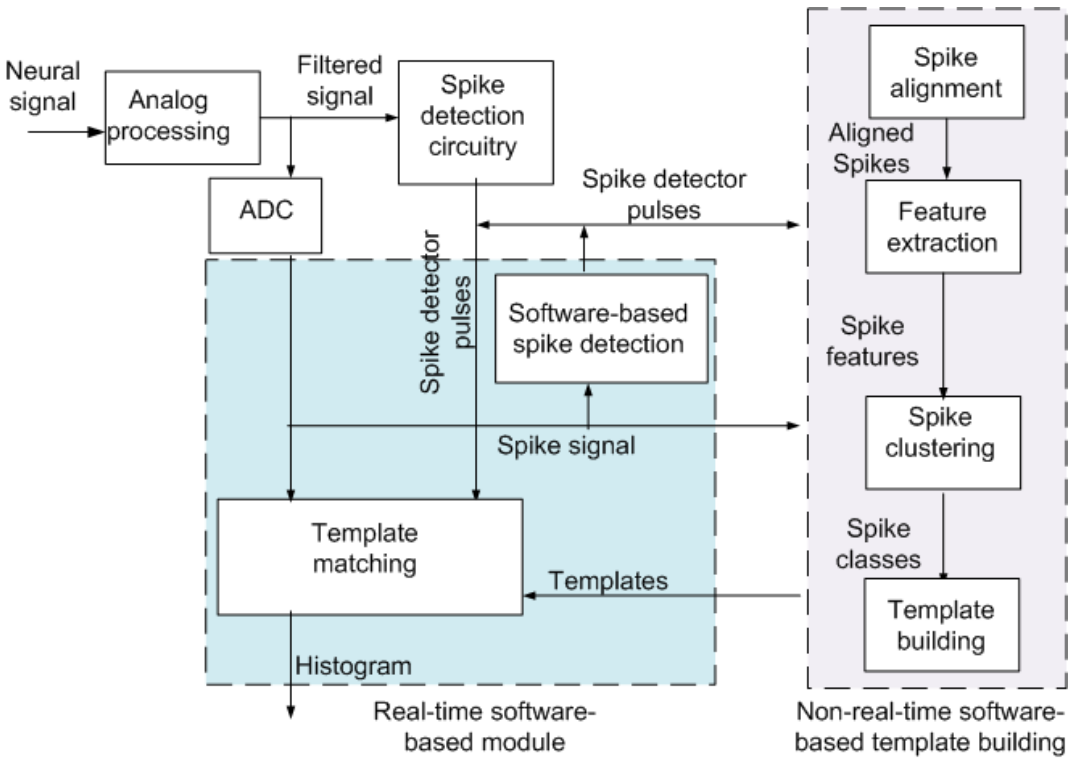

Figure 2.5: Implementation steps of selected algorithms for the software modules.

CHAPTER 3

ANALOG PROCESSING MODULE DESIGN

This chapter discusses the analog PCB hardware structure which is upgraded from existing works in the Higgins laboratory. The main duties of this PCB are filtering out the noise, amplifying signals, and detecting spikes. Because this module design contains several configurable devices with a serial bus interface, the last section of this chapter is dedicated to a short description of how to program these devices. The serial bus interface has been developed under several names such as TWI and I^2C though in this document we only use I^2C as a general name for this type of protocol.

3.1 Functional Diagram Design

The analog PCB can be viewed as a series of five processing stages (Figure 3.1). Each stage handles a different function. The first stage has a differential input preamplifier. The second stage has a high-pass filter (HPF) while the third one has a low-pass filter (LPF). Both of them are designed to remove noise and undesired frequency components. The fourth stage is a notch filter to remove the electricity interference. The last stage is a spike detector.

The total gain of the module is contributed from several stages due to gain bandwidth restrictions of the available components. The available amplifiers often offer a gain-bandwidth product (GB) limited to $1M$ Hz [5]. The maximum gain that can be extracted from a single amplifier is 100. In the meanwhile, the operation bandwidth of the module should be around 10 k Hz [40]. Thus, to get a total gain of around 1000, the signal needs to be amplified by a number of stages.

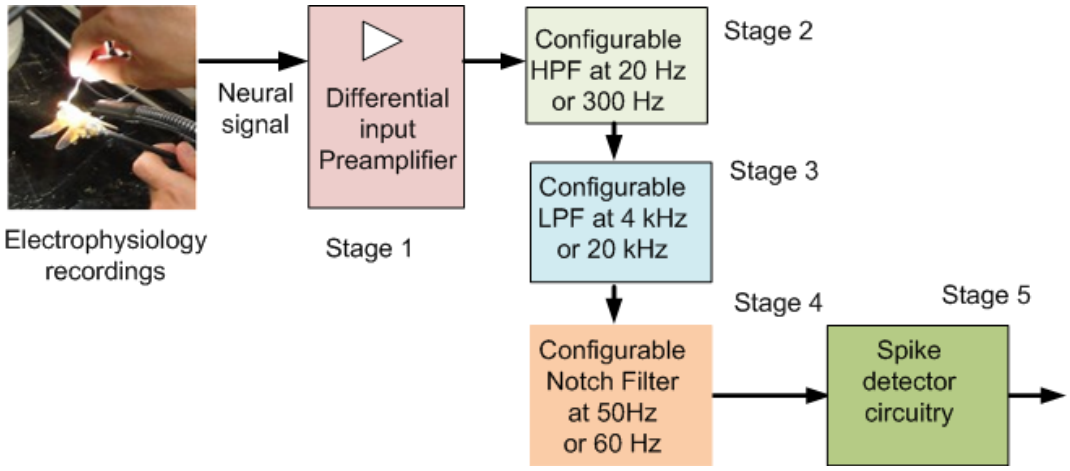

Figure 3.1: Five stages of the analog signal processing module.

3.2 Component-Level Design

We choose main components for the proposed design according to desired functional tasks (Section 2.5) and availability of suitable products.

3.2.1 Differential Input Preamplifier

The main idea of this stage is to transfer states from a very high microelectrode resistance to a very low output resistance while maintaining the signal undistorted [5]. This amplifier should have a high input resistance and a low input capacitance [51]. Also, the component should have a high common-mode rejection ratio (CMRR) [40]. From the design by Ortiz [5], the gain of this stage should be not too large to avoid amplifying the DC offset (causing saturation). We select a INA111 [52] as a suitable amplifier with a very good performance (Figure 3.2) and set the stage gain to be 10.

3.2.2 Configurable High-Pass Filter with Non-Unity Gain

This stage not only works as a high-pass filter, but also provides the rest of the required gain for the whole system. The configurable cut-off frequency of this HPF

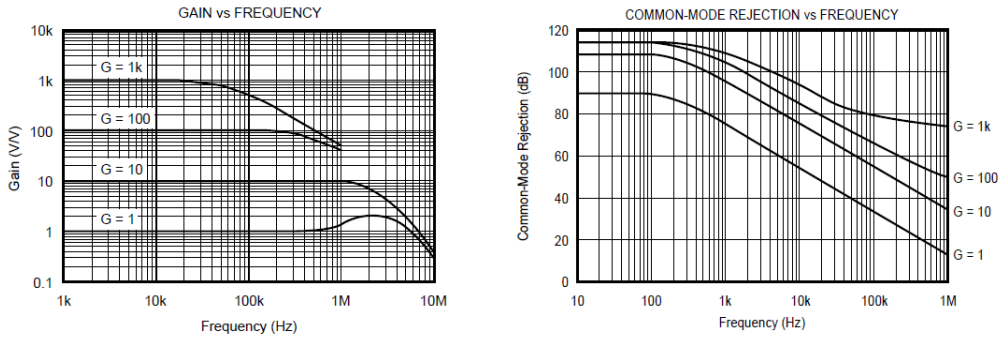

Figure 3.2: Typical performance curves of selected preamplifier [52]. The INA111 has a suitable gain response (left) and a high CMRR (right).

is provided by digital switches and potentiometers which can vary capacitance values and resistance values under the control of I^2C signals (Figure 3.3). In this diagram, when R_1 and R_3 values are fixed, cut-off frequencies $f_{0_1} = \frac{1}{2\pi R_1 C_1}$, $f_{0_2} = \frac{1}{2\pi R_3 C_2}$ are controlled by switches on C_1 and C_2, and gains $G_1 = -\frac{R_2}{R_1}$, $G_2 = -\frac{R_4}{R_3}$ are controlled by potentiometers on R_2 and R_4.

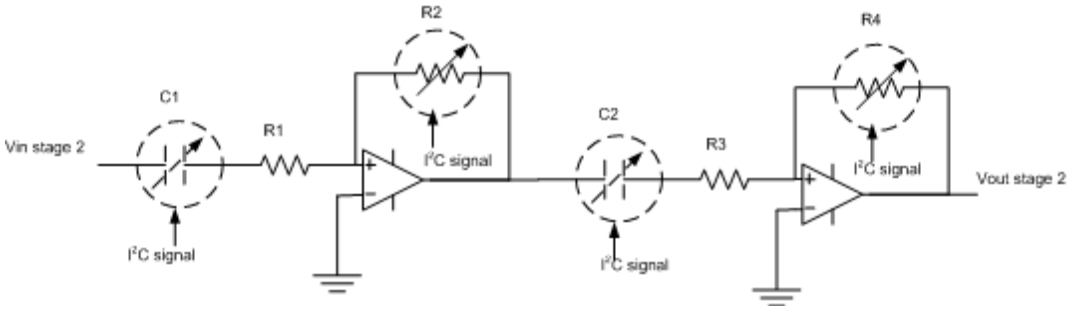

Figure 3.3: Second-order high-pass filter at w_0 with gain G. R_2, R_4 can be variable resistors. C_1, C_2 are selected capacitors by digital switches. R_2, R_4 , C_1, C_2 will be set by I^2C signals.

3.2.3 Configurable Low-Pass Filter

To block undesired high frequencies, a second-order Butterworth low-pass filter using the Sallen and Key topology [53] (Figure 3.4) was chosen [5]. The cut-off frequency

is designed as 6 k Hz or 10 k Hz [54] by equation $f_0 = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$ [53]. To get a maximally flat pass-band frequency response, a quality factor $Q = \frac{\sqrt{R_1 R_2 C_1 C_2}}{C_2(R_1+R_2)}$ is set to $1/\sqrt{2}$ or equivalently $C_1 = 2C_2$ with $R_1 = R_2$. Therefore, the cut-off frequency is controlled by the values of C_1 and C_2.

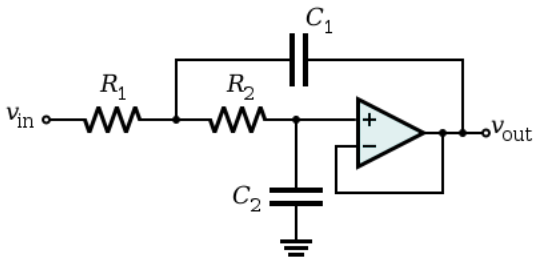

Figure 3.4: Second-order Butterworth low-pass filter [53].

3.2.4 Configurable Notch Filter

When we collect an electrophysiological signal in a very wide range of bandwidth (e.g., from 20 Hz to 10 k Hz), the power line interference at 50 Hz or 60 Hz should be removed. To build this notch filter, we combine a first order low-pass filter and a first order high-pass filter (Figure 3.5) [5]. If $C = 15\ nF$, R should be 424 $k\Omega$ or 353 $k\Omega$ to get the notch at 50 Hz or 60 Hz, respectively.

3.2.5 Spike Detector

As mentioned in Section 2.5, this stage provides an optional implementation to detect spikes. This feature of the system is necessary for the case of lacking a high-speed microprocessor to execute several software modules concurrently. Since the output of this stage is a chain of pulses (i.e. in a digital format), this output can be used by an FGPA-based microprocessor in the next steps. The proposal for this stage consists of two blocks: spike enhancement and spike detection (Figure 3.6). While the first block increases the SNR at expected frequency contents, the second one works as a threshold comparator.

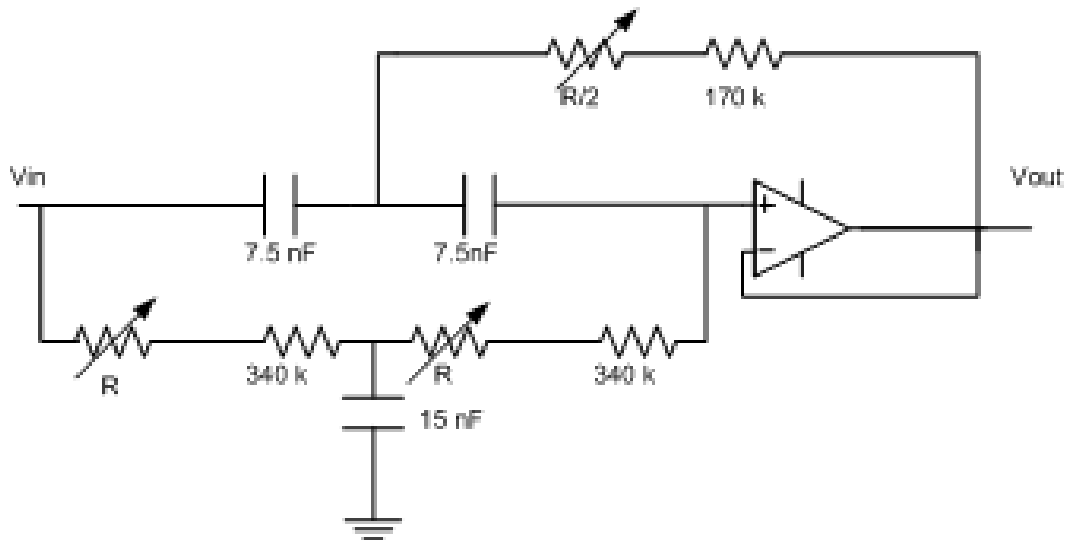

Figure 3.5: Configurable notch filter for 50 Hz network or 60 Hz network option.

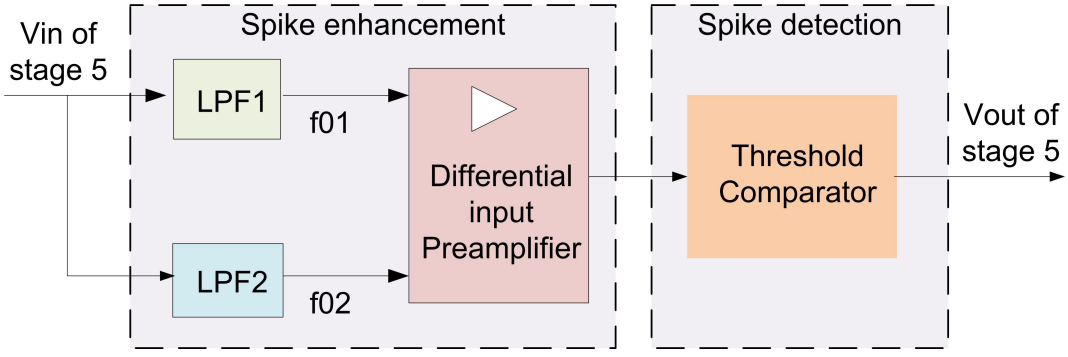

Figure 3.6: Functional blocks for a spike detector .

To emphasize a detected spike, two LPFs are set at different cut-off frequencies in order to subtract the lower frequency content from the higher one. The values of these two cut-off frequencies are decided upon by the type of recorded data and the noise of the recording environment. Ortiz [5] suggested an operational transconductance amplifier (OTA) to vary these cut-off frequencies. An OTA circuit can work as a voltage controlled filter (Figure 3.7) where the cut-off frequency is defined by equation 3.1 [55].

$$\frac{2\pi f_0 C}{g_m} = \frac{R_A}{R + R_A} \tag{3.1}$$

where $g_m = 19.2\ I_{ABC}$ at a room temperature (thermal voltage = 25.8 mV) and I_{ABC} is the amplifier bias current.

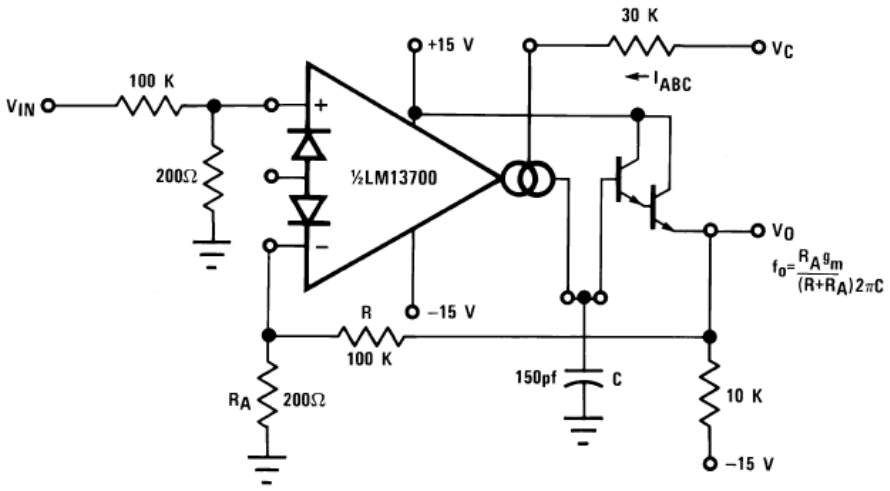

Figure 3.7: Example of a voltage controlled low pass filter. The cut-off frequency is defined in equation 3.1 [55].

When varying I_{ABC}, g_m is tuned and consequently f_0 is set to a new value. In our design, 1.4 k Hz and 5.3 k Hz (suggested by [56]) are set as default values for these two LPFs.

The spike detection block is basically a model of a non-inverting Schmitt trigger (Figure 3.8 [57]). In this positive feedback model, the output retains its value (high or low) until the input goes over a level to trigger a change. In Figure 3.8, the

threshold level is controlled by the proportion between R_1 and R_2 as follows.

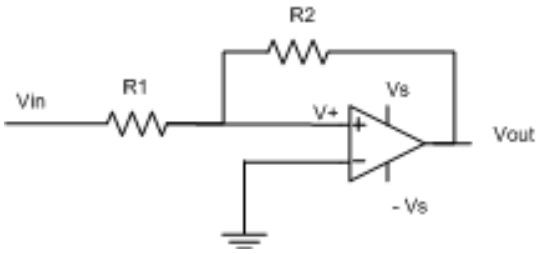

Figure 3.8: Model of a Schmitt trigger [57].

If the trigger is currently at a high state ($V_{out} = +V_S$), the input V_+ is given by:

$$V_+ = \frac{R_2}{R_1 + R_2} V_{in} + \frac{R_1}{R_1 + R_2} V_s. \tag{3.2}$$

The comparator will switch to $V_{out} = -V_S$ when V_{in} goes over a low threshold level of $-\frac{R_1}{R_2} V_s$. To switch back to the high state, V_{in} needs to go over a high threshold level of $+\frac{R_1}{R_2} V_s$. In our spike detection block, a flexible reference voltage (V_{ref}) is applied (Figure 3.9) in order to accommodate a particular recording experiment. V_{ref}, which is controlled by a potentiometer, determines two switching levels V_{H-to-L} and V_{L-to-H} as in equations 3.3 and 3.4.

$$V_{H-to-L} = V_{ref}(1 + \frac{R_1}{R_2}) - V_s \frac{R_1}{R_2}. \tag{3.3}$$

$$V_{L-to-H} = V_{ref}(1 + \frac{R_1}{R_2}) + V_s \frac{R_1}{R_2}. \tag{3.4}$$

3.3 System-Level Design

Though the complete analog PCB is designed for eight channels (Figure 3.10), all channels have a similar structure. Therefore, the detailed diagram of only one channel is shown in Figure 3.11. Table 3.1 lists core components of the analog PCB with their operating conditions. To fit well into a small rack of a hybrid bio-robot,

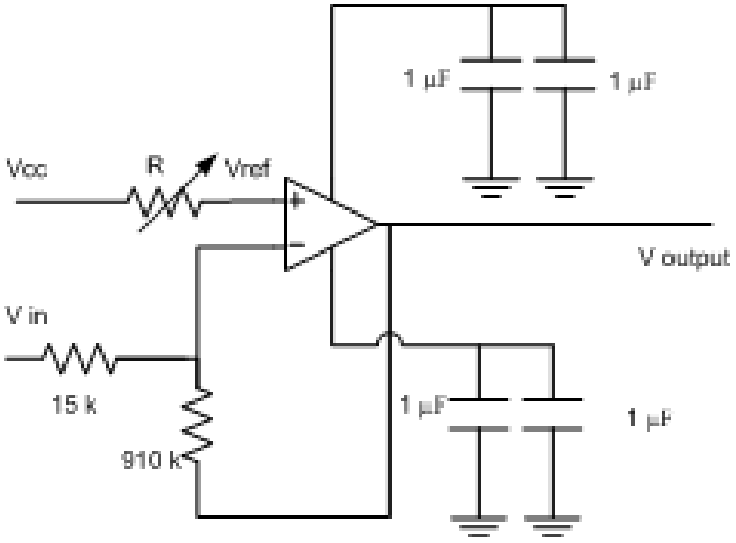

Figure 3.9: Threshold setup with a flexible reference voltage.

the PCB should be very thin and small. Figure 3.12 and Figure 3.13 capture two sides of a recent fabricated version in our laboratory. Table 3.2 shows its main specifications.

3.4 Two-Wire Bus Interface Programming

The Inter-Integrated Circuit (IIC, I^2C, or I2C) interface is a multi-master serial single-ended computer bus invented by Philips [58]. Two Wire Interface (TWI) or Two-Wire Serial Interface (TWSI) is defined by Atmel [59] and other vendors to avoid conflicts with trademark issues. TWI devices are considered compatible to I^2C devices except for some particularities like general broadcast or 10 bit addressing [58]. In the later part of the thesis, we use "I^2C" as a general name for this type of protocol.

The I^2C protocol involves two bidirectional open-drain lines to send and receive data: serial data line (SDA) and serial clock line (SCL). While SDA is used to transfer data between master devices and slave devices, SCL defines a clock signal that allows devices take turns in communicating. At the rising edge of a clock signal, a bit of information is transferred from a master to a slave over the SDA line. At

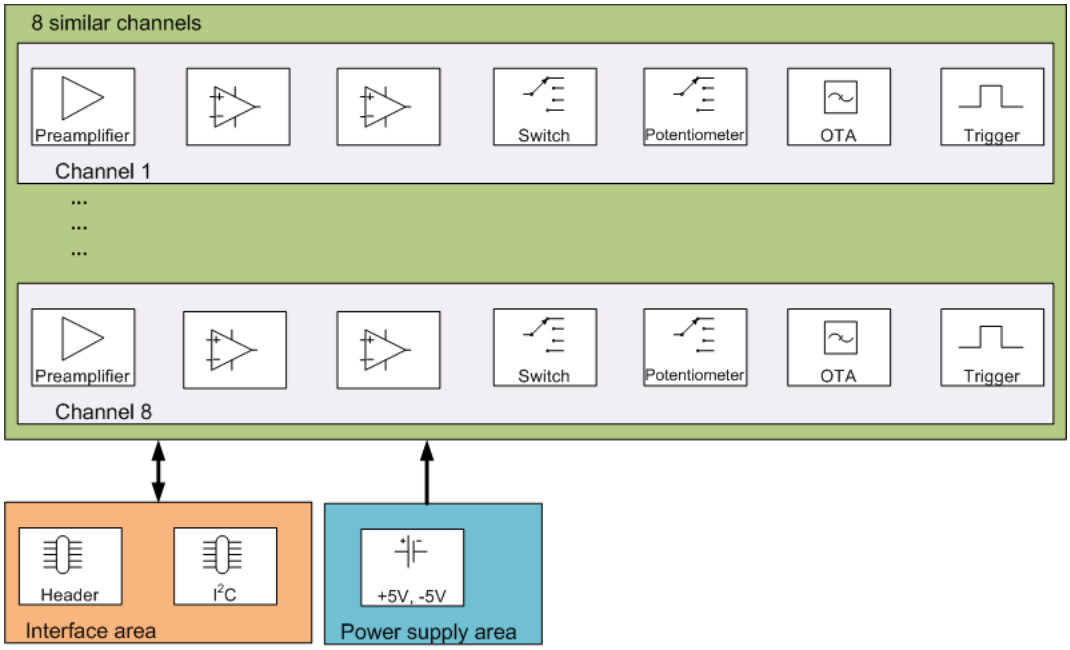

Figure 3.10: Schematic of the complete analog processing module.

Table 3.1: Operating conditions of the analog PCB.

Component	Quantity	Operating temperature	Pins	Voltage supply range
INA	8	$-40°C : +85°C$	8	$\pm 6\ V : \pm 18\ V$
OPA2111	8	$0°C : +70°C$	8	$\pm 18V DC$
OPA705	16	$-40°C : +85°C$	14	$\pm 2\ V : \pm 6\ V$
ADG	8	$-40°C : +85°C$	32	V_{DD} $-0.3V$: $+15V$. V_{SS} $+0.3V$: $-7V$. V_L $-0.3V$: $+7V$.
X9258	16	$-40°C : +85°C$	24	$V_+ - V_- < 12V$. $V_{SDA,SCL}$ $-1V : +7V$.
LM13700	8	$0°C : +70°C$	16	$\pm 18V$

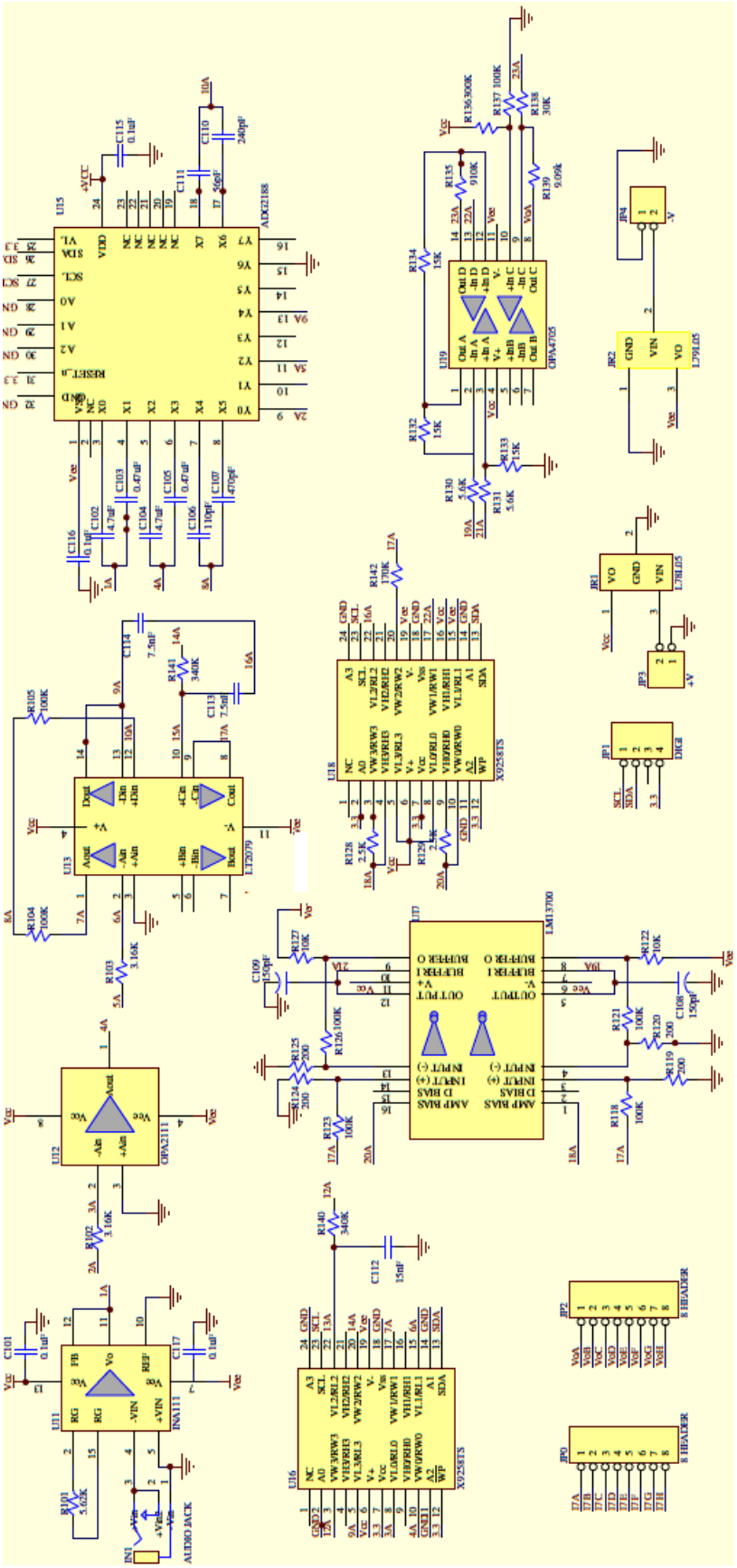

Figure 3.11: Detailed diagram represents one channel.

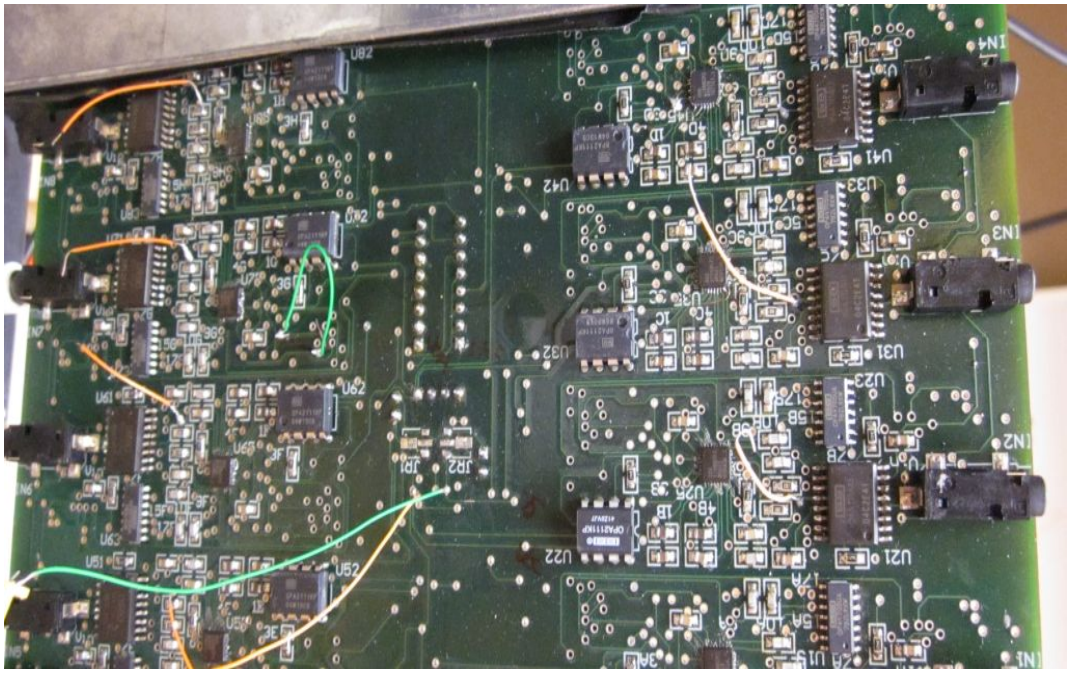

Figure 3.12: Fabricated analog PCB in the Higgins laboratory- side A.

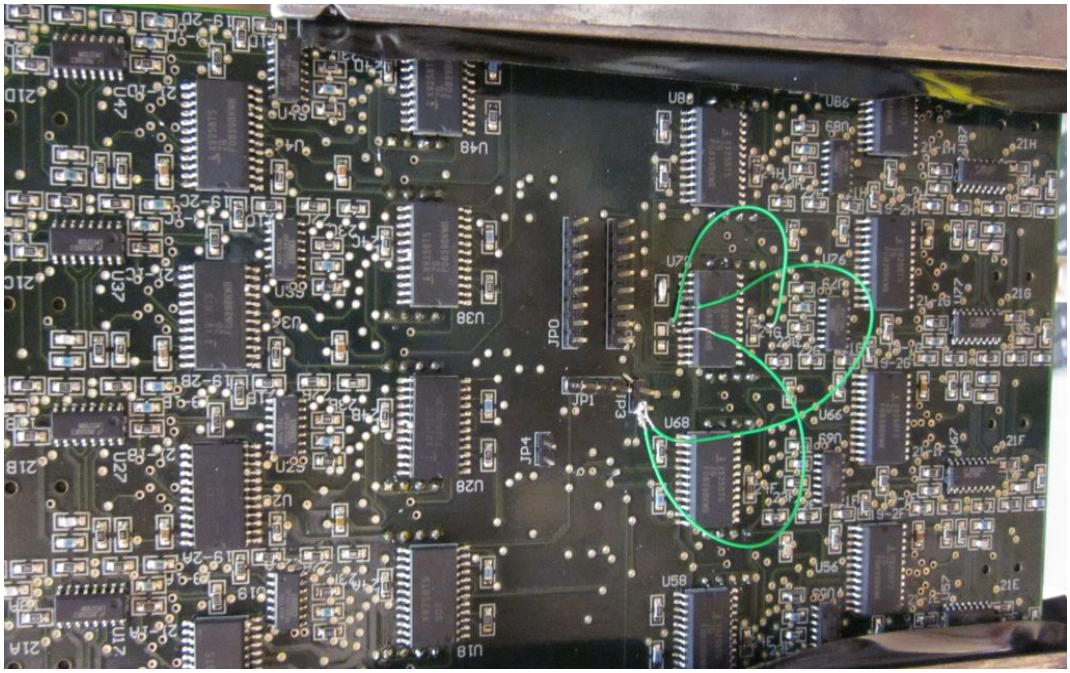

Figure 3.13: Fabricated analog PCB in the Higgins laboratory- side B.

Table 3.2: Specifications of the fabricated analog PCB.

Size	7.5" x 4" x 0.062"
Weight	< 1 lb.
Voltage supplies	+3.3 V; $\pm$5V
Power consumption	0.5 W $\rightarrow$ 0.7 W
Bandwidth gain	100x, 10000x
Notch cut-off frequencies	50 Hz or 60 Hz
Notch rejection	< -40 dB
LPF cut-off frequencies setting	4.7 k Hz or 20 k Hz
HPF cut-off frequencies setting	10 Hz or 100 Hz

the falling edge of a clock signal, the slave transmits data back to the master over the same line. Figure 3.14 shows the network of I^2C communication in the analog PCB.

Because of sharing the same bus, all I^2C devices must have unique addresses. Together with hard-wired addressing bits (manufacturer-address bits), designers use user-defined bits to name these devices. For example, with potentiometers of Intersil, four bits "0101" indicate the maker and four bits (A3A2A1A0) are used to name sixteen sixteen potentiometer-chips [60]. With switches of Analog Devices, the manufacturer uses only seven bits for addressing; four hard-wired bits of "1110" and three user's bits of "A2A1A0" to name eight switches on board [61]. I^2C messages have a format of four parts. The first part defines whether an 8-bit address or a 7-bit address type is used. The second one specifies address of the I^2C device. The third one names a particular component inside the device. The last part declare a value of setting.

Each time we program these types of devices, we often want to assign a particular value for the devices. For example, if we want to turn on or off digital switches to connect the desired capacitor in the circuit, we send a message over the I^2C communication network. This message contains the address bits and instruction codes. Table 3.3 [61] lists several instruction codes for this programming example.

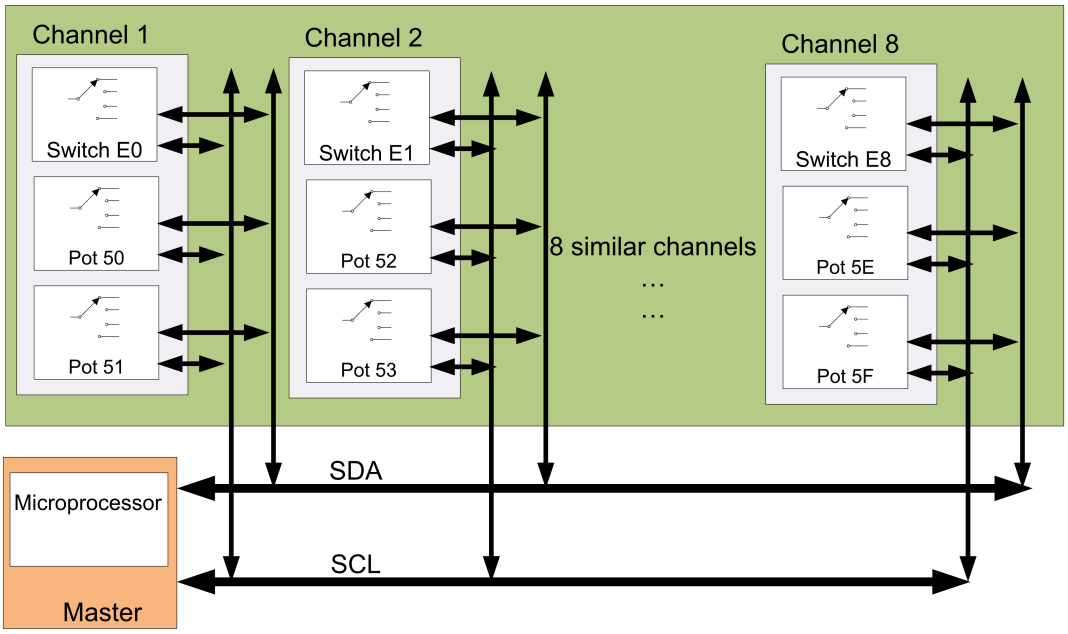

Figure 3.14: Network structure of the I^2C communication in the analog PCB. Each channel has one switch and two potentiometers. Eight channels share the same SDA and SCL buses with the master.

Table 3.3: Instructions for the I^2C code to turn on/off switches.

Switch location	"on" (hex)	"off" (hex)
X0 and Y0	80	00
X2 and Y2	92	12
X1 and Y0	88	08
X3 and Y2	9A	1A
X4 and Y4	A4	24
X7 and Y6	CE	4E
X5 and Y4	AC	2C
X6 and Y6	C6	46

CHAPTER 4

DIGITAL PROCESSING MODULE DESIGN

This chapter discusses the digital PCB design. Key tasks of the digital PCB include converting analog signals to digital signals, configuring the whole system, executing a spike-sorting application, and transmitting data. In the last section, a brief description of how to set up the digital PCB for experiments is also given.

4.1 Functional Diagram Design

The digital PCB is described with its functional blocks in Figure 4.1. The analog-to-digital converter (ADC) converts the flow of analog signal inputs into their digital format versions. The microprocessor executes software modules. To support this microprocessor, there are blocks of clock-generating, memory, and digital power supplies. Followings are details of some principal blocks.

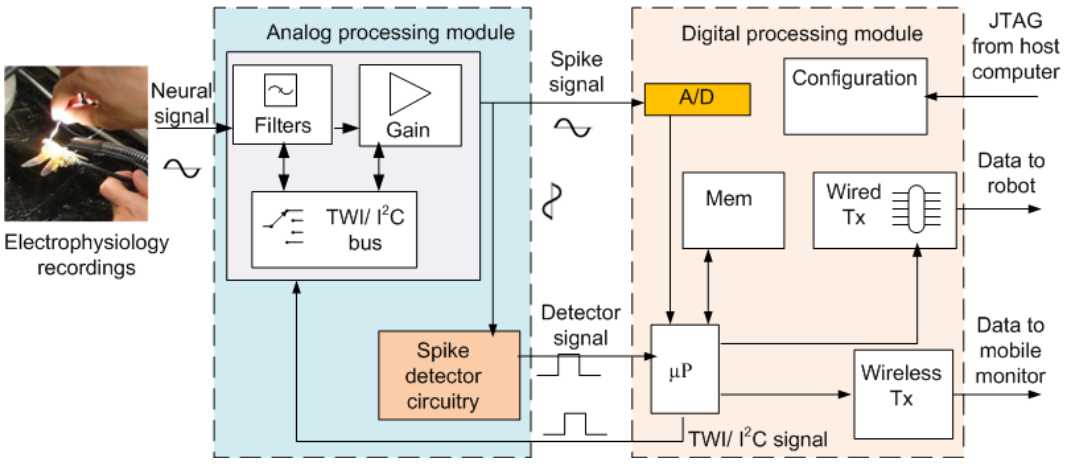

Figure 4.1: Functional blocks of the digital PCB.

4.2 Component-Level Design

We selected digital components upon some specific design criteria and availability of suitable products. Following are details of selected components.

4.2.1 Analog-to-Digital Converter

There are several design requirements for a selected ADC. First, the ADC should support a voltage range of bipolar 5 V to deal with both positive and negative action potentials. To sort spikes more accurately, the ADC should provide high-resolution data. Because neural signals have high frequency components [62], the ADC should operate with high sampling rates in the 50 k Hz range. Other requirements such as small space occupation and low power consumption are also critical. In this project, an 8-channel 12-bit ADC with ± 5 V analog input range was chosen with a functional diagram illustrated in Figure 4.2 [63]. The basic operation of this ADC is described briefly below.

To preserve phase information across the multichannel ADC, all input channels have dedicated amplifiers for track and hold (T/H). The T/H circuit is controlled by an input CONVST. When CONVST is low, the T/H circuit tracks the analog input. When CONVST is high, the T/H circuit holds the analog input.

A rising edge of CONVST is a sampling instant of the analog input. To get 12-bit accuracy, CONVST must be low for at least 100 ns and an acquisition time (t_{ACQ}) should be limited from 100 ns to 1 ms [63]. To start a conversion, the ADC receives a low CONVST signal for a duration of t_{ACQ}. The T/H circuit acquires the signal while CONVST is low, and the conversion begins on the rising edge of CONVST. The end-of-conversion signal (EOC) is low whenever a conversion result becomes available to be read. The end-of-last-conversion signal (EOLC) goes low when the last conversion result is available. Until the last conversion is read, CONVST is high to avoid aborting the current conversion. Before CONVST is low, there must be a period of bus inactivity (t_{Quiet}) for at least 50 ns [63]. The timing of a conversion for eight channels is illustrated in Figure 4.3.

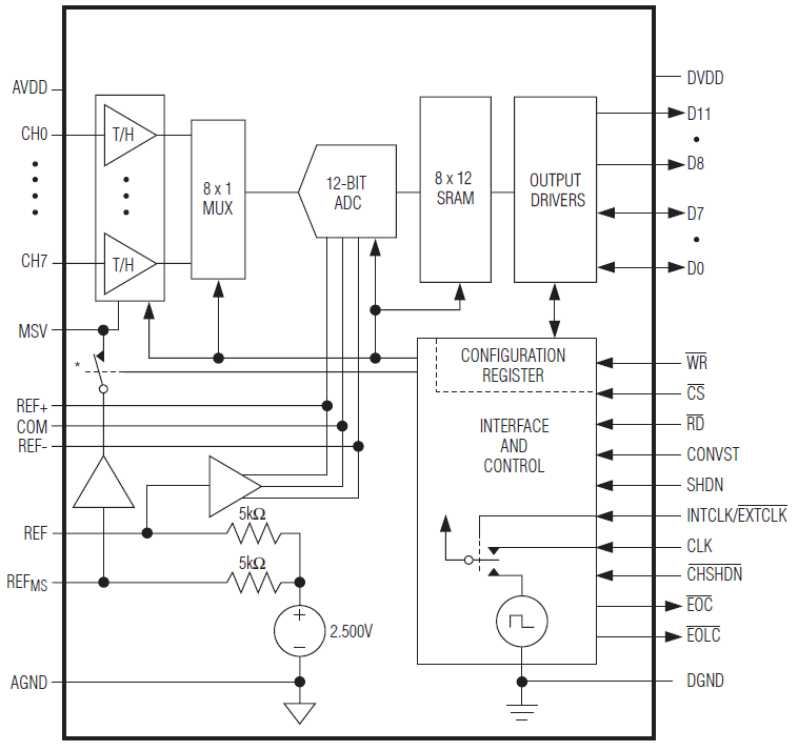

Figure 4.2: Functional diagram of the selected ADC [63].

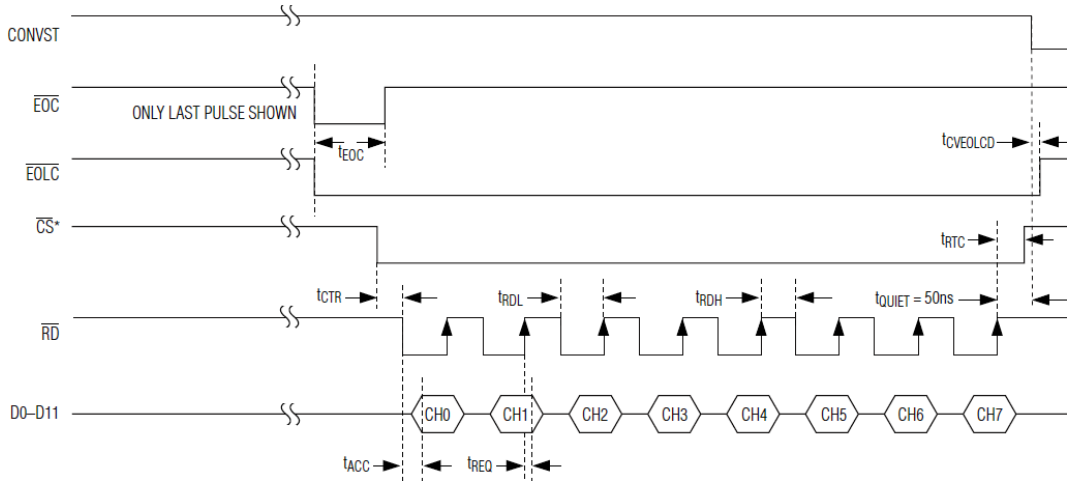

Figure 4.3: Timing of an A/D conversion for eight input channels [63].

Under typical operating conditions, when $f_{CLK} = 15\ M$ Hz (internal clock mode), $t_{ACQ} = 100\ ns$, and $t_{Quiet} = 50\ ns$, the number of clock cycles until reading the last result is 33 *cycles* (total conversion time = 2.2 μs) and the data throughput Φ will be 413 *ksps/channel* (from Equation 4.1 [63]).

$$\Phi = \frac{1}{t_{ACQ} + t_{QUIET} + \frac{12+3(N-1)+1}{f_{CLK}}} \tag{4.1}$$

where N is the number of active channels and f_{CLK} is the clock speed.

4.2.2 Data Processing

The spike data processing involves a soft-core processor and a memory subsystem with basic datapaths as shown in Figure 4.4. The soft-core processor is generated in a hardware description language (HDL). Most FPGA devices have on-chip memories. These memories' capacities are often not adequate for multi-channel neural signal processing tasks. Hence, it is necessary to implement external memories (SRAM or Flash). The parallel input/output interface and serial input/output interface enable the FPGA system to communicate with other parts of the system. An initial configuration step is required for FPGA-based systems whenever loading a HDL-coded design.

Because of the mobility requirement and convenience for power saving, an automatic configuration ability of this FPGA device is preferred. From the recommendation of the FPGA manufacturer [64], we use an embedded memory as a configuration memory. It can support the active parallel (AP) configuration with a recommended connection diagram between the processor and the memory (Figure 4.5).

4.3 System-Level Design

Similar to the system-level design requirements of the analog PCB, the digital PCB also needs to have small dimensions, consume little power supply, and introduce as little noise as possible. Table 4.1 lists key components with their operating

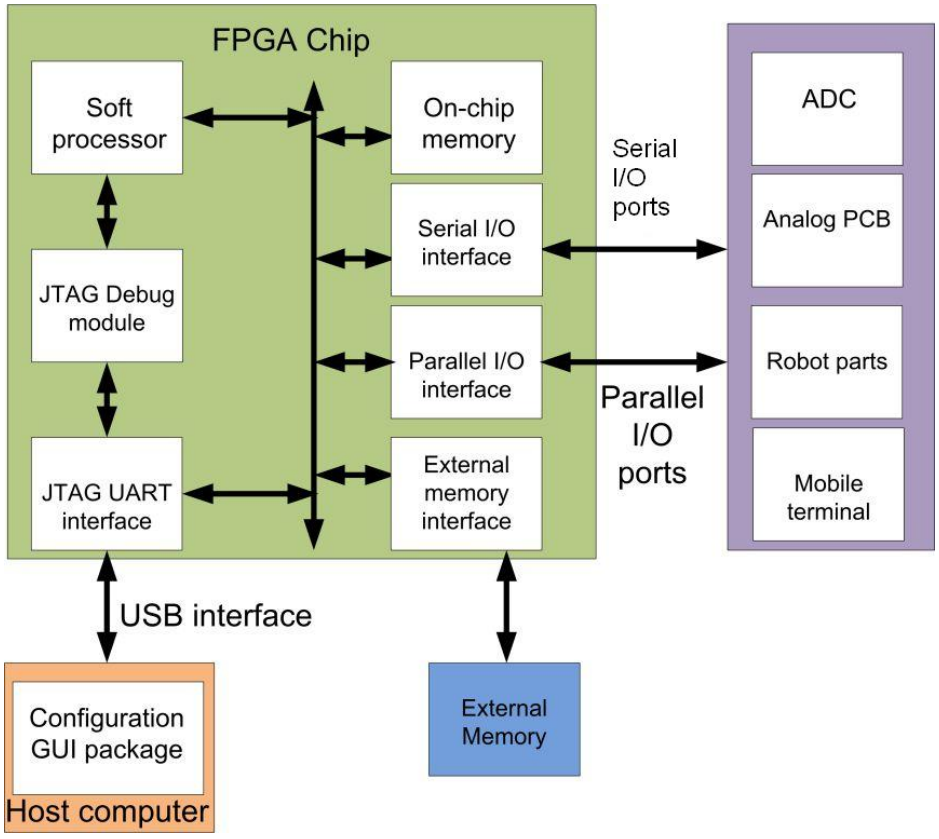

Figure 4.4: Datapaths for data processing in the digital module.

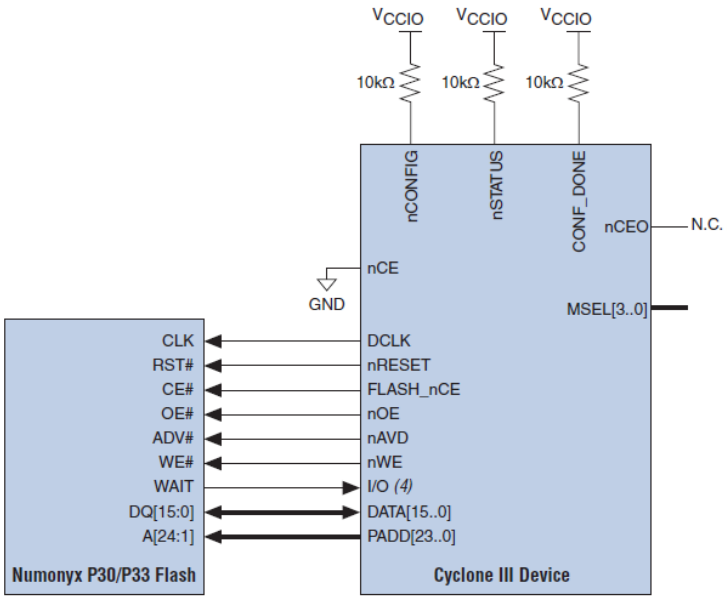

Figure 4.5: Recommended connection for the embedded memory to support the AP mode [64].

conditions. Figure 4.6 captures a recent fabricated version of the digital PCB in the Higgins laboratory.

4.4 FPGA-Based System Configuration

To launch a software application on an FPGA system, a tool of system integration (SI) made by the manufacturer of the FPGA is often utilized. The recommended system design flow is described in Figure 4.7. The tool generates two kinds of outputs: HDL files for the hardware flow and a system description, also called board support package (BSP), for the software flow [64].

A high-level description of the system architecture can be illustrated in individual blocks (Figure 4.8) or in a single block (Figure 6.6). Blocks are designed at a high level of abstraction using HDL components. The system fabric that connects the design blocks together is generated automatically by a SI tool. Once the soft-core processor is generated, the first-time FPGA configuration is performed with a

Table 4.1: Operating conditions of main components in the digital PCB.

Component	Operating temperature	Pins	Power supply	Footprint (mm)
ADC	$-40°C : +85°C$	48	$5V 1.3mA$	7×7
Flash	$-40°C : +85°C$	64	$2.3V \rightarrow 3.6V$	10×13
FPGA	$0°C : +85°C$	484	internal logic: $V_{CCINT} = -0.5V \rightarrow 1.8V$. output buffer: $V_{CCIO} = -0.5V \rightarrow 3.9V$. PLL: $V_{PLL} = -0.5V \rightarrow 1.8V$.	23×23
Bluetooth	$-40°C : +85°C$	-	$3V \rightarrow 3.6V$	13.4×25.8

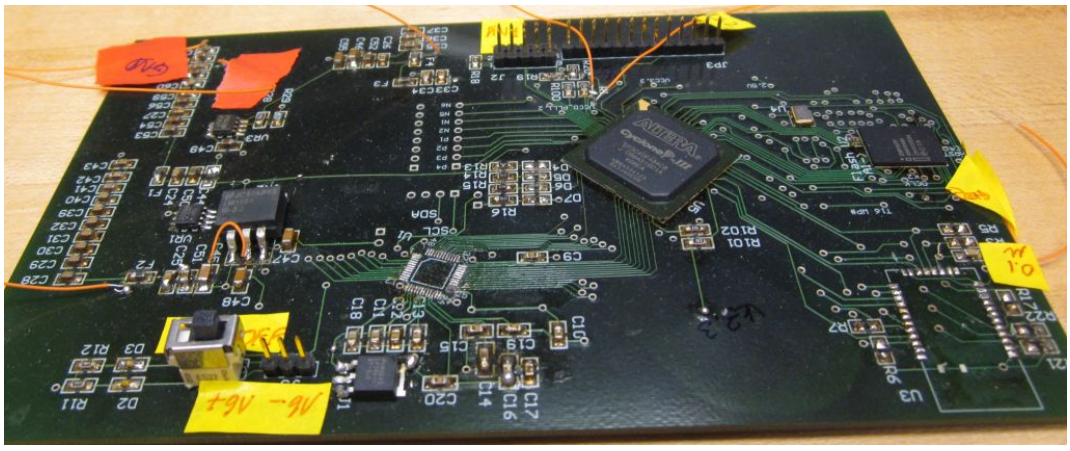

Figure 4.6: Recently fabricated digital PCB version in Higgins laboratory.

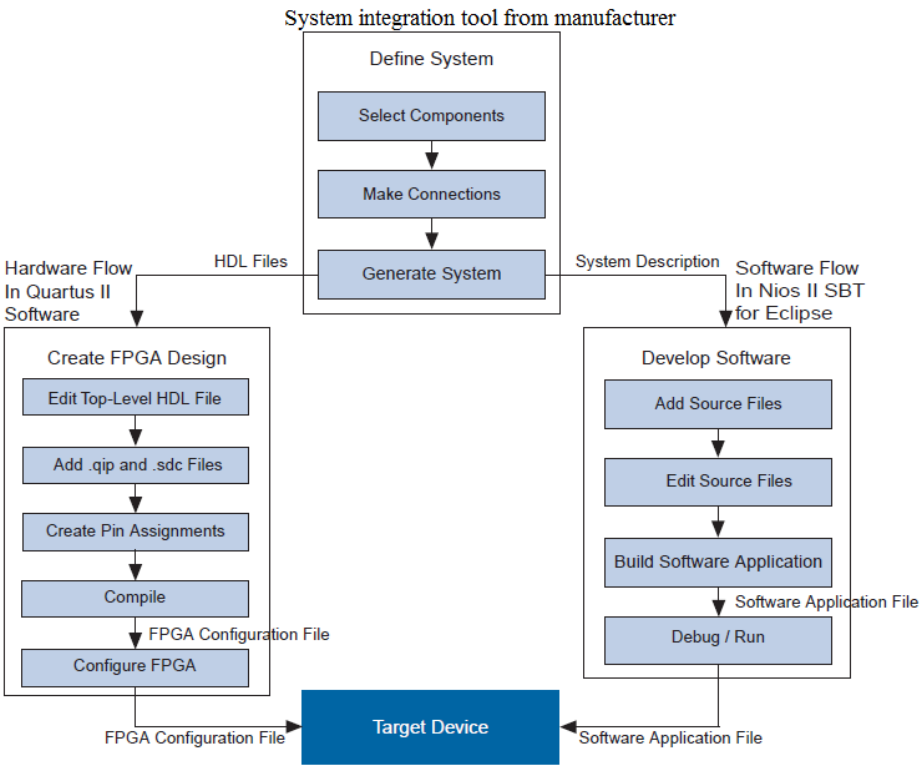

Figure 4.7: Recommended system design flow for FPGA-based systems [64].

desktop computer. Then, an application is loaded into the memory and is executed by the configured FPGA.

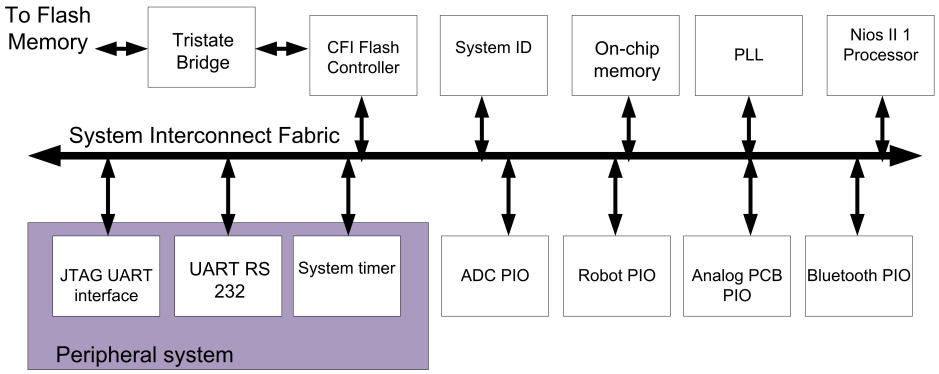

Figure 4.8: Block diagram of a soft-core processor at high level.

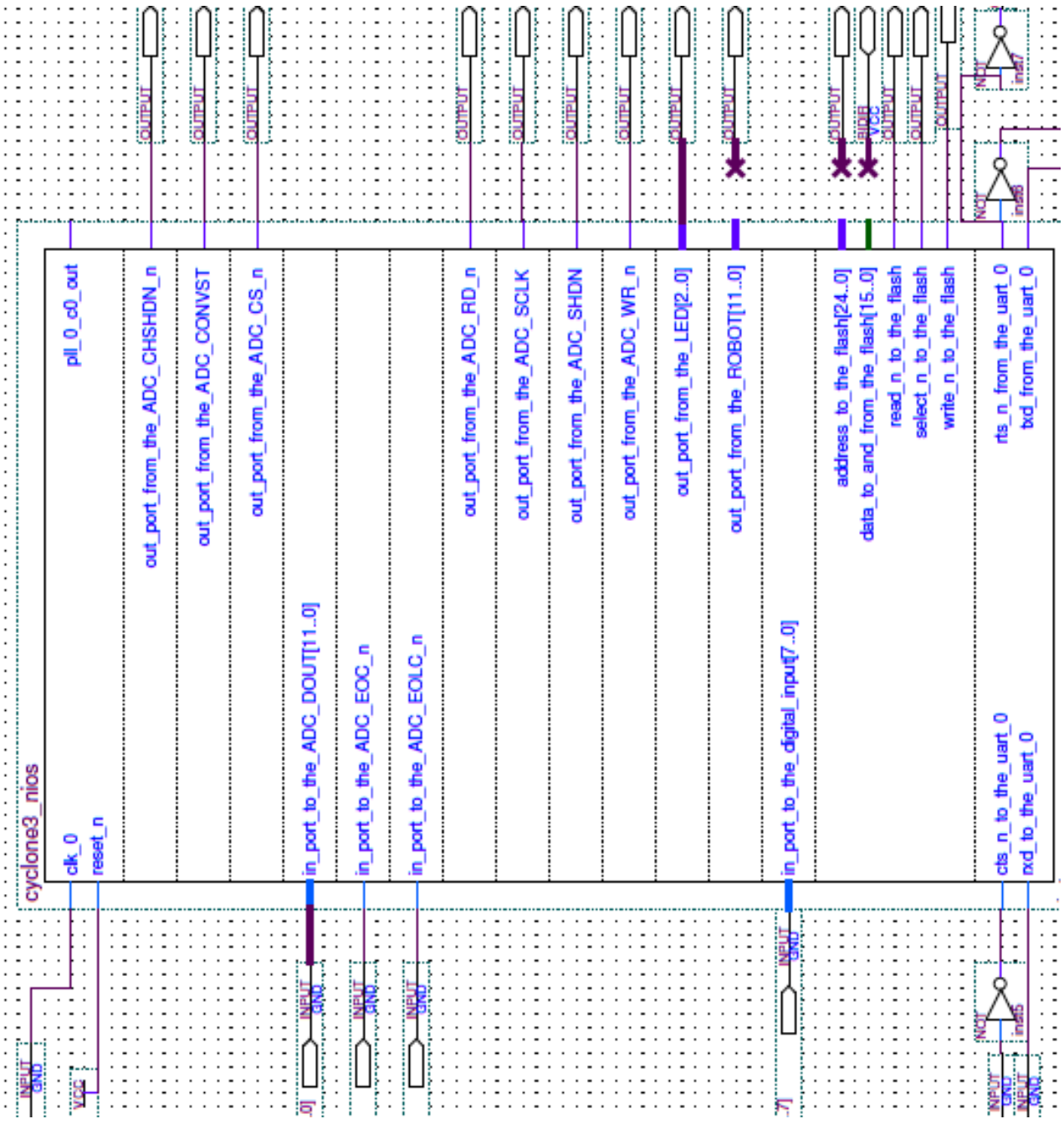

Figure 4.9: Black-box view for the soft-core processor at high level.

CHAPTER 5

SPIKE-SORTING SOFTWARE APPLICATION

This chapter builds a software application module for the discussed hardware structures in the two previous chapters. This software application is designed to sort out spikes from neural signals recorded from dragonflies. After discussing the neural information interpretation, the main data processing procedures are presented. The data flow for the whole module is then depicted, followed by pseudo-code of the principal procedures.

5.1 Neural Information Interpretation

As discussed in Chapter 2, the eight descending neurons called TSDNs contain the visual "sensing" signals from the brain when a dragonfly is seeing a moving target. If we can collect electrophysiological signals from these TSDNs, we can use the visual receptive field properties discovered by Frye et al. [9] to guide a robot platform. For that purpose, we need to detect when and which TSDN fires a spike from given recording signals. A time instant of firing can be found by applying a spike-detection algorithm (discussed in Section 2.4). To know which one of the eight TSDNs a spike belongs to, we use a template-matching method. Specifically, if a detected spike form matches very well (quantified by a high correlation coefficient) with a spike-template of a TSDN, that spike is considered belonging to that TSDN.

We build spike-templates for each of the eight TSDNs by giving special visual stimuli then collecting appropriate spike-form samples. These special visual stimuli are designed to strongly stimulate a particular TSDN according to given properties. For example, in experiments by Frye et al [9], if a moving target starts from the bottom right corner of a stimulus screen with anterior orientation, there will be spikes that mostly come from the MDT2 neuron. After detecting all spikes in this

Table 5.1: Visual stimulation arrangement and corresponding templates.

Orientation	Starting at	Segment	Templates
Posterior	Bottom far right	Early	DIT3
Posterior	Bottom far right	Later	DIT1
Anterior	Bottom far right	Whole	MDT2
Dorsal	Top center	Whole	MDT1
Dorsal	Top left	Whole	MDT3
Ventral	Bottom right	Whole	MDT4

Table 5.2: Stimulus orientation with respect to the animal is denoted by angle in experiments of the Higgins laboratory.

Direction	Degree
Dorsal	270^o
Ventral	90^o
Posterior	0^o
Anterior	180^o

stimulation case, we can cluster them to find the largest group of similar spike samples. The average spike form of this group is then considered a spike-template of MDT2. With this approach, the thesis builds a total of six templates and names them after the respective TSDNs. Five templates are for all strongly directionally selective neurons (MDT1, MDT2, MDT3, MDT4, and DIT1). The sixth template is for DIT3, serving as an example of a neuron that is not strong directionally selective.

These six spike-templates are built upon the specifications of visual stimulation arrangements (Table 5.1). In our experiments, to record electrophysiological signals from the ventral nerve cord of a dragonfly, the dragonfly's eyes are upside down (Figure 5.1). Hence, we denoted the orientation of a stimulus with respect to the animal by angle (Table 5.2).

Referring to Section 2.1 for the visual receptive field properties, when analyzing the 270^o stimuli starting at the left corner of the stimulus screen, spikes that come from MDT3 would dominate the observation. Therefore, if we cluster all detected

Figure 5.1: The dragonfly's eyes are upside down in experiments of the Higgins laboratory.

spikes, the largest class would present MDT3 spikes. Similarly, with the 270^o stimuli starting at the right corner, the largest class of spikes after clustering would present MDT1 spikes. In the same way, we find MDT4, MDT2, DIT3 and DIT1 from the data sets containing stimuli of 90^o , 0^o, and 180^o.

Once these templates are built, they will be used to sort out spikes by the template-matching method. Finally, the information of when and which TSDN fired a spike will be used to control a robot platform.

5.2 Neural Signal Processing

From the above interpretation, in the software module, there are two main steps of data processing: building spike-templates and sorting out spikes. The former constructs templates from several stored data sets. Thus, this phase is not executed in real time. By contrast, the latter matches these templates with detected spikes in real time. Hence, to provide the desired functionality, the software application module is designed to perform both non-real-time procedures (shown in the diagram of Figure 5.2) and real-time procedures (shown in the diagram of Figure 5.3).

In the data flow of the non-real-time procedures, there are four chief software-based functions: detecting spikes with the NEO algorithm, extracting spike features with the DWT algorithm, clustering spikes with the SPC method, and building templates with the averaging method. Signals from the stored data set are analyzed with the NEO method to extract spikes. Then, spike features are found by using

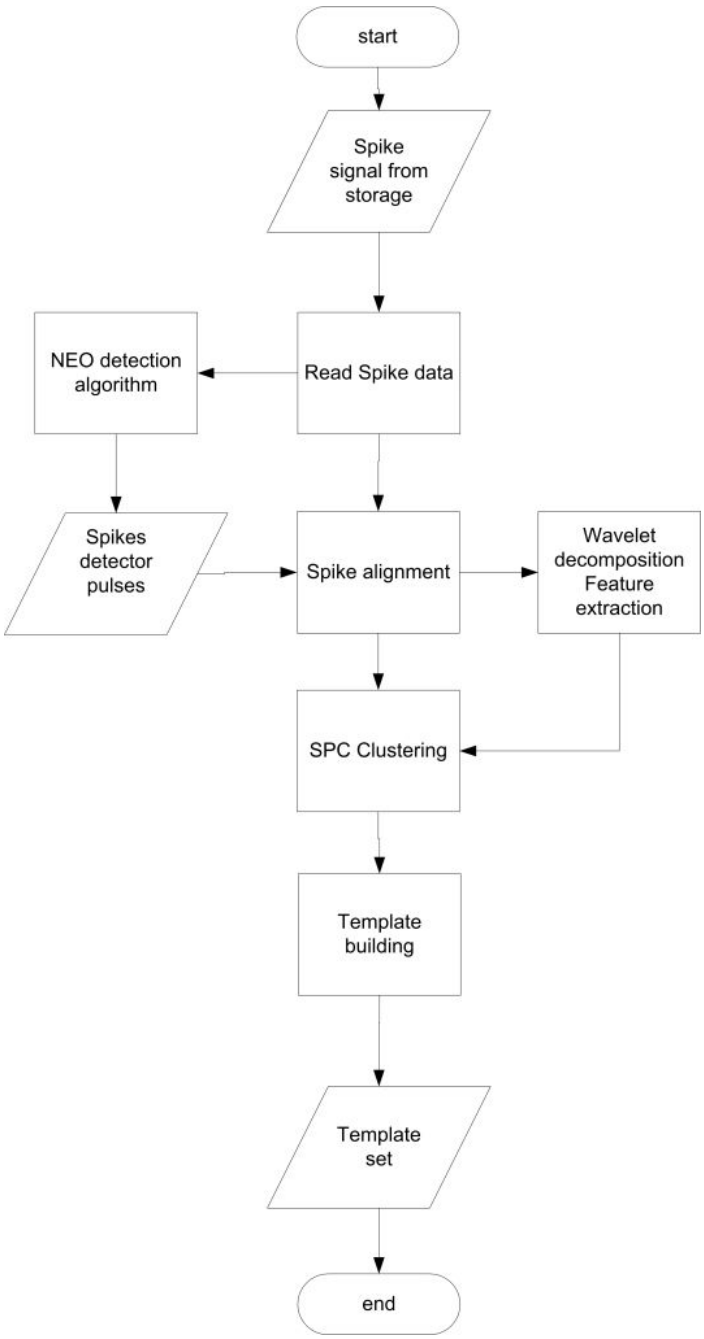

Figure 5.2: Data flow diagram of non-real-time procedures to build the templates from the data sets of the special stimuli.

the wavelet decomposition method. Based on these features, spikes are clustered with the SPC method into classes. The corresponding spike-template is built by averaging spike-forms of the largest class.

In the data flow of the real-time procedures, spike detection can be performed by a circuit in the analog PCB or by the software-based NEO method. Then, spikes are extracted from real-time recorded signals. Finally, these spikes are compared with predefined spike-templates. If a spike matches well with a spike-template, the time instant and the corresponding TSDN are written in a histogram.

5.2.1 Spike Detection Pseudo-Code (Algorithm 1)

As we described in the review in Section 2.4, we choose to detect spikes with the software-based NEO method. From our experiments, the adjacent time parameter is set to $\delta = 4$ in equation 5.1 to find the NEO data:

$$\psi\left[x\left(n\right)\right] = x^2\left(n\right) - x\left(n+4\right)x\left(n-4\right) \tag{5.1}$$

where $x\left(n\right)$ is the spike signal at the time instant of n.

The threshold level for the NEO data is chosen with the parameter C set to $C = 8$ in equation 5.2

$$\Theta = 8\frac{1}{N}\sum_{n=1}^{N}\psi[x(n)] \tag{5.2}$$

where N is the number of samples.

In Algorithm 1, x is a neural signal and y is the output spike data. Whenever the NEO data of x suddenly jumps over the threshold Θ, one spike is counted and y is copied from x for a spike duration M.

5.2.2 Spike Extraction Pseudo-Code (Algorithm 2)

This function is used only in the non-real-time data flow to build spike-templates for TSDNs. From Section 2.4, we create a wavelet decomposition vector for each aligned

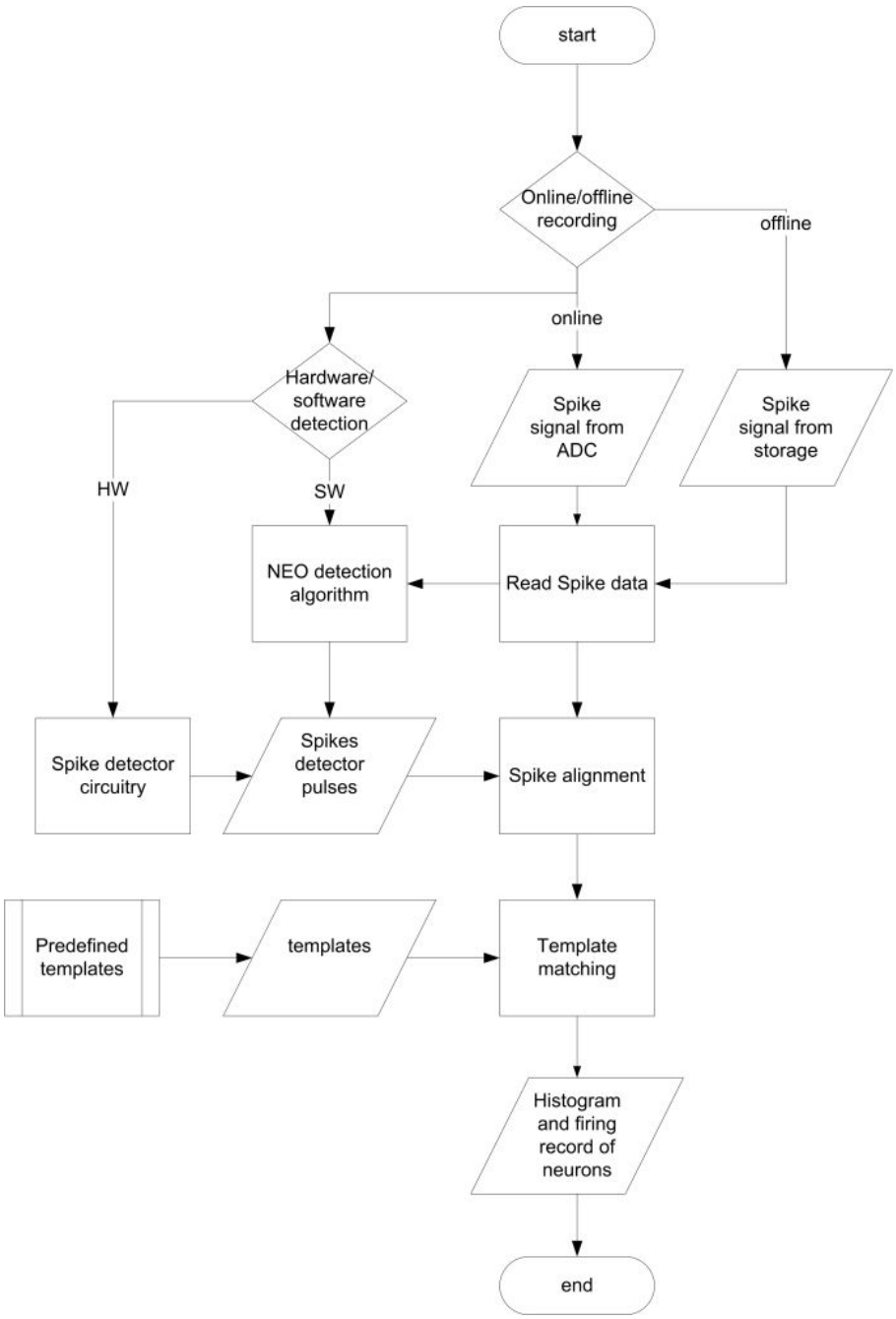

Figure 5.3: Data flow diagram of real-time operation in the system. The templates built from the non-real-time procedures are used to match a spike.

Algorithm 1 Spike detection with the NEO method.

Input:
f_s, % sampling rate
x; % neural signal
Constant:
M, % spike length
δ, % adjacent time interval
C, % constant varies threshold
Output:
y, %spike signal
count; % number of spikes detected
Begin
$N = length\,(x)$;
FOR $n = \delta + 1 \rightarrow N - (\delta - 1)$
$\psi\,[x\,(n)] = x^2\,(n)\; - x\,(n+\delta)\,x\,(n-\delta)$;
ENDFOR
$T = C\frac{1}{N}\sum_{n=1}^{N}\psi[x(n)]$;
WHILE (! end of x)
IF($(\psi_x(n-1) < T)$ & $(\psi_x(n) > T)$)
Copy y from x during a spike length;
count ++;
ENDIF
ENDWHILE
End;

spike to extract spike features. These spike features are stored in a matrix for every spike.. Spikes having similar features will belong to the same group of spike-forms. In our experiments, to save computation time, we reduce the dimension of the vector by selecting the top ten most significant coefficients.

Algorithm 2 Feature extraction with the DWT algorithm.

Input:
N, % number of spikes;
x; % is an aligned spike set matrix
Constant:
L, % spike length
Output:
y; % is a matrix of feature coefficients
Begin
FOR $i = 1 \rightarrow N$
$C_{(i,1\rightarrow L)}$ = wavelet decomposition of i^{th} spike;
ENDFOR
Take the top ten most significant coefficients % Reduce the matrix dimension
End;

In Algorithm 2, the input x is an aligned-spike-set matrix and y is the output matrix of feature coefficients. The vector of coefficients $C_{(i}$ is found from the wavelet decomposition of the i^{th} spike where i is an integer index. Finally, the dimension of the coefficient matrix is reduced to save computation time.

5.2.3 Template Building Pseudo-Code (Algorithm 3)

This function is the final step of the non-real-time data flow to build spike-templates. After finding features of all spikes, we cluster them into groups. The largest group often provides the suitable information to build an appropriate template. Though we have groups clustered automatically, we assess reasonable template information manually. Hence, the performance of the whole application depends not only on the special visual stimuli we set up, and on the rightness of the receptive field properties discovered by Frye et al. [9] but also on the experience of the person who assesses the classes for template information gathering.

In Algorithm 3, the input x is a stored data set and *temp* is a template we want

Algorithm 3 Building templates.

Input:
x; % raw data from data set of special stimuli
Output:
$information$, % template information sets
$temp$; % a template we want to build
Begin
y= NEO detection(x);
z= Spike alignment(x,y);
c= Feature extraction(z);
$classes$= Clustering(c);
$information$ =Saving ($classes$,z);
$temp$= Select class($classes$,y);
End;

to build. After finding spike features of x, a *clustering* function will return classes of spikes. We select a suitable class as a basis for the corresponding template *temp*. This function is repeated with different data sets for every template we want to construct.

5.2.4 Template Matching Pseudo-Code (Algorithm 4)

This function is executed in the real-time data flow. It uses the templates that are previously built to match every spike that the system detects. The relevant neuron corresponds to the best-matched template, i.e., the one that has the highest correlation coefficient with the spike (Equation 5.3). The highest correlation coefficient should also be higher than a level C that we expect the assignment is true.

$$r_{X_i,Y_j} = \frac{\mathcal{C}\{X_i, Y_j\}}{\sigma_{X_i}\sigma_{Y_j}} \tag{5.3}$$

where r_{X_i,Y_j} is the correlation coefficient between the i^{th} spike of a spike set X and the j^{th} template of a template set Y. $\mathcal{C}\{X_i, Y_j\}$ is the covariance of X_i and Y_j. σ_{X_i} and σ_{Y_j} are the variances of X_i and Y_j, respectively.

In Algorithm 4, the input X is an aligned-spike-set matrix and the input Y is a set of templates. r_{X_i,Y_j} is the correlation coefficient between the i^{th} spike of X

and the j^{th} template of Y. r_{X_i,Y_j} is calculated for every detected spike and every template. When comparing a spike with every spike-template, the largest coefficient is chosen and stored as z. C is the level to assume a template matching is right. In most cases, we set $C = 0.9$. If z is greater than C then the information of the right template and time instant is written in a histogram.

Algorithm 4 Template-matching.

Input:
X, % is an aligned spike set matrix
N, % number of spikes
L, % spike length
C, % expected correlation level
Y; % is a set of templates
Output:
histogram;
Begin
FOR $i = 1 \rightarrow N$
FOR $j = 1 \rightarrow L$
$r_{X_i,Y_j} = \frac{\mathcal{C}\{X_i,Y_j\}}{\sigma_{X_i}\sigma_{Y_j}}$;
$match(i,j) = r_{X_i,Y_j}$;
ENDFOR
$z = \max\{match(i,:)\}$;
IF $z \geq C$
FOR $j = 1 \rightarrow L$
IF $match(i,j) = z$
$match(i,j+1) = j$;
ENDIF
ENDFOR
ENDIF
ENDFOR
Record name of template and time instant;
End;

In summary, the above procedures of the software module provide two stages of neural signal processing for dragonflies to interpret neural information to electrical control signals. This software module is capable of accommodating future modifications for different applications of the proposed system.

CHAPTER 6

EXPERIMENTAL RESULTS

This chapter illustrates the main results of the two sub-projects within this thesis: validating the two fabricated PCBs and testing the spike-sorting software application with biological data sets. Each section presents information and methods along with the observed results.

6.1 Validating Two Fabricated PCBs

To characterize the analog PCB, we checked the gains and the cut-off frequencies of the analog PCB by using artificial signals and tested the functionality of the spike detection circuitry by using a biological analog signal. The analog PCB was verified independently from the digital PCB by using an I^2C-USB converter to communicate through a desktop computer. With this tool, we can send I^2C messages to the analog PCB and set configurable parameters.

By programming I^2C devices (Section 3.4 for more details), we validated the optional cut-off frequencies for the HPF and the LPF. For example, we set two switch cut-off frequencies for the HPF, 10 Hz or 100 Hz. For the case of 10 Hz, we turned on switches X2-Y2 and X0-Y0 while X3-Y2 and X1-Y0 were off. For the case of 100 Hz, we turned on X3-Y2 and X1-Y0 while X2-Y2 and X0-Y0 were off. For the notch filter setting, we set two cases of notch frequencies, 50 Hz or 60 Hz. We programmed potentiometers to have the settings of 84 $k\Omega$ and 42 $k\Omega$ for the case of $f_{notch} = 50$ Hz. For the case of $f_{notch} = 60$ Hz we programmed potentiometers to have the settings of 13 $k\Omega$ and 6.5 $k\Omega$.

The two procedures we used to validate the digital PCB are checking the power consumption and configuring the FPGA. We supplied this PCB with a DC voltage supply of $\pm 9V$ and plugged into a host computer through a USB-Blaster download

cable which sends configuration data from the computer to a standard 10-pin header connected to the FPGA. We used the Quartus version 10.0 software as an SI tool to build the soft-core processor for the FPGA. Finally, we tested this FPGA with an application that was implemented in the C programming language.

6.1.1 Analog PCB Frequency Response

We used a function generator to find the frequency responses of the analog PCB (Figure 6.1, Figure 6.2, and Figure 6.3). With switch settings of $f_{HPF} = 10$ Hz and $f_{notch} = 50$ Hz, Figure 6.1 showed a notch with a narrow stop-band at 40 dB of around 10 Hz. Because the notch is very close to the cut-off frequency, the measured $3dB$ point fell at 270 Hz.

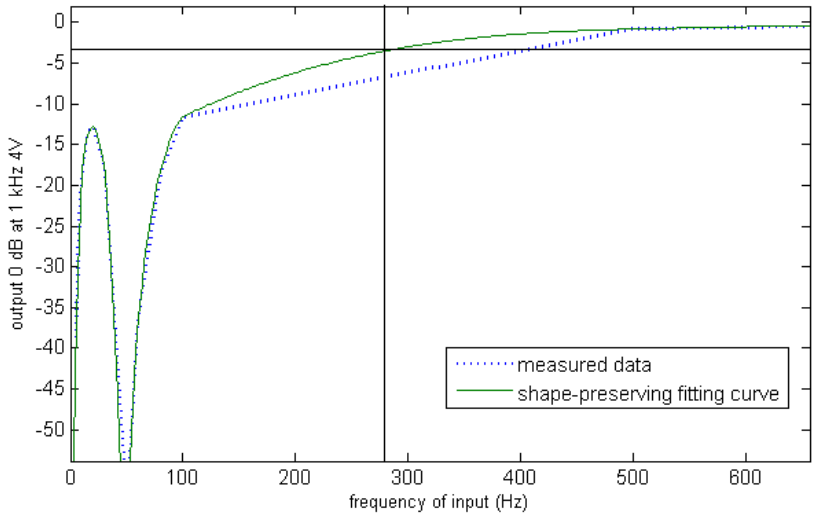

Figure 6.1: Frequency response with setting $f_{HPF} = 10$ Hz and $f_{notch} = 50$ Hz. A notch at $f_{HPF} = 50$ Hz was very clear with a narrow stop-band of around 10 Hz width at 40 dB. The measured 3 dB point fell at 270 Hz.

Figure 6.2 illustrates the frequency response with switch settings of $f_{HPF} = 100$ Hz and $f_{LPF} = 5$ k Hz. The measured 3 dB points for the HPF and the LPF were around 250 Hz and 3800 Hz, respectively. With switch settings of $f_{HPF} = 100$ Hz and $f_{LPF} = 20$ k Hz, the measured 3 dB points for the HPF and the LPF were around 250 Hz and 12000 Hz, respectively (Figure 6.3). Though the cut-off

frequencies did not exactly match the theoretical settings, this fabricated PCB still meets the demands of the research.

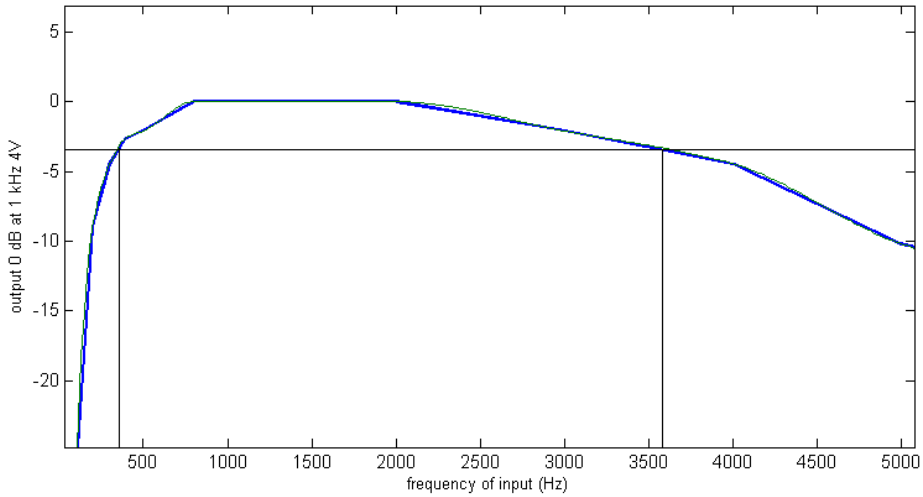

Figure 6.2: Frequency response with switch settings of $f_{HPF} = 100$ Hz and $f_{LPF} = 5$ k Hz. The measured 3 dB points for the pass-band were around 250 Hz and 3800 Hz.

6.1.2 Hardware-Based Spike Detection Functionality

We tested spike detection in the analog PCB with a biological signal (Figure 6.4). Figure 6.5 shows a photo of the oscilloscope during the detecting period by the circuit. The circuit detected spikes at very high speed, but because of the using very simple threshold algorithm it had a quite high false positive rate.

6.1.3 FPGA Configuration for the Digital PCB

Immediately after power-up, the digital PCB consumed a low electric current (around 70 mA). After the soft-core processor was generated successfully, the configuration file was loaded to the FPGA. During the process of configuration, the electric current of the digital PCB was up to about 130 mA. When the process finished, the current returned to the initial value.

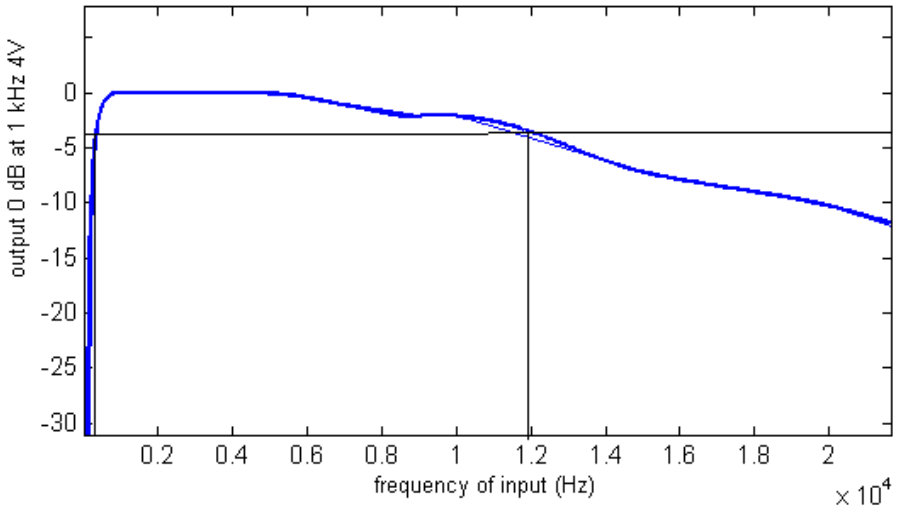

Figure 6.3: Frequency response with switch settings of $f_{HPF} = 100$ Hz and $f_{LPF} =$ 20 k Hz. The measured 3 dB points for the pass-band were around 250 Hz and 12000 Hz.

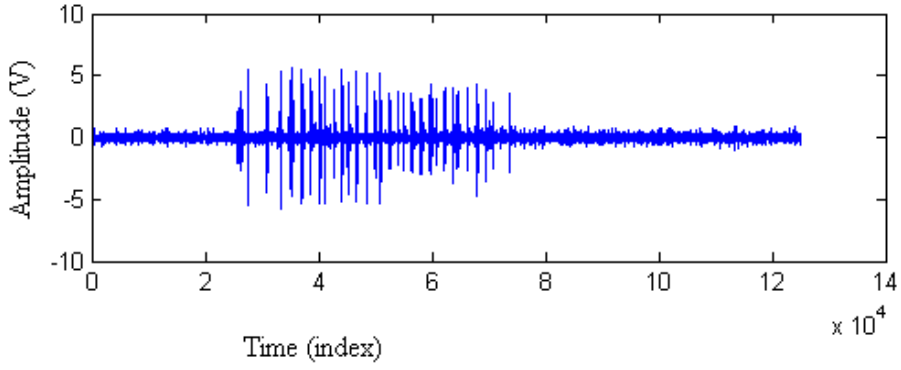

Figure 6.4: Biological data set used to verify the spike detection circuit. Index = *time* × *sampling rate*.

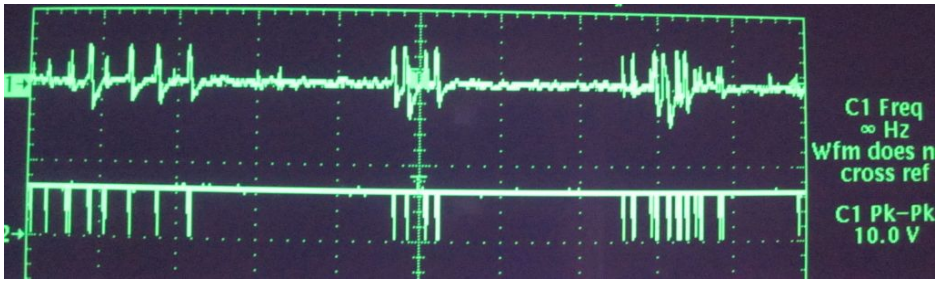

Figure 6.5: Oscilloscope snapshot during verification of the spike detection circuit. The top signal is the raw input signal. The bottom signal presents the detected pulses at the output of the circuit.

From the proposed design for the data processing module (Section 4.2.2), we built a soft-core processor (Figure 6.6) for the FPGA with the manufacturer's system integration tool. This tool generated two kinds of outputs: HDL files for the hardware flow and a system description (also called board support package (BSP)) for the software flow. Using a host computer and the JTAG communication ("Joint Test Action Group" standard), we loaded these output files into the FPGA for the configuration step.

After configuring the FPGA successfully, we started loading a software application for this processor. The application needed to be programmed in C language. However, we utilized the Matlab programming tool to design the spike-sorting application module of the proposed system (Chapter 5). Hence, in order to test the system further, we had to convert the current application module to the C language version. When finishing this task, the application is loaded into the memory and is executed by this configured FPGA. We have not passed this step of testing because we have not been able to load the testing application into the memory device of the current fabricated digital PCB. The problem might be some damage in some internal parts of the components when we tried to fabricate and populate the boards by ourself. We should have them fabricated by a professional service provider. This problem can be solved in future projects of the Higgins laboratory.

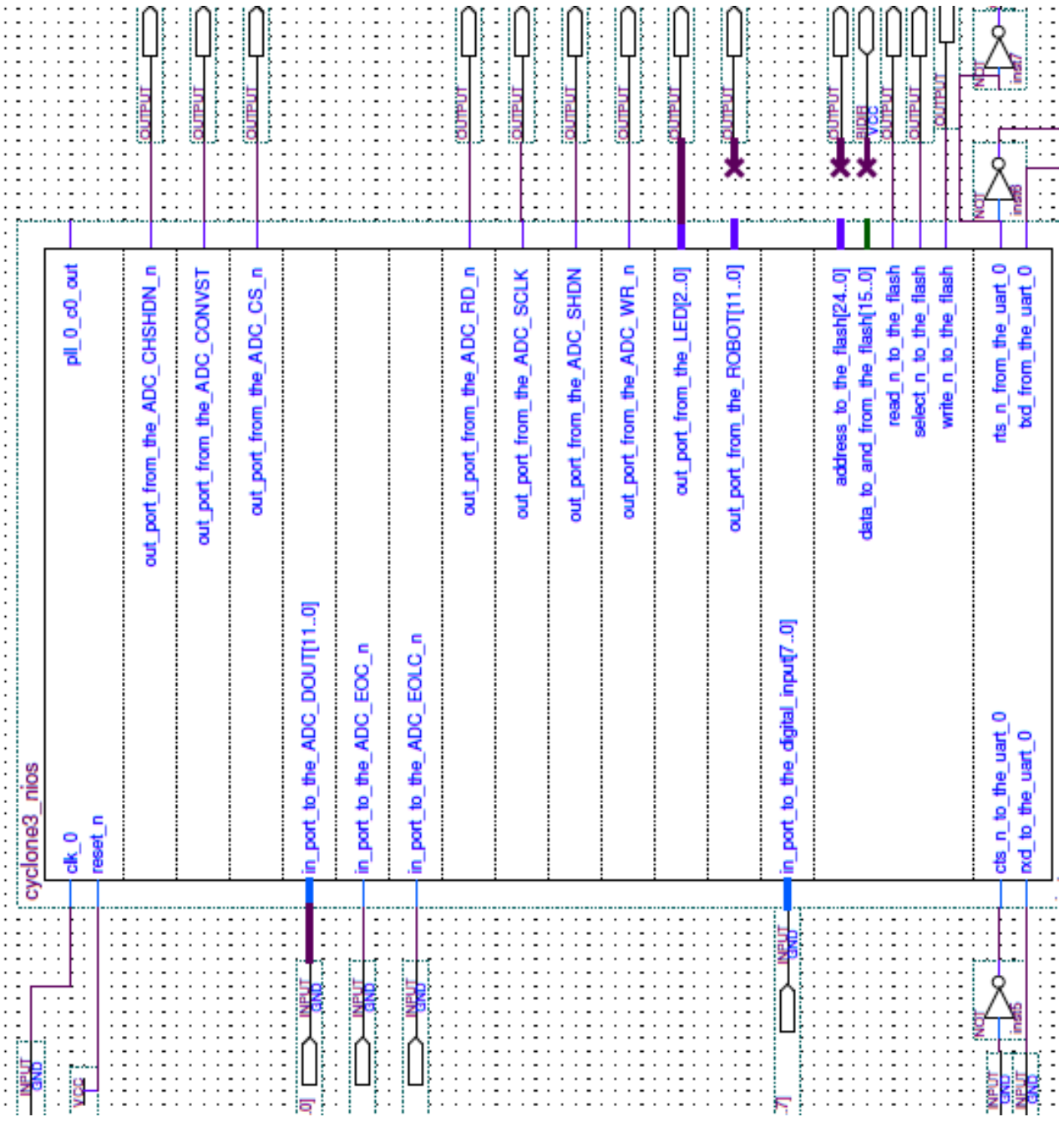

Figure 6.6: Black-box view for the soft-core processor at high level.

6.2 Testing the Spike-Sorting Software Module

We tested the spike-sorting module with raw data sets collected from dragonflies in the Higgins laboratory. The settings of the preparations for these raw data sets are described as follows. The dragonfly's eyes were upside down (Figure 6.7) to make recordings from the ventral nerve cord more conveniently. The stimulus screen had a size of 1440×900 *pixels* and was put next to the dragonfly. An example of the stimulus's starting point location (co-ordinates (x, y)) and its moving direction is shown in Figure 6.8. These recordings were implemented with a single hook electrode around the ventral nerve cord. The recorded signal was then stored in a digital format at a sampling rate of 50 k Hz.

We first present the template set that was built with the above data sets. We then show results of sorting out spikes with arbitrary stimuli.

Figure 6.7: The dragonfly's eyes are upside down in experiments of the Higgins laboratory.

6.2.1 Templates for TSDNs

According to the neural information interpretation discussed in Section 5.1, we built templates for TSDNs with data sets that have corresponding visual stimulus arrangements as described in Table 6.1.

We implemented the non-real-time data flow (Section 5.2) to build templates. Referring to Section 5.2.3 for how to collect information for templates from given data sets, we gathered spike-pattern information for all six templates (Figures 6.9

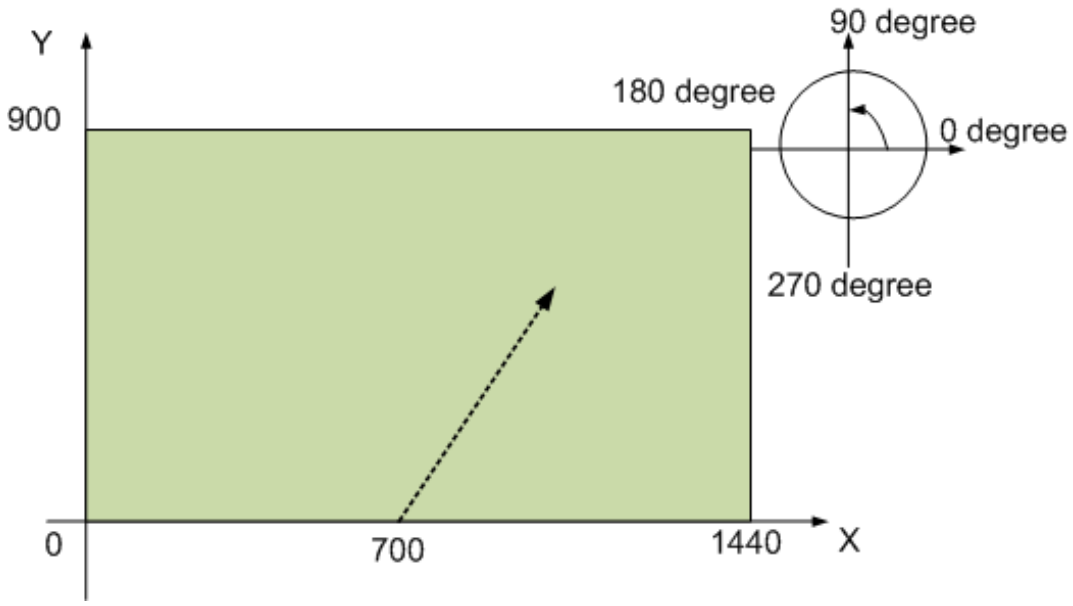

Figure 6.8: Example of a small moving target visual stimulus starting at co-ordinates $(x, y) = (700, 0)$ and moving at an angle of 70^o.

Table 6.1: Selected data sets to build templates.

Corresponding templates	Data set name	Stimulus Orientation	Starting location (pixels)	Data segment (index)
DIT3	S102	Posterior	1440×200	20000-50000
DIT1	S102	Posterior	1440×200	50000-80000
MDT2	S410	Anterior	1300×200	30000-90000
MDT1	S12705	Dorsal	902×900	33000-100000
MDT3	S12714	Dorsal	263×900	33000-100000
MDT4	S12307	Ventral	1000×0	30000-80000

Table 6.2: List of figures showing information for each template from spike patterns.

Figure number	Template information
6.9	Information to build the MDT3 template.
6.10	Information to build the MDT1 template
6.11	Information to build the MDT4 template
6.12	Information to build the MDT2 template
6.13	Information to build DIT3 template
6.14	Information to build DIT1 template

to 6.14, listed in Table 6.2).

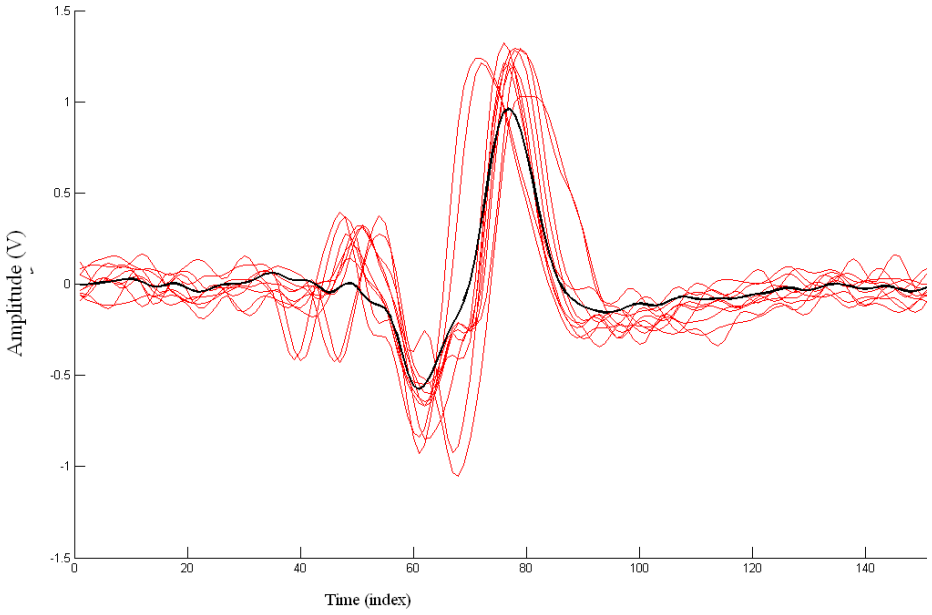

Figure 6.9: The MDT3 template and the waveforms used to build it.

Finally, we had the whole template set as shown in Figure 6.15 and Figure 6.16. We found that DIT1 and DIT3 had a similar amplitude range but they were different in the first half of their shapes. All the MDT group had a smaller amplitude range (just about $\pm 1\ V$) than the DIT group had ($\pm 3.2\ V$).

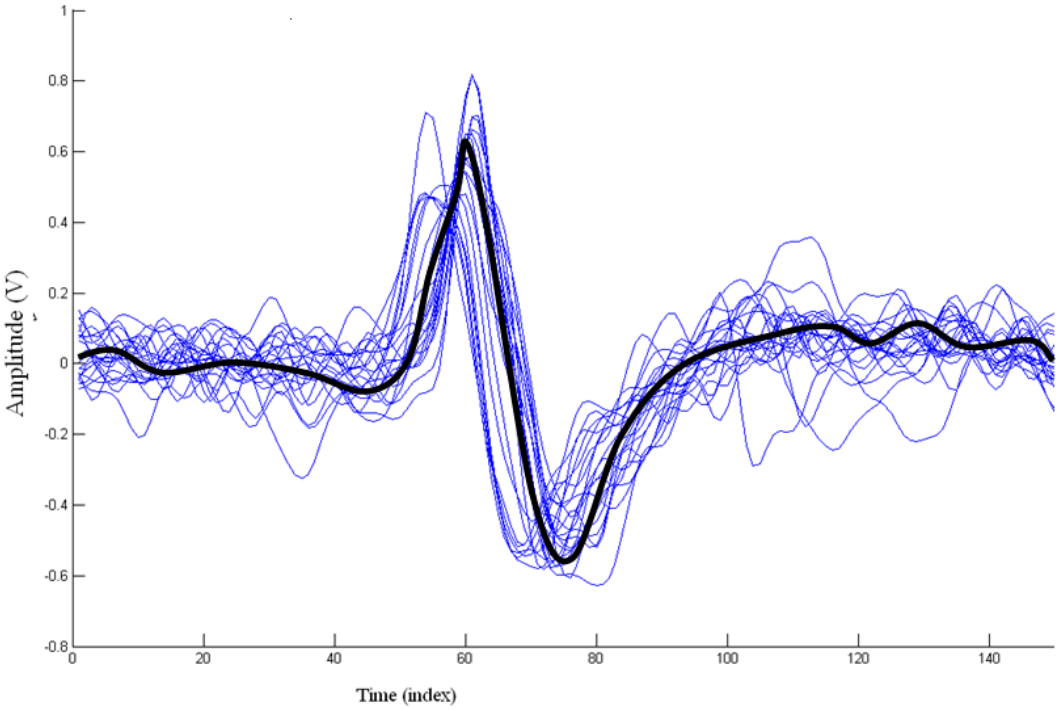

Figure 6.10: The MDT1 template and the waveforms used to build it.

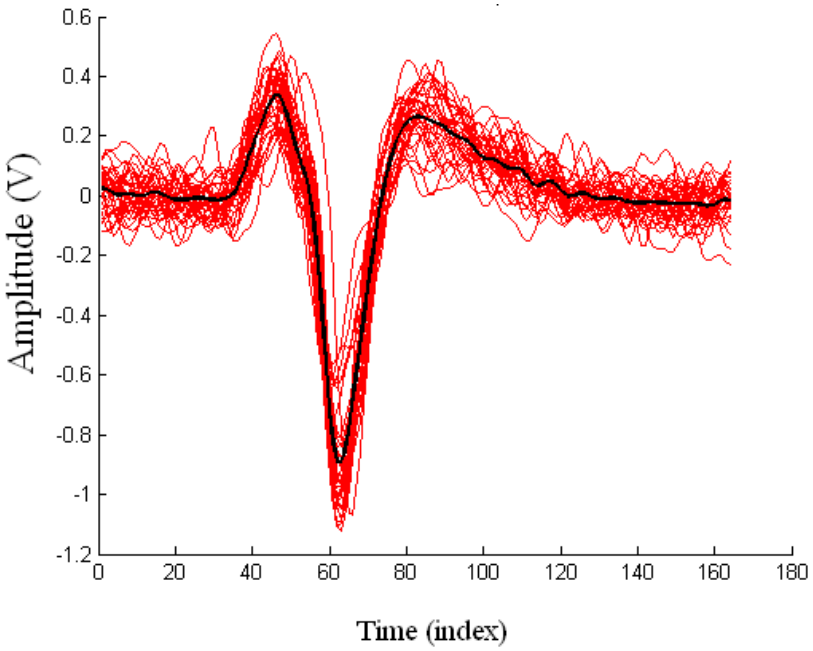

Figure 6.11: The MDT4 template and the waveforms used to build it.

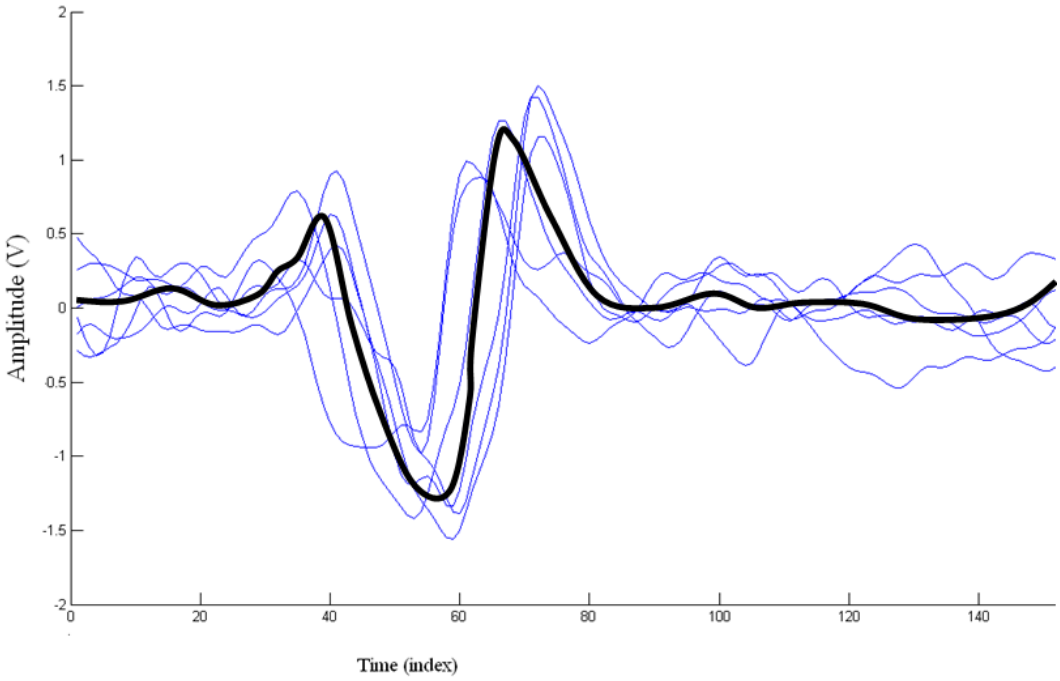

Figure 6.12: The MDT2 template and the waveforms used to build it.

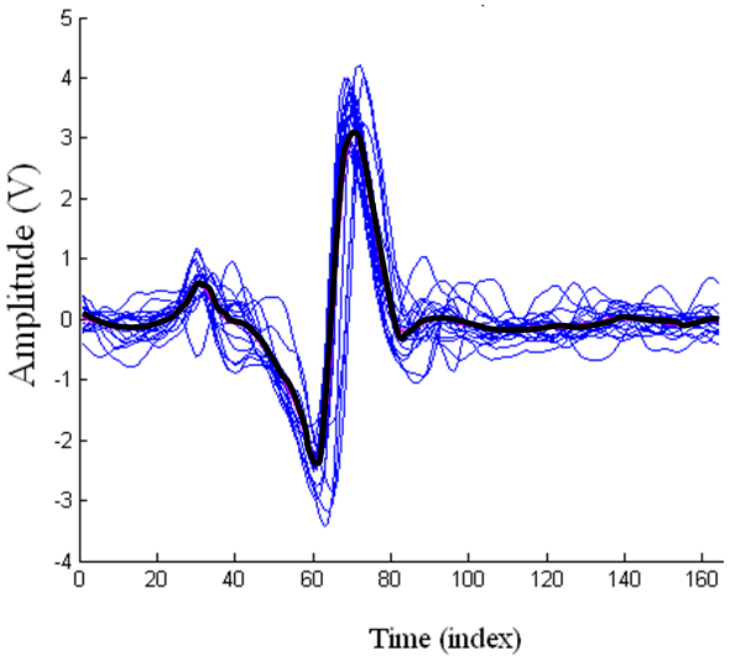

Figure 6.13: The DIT3 template and the waveforms used to build it.

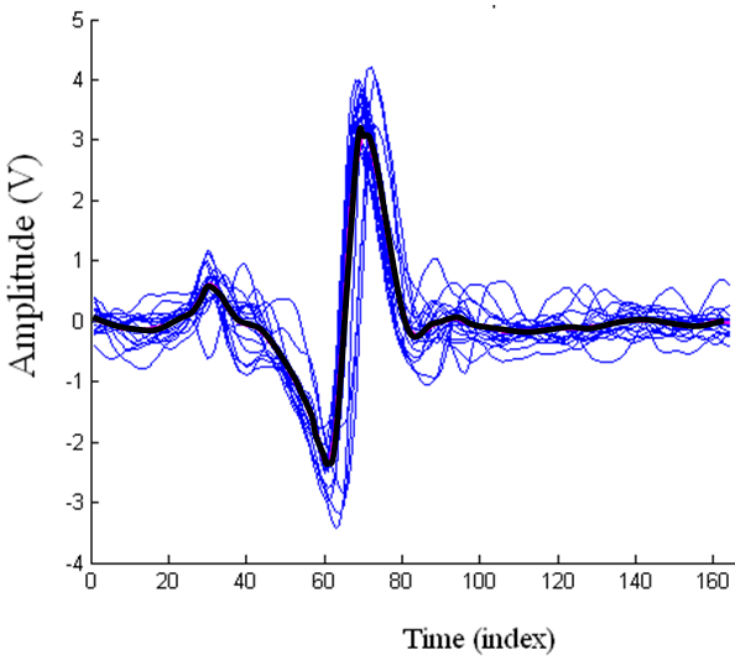

Figure 6.14: The DIT1 template and the waveforms used to build it.

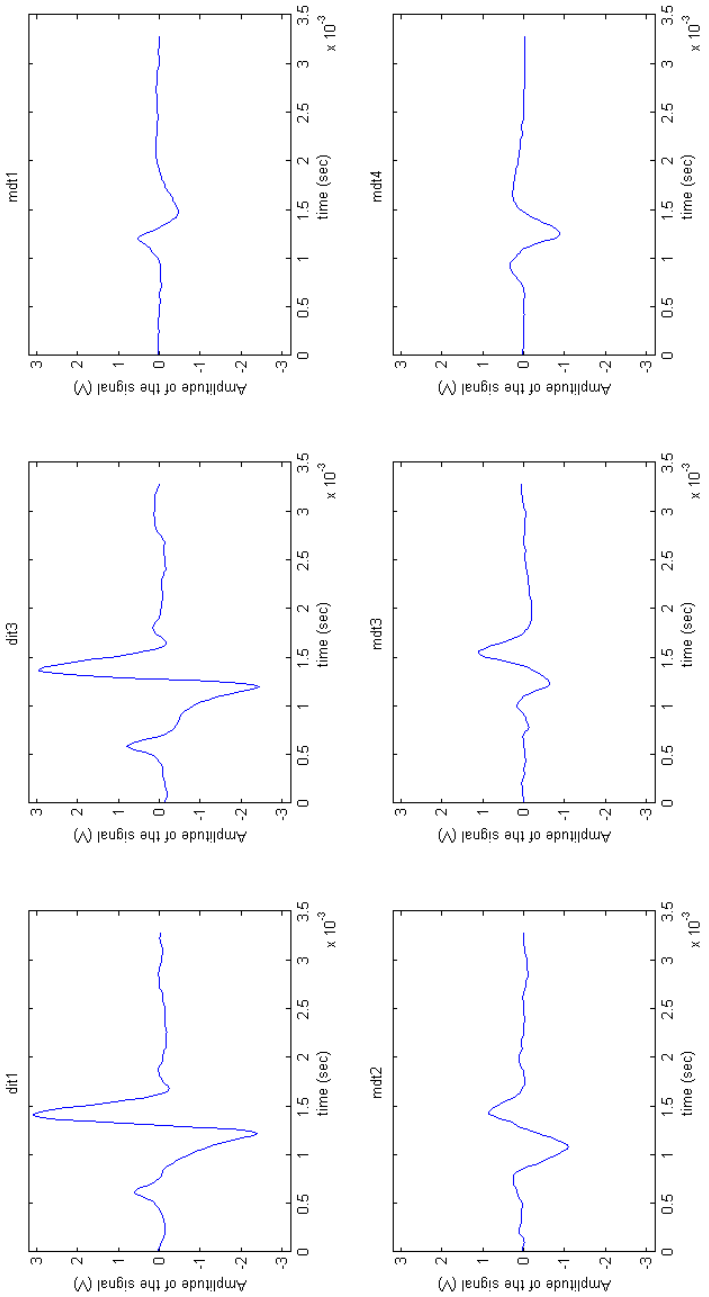

Figure 6.15: The whole set of templates previously shown in separate subplots to compare their amplitude ranges. Each subplot shows a spike length in time of 3.5 ms (horizontally) and an amplitude range of ± 3.5 V (vertically). The top left is DIT1. The top center is DIT3. The top right is MDT1. The bottom left is MDT2. The bottom center is MDT3. The bottom right is MDT4.

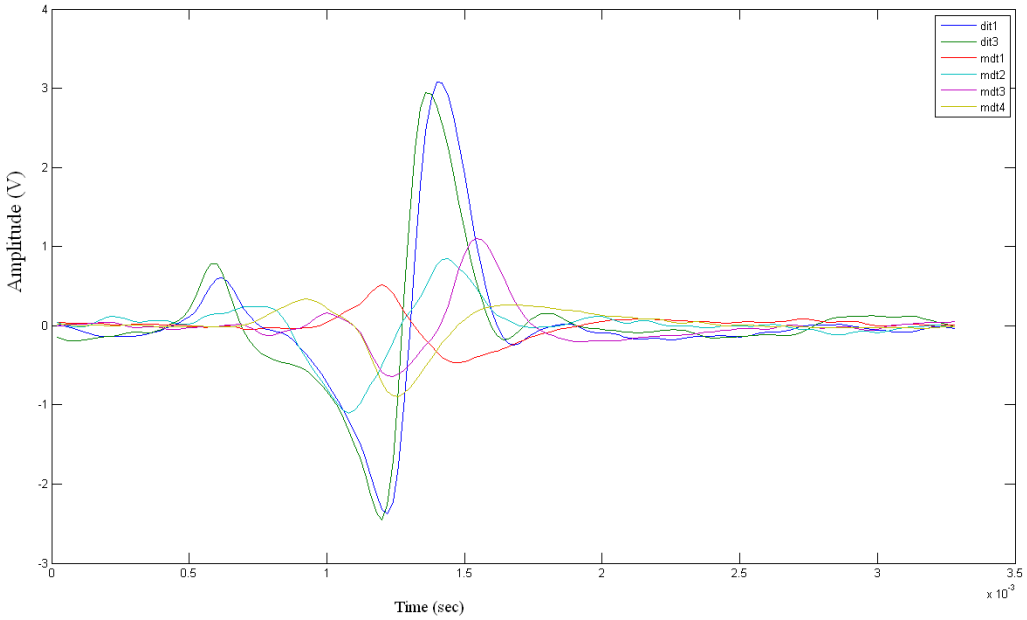

Figure 6.16: The whole set of templates showed in one plot to compare their shapes.

6.2.2 Spike Sorting with Arbitrary Stimuli

In this section, we applied the template-matching method (discussed in real-time data flow of the software application module (Section 5.2) that uses the above template set to sort out spikes from arbitrary stimuli. To validate the whole template set, we tested again with new data sets that had similar settings to the ones we used to build templates, but were recorded in different preparations. We tested several different settings of stimuli starting at 700×0, 902×900, 263×900, 1440×200, and 868×0 moving in directions of 90^o, 270^o, and 180^o. Results are shown in Figures 6.17 to 6.21.

When testing with the stimulus having orientation of 90^o, more MDT4 spikes appeared (with a high correlation level of 0.9, Figure 6.20) if the stimulus started at 868×0 co-ordinates than starting at the bottom center of the screen (e.g. 700×0, Figure 6.17). When the angle was 270^o, spikes from MDT1 dominated the plot if the stimulus started at the top center (e.g. 902×900, Figure 6.18) while more spikes

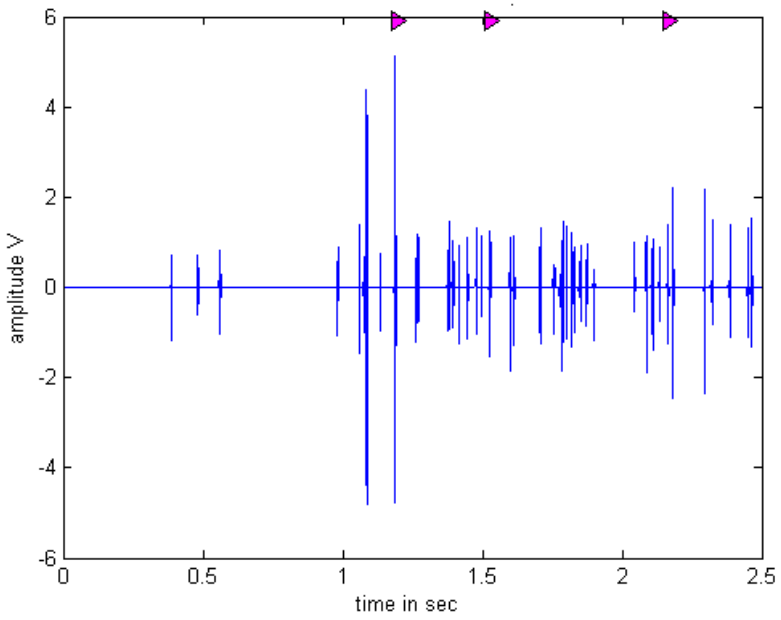

Figure 6.17: Testing the template set with data set S808, 90^o start at 700×0. Some MDT4 were indicated (with markers) with a correlation of 0.9. Markers $>$ are for detected MDT4.

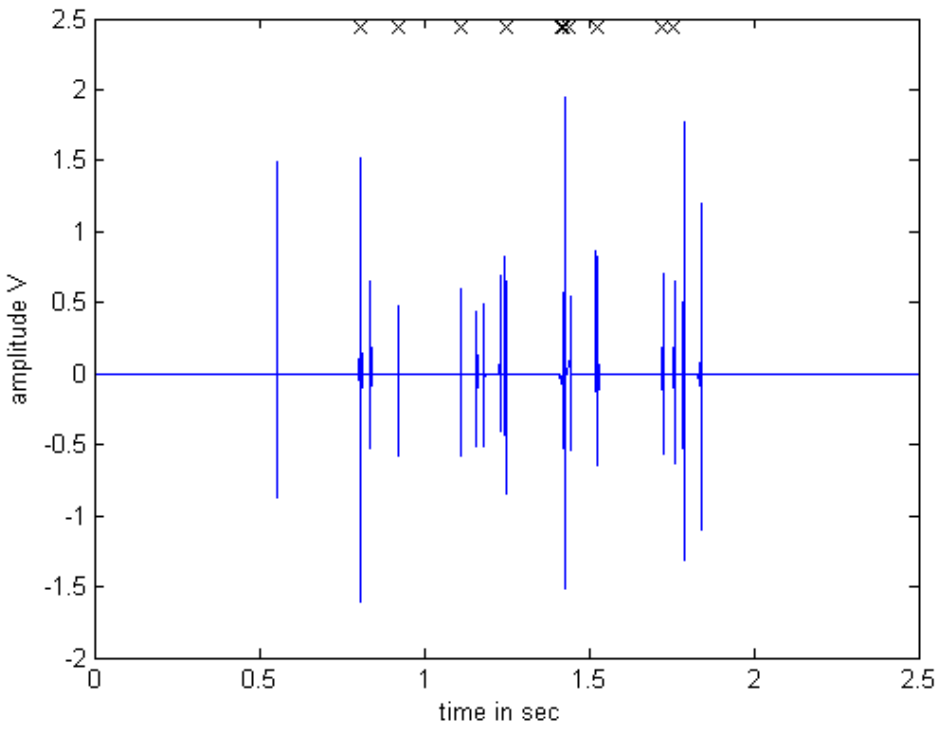

Figure 6.18: Testing the template set with data set S12705, 270^o start at 902×900. Several MDT1s were indicated (with markers) with a correlation of 0.9. Markers x are for detected MDT1.

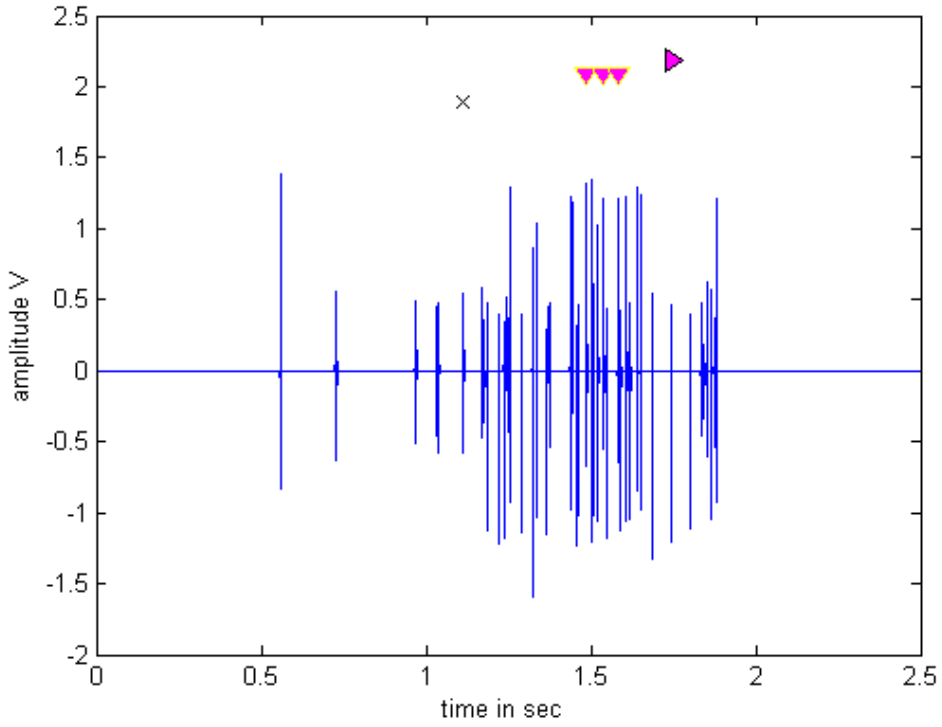

Figure 6.19: Testing the template set with data set S12714, 270^o start 263×900. MDT3 dominated among DIT1 and MDT4 (with markers) with a correlation of 0.9. Markers for detected neurons: x for MDT1, v for MDT3, and > for MDT4.

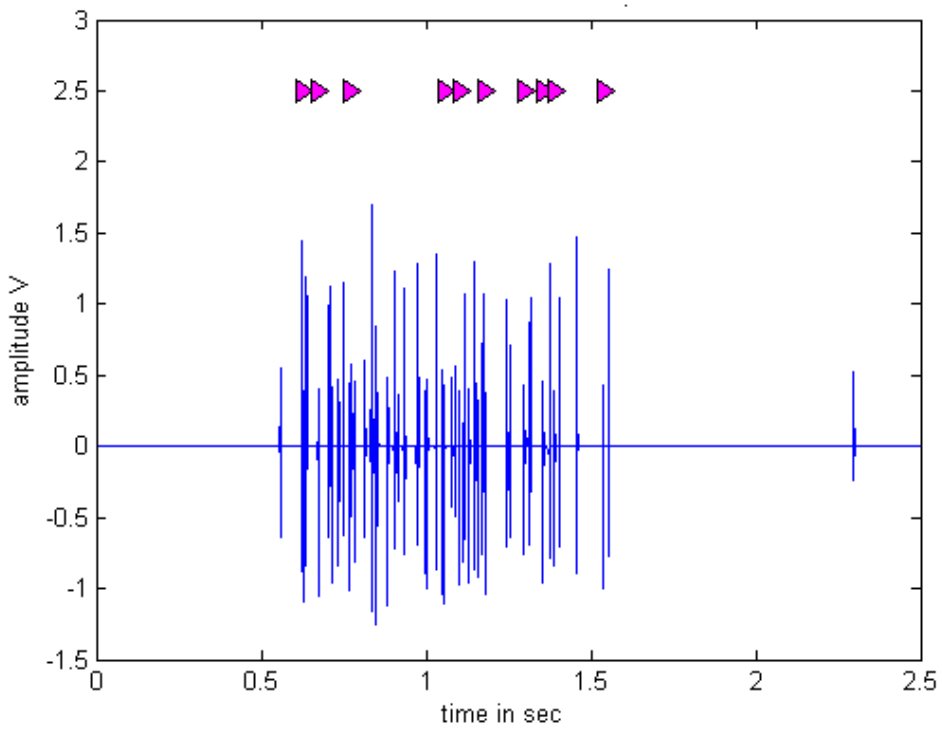

Figure 6.20: Testing the template set with data set S12519, 90^o start 868×0. When the angle is 90^o and closer to the right, more MDT4s appeared (with markers) with a correlation of 0.9. Markers $>$ are for detected MDT4.

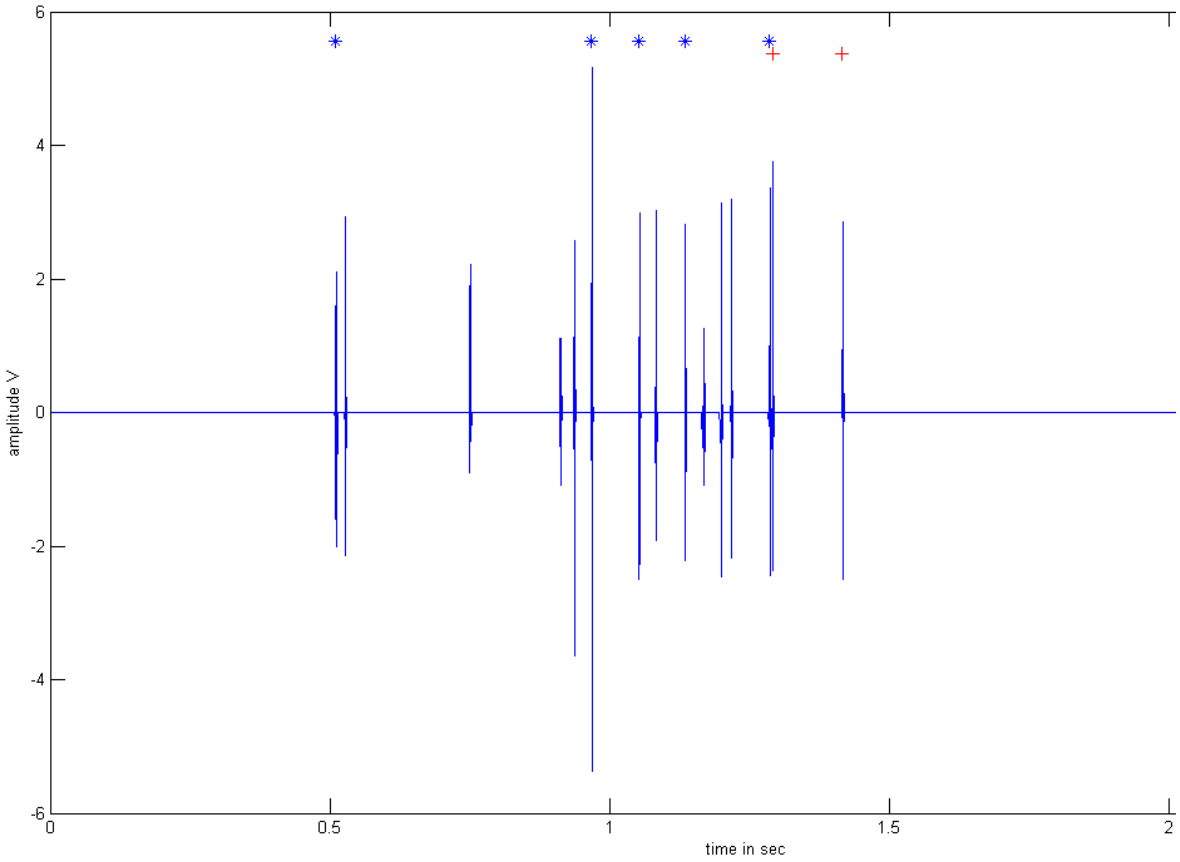

Figure 6.21: Testing the template set with data set S101, 180^o from 1440×200. More DIT3s were indicated than DIT1s (with markers) with a correlation of 0.9. Markers for detected neurons: + for DIT1 and * for DIT3.

from MDT3 along with few spikes from MDT4 and few spikes from MDT1 appeared in the case of starting at the top left (e.g. 263×900, Figure 6.19). When the angle was 180^o and starting from the bottom at far right (e.g. 1440×200), more spikes from DIT3 appeared than from DIT1 (Figure 6.21). These observations strongly agree with the visual receptive field properties of dragonflies presented by Frye et al. [9]. This fact validates the template set built in the above section for the sorting purpose.

We sorted out spikes with three different arbitrary settings of the stimulus starting at 700×0 and 600×0 moving in directions of 48^o, 55^o, and 70^o. Results are shown in Figures 6.22 to 6.24. When the angle was small but not orthogonal to the axes of the screen (e.g. 48^o and 55^o), only DIT3s were indicated (Figure 6.23, 6.22). If the angle was also not orthogonal to the axes of the screen but closer to 90^o (e.g. 70^o), DIT1s appeared but still less than DIT3 (Figure 6.24). All of these observed results also agree with the investigation of visual receptive field properties by Frye et al. [9].

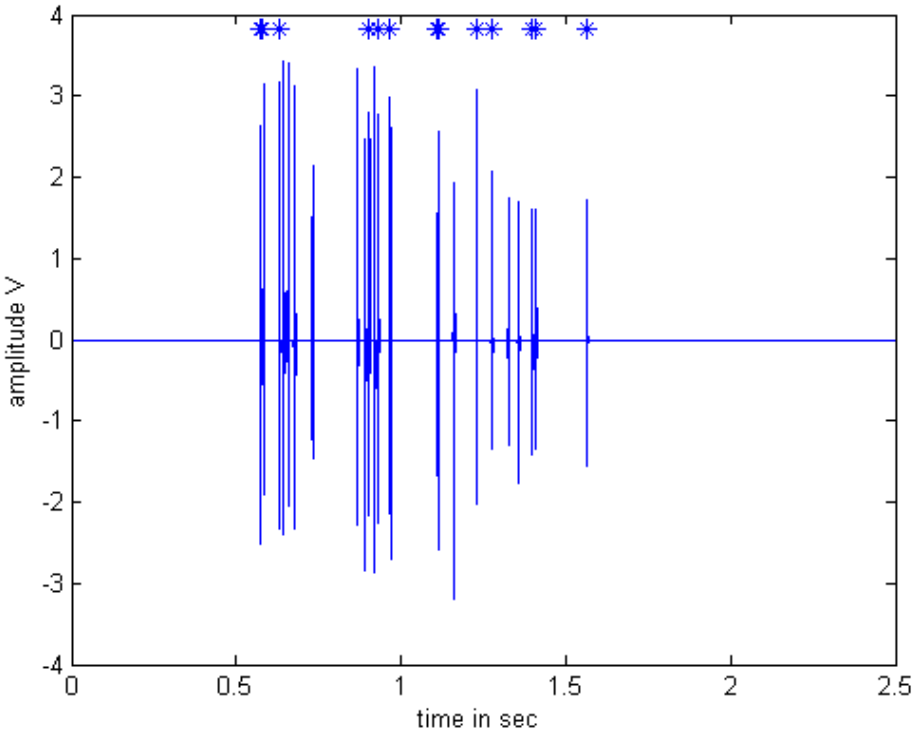

Figure 6.22: Testing the template set with data set S201, 48^o from 700×0. When the angle was small but not orthogonal, only DIT3s were indicated (with markers) with a correlation of 0.9. Markers * are for detected DIT3.

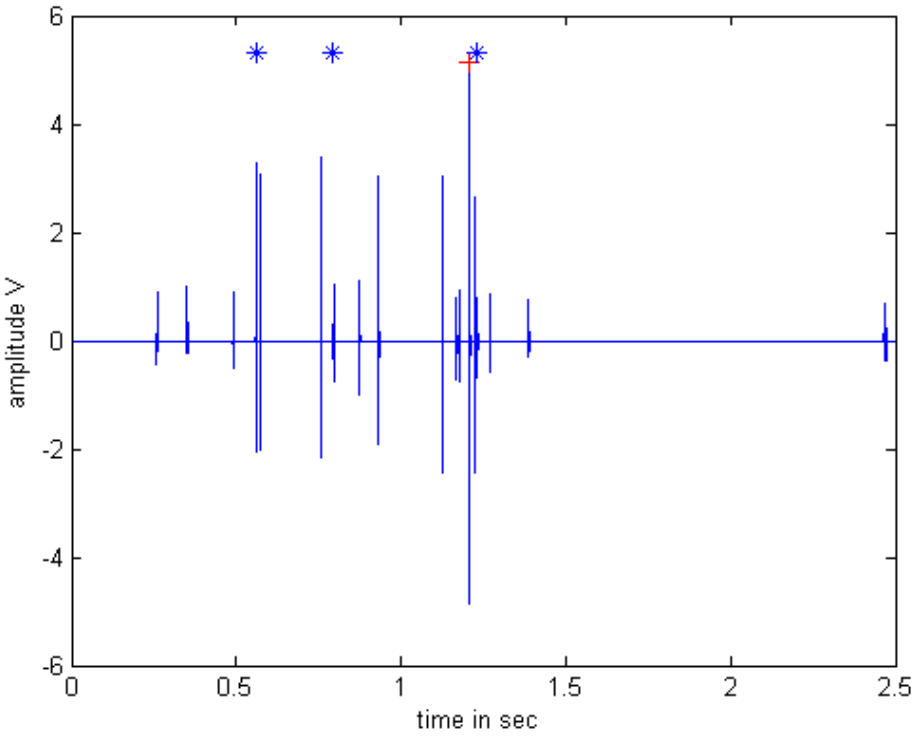

Figure 6.23: Testing the template set with data set S316, 55^o from 700×0. When the angle was not orthogonal but closer to 90^o, more DIT3s than DIT1 were indicated (with markers) with a correlation of 0.9. Markers for detected neurons: + for DIT1 and * for DIT3.

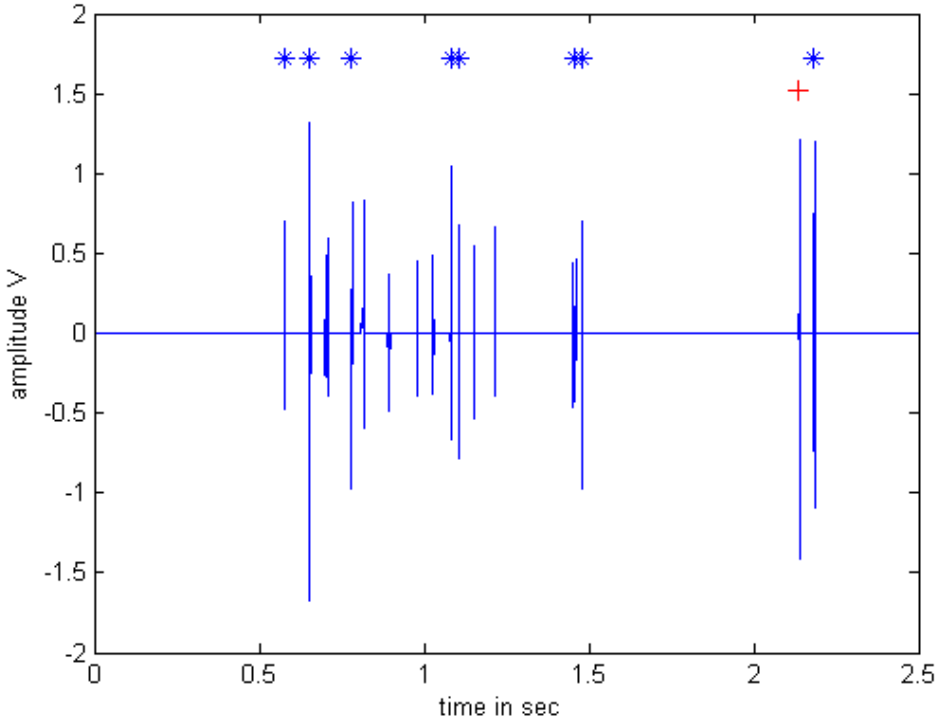

Figure 6.24: Testing the template set with data set S12202, 70^o from 600×0. When the angle was not orthogonal but closer to 90^o, more DIT3s than DIT1 were indicated (with markers) with a correlation of 0.9. Markers for detected neurons: + for DIT1 and * for DIT3.

CHAPTER 7

SUMMARY AND FUTURE WORK

7.1 Summary

With an approach using hybrid bio-robotics, this project proposes a real-time neural signal processing system for dragonfly ventral nerve cord recordings. The visual receptive field properties of eight dragonfly TSDNs were used to construct the method for spike sorting in the proposed system and to verify the observed experimental results.

The proposed architecture consists of different modules, including analog and digital PCBs, and a software application. The analog PCB has five functional blocks: a preamplifier, a HPF, a LPF, a notch filter, and a spike detector. The digital PCB is comprised of an analog-to-digital converter, a data processor with an FPGA-based microprocessor, and a data transmitter (with wired and wireless methods). The application module contains real-time and non-real-time procedures to build templates and to create a histogram of spike sorting.

We validated the fabricated PCBs and tested the spike-sorting software application with real neural data sets. From observed results, we found that the DIT1 template and the DIT3 template had a similar amplitude range. But DIT1 was different from DIT3 in the first half of the shapes. The MDT group had a smaller amplitude range than the DIT group had. In the final testing session, the whole template set was matched with spikes detected from arbitrary stimuli. Results confirmed that the template set agrees with the visual receptive field properties and can be used for spike sorting. In summary, the proposed system provides a module to process visual information from dragonfly TSDNs in real time for dragonfly hybrid bio-robots.

7.2 Future Work

Several projects can be developed from this study. There are some limitations found in the fabricated PCBs when we test the proposed system design. Firstly, though the size of the current PCBs are small enough for the robot platform in Higgins laboratory, the PCB should be smaller in order to fit a newer robot platform design in the future. We can reduce the size of the analog PCB by choosing new models of digital switches and potentiometers or finding a new method to vary the capacitance and resistance components in the circuit. Currently, these components cause a complicated I^2C communication network and a large space consumption. Additionally, we can further investigate the real-time operation of the system. We have not tested the digital PCB with the spike-sorting application in real time due to the disability of loading applications to the memory in the current fabricated digital PCB. This problem might come from some damage in some internal parts of the components when we tried to fabricate and populate the boards. We should have them fabricated by a professional service provider. These problems can be solved in future projects of the Higgins laboratory.

Though this system is aimed to propose a real-time visual motion detection module for dragonfly hybrid bio-robots, the design can be modified for another kind of animal. Hence, the system may help better understand the representations and computational architectures used by different biological systems in neuroscience. Additionally, different spike-sorting algorithms can also be applied to the proposed architecture. Consequently, the system can be used to evaluate recent spike-sorting algorithms with real neural signals instead of simulated ones.

REFERENCES

[1] K. Jeong, J. Kim, and L. P. Lee, "Biologically inspired artificial compound eyes," *Science, Materials and Biology*, vol. 310, pp. 557–560, 2005.

[2] L. P. Lee and R. Szema, "Inspirations from biological optics for advanced photonic systems," *Science, Materials and Biology*, vol. 310, pp. 1148–1150, 2005.

[3] C. M. Higgins, "Living insects become 'eyes' for robots," *Arizona Daily Star (special supplement), School of mind, brain, and behavior, University of Arizona*, 2010.

[4] T. Melano, "Insect-machine interfacing," *PhD thesis (Advisor: Charles M. Higgins), Biomedical Engineering, The University of Arizona*, January 2011.

[5] L. I. Ortiz, "A mobile electrophysiology board for autonomous biorobotics," *MS thesis (Advisor Charles M. Higgins), Department of Electrical and Computer Engineering, The University of Arizona*, December 2006.

[6] J. Routh, J. Methvin, J. Radtke, J. Alstine, and B. Richardson, "Biological signal interface board." Interdisciplinary Engineering Design Program Report, The University of Arizona, 2009.

[7] J. Kien and J. S. Altman, "Descending interneurones from the brain and suboesophageal ganglia and their role in the control of locust behaviour," *Insect Physiology*, vol. 30, pp. 59 – 72, 1984.

[8] R. M. Olberg, "Identified target-selective visual interneurons descending from the dragonfly brain," *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, vol. 159, pp. 827–840, 1986.

[9] M. A. Frye and R. M. Olberg, "Visual receptive field properties of feature detecting neurons in the dragonfly," *Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, vol. 177, pp. 569–576, 1995.

[10] S. Gozani and J. Miller, "Optimal discrimination and classification of neuronal action potential waveforms from multiunit, multichannel recordings using software-based linear filters," *IEEE Transactions on Biomedical Engineering*, vol. 41, pp. 358 –372, 1994.

[11] K. S. Guillory and R. A. Normann, "A 100-channel system for real time detection and storage of extracellular spike waveforms," *Journal of Neuroscience Methods*, vol. 91, no. 1-2, pp. 21 – 29, 1999.

[12] R. Chandra and L. Optican, "Detection, classification, and superposition resolution of action potentials in multiunit single-channel recordings by an online real-time neural network," *IEEE Transactions on Biomedical Engineering*, vol. 44, pp. 403 –412, 1997.

[13] I. Obeid and P. Wolf, "Evaluation of spike-detection algorithms for a brain-machine interface application," *IEEE Transactions on Biomedical Engineering,*, pp. 905 –911, 2004.

[14] J. Kaiser, "On a simple algorithm to calculate the 'energy' of a signal," in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, vol. 1, pp. 381 –384, 1990.

[15] I. Bankman, K. Johnson, and W. Schneider, "Optimal detection, classification, and superposition resolution in neural waveform recordings," *IEEE Transactions on Biomedical Engineering*, vol. 40, pp. 836 –841, 1993.

[16] X. Yang and S. Shamma, "A totally automated system for the detection and classification of neural spikes," *IEEE Transactions on Biomedical Engineering*, vol. 35, pp. 806 –816, 1988.

[17] E. M. Glaser and W. B. Marks, "Separation of neuronal activity by waveform analysis," *Advances in Biomedical Engineering*, vol. 5, pp. 137–156, 1968.

[18] E. M. Glaser, "On-line separation of interleaved neuronal pulse sequences," *Data Acquisition Process. Biol. Med.*, vol. 1, pp. 77–136, 1971.

[19] M. Abeles and M. H. Goldstein, "Multispike train analysis," *Proceedings of the IEEE*, vol. 65, pp. 762 – 773, 1977.

[20] J. C. Letelier and P. P. Weber, "Spike sorting based on discrete wavelet transform coefficients," *Journal of Neuroscience Methods*, vol. 101, no. 2, pp. 93 – 106, 2000.

[21] E. Hulata, R. Segev, and E. Ben-Jacob, "A method for spike sorting and detection based on wavelet packets and Shannon's mutual information," *Journal of Neuroscience Methods*, vol. 117, no. 1, pp. 1 – 12, 2002.

[22] R. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Comput.*, vol. 16, pp. 1661–1687, 2004.

[23] S. Takahashi, Y. Anzai, and Y. Sakurai, "A new approach to spike sorting for multi-neuronal activities recorded with a tetrode–how ICA can be practical," *Neuroscience Research*, vol. 46, no. 3, pp. 265 – 272, 2003.

[24] A. M. Mamlouk, H. Sharp, K. M. Menne, U. G. Hofmann, and T. Martinetz, "Unsupervised spike sorting with ICA and its evaluation using GENESIS simulations," *Neurocomputing*, vol. 65-66, pp. 275 – 282, 2005. Computational Neuroscience: Trends in Research 2005.

[25] A. Snellings, D. J. Anderson, and J. W. Aldridge, "Improved signal and reduced noise in neural recordings from close-spaced electrode arrays using independent component analysis as a preprocessor," *Journal of Neuroscience Methods*, vol. 150, no. 2, pp. 254 – 264, 2006.

[26] V. Karkare, S. Gibson, and D. Markovic, "A 130 μW, 64-channel spike-sorting DSP chip," in *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, pp. 289 –292, 2009.

[27] Z. Yang, Q. Zhao, and W. Liu, "Improving spike separation using waveform derivatives," *Journal of Neural Engineering*, vol. 6, p. 046006 (12pp), 2009.

[28] A. Zviagintsev, Y. Perelman, and R. Ginosar, "Low-power architectures for spike sorting," in *Neural Engineering, 2005. Conference Proceedings. 2nd International IEEE EMBS Conference on*, pp. 162 –165, 2005.

[29] M. Delescluse and C. Pouzat, "Efficient spike-sorting of multi-state neurons using inter-spike intervals information," *Journal of Neuroscience Methods*, vol. 150, no. 1, pp. 16 – 29, 2006.

[30] E. Chah, V. Hok, A. Della-Chiesa, J. H. Miller, S. M. O'Mara, and R. B. Reilly, "Automated spike sorting algorithm based on Laplacian eigenmaps and k -means clustering," *Journal of Neural Engineering*, vol. 8, p. 016006 (9 pp), 2011.

[31] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman, "Autoclass: A Bayesian classification system," *Proc of the Fifth Intl Workshop on Machine Learning*, pp. 54–64, 1988.

[32] J. J. Capowski, "The spike program: a computer system for analysis of neurophysiological action potentials," *Computer Technology in Neuroscience ed P P Brown (Washington DC: Hemisphere)*, pp. 237–288, 1976.

[33] J. S. Oghalai, W. Street, and W. S. Rhode, "A neural network-based spike discriminator," *Journal of Neuroscience Methods*, vol. 54, no. 1, pp. 9 – 22, 1994.

[34] M. Blatt, S. Wiseman, and E. Domany, "Superparamagnetic clustering of data," *Phys. Rev. Lett.*, vol. 76, pp. 3251–3254, 1996.

[35] C. Vargas-Irwin and J. P. Donoghue, "Automated spike sorting using density grid contour clustering and subtractive waveform decomposition," *Journal of Neuroscience Methods*, vol. 164, no. 1, pp. 1 – 18, 2007.

[36] S. Gibson, J. W. Judy, and D. Markovic, "Comparison of spike-sorting algorithms for future hardware implementation," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pp. 5015 –5020, 2008.

[37] M. Scanziani and M. Hajsser, "Electrophysiology in the age of light," *Nature*, vol. 461, pp. 930–939, 2009.

[38] J. S. Buchwald, S. B. Holstein, and D. S. Weber, "Multiple unit recording: Technique, interpretation, and experimental applications," *Bioelectric Recording Techniques, Vol. I of Methods in Physiological Psychology, chapter 1*, pp. 201–238, 1973.

[39] M. A. Dichter, "Intracellular single unit recording," *Bioelectric Recording Techniques, Vol. I of Methods in Physiological Psychology, chapter 1*, pp. 3–21, 1973.

[40] R. Purves, "Microelectrode methods for intracellular recording and ionophoresis," *Academic Press, Inc., London*, 1981.

[41] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Computation in Neural Systems*, vol. 9, no. 4, pp. R53–R78, 1998.

[42] S. Gibson, R. Chandler, V. Karkare, D. Markovic, and J. Judy, "An efficiency comparison of analog and digital spike detection," in *Neural Engineering, 2009. NER '09. 4th International IEEE/EMBS Conference on*, pp. 423 –428, 2009.

[43] M. B. Priestley, *Spectral analysis and time series*. London: Academic Press, 1981.

[44] S. Mukhopadhyay and G. Ray, "A new interpretation of nonlinear energy operator and its efficiency in spike detection," *IEEE Transactions on Biomedical Engineering*, vol. 45, pp. 180 –187, 1998.

[45] K. H. Kim and S. J. Kim, "Neural spike sorting under nearly 0-dB signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier," *IEEE Transactions on Biomedical Engineering*, vol. 47, pp. 1406 –1411, 2000.

[46] M. Sarna, P. Gochin, J. Kaltenbach, M. Salganicoff, and G. Gerstein, "Unsupervised waveform classification for multi-neuron recordings: a real-time,

software-based system. II. performance comparison to other sorters," *Journal of Neuroscience Methods*, vol. 25, no. 3, pp. 189 – 196, 1988.

[47] M. Salganicoff, M. Sarna, L. Sax, and G. Gerstein, "Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. I. algorithms and implementation," *Journal of Neuroscience Methods*, vol. 25, no. 3, pp. 181 – 187, 1988.

[48] R. B. Potts, "Some generalized order-disorder transformations," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 48, pp. 106–109, 1952.

[49] M. Blatt, S. Wiseman, and E. Domany, "Data clustering using a model granular magnet," *Neural Comput.*, vol. 9, pp. 1805–1842, 1997.

[50] U. Wolff, "Comparison between cluster Monte Carlo algorithms in the Ising model," *Physics Letters B*, vol. 228, pp. 379 – 382, 1989.

[51] M. D. Snodderly, "Extracellular single unit recording," *Bioelectric Recording Techniques, Vol.I of Methods in Physiological Psychology, chapter 6*, pp. 137–162, 1973.

[52] Texas Instrument, http://www.ti.com, *INA datasheet*, 2010.

[53] R. Sallen, "A practical method of designing RC active filters," *IRE Transactions on Circuit Theory 2*, pp. 74–85, 1955.

[54] I. Obeid, J. Morizio, K. Moxon, M. Nicolelis, and P. Wolf, "Two multichannel integrated circuits for neural recording and signal processing," *IEEE Transactions on Biomedical Engineering*, vol. 50, pp. 255 –258, 2003.

[55] National Semiconductors, http://www.national.com, *LM13700 datasheet*, 2010.

[56] C. Rogers and J. Harris, "A low-power analog spike detector for extracellular neural recordings," in *Electronics, Circuits and Systems, 2004. ICECS 2004. Proceedings of the 2004 11th IEEE International Conference on*, pp. 290 – 293, 2004.

[57] O. H. Schmitt, "A thermionic trigger," *Journal of Scientific Instruments*, vol. 15, pp. 24–26, 1938.

[58] D. Paret, *The I2C Bus : From Theory to Practice*. Wiley (Chichester and New York), 1997.

[59] Information on NXP's (Philips') I2C bus, http://www.i2c-bus.org/twi-bus.

[60] Intersil Corporation, www.intersil.com, *X9258 datasheet*, 2010.

[61] Analog Devices Inc, www.analog.com, *ADG2188 datasheet*, 2010.

[62] D. J. Banks, W. Balachandran, P. R. Richards, and D. Ewins, "Instrumentation to evaluate neural signal recording properties of micromachined microelectrodes inserted in invertebrate nerve," *Physiological Measurement*, vol. 23, pp. 437–448, 2002.

[63] Maxim Integrated Products, www.maxim-ic.com, *MAX1308 datasheet*, 2010.

[64] Altera Corporation, www.altera.com, *Cyclone III Datasheet*, 2010.