A HARDWARE-EFFICIENT VLSI NEURAL SIGNAL PROCESSOR FOR IMPLANTABLE HIGH-CHANNEL-COUNT BRAIN MACHINE INTERFACES

By

Yuning Yang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering ─ Doctor of Philosophy

2016

ABSTRACT

A HARDWARE-EFFICIENT VLSI NEURAL SIGNAL PROCESSOR FOR IMPLANTABLE HIGH-CHANNEL-COUNT BRAIN MACHINE INTERFACES

By

Yuning Yang

Many brain machine interfaces (BMIs) aim to assist paralyzed subjects to control real-time man-made devices by translating human neural activities into machine commands. Neural activities can be recorded through implantable microelectrode arrays (MEAs) that provide the highest spatial and temporal resolutions compared to other recording techniques. In order to provide neural control of advanced prosthetic limbs with many degrees of freedom, next-generation BMIs will demand simultaneous recording of thousands of neurons from high-channel-count MEAs., Furthermore, next-generation BMIs must be fully implantable wireless neural microsystems to eliminate infection risks and reduce the mechanical vulnerabilities. Such a system would generate and transmit a vast amount of neural data within an environment where power and heat dissipation are tightly constrained. To prevent tissue damage, the neural microsystem must incorporate a neural signal processor (NSP) to reduce neural data streams by preserving only sequences of spikes fired by each active neuron and discarding noise when neurons are inactive. However, this data reduction method is challenged by the fact that each microelectrode can observe activities of multiple neurons which must be individually processed to accurately translate neuron activities into machine commands. Thus, the NSP should be able to map each recorded spike to its source neuron. The goal of this research is to develop an implantable hardware-efficient NSP that is capable of preserving and identifying useful spike information from raw neural signals. In this work, three successive processing steps were developed for reducing neural data rate: A new spike detection method was developed that can

automatically and adaptively observe as many true spikes as possible from noisy neural signals. Automatic spike detection eliminates the need for manually parameter setting and enables real-time high-channel-count neural recording. To identify the source neuron for each detected spike without compromising the power or area budget of the NSP, a new feature set was created to reserve spike information. Based on an analysis of the neural signal energy spectrum, the new feature set enables accurate neural recording with high tolerance to noise variance. Furthermore, a new method was designed to classify spikes, providing high performance while reducing hardware resources for a 50% area reduction. Finally, these design concepts were integrated into a compact energy-efficient hardware NSP platform. The new NSP platform is scalable to high-channel-count and preserves useful information over a wide range of signal to noise ratio while achieving 15% higher accuracy on average than the current existing NSP with only consuming 0.75 μW power and 0.023 mm^2 area per channel. The innovations of this research contribute to overcoming the challenges of developing next-generation fully implantable wireless neural microsystems.

ACKNOWLEDGEMENTS

Looking back at my seven-year life as a PhD student, I have been through good times and tough times. Exploring and investigating several research areas in the first four years led to my thesis work in the last three years. This work would not be possible without the support of professors, fellow graduate students and my family. At this moment, I would like to express my gratitude and appreciation to them all.

Prof. Andrew J. Mason, my advisor, has been a great help to me through the long journey of my PhD research. Not only has he provided me with insightful and valuable suggestions to improve the quality of my work, but he has spent a considerable amount of time on enhancing my writing and presentation skills. He was very patient and amenable when I detoured in my research path. I am thankful to have pursued my PhD degree under his guidance. I believe what I learned from him will give me lifelong benefit.

I would like to thank Prof. Subir Biswas, Prof. Rama Mukkamala and Prof. Juyang Weng for serving on my committee. They gave me support all the time and provided me with useful feedback for my research work.

I am very grateful to former and current colleagues in AMSaC lab: Dr. Yue Huang, Dr. Awais Kamboh, Dr. Xiaowen Liu, Dr. Xiaoyi Mu, Dr. Lin Li, Dr. Haitao Li, Sam Boling, Heyu Yin, Sina Parsnejad, Ehsan Ashoori and Sylmarie Montero. I would like to give special thanks to Haitao for the tough times we have been through when we got lost in PhD life but figured out a way together. I also appreciate Sam for his generous help on experimental setup to validate my research work.

My utmost thanks go to my family. My parents consistently support and encourage me in this long journey. My wife and her parents have given me courage and confidence when I was

depressed. This thesis dissertation is dedicated to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

1.1 Brain Machine Interface for Neurological Paralysis

More than five million Americans suffer from neurological paralysis and have difficulty speaking and/or moving their arms or legs [1]. Neurological paralysis is caused by damage to the central nervous system (CNS), often resulting from stroke or spinal cord injury (SCI). It has been reported that stroke leads to 550,000 hospitalizations each year [2], and more than 200,000 individuals in the United States alone live with SCI, mainly due to motor vehicle accidents[3].

Because the CNS cannot recover by itself, neurosurgeons have made considerable efforts to achieve biological restoration. Stem cell transplantation is considered a promising technology for stroke, SCI and other CNS damage. Despite encouraging preliminary clinical results, several key issues need to be resolved before such treatments may be clinically available for CNS disorders [4]. On the other hand, neurophysiology researchers in the early 1960s demonstrated that brain activities can be read and interpreted. Walter showed that brain signals recorded from the scalp can be used to control a slide projector in 1964 [5]. Evarts found a relationship between the force of arm movement and neuron activity in the motor cortex in 1964 [6]. These discoveries of the neurophysiology indicate that a direct functional interface can be built between a brain and artificial devices to bypass the spinal cord lesions and to help paralyzed patients interact with their surroundings. An interface that can read brain signals and convert them into control and communication signals has become popularly known as a brain machine interface (BMI).

Fig. 1.1 illustrates the concept of a BMI to assist paralyzed people. Brain signals are

recorded by a neural recording system. The recording system consists of an electrode array to collect neural signals from the brain and a signal conditioning circuitry to process neural signals. The neural signals are then transmitted wirelessly to a device where they are decoded and translated into commands. Commands can be used to control different devices and to realize thoughts from patients. The applications of BMIs include simple two-dimensional control, such as selecting letters on a screen to type messages for communication or directing an automated wheelchair for navigation. More promising applications involve controlling a robot to move an object from one location to another or controlling a prosthetic arm to behave as a natural arm in a three-dimensional space. The applications of BMIs can be also extended beyond the medical field in the future, such as remotely piloting an aircraft in military or enhancing the control abilities for astronauts in space flight.

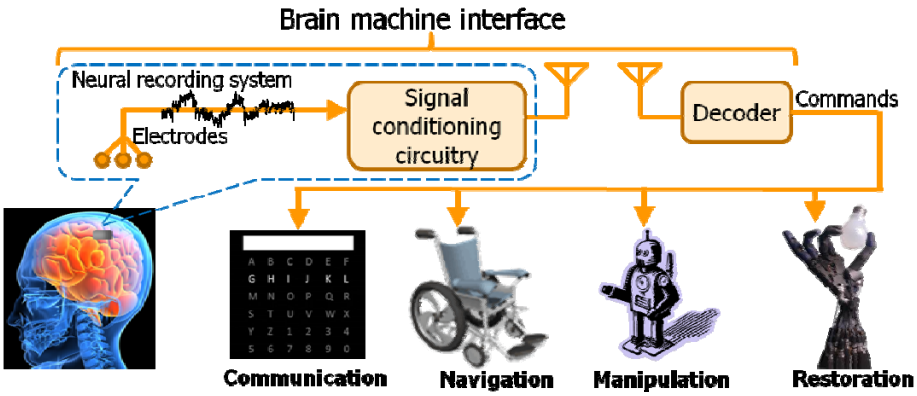


Figure 1.1. An illustration of different applications of BMI.

1.2 Future BMI requirements

As shown in Fig. 1.1, BMIs place electrodes near the brain to monitor neural activities. Electrodes can be placed either extracranially or intracranially. Extracranial electrodes are placed on the scalp and intracranial electrodes are placed on the cortical surface or inside the cortex. BMIs with electrodes placed on the scalp or on the cortical surface can be

used to control devices in a two dimensional space [7, 8]. Because brain activities originate within the underlying cortical tissue, there is a significant distance between recording electrodes placed on the scalp or the cortical surface and the underlying cortical tissue. Electrodes in these two cases record averaged brain activities over time and space. Thus, with electrodes placed on the scalp or the cortical surface, it is difficult to achieve sufficient temporal and spatial resolution to control advanced devices in a three dimensional space such as neural prostheses.

In order to control devices like neural prostheses, electrodes need to be place inside the cortex where brain activities within the underlying cortical tissue can be recorded [9, 10]. BMIs with electrodes placed inside the cortex are referred to as intracortical BMIs. Intracortical BMIs insert electrodes inside the cortex, with each electrode sized to the order of tenths of micrometers. These electrodes form an array structure referred to as microelectrode array (MEA). Each electrode of the MEA collects one channel of neural signal from brain activities. Recent research has shown that intracortial BMIs in both non-human primates and human patients can effectively control prostheses: [11] demonstrated that people with tetraplegia implanted with a 96-channel MEA can manipulate a robotic arm to take reach-and-grasp actions and to drink coffee from a bottle; [12] further showed that people with tetraplegia implanted with two 96-channel MEAs can control a prosthetic limb with seven degrees of freedom (DOFs). Current intracortical BMIs mount a connector on a patient's head. Cables are connected percutaneously from the connector to a commercial neural recording system.

To make prostheses feel like natural human limbs, future BMIs demand higher DOFs. To achieve high DOFs, recent studies have shown that intracortical BMIs with a large

number of recording channels (more than one thousand) are needed [13]. Furthermore, future BMIs need to fully implant the neural recording system with a small area on the brain to minimize the risk of infection. Thus, the neural recording system must wirelessly transmits data outside to an external decoder as shown in Fig. 1.1. Implantation avoids heavy cables connecting high channel count electrode channels to a neural recording system. Wireless transmission avoids cables connecting the recording system to an external decoder and reduces risk of infection.

For future implementation of practical BMI implants consisting of an MEA and neural recording electronics, BMI implants need to provide both high channel-count and wireless interface. The neural recording system must be capable of processing neural data online for real-time decoding. Because the neural recording system would be surgically implanted in the brain, the size of the system needs to be small. To prevent tissue damage due to temperature increases, the system must consume low power. Low power consumption also enables long term operation of the system.

1.3 Challenges

To date, several research groups have developed wireless intracortical neural recording systems and tested them in animals [14-16]. These systems can record tens of channels simultaneously and consume around 100mW power. Due to the high power consumption, they can only operate a few hours continuously. For example, [16] first presented a fully implantable device with 90.6mW power consumption for one hundred channels in moving primates. Because of the high power consumption, it can only operate for 6.6 hours and has to apply active cooling to mitigate the heat generation. The main reason for the high power consumption is the wireless transmission of raw signals for all one

hundred channels. In this system, the total data rate is 24Mbps, and wireless transmission of this amount of data consumes 80% of the total power. If the amount of data for wireless transmission can be greatly reduced, significant power consumption can be saved significantly.

Future BMIs with hundreds of intracortical neural recording channels will generate a data rate of hundreds of Mbps. Wireless transmission of such tremendous data without any reduction will further increase the heat dissipation and decrease the operation time, thus prohibiting the clinical application of BMIs. To ease the heat dissipation and extend the operation time, wireless power consumption must be minimized. Thus, a data reduction technique needs to be incorporated into the recording system to perform online and real-time data reduction. The reduction ratio must be over two hundred to reduce the data rate to around 1Mbps, thus reducing power for wireless transmission to less than 10mW. From the system point of view, several challenges remain in order to achieve such a high data rate reduction ratio.

The first challenge for data reduction in a high-channel-count BMI lies in **maintaining high information quality** for decoding in the presence of variable background noise levels. Maximization of information quality provides accurate decoding performance. Existing methods are sensitive to noise which results in information loss. Robust algorithms need to be developed to minimize the noise effect on information quality.

The second challenge for data reduction in a high-channel-count BMI lies in the fact that the data reduction technique needs to be **automatic** and capable of **real time implementation**. Current data reduction techniques require several parameters to be experimentally determined, channel by channel, in order to preserve information quality.

As background noise varies, these parameters need to be updated in real time, which is almost impossible when the number of recording channels is scaled up to one thousand. Thus, it is necessary to develop automatic parameter estimation methods that are insensitive to the variability of noise.

The third challenge for data reduction in a high-channel-count BMI lies in the **hardware-efficient implementation** of the data reduction technique in terms of low power and area consumption. Because the neural recording system is implanted, hardware implementation must be low power for low heat dissipation without tissue damage, and it must be low area for surgical implantability.

1.4 Goals

The primary goal of this research overcome challenges to next-generation BMIs through the development of hardware-efficient data reduction algorithms that enhance the quality of information while achieving high data rate reduction ratio for a high-channel-count BMI. As explained in Chapter 2, the data reduction method with the highest data rate reduction ratio is referred to as *spike sorting* which consists of three steps. Thus, this thesis specially seeks to:

- Develop algorithms in each step of spike sorting that can maintain high quality information even when the noise level of neural signals degrades.
- Develop hardware-efficient design methods for each step of the spike sorting algorithms and scalability to a high-channel-count BMI.
- Efficiently integrate hardware designs in each step of spike sorting together as a data reduction scheme for processing high-channel-count neural signals.

1.5 Outline

Chapter 2 describes backgrounds of this research. Neurophysiology and neural recording techniques are discussed. Different data reduction techniques are discussed and hardware-efficient methods for the best data reduction technique, spike sorting, are reviewed. Chapter 3 designs the first step of spike sorting to preserve useful information by analyzing the statistic of neural background noise. For the second step of spike sorting, Chapter 4 introduces a new method to extract key points from useful information obtained in the first step. Chapter 5 designs the third step of spike sorting by introducing a quantized oblique decision tree model. Chapter 6 integrates all of the three steps together and implements a neural signal processing system for high-channel-count application. Finally, Chapter 7 summarizes the thesis work and contributions, and it outlines future work related to this research.

Chapter 2 Background

2.1 Neurophysiology and Electrophysiology

2.1.1 Neurons and Action Potentials

When a person tries to move their body, their brain sends messages to certain muscles. How does the brain send these messages to the rest of the body? The messages are transmitted from the brain to the body by nerve cells, or neurons. It is estimated that the adult human brain contains 10^{10} neurons [17]. Communications between neurons construct a massive network that extends throughout the human body. The brain sends messages across this network, which enables people to think, move and feel.

The structure of a typical neuron is shown in Figure 2.1 [17]. The cell body contains genetic materials and makes proteins. Dendrites are the regions where a neuron receives information from other neurons. The received information is converted into an electrical impulse which begins at the axon hillock and propagates along the axon. The myelin sheath insulates the axon and helps the electrical impulse to transmit more quickly. When the electrical impulse travels down to the axon terminals, neurotransmitters are released and the neuron activates the receptors of other neurons in a process known as synaptic transmission.

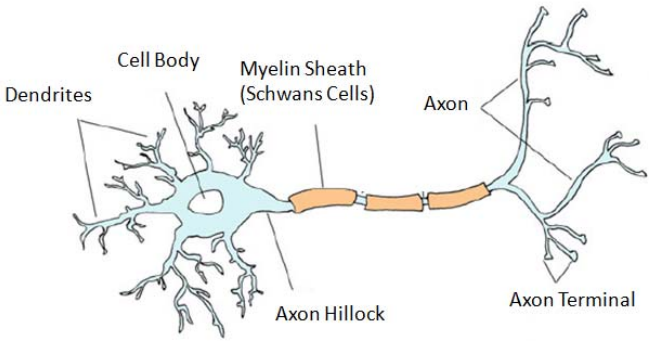


Figure 2.1. The structure of a neuron adapted from [17].

The electrical impulse carrying neural information is called an action potential (AP), which is the difference in voltage between the inside and outside of a neuron. Neurons use sodium (Na^+) and potassium (K^+) ions to create the voltage difference across the neuronal membrane. A typical waveform of an AP is shown in Figure 2.2 [17]. According to ion flow through ion channel proteins in the neuronal membrane, an AP can be described in four phases: 1) *resting state* where ions around the membrane are in an equilibrium condition, 2) *depolarization* where sodium ions flow into the neuron through sodium channels and an AP is initiated and reaches to its peak, 3) *repolarization* where potassium ions flow out of the neuron through potassium channels and the AP goes back towards to the resting potential and 4) hyperpolarization where potassium channels remain open and allow potassium for efflux after the AP reaches the resting potential, which results the potential of the AP keeps decreasing and becomes more negative than the resting potential.

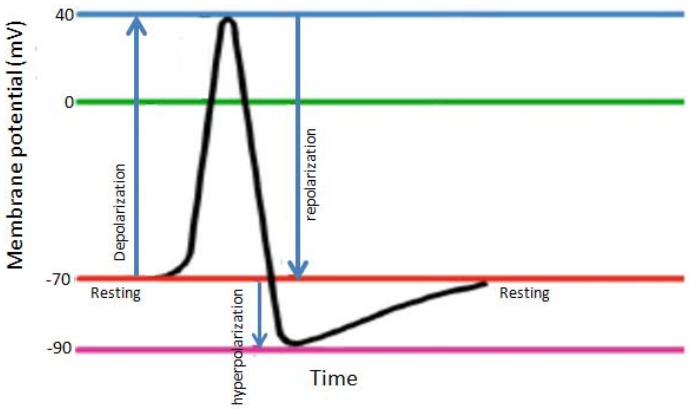


Figure 2.2. A typical waveform of an action potential adapted from [17].

Early studies on the electrophysiology of brain unveiled the relationship between the APs of neurons and the kinematic limb movements in the motor cortex of monkeys [18-20]. Since then, there has been enormous interest in research to link brain activities to

man-made devices to enable communication and control among severely disabled patients. The devices that can read brain signals and convert them into control and communication signals are called brain machine interface (BMI) [5].

2.1.2 Neural Recording Techniques

Current BMI electrical recording techniques range from noninvasive to invasive modalities. Electroencephalography (EEG) is a non-invasive method that places electrodes on the scalp. This approach has proved its capability of helping paralyzed patients to control a computer screen cursor [7] or a communication device [21]. One study further demonstrated that human subjects are capable of controlling a robotic quadcopter in a 3D physical space [22]. The primary advantage of EEG based BMIs is its noninvasive nature. However, because each EEG electrode samples average activities of neurons on the order of 10^5-10^8 [23], EEG signals provide poor spatial and temporal (less than 70 Hz [24]) resolutions. As a result, the information that EEG conveys is non-specific and thus EEG based BMI requires intensive training of weeks and suffer high error rates [25]. The information transfer rate (ITR) measuring the time required for conveying commands for EEG based BMIs is less than 0.5 bits/sec [26]. This limits the number of degrees of freedom (DOFs) that can be controlled using EEG.

To have better spatial and temporal resolutions, a minimally invasive method using electrocorticography (ECoG) was developed. ECoG records the electrical activities from the surface of the brain, beneath the skull, by integrating signals from 10^2-10^3 neurons [23]. ECoG can record higher frequency (up to 200 Hz [24]) neural signals than EEG. The higher frequency contents in signals provide additional cognitive and motional information for BMI control that is not accessible with EEG [27]. Experiments have

shown that subjects can learn to control a computer cursor less than half an hour by modulating the ECoG signals [8, 28]. Another experiment even showed that a paralyzed individual can operate ECoG based BMI to control a robot arm to hit targets and touch hands in a 3D space [29]. However, the ITR for ECoG based BMIs are limited to only 1.5 bits/sec [30]. The quality of ECoG control appears to be still considerably poor and unreliable [31].

The highest spatial and temporal (up to 5-10 kHz [24]) resolutions of electrical activity can be acquired by inserting microelectrodes into the cortex, the outer 2-4mm of the brain where most thoughts are believed to take place, to record the electrical activities from individual neurons. The method of recording signals from the cortex is called intracortical recording. Several research studies have used this method for communication and control in non-human primates [9, 32-34]. One research reported control of intracortical BMI device by monkeys with ITR up to 6.5 bits/sec [32]. One group showed its application on a human with tetraplegia who is capable of controlling a prosthetic limb to reach and grasp objects with seven DOFs (3D translation, 3D orientation, 1D grasping) [12]. The same group recently demonstrated that this person with tetraplegia can control the prosthetic limb with ten DOFs (3D translation, 3D orientation, 4D hand shaping) [35]. Another group showed that a paralyzed individual was able to use their neural prosthetic system to type words at a speed of approximately 6 words/min [36]. It has been shown that the BMI performance increases with the size of the neuronal recording ensemble [37]. In contrast to the EEG and ECoG methods, intracortical recording is the only method that provides the neuronal recordings scale that will enable future BMIs with the high DOF necessary for prostheses with natural

movement dexterity.

2.1.3 Characteristics of Intracortical Recording

The goal of intracortical recording is to obtain the sequences of APs that represent threads of information referred to as neural codes [38, 39]. For example, in controlling the movement of a prosthetic limb, the sequences of APs can be used to estimate the coordinates and orientation in the three-dimensional space [12]. Two main principles schemes for encoding APs are used by the neuroscience community. One is called *rate coding ,* and it estimates the average firing rate of neurons by counting the number of APs in a time interval. The other is called *temporal coding*, which stems from the neuroscience theory that precise relative timing of individual APs carries the information. In order to decode and interpret neural codes using either of these coding schemes, the first and critical step is to record AP events.

The occurrence of APs can be recorded within neurons, which is referred to as *intracellular recording*, or outside of neurons, which is referred to as *extracellular recording*. Intracellular recording is achieved by inserting an electrode into a neuron, allowing morphological identification of the recorded neurons. This form of recording can only be done on a few neurons simultaneously and the neurons must remain physically unmoving. Thus, to obtain high signal quality in animal studies, intracellular recordings can only be done on anesthetized animals. When recording in freely moving animals, the recording is often lost due to movements and thus the recording duration is limited to only one hour [40]. Extracellular recording places electrodes in close proximity (<60μm) of neurons and permits long term recording in freely moving animals. Because recent technological advances in the extracellular electrodes have made it possible to

record simultaneously large number of neurons, extracellular recording has become the main technique for neuroscientists to study complex brain functions.

Figure 2.3 shows the signals of intracellular and extracellular recording. Intracellular recording provides higher amplitude, on the order of 100mV. However, due to the low-pass filtering property of the extracellular space, the amplitude of extracellular recording is quite small, on the order of 100μV [41]. Extracellular recordings typically contain two types of signals: local field potential (LFP) and neural spikes. The LFP arises from the sum of activity of neurons distant to the electrode and present as low frequency oscillations below 250 Hz [42]. The neural spikes indicate an AP fired by neurons in close proximity to the electrode. Spikes have energy concentrated in the high frequency range (typically from 400 to 3000 Hz [42]). In many applications, it is desirable to separate LPF and spikes using signal processing filters of different frequency bands. The middle waveform in Figure 2.3 shows a filtered signal in which LPF is removed. It consists of background noise and spikes. The main source of background noise is biological noise which reflects the activities of neurons far away from the electrode. Another source of noise is the thermal noise which is attributed to the recording instruments. Spikes recorded by an electrode can include indications of APs from multiple neurons. Identification of source neurons that are firing spikes recorded by an electrode is thus necessary before decoding. It is widely believed that spikes observed by an electrode from different neurons exhibit different shapes because of their relative positions to the electrode and the distributions of conductive ion channels [43]. Source identification thus can be performed based on the shapes of spikes.

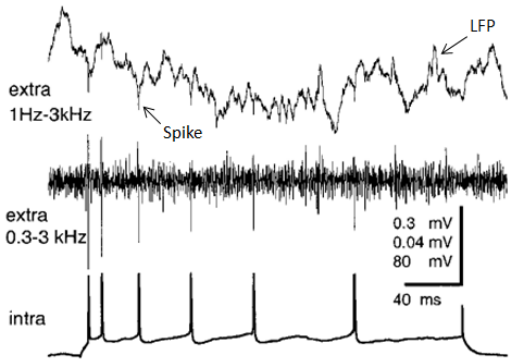


Figure 2.3. Intracellular and extracellular signals from [44]. Top: wideband extracellular signals from 1 Hz to 3 kHz. Middle: highpass filtered extracellular signals from 0.3 to 3 kHz. Bottom: intracellular signals.

2.2 Intracortical Neural Recording Systems

In order to record activities of individual neurons using extracellular methods, an intracortical neural recording system must be capable of obtaining spikes with small amplitude. In the early days of neural recording, from the 1970s to the middle of 1980s, extracellular signals recorded from microelectrodes were amplified and filtered using discrete electronic components [20, 45-47]. The conditioned signals were either displayed on an oscilloscope or connected to a device capable of indicating the firing rate of APs. From the 1980s to the 1990s, continuous improvements of CMOS fabrication technology enabled the minimization and implantation of recording systems. Different monolithic systems were designed to integrate the amplifier and the filter into a single chip [48-51]. Some of these systems could process up to ten channels simultaneously. During this period, data transfer between the front-end recording amplifiers and an external spike analysis device was still achieved by interconnect wires. For chronic recording, these wires significantly limited mobility and impeded the study of freely moving subjects and the development of BMIs for human use. Thus, efforts to integrate biotelemetry into the system for data transfer became central to neural interface research. From the late 1990s

to the middle of 2000s, early implementations of wireless recording systems consumed a significant amount of power and were only able to manage around four signals channels [52-55]. In the past decade, research in wireless neural recording has focused on head-mounted multichannel designs that can be tested in the free living animals while recording up to 100 channels. Table 2-1 lists performance metrics of recent wireless intracortical neural recording systems. It can be seen that power consumption is a critical issue when transmitting raw data with enormous data rate [15, 16]. Some of these systems perform data reduction techniques to ease power consumption [56, 57]. these early data reduction techniques compromised the data integrity necessary to separate activities of individual neurons. In contrast, future BMIs will require large-scale recording neuronal populations [58] that will yield an aggregate data rate in the order of tens of Mbps. The ultimate goal of future high performance intracortical BMIs is to develop a fully implantable wireless recording system. Such a system would benefit from eliminating the infection risk and reducing mechanical vulnerability. However, wireless transmission of a vast amount of neural data would consume considerable power and dissipate significant heat. To prevent tissue damage, new neural signal processing techniques are needed that can simultaneously minimize the data rate and power consumption in real time while preserving high data integrity.

Table 2-1 Comparison of performance metrics for intracortical neural recording systems

Reference	[14]	[56]	[57]	[15]	[16]
No. of channels	96	100	64	32	100
Data (Mbps)	1	0.35	2	24	24
Power (mW)	100	10	14.4	142	90.6
Year	2007	2009	2009	2010	2013

The components of a complete wireless intracortical neural recording system are shown in Figure 2.4. Extracellular signals are usually recorded by a microelectrode array (MEA) that is implanted in the cortex. Weak (low amplitude) neural signals recorded by the MEA are first conditioned using amplifiers and filters to eliminate the background LFP and any electrode offset between electrodes. The conditioned signals are then converted into digital signals by analog-to-digital converters (ADC). The ADC sampling rate of signals is usually between 20 kHz and 40 kHz to cover the energy spectrum of APs and minimize distortion of spikes. A neural signal processor (NSP) analyzes and compresses the digitized signals to retain the meaningful neurological information within a minimized data set. The relevant information is then transmitted out to an external data processing system by a wireless transceiver. Because the power consumption of the neural recording system is dominated by the wireless data transmission [59], the NSP block plays a critical role in high-channel-count recording by performing data reduction. The schemes of neural data reduction feasible for implantable hardware implementation are reviewed in the following section.

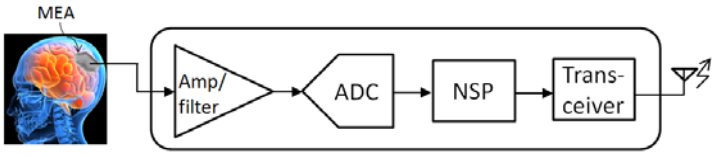


Figure 2.4. The functional block diagram of a wireless intracortical neural recording system.

2.3 Neural Data Reduction Techniques

2.3.1 Spike Timestamps and Spike Waveforms

The firing rate of a neuron in the cortex may be up to 100 spikes/sec [60]. Because the information needed to decode neural codes is contained within the occurrence rate of spikes, data reduction can be achieved by capturing timestamps of each spike occurrence while ignoring the data between spikes. Spikes can be separated from the background by comparing the amplitude of filtered signals with a predetermined threshold. If the signal amplitude crosses the threshold, a spike event is located. Spikes have been captured in the analog domain, where the output of each channel is a binary stream with 'one' representing that a threshold has been crossed [61]. The address of the electrode channel has been transmitted when a spike is found in the analog [62] or digital [63] domains within a specific channel.

The timestamp method provides very conservative use of hardware resources that minimize power consumption within the front-end implanted device. It provides a high data reduction rate with binary outputs. However, because all spikes at a given electrode are reduced to binary events, it is impossible to discriminate spikes originating from different sources. As mentioned in Section 2.1.3, spikes recorded by an extracellular electrode can originate from multiple neurons. Thus, it is impossible to discriminate spikes fired by different neurons using timestamps, which limits the performance of this method in neural decoding.

Given that each neuron fires spikes of a particular shape [64], an alternative method is to capture the entire spike waveform and transmitted out waveform information for further signal processing. This method requires a larger amount of memory cells than timestamp method because spike samples before the threshold is crossed need to be stored. One implementation utilizes 16 byte first-in first-out buffers to store spike samples before the threshold is crossed and outputs totally 64 samples per spike [65]. The data reduction rate of the spike waveform method varies from 3.5 to 48 and decreases dramatically as the neuronal firing rate increases [65]. For a typical firing rate of 60 spikes/sec, it shows that the data reduction ratio of eight can be achieved [65].

2.3.2 Transform Compression

Transformation compression is another method used for neural data reduction. Figure 2.5 shows the procedure of this method. Neural signals are transformed into a new space by convolving or multiplying with an orthogonal basis. The orthogonal basis can be considered as fundamental vectors that represent the new space. Neural data are transformed into new values referred to as transform coefficients in the new space. If the basis is properly selected, neural spikes will have transform coefficients with large values while neural noises between spikes will have transform coefficients with small values. In the thresholding step of the compression procedure, transform coefficients are compared with a threshold value such that coefficients less than the threshold are set to zero. If an appropriate threshold value is chosen, most insignificant transform coefficients related to neural noises will become zero. The final run-length encoding (RLE) step passes values above the threshold while compressing the sequences of zero coefficients and storing them as coded data values representing the zero count in each sequence.

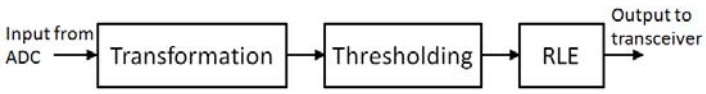


Figure 2.5. Diagram of the procedure for the transform compression method.

One transform algorithm that has been utilized with neural signals is the discrete wavelet transform (DWT). DWT decomposes a signal into different levels that are related to binary-scaled frequency bands while preserving temporal structure of the original signal. Figure 2.6 shows the frequency domain representation of the DWT. In the j^{th} level of decomposition, the approximation coefficients a_j are convolved with a half-band lowpass filter h_0 generating the approximation coefficients a_{j+1}, and a_j are convolved with a half-band highpass filter g_0 generating detail coefficients d_{j+1} for the next level. The DWT decomposition process can be described as

$$\begin{aligned} a_{j+1}(k_{DWT}) &= \sum_{n_f} h_0(n_f - 2k) a_j(k_{DWT}) \\ d_{j+1}(k_{DWT}) &= \sum_{n_f} g_0(n_f - 2k) a_j(k_{DWT}) \end{aligned} \tag{2-1}$$

where n_f is the index of half-band filter coefficients and k_{DWT} is the index of transform coefficients. The approximation coefficients a_{j+1} and detail coefficients d_{j+1} are downsampled, or in other words, every other wavelet coefficient is discarded to eliminate information redundant with the previous decomposition level.

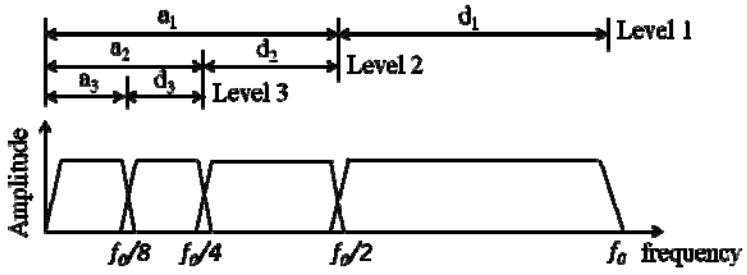


Figure 2.6. Frequency domain illustration of a three-level DWT decomposition.

After DWT transformation, a threshold value is applied to the detail coefficients at

each level. Because the energy of neural signals are concentrated in the low frequency bands, most detail coefficients in the lower levels are related to high frequency noise and will become zero after thresholding.

Different wavelet bases have been used for DWT-based transformation compression such as a simple 'haar' wavelet [66] and a 'symmlet4' wavelet [67, 68]. The data reduction ratio of DWT is reported to be 61 and 62 for 'haar' and 'symmlet4', respectively [69, 70]. Because the 'symmlet4' wavelet basis is more similar to the shape of spikes than 'haar' wavelet, it provides less distortion of neural spikes than 'haar' wavelet.

Another transform algorithm called Walsh–Hadamard Transform (WHT) projects signals into a new space by multiplying a signal with matrix H_m, such that

$$Y = H_m * X \tag{2-2}$$

where m is an integer, H_m is a $2^m \times 2^m$ matrix, X is a $2^m \times 1$ input vector and Y is a $2^m \times 1$ output vector. The H_m matrix can be recursively defined as

$$H_m = \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}, H_0 = 1 \tag{2-3}$$

It can be seen from (2-3) that the elements of WHT matrix only consists of 1 and -1, which indicates low computation complexity of WHT.

A WHT-based transformation compression was implemented with m equal to 3 [70]. WHT provides the data reduction ratio of 63 when the firing rate is low and the data reduction ratio decreases to around 5 when the firing rate is as high as 100 spikes/sec Because WHT requires no multiplication operations, its computation complexity is less

than DWT.

2.3.3 Spike Sorting

As mentioned in section 2.1.3, the spikes observed at an electrode originating from different neurons exhibit different shapes. Spike sorting is a process that groups the recorded spikes based on the similarity of their shapes and thereby provides the information about individual neuron firing needed for neural decoding. Because the goal of spike sorting is to assign a source-label to each spike, a spike can be represented by a single spike-label data value. As such, spike sorting provides higher data reduction ratio than spike timestamps, spike waveforms or transform compression methods. Thus, spike sorting based NSP has become the main technique for high-channel-count neural recording system. The strategies for spike sorting can be divided into template matching and feature-space based methods.

2.3.3.1 Template Matching based Spike Sorting

Template matching methods first derive spike templates from all detected spike waveforms during a training phase. Using these spike templates, different spike sorting discriminators can be constructed. One discriminator was constructed using matched template filters (MTFs) that compare a neural signal with spike templates [71]. If the difference between the neural signal and an MTF is less than a threshold, a spike is simultaneously detected and classified. Alternatively, optimal linear filters [72] and neural networks [71] have been constructed based on spike templates such that only one linear filter or one neural network output provides the largest response to spikes fired by a specific neuron and severely attenuates others. One recent work has integrated template matching based spike sorting into a head-mounted wireless recording system which can

record up to 512 channels in monkeys [73]. However, the system consumes approximately 2mW power per channel and only allows continuous operation for 30 hours.

2.3.3.2 Feature based Spike Sorting

Figure 2.7 shows the basic steps of feature-based spike sorting. Spikes are detected from the noisy input signal and aligned to a common temporal point of the spike waveform. This step is similar to the spike timestamp method. In the feature extraction step, the aligned spikes are projected into a lower dimensional space where each spike is represented by significantly fewer data points than the original spike waveform. The last step uses the extracted features to classify spikes into different neuron source groups. Notice that the feature extraction step inherently provides data reduction. As a result, some neural recording systems transmit spike features to ease the communication bandwidth [74, 75].

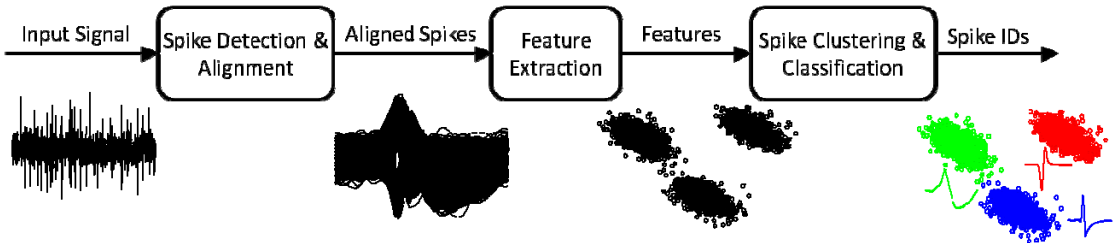


Figure 2.7. Basic steps of feature-based spike sorting algorithms and the output signals at each step.

Template matching methods require intervention from a human operator to form templates [76], and they demand a large amount of memory to store the templates. These properties make these methods impractical for high-channel-count applications. Compared to template matching methods, feature-based methods provide

computationally efficient and memory efficient characteristics for multichannel spike sorting. The algorithmic complexity of feature-based spike sorting determines its accuracy and the hardware cost. For the implantable application, a feature-based spike sorting NSP needs to meet both low power-area and high reliability requirements. In the following section, computationally efficient methods for each step of feature-based real-time spike sorting procedure will be reviewed.

2.4 Computationally Efficient Real-time Spike Sorting Methods

2.4.1 Spike Detection

Early spike detection techniques developed for the spike timestamps method applied thresholds to raw neural recording signals. Thresholding of raw signals is easy to implement in hardware, but detection performance is sensitive to the background noise. Experiments have shown that up to 40% of recording sites generate signal with high background noise, and this can increase to nearly 60% during recording sessions [77]. Therefore, some preprocessing methods have been proposed to enhance the signal-to-noise ratio (SNR) of neural signals, allowing detection performance to be improved by applying a thresholding technique to the preprocessed signal.

The most popular preprocessing method for spike detection is the nonlinear energy operator (NEO). The NEO algorithm was first proposed to track the instantaneous energy of signals composed of several different frequency components [78]. It was applied to the neural spike detection because a spike can be considered as the instantaneous increment of the energy level of a neural signal [79]. The NEO operator in continuous time, $\Psi(t)$, can be expressed

$$\Psi(t) = (\frac{dx(t)}{dt})^2 + x(t)(\frac{d^2x(t)}{dt^2}) \quad (2\text{-}4)$$

where *x(t)* is the input signal. In discrete time, the NEO can be defined as

$$\Psi(n) = (x(n))^2 + x(n-1)x(n+1) \quad (2\text{-}5)$$

One extension of the NEO reported to exhibit less sensitivity to high noisy peaks when the SNR is low is described by [80]

$$\Psi_k(n) = (x(n))^2 + x(n-k)x(n+k), k\ is\ an\ integer \quad (2\text{-}6)$$

where *k* is a tuning parameter.

Once the NEO operators have been determined, thresholding can be applied to separate spikes from noise. The threshold value can be set as a scaled value of the local average of NEO values, as given by

$$Thr = C\frac{1}{N}\sum_{n=1}^{N}\Psi(n) \quad (2\text{-}7)$$

where *N* is the number of samples in the signal, and *C* is a scaling factor that needs to be determined experimentally [81]. It can be seen from (2-5) that the NEO operation only requires two multiplications and one addition, thus making it suitable for implantable hardware realization. NEO-based spike detectors have been implemented in both analog [82, 83] and digital domains [75].

Another well-known method is matched filter based spike detection where the neural signal is convolved with a predefined spike template. Because the spike template is unknown a priori, it requires users to select spikes from the test data and average those spikes to form a template. Matched filter based spike detection has been shown to be difficult for real-time implementation because of its relatively large computational

requirements [84].

Wavelet methods have also been applied for spike detection. The stationary wavelet transform (SWT) is more commonly used for detection than DWT because it provides better overall performance. SWT decomposes signals into different levels corresponding to different frequency bands of signals, which is similar to DWT, but SWT does not decimate coefficients at each level. Selected levels of detail coefficients are utilized for spike detection by applying different threshold values at each level. The threshold value is usually calculated as a scaled version of the standard deviation σ_j of the detail coefficients at each level j. σ_j can be estimated by the median absolute deviation of detailed coefficients, as given by:

$$\sigma_j = median\left|d^s{}_j\right|/0.6745 \tag{2-8}$$

where $d^s{}_j$ is the SWT detail coefficients at j^{th} level and 0.6745 is a constant of median absolute deviation [85]. The non-decimation property allows the SWT to be insensitive to the sampling phase and thus provides more reliable detection performance than the DWT [85, 86]. Different wavelet bases have been selected for SWT-based spike detection, such as 'coiflet' basis [87], 'symmlet7 'basis [85] and 'symmlet4' basis [88]. A recent study shows that 'symmlet2' basis provides a better balance between hardware resources and detection accuracy [86].

A new method to detect spikes in Hilbert space has recently been proposed [89]. It shows that the probability density function (PDF) of neural signals after Hilbert transform can be decomposed into two components: one is the exponential component (EC) and the other is the power-law component (PC). The PDF of the background noise follows EC distribution while the PDF of spikes follows the PC distribution. Thus a spiking

probability map can be generated based on the PDFs of the noise and spikes. Spikes are detected by thresholding the spiking probability map. This method has been shown to be robust for spike detection and feasible for hardware implementation [90].

2.4.2 Alignment

Following spike detection, spikes are often aligned to a common temporal point to minimize variation between spikes from the same neuron caused by sampling jitter. When a spike is detected, it is automatically aligned to the threshold crossing point, which is sensitive to noise. For a more precise alignment, the most common method is to align spikes to their maximum amplitude [91, 92], where spikes are usually upsampled by interpolation and then downsampled to the original rate to avoid sampling jitter misalignment. Another alignment method that uses maximum slope as a reference point [71] and has been realized in hardware [75] because the maximum slope is considered to have more biological significance than the maximum amplitude. Although the alignment to the maximum amplitude or slope provides better sorting performance, they both require significantly more computation and hardware, particularly memory, than simple alignment to the threshold crossing.

2.4.3 Feature Extraction

When a spike is detected and aligned, it is represented by a few dozens of samples. The goal of the feature extraction step is twofold: extracting the most informative features to separate different types of spikes and using as less number of features as possible to minimize the computation complexity.

Conventional features such as the spike maximum and minimum amplitudes and spike width [92] are sensitive to noise and provide poor discrimination among spikes. To

enhance the feature robustness to noise, a method called principle component analysis (PCA) is used to project spike waveforms into another space using principle component (PC) vectors [92]. The PCs are obtained by calculating the eigenvectors of the covariance matrix of spike waveforms in a training set. Then the projection coefficient C_i of each spike on the i^{th} PC vector can be computed as

$$C_i = \sum_{n=1}^{N} PC_i(n)S(n) \tag{2-9}$$

where S represents a spike and N is the number of samples in a spike. Usually, the first two or three PC vectors are used because these features provide the most informative information to separate spikes. Another sophisticated method robust to noise performs DWT on the spike waveforms and selects the DWT coefficients as features based on Lilliefors test [91]. Both PCA and DWT methods request a huge amount of hardware resources, but they can be considered as gold standards for comparison when other features developed for the implantable application are evaluated.

Computationally efficient feature extraction methods can be divided into two categories. One performs the integral and the other performs the discrete derivative (DD) on spike waveforms. The integral methods, such as integral transform (IT) [93] and zero crossing features (ZCF) [94], rely on the area under positive and negative lobes. Figure 2.8 illustrates two methods. The IT method can be described as

$$I_A = \frac{1}{N_A}\sum_{n=n_A}^{n_A+N_{A-1}} S(n)\,, I_B = \frac{1}{N_B}\sum_{n=n_B}^{n_B+N_{B-1}} S(n) \tag{2-10}$$

where n_A is the first sample of the first lobe, N_A is the number of samples in the first lobe, n_B is the first sample of the second lobe, N_B is the number of samples in the second lobe. The ZCF method can be expressed as

$$ZC1 = \sum_{n=0}^{Z-1} S(n), ZC2 = \sum_{n=Z}^{N} S(n) \quad (2\text{-}11)$$

where Z is the index of the first zero after a spike is detected. The parameters of the IT method requires the offline training while the ZCF method only depends on the zero-crossing point which can be adaptively obtained online.

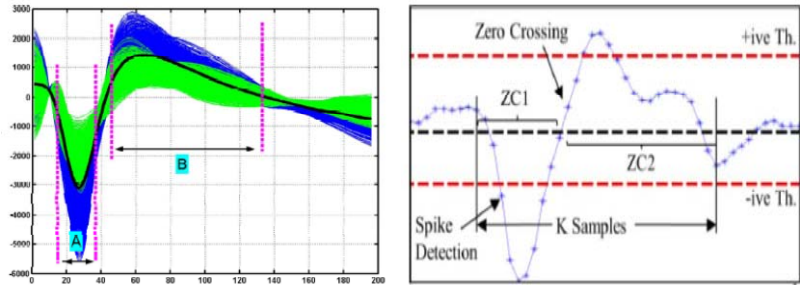


Figure 2.8. Illustration of IT (left) [93] and ZCF (right) [94] feature extraction methods.

The DD methods compute the slope of each spike sample over a number of different time scales. It can be described as

$$DD_{\delta} = S(n) - S(n - \delta) \quad (2\text{-}12)$$

where δ is the scaling factor and is chosen as 1, 3 and 7. This yields three times more expansion coefficients than the original spike. Different tests such as Lilliefors test, Hartigan's dip test and the maximum-difference test are operated to select a subset of coefficients as features [95]. It shows that 10 DD coefficients provide the acceptable classification accuracy based on the maximum-difference test. A recent study tested the permutations of DD extremas at all the time scales and shows that using the extremas for δ equal to 3 and 7 provides the best separation [96]. Another similar approach uses the first derivative (FD) and second derivative (SD) extrema (FDSE) of spike waveforms as features [97]. The FD and SD can be computed as

$$FD(n) = s(n) - s(n-1), SD(n) = FD(n) - FD(n-1) \quad (2\text{-}13)$$

It shows that the positive peak of the FD together with the positive and negative peaks of the SD achieves the best classification accuracy [97].

All the recent efforts for feature extractions like ZCF, DD extrema and FDSE target at the online processing of spike waveforms and reduce the number of features for each spike to less than four. These recent methods enable the on-chip implementation of high-channel-count spike sorting for the future BMI application.

2.4.4 Clustering & Classification

The purpose of clustering & classification is to divide all the detected spikes recorded from an electrode into different number of spike classes because one electrode can record activities of multiple neurons. The clustering & classification method can be implemented either offline or online. The offline implementation provides better sorting performance than online but it cannot process data in real time. The online implementation allows real-time operation but at the cost of significant hardware usages.

2.4.4.1 Offline Clustering

For offline processing, clustering algorithms are used for both identification of number of spike clusters and classification of spikes. One benchmark method for offline clustering is called k-means which iteratively updates cluster centroids based on the data points closest to each centroid [98]. Another offline method considers the distribution of spikes as the mixture Gaussian model [99, 100] or t-distribution model [101] of which the model parameters are derived by the expectation-maximization algorithm. Different penalty functions are used for these methods to discourage a large number of clusters and determine the correct cluster number. Superparamagnetic clustering (SPC) is also one common offline method which is based on the interaction between each data point and its

k-nearest neighbors [91]. SPC introduces a parameter called temperature. By scanning the temperature from low to high, new clusters will be created with different cluster sizes. The number of clusters can be determined when the cluster size of a new cluster is less than a threshold. The drawbacks of all above clustering algorithms are high computation complexity, high memory storage and off-line processing.

2.4.4.2 Online Clustering & Classification

For online processing, a clustering algorithm is used to determine the number of spike clusters and a classification algorithm is used to assign each detected spike a label. One computationally efficient clustering algorithm suitable for online application is Osort clustering & classification [102]. For each coming spike pattern, this method computes the distances between this spike and current cluster centroids. If the maximum distance is greater than a sorting threshold, a new cluster will be created. Otherwise, assign the input spike to the nearest cluster and updates this cluster centroid. Then this new updated centroid is compared with other centroids. If the minimum distance is less than a merging threshold, two clusters will be merged. The Osort method is the only one clustering method that has been realized for on-chip implementation at this time [103]. The hardware implementation divides this method into two phases. The clustering phase is first trained to determine the number of clusters and their centroids channel by channel. The classification phase assigns each detected spike into a class based on the metric of ℓ_1 norm distance in real time. Although OSort has been implemented on chip, the clustering phase still requires a huge amount of memory. As a result, it only supports clustering one channel at a time. For multichannel application, the clustering phase has to be trained channel by channel in sequence.

2.5 Spike Sorting Processors

A comparison of on-chip implementations of spike sorting processors is given in Table 2-2. Some of the designs output spike features while others output spike labels for data compression. One design extracts features directly from raw spike waveforms, and these features are sensitive to the noise level [104]. One design chose the DD method to trade off between sorting accuracy and computational complexity [75]. Another design uses peaks of raw spike waveforms and the extrema of the first derivative as features [74]. However, its implementation is too power hungry for multi-channel applications. The only designs that perform classification functions at this time are in [103, 105]. One work implemented the training phase of the spike clustering on-chip [103] while the other work trained classification parameters offline [105]. They both perform classification directly on raw spike waveforms without using feature extraction to reduce the data dimensionality. This approach is area-inefficient when scaled for high-channel-count applications. Only two studies reported the overall spike sorting performance for neural signals with different SNRs [75, 103]. None of current spike sorting processors provide detailed analysis on the effect of each spike sorting step on the spike sorting performance. The review of current work concludes that, to date, no spike sorting processor has been implemented achieving both power-area efficiency and high sorting accuracy targeting high-channel-count applications. For future BMI applications requiring a fully implantable wireless neural recording system, it is promising to develop such a power-area efficient and highly accurate spike sorting processor.

Table 2-2 Comparison of on-chip spike sorting processors

Reference	[104]	[74]	[75]	[103]	[105]
Process (nm)	90	350	90	65	FPGA
Voltage (V)	1.08	3.3	0.55	0.55	-
Power (μW/channel)	14.6	100	2.03	4.68	-
Area (mm^2/channel)	0.01	1.58	0.06	0.07	-
Template/Feature	Feature	Feature	Feature	Template	Template
Clustering	No	No	No	Yes	Yes
Year	2009	2009	2011	2013	2014

Chapter 3 Hardware Efficient Automatic Thresholding for NEO-Based Neural Spike Detection

3.1 Advantages and Challenges of NEO based Spike Detection

In Chapter 2, several sophisticated algorithms to enhance the SNR of neural signals for spike detection were reviewed. These signal processing algorithms help to identify spike events when the noise level is high. Among these algorithms, NEO based spike detection has been commonly implemented in hardware designs [74, 75, 82, 104, 105] due to its favorable balance between detection performance and hardware implementation complexity. Figure 3.1 illustrates use of the NEO algorithm to enhance the SNR of a neural signal and shows that the amplitudes of NEO coefficients are much larger where spikes occur than where only noise exists in the raw signal. If a threshold can be set appropriately on the NEO coefficients as shown in Figure 3.1, all spikes in the figure are detected and no noise is identified as a spike.

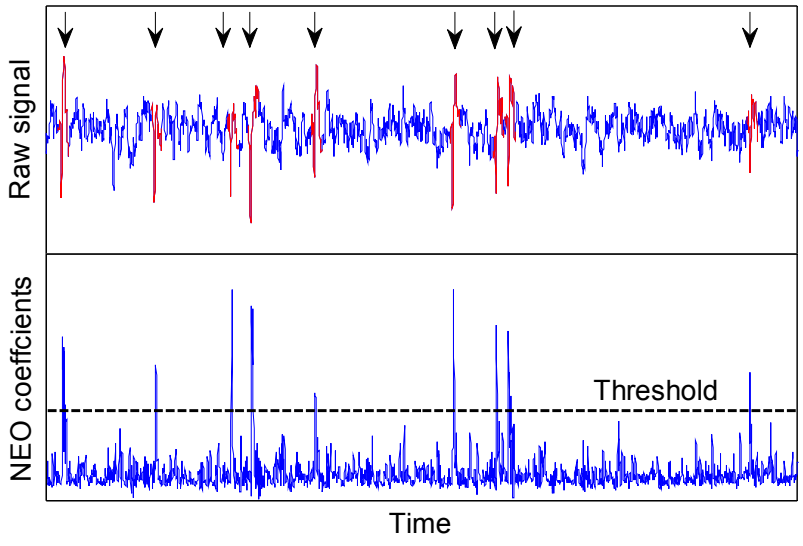


Figure 3.1. An example of the NEO method for enhancing spike events. The arrows indicate when true spikes occur in the raw signal.

Typically, the NEO threshold value Th_{Ψ} is set to a scaled mean of NEO coefficients

$$Th_{\Psi} = C\frac{1}{N_c}\sum_{n=1}^{N_c}\Psi(n) \quad (3\text{-}1)$$

where N_c is the number of samples of $\Psi(n)$, $\Psi(n)$ is a NEO coefficient, and C is a scalar and determined experimentally. The threshold value using (3-1) is sensitive to the spike firing rate, which varies between 10 and 100 spikes per second [60]. Thus, this method prevents adjustment of the threshold in real time as the firing rate changes. One approach in the literature utilized a Q_n estimator for unbiased threshold estimation [106], but this method requires extensive computational resources that challenge hardware realization.

In this chapter, a new method is presented to automatically estimate the NEO threshold value in a manner that is robust to changes in the spike firing rate. The automatic and robust NEO thresholding method enables on-chip threshold estimation for a high-channel-count spike sorting system. This method eliminates the need to periodically transmit neural data for offline manual threshold updates and hence reduces the wireless power consumption.

3.2 Analysis of NEO Threshold Estimation

3.2.1. Statistical Analysis of NEO Coefficients

The neural signal *x(n)* can usually be considered as a spike train *s(n)* corrupted by the background noise *v(n)* such that:

$$x(n) = s(n) + v(n) \quad (3\text{-}2)$$

Assuming the spike train and the noise are uncorrelated, it has been shown that the expectation of NEO coefficients $\Psi(n)$ can be expressed as [81]

$$E[\Psi(x(n))] \approx E[\Psi(s(n))] + E[\Psi(v(n))] \quad (3\text{-}3)$$

Because NEO is an instantaneous estimation of signal energy and spikes can be

considered as instantaneous increments of signal energy, the NEO coefficients of spikes are larger than those of noise [79]. Spike detection requires minimizing the number of missed true spikes while keeping the number of false detections to a reasonable limit [81]. (4) indicates that, when $E[\Psi(x(n))]$ is compared to a threshold value Th_{Ψ}, the threshold can be set such that the probability that $E[\Psi(v(n))]$ is greater than the threshold is within a reasonable limit of false detections. Setting an appropriate threshold requires studying the probability density distribution (pdf) of $E[\Psi(v(n))]$.

If we assume the background noise can be represented by a sum of signals with different frequency components, noise can be given by

$$v(n) = \sum_{k=1}^{K} B_k cos\Omega_k n\,;\ \Omega_k = 2\pi \frac{f_k}{f_s} \tag{3-4}$$

where K is the number of frequency components, B_k is the amplitude of the k^{th} frequency component f_k, and f_s is the sampling rate. It has been shown that $E\left[\Psi\left(v(n)\right)\right]$ under the form of (3-4) can be approximated as [107]

$$E\left[\Psi\left(v(n)\right)\right] \approx \sum_{k=1}^{K} B_k^2 \Omega_k^2 = \sum_{k=1}^{K} B_k^2 \times \frac{\sum_{k=1}^{K} B_k^2 \Omega_k^2}{\sum_{k=1}^{K} B_k^2} \tag{3-5}$$

The first term $\sum_{k=1}^{K} B_k^2$ in (3-5) is related to the energy of the signal computed in the frequency domain, and, using Parsevel's theorem, it can be expressed as the energy computed in the time domain

$$\sum_{n=0}^{N-1}[v(n)]^2 = \frac{1}{N}\sum_{l=0}^{N-1}[V(l)]^2 = \frac{N}{2}\sum_{k=1}^{K} B_k^2 \tag{3-6}$$

where N is the number of samples of $v(n)$ and $V(l)$ is the discrete Fourier transform (DFT) of $v(n)$. The derivation of (3-6) is described as follows:

$$V(l) = DFT\big(v(n)\big) = \sum_n v(n) e^{-j\frac{2\pi l}{N}n} = \sum_n \sum_k B_k cos(\Omega_k n)\, e^{-j\frac{2\pi l}{N}n}$$

$$= \sum_k B_k \sum_n cos(\Omega_k n)\, e^{-j\frac{2\pi l}{N}n} = \sum_k \frac{B_k}{2} \sum_n \left[e^{-j\left(\frac{2\pi l}{N} - \Omega_k\right)n} + e^{-j\left(\frac{2\pi l}{N} + \Omega_k\right)n} \right]$$

$$\xrightarrow{assuming\ \frac{N\Omega_k}{2\pi}\ is\ integer} \sum_k \frac{NB_k}{2} \left[\delta\left(\frac{2\pi l}{N} - \Omega_k\right) + \delta\left(\frac{2\pi l}{N} + \Omega_k\right)\right]$$

$$\sum_n [v(n)]^2 = \frac{1}{N} \sum_l [V(l)]^2 = \frac{1}{N} \sum_k \left(\frac{N^2 B_k^2}{4} + \frac{N^2 B_k^2}{4}\right) = \frac{N}{2} \sum_k B_k^2 \tag{3-7}$$

The second term $\frac{\sum_{k=1}^{K} B_k^2 \Omega_k^2}{\sum_{k=1}^{K} B_k^2}$ in (3-5) is a weighted sum of the square of radial frequencies and is referred to as the RMS frequency. If this second term is denoted as Ω_{rms}^2, then can be expressed as

$$E\big[\Psi\big(v(n)\big)\big] = \frac{2}{N} \sum_{n=1}^{N} [v(n)]^2 {\Omega_{rms}}^2 \tag{3-8}$$

Because $v(n)$ is assumed to have a zero mean and a Gaussian distribution [92], $\frac{\sum_{n=1}^{N}[v(n)]^2}{\sigma_n^2}$ follows the chi-square distribution $\chi^2(N)$ with N degrees of freedom:

$$\frac{\sum_{n=1}^{N}[v(n)]^2}{\sigma_n^2} \sim \chi^2(N) \tag{3-9}$$

where σ_n is the Std of the background noise $v(n)$. Thus $E\big[\Psi\big(v(n)\big)\big]$ can be translated into a chi-square distribution by substituting (3-8) into (3-9)

$$Z_{noise} \equiv \frac{N}{2\sigma_n^2 \Omega_{rms}^2} E\big[\Psi\big(v(n)\big)\big] \sim \chi^2(N) \tag{3-10}$$

Figure 3.2 shows the pdf of Z_{noise} and compares it with $\chi^2(N)$ for N equal to 10. It can be seen that the pdf of Z_{noise} provides a good approximation of $\chi^2(N)$. Because NEO is a measure of instantaneous energy that is equal to the product of the square of the signal's

amplitude and the square of the signal's frequency [78], Th_Ψ can be expressed by

$$Th_\Psi = C_0 \sigma_n^2 \Omega_{rms}^2 \tag{3-11}$$

where C_0 is a scalar. In this case, Th_Ψ reflects the level of noise energy. Thus, we obtain

$$\begin{aligned} &Prob\{E[\Psi(v(n))] > Th_\Psi\} = Prob\{E[\Psi(v(n))] > C_0 \sigma_n^2 \Omega_{rms}^2\} \\ &= Prob\left\{\frac{N}{2\sigma_n^2 \Omega_{rms}^2} E[\Psi(v(n))] > \frac{NC_0}{2}\right\} = 1 - F_{\chi^2}(\frac{NC_0}{2}, N) \end{aligned} \tag{3-12}$$

where F_{χ^2} is the cumulative distribution function for the chi-square distribution. (3-12) shows that, when the threshold value is chosen to be proportional to the neural noise level, setting $Prob\{E[\Psi(v(n))] > Th_\Psi\}$ within a reasonable limit of false detections depends only on parameters of C_0 and N and is independent of the neural background noise.

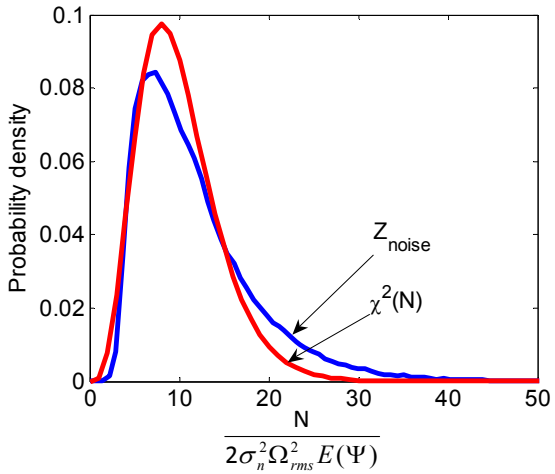


Figure 3.2. pdf of Z_{noise} and $\chi^2(N)$ for N equal to 10.

3.2.2. Parameters Settings for NEO Thresholding

The parameter C_0 determines the ratio of the threshold value to the background noise level. If C_0 is too small, NEO coefficients associated with noise will be detected as false

spikes. If C_0 is too large, NEO coefficients associated with spikes will be missed. To determine the best value of C_0, a hit-to-false-alarm rate (HFR) metric is used, defined as

$$HFR = \frac{F_{spike}(Th_\Psi)\times Firing\_rate}{Firing\_rate + F_{noise}(Th_\Psi)\times f_s} \tag{3-13}$$

where $F_{spike}(Th_\Psi)$ and $F_{noise}(Th_\Psi)$ are the probability of noisy spike NEO samples and noise NEO samples exceeding the threshold value Th_Ψ, respectively. The numerator represents the number of hits and $F_{noise}(Th_\Psi) \times f_s$ in the denominator represents the number of false alarms. To maximize the detection performance, the number of hit needs to be maximized and the number of false alarms needs to be minimized. As defined in (3-11), because Th_Ψ is a function of C_0, C_0 needs to be set to maximize the HFR. Figure 3.3 shows the averaged HFR at different SNRs as a function of C_0 by studying the statistics of synthetic neural signals with sampling frequency of 20 kHz based on real extracellular recordings. The construction of datasets will be described later in the result section. It can be seen that C_0 can be set to 9.5 to maximize the HFR.

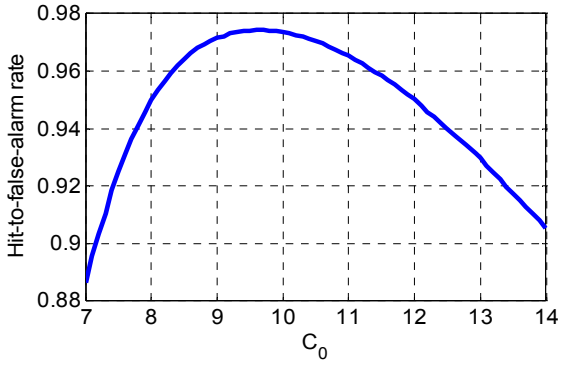


Figure 3.3. The hit-to-false-alarm rate as a function of C_0.

The estimate of $E[\Psi(x(n))]$ can be calculated using a moving average filter with a length of *N* as indicated by (8). The parameter *N* determines the degree of smoothing within NEO coefficients. A factor called "Smoothing effect" (SE) for a NEO signal

$\Psi(x(n))$ is defined as

$$SE[\Psi(x(n))] = \frac{Mean(peaks\ of\ \ E[\Psi(x(n))])}{Mean(peaks\ of\ \Psi(x(n)))} \tag{3-14}$$

A large value of SE provides good approximation of (3-3) and (3-5) and requires a large value of *N*. However, if *N* is too large, the NEO coefficients related to neural spikes will be severely smoothed and become too small to be detected. To provide a balance between the approximation accuracy and the smoothing of spikes, we defined another factor called "Smoothing quality" that describes the ratio between the SE of NEO spikes and the SE of NEO noise, given by

$$Smoothing\ quality = \frac{SE(\Psi(s(n)))}{SE(\Psi(v(n)))} \tag{3-15}$$

The *Smoothing quality* can be used as a criterion to select *N*. Figure 3.4 shows the normalized *Smoothing quality* as a function of a moving average filter of length *N*. It can be seen that the *Smoothing quality* degenerates as *N* is greater than 15, which means NEO spikes are smoothed more than NEO noise. Thus, the best choice of *N* is 15 from Figure 3.4.

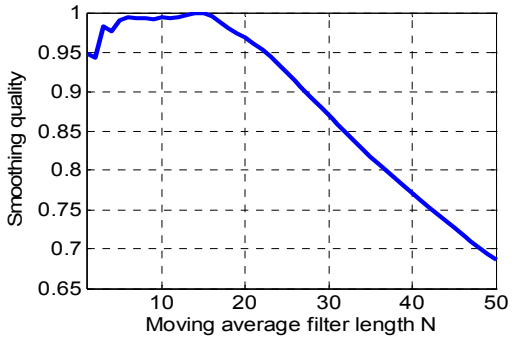


Figure 3.4. Normalized smoothing quality as a function of moving average filter with a length of *N*.

3.3 Hardware Design for Automatic NEO Thresholding

The analysis in section II suggests that an automatic NEO thresholding method could be implemented by first computing smoothed NEO coefficients, $E[\Psi(v(n))]$, and then comparing it to the NEO threshold, Th_{Ψ}. From (3-11), it can be observed that the calculation of Th_{Ψ} requires an estimate of RMS frequency, Ω_{rms}, and the Std of the neural background noise σ_n. Because the neural signal consists of spikes and background noise, any estimation of Ω_{rms} and $\sigma_n$will be affected by the firing rate of spikes. Thus, robust methodologies need to be developed to minimize the impact of the firing rate on the estimate of Ω_{rms} and σ_n. Furthermore, the power and area efficiency of the hardware realization of $E[\Psi(v(n))]$ and of Ω_{rms} and σ_n estimation must be carefully considered so that this threshold estimation function can be implemented in a real-time and fully implantable neural recording microsystem.

3.3.1 Hardware Implementation of $E\left[\Psi\left(x(n)\right)\right]$

Hardware implementation of $E\left[\Psi\left(x(n)\right)\right]$ can be achieved by a moving average filter comprised of several memory registers, each with twice the word length of a neural data sample. Some studies used a Barlett window [81] or Hamming window [106] with five or six registers to approximate the moving average filter estimation of $E\left[\Psi\left(x(n)\right)\right]$. To minimize the circuit area, we proposed to estimate $E\left[\Psi\left(x(n)\right)\right]$ using an infinite impulse response (IIR) exponential filter, which needs only one register. The IIR response, y_{IIR}, to NEO coefficients $\Psi\left(x(n)\right)$ can be expressed by

$$y_{IIR}(n) = \alpha\Psi(x(n-1)) + (1-\alpha)y(n-1) \tag{3-16}$$

where α is the IIR coefficient ranging from zero to one that determines the smoothing

strength of NEO coefficients. To provide the best approximation of a moving average filter, (3-16) can be expanded as

$$y_{IIR}(n) = \sum_{m=0}^{\infty} \alpha(1-\alpha)^m \Psi(n-m) \quad (3\text{-}17)$$

Because a finite impulse response (FIR) moving average filter can be expressed as

$$y_{FIR}(n) = E\left[\Psi\left(x(n)\right)\right] = \sum_{m=0}^{N-1} \frac{1}{N} \Psi(n-m) \quad (3\text{-}18)$$

the error in approximating the FIR filter with the IIR expansion can be determined by the mean-square difference, $J(\alpha)$, between (3-17) and (3-18)

$$J(\alpha) = \sum_{m=0}^{N-1} (\alpha(1-\alpha)^m - \frac{1}{\mathrm{N}})^2 + \sum_{m=N}^{\infty} \alpha(1-\alpha)^m \quad (3\text{-}19)$$

To minimize the approximation error $J(\alpha)$ for a value of N, α can be swept between zero and one. The value of α resulting in the minimum $J(\alpha)$ is selected to approximate the FIR filter with the length of N.

Figure 3.5 shows that α equal to 3/32 provides the best approximation of a moving average filter with N = 15while also enabling an efficient hardware realization for coefficient multiplications.

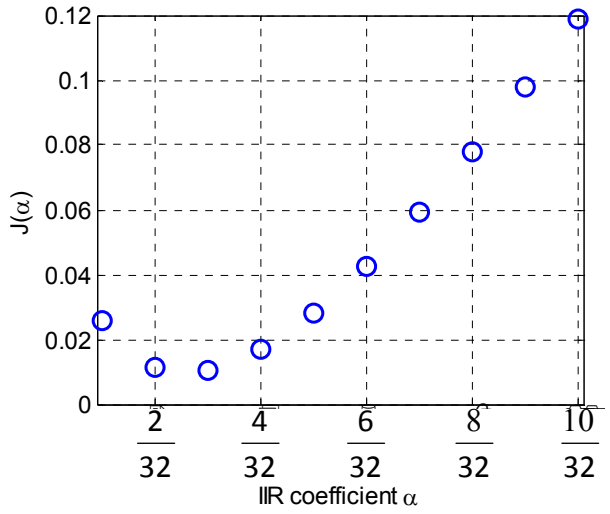


Figure 3.5. The approximation error between the IIR exponential filter and the FIR moving average filter at different values of α.

3.3.2 Estimate of Ω_{rms}

Conventionally, Ω_{rms} can be expressed as:

$$\Omega_{rms}^2 = \frac{\int_0^{\pi} \Omega^2 X^2(\Omega) d\Omega}{\int_0^{\pi} X^2(\Omega) d\Omega} \tag{3-20}$$

where $X(\Omega)$ is the Fourier transform (FT) of neural signal *x(n)*. This method of estimating Ω_{rms} will be greatly affected by the spike firing rate. Moreover, on-chip implementation of (3-20) requires prohibitively expensive computation and memory for the FT and the square operation. For an alternative method, we observe that Ω_{rms} has been shown to be a kind of average measure of the zero-crossing frequency for a zero-mean Gaussian process [108]. The zero-crossing frequency is the number of zero crossings in a neural signal divided by twice the length of the neural signal. Figure 3.6 plots relative Ω_{rms} estimation error vs. firing rate using the conventional method (3-20) and the zero-crossing method and shows that zero-crossing reduces the interference by at least 5% when the firing rate is greater than 50.

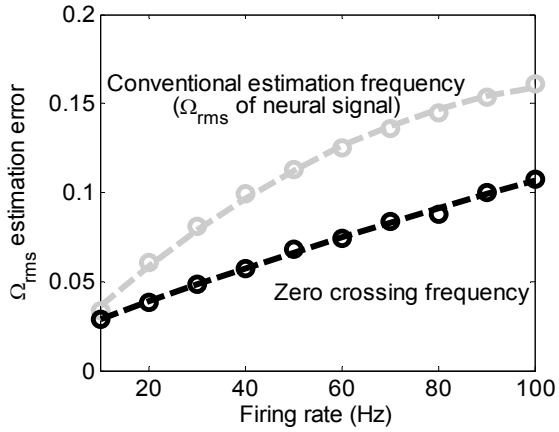


Figure 3.6. Estimation of the RMS frequency using the conventional method and the zero-crossing method at different firing rates with SNR = 4.

Figure 3.7 shows a hardware efficient block diagram for calculating Ω_{rms} using the zero-crossing method. The XOR gate compares sign bits of two consecutive neural samples such that it outputs logic 1 when a zero crossing occurs. The XOR output is accumulated in a register for 2^{N_z} clock cycles. For a final output of the Ω_{rms} calculator of n_z, Ω_{rms} can be determined by $\pi \frac{n_z}{2^{N_z}}$.

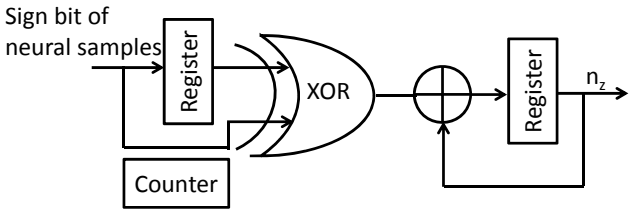


Figure 3.7. Block diagram of zero-crossing Ω_{rms} calculator.

3.2.3 Estimate of σ_n

The conventional way of estimating σ_n (the Std of background noise) is by calculating the Std of a neural signal (spikes and background noise). It has been shown that the conventional estimate is sensitive to the spike firing rate [91]. One study has shown that an estimate for σ_n can be obtained from the median absolute deviation (MAD) [91]:

$$\sigma_n^{MAD} = \frac{median(|x(n)|)}{0.6745} \tag{3-21}$$

The MAD method was demonstrated to be much more weakly affected by spike firing rate than the conventional estimate. However, hardware realization of the *median* operator requires sorting a large array of neural samples. Figure 3.8 shows the relative estimation error for MAD vs. number of neural samples with the absence of spikes. It can be observed that at least one hundred samples are required for the average of accurate estimation and even more for small variance of estimation error. When the spikes in a real neural signal are included, many more samples need to be stored for robust estimation. Thus, the MAD method demands a large memory size and would be area-inefficient for on-chip implementation.

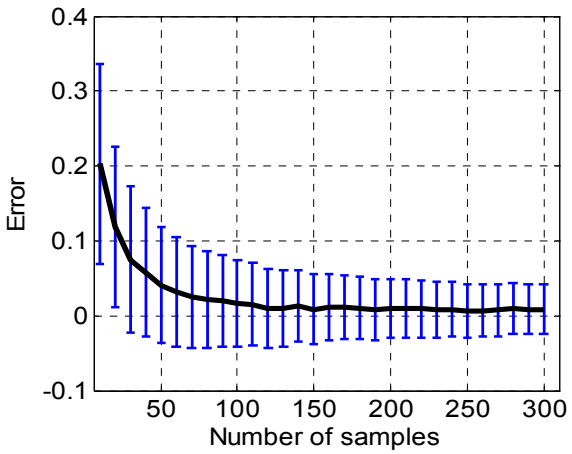


Figure 3.8. The σ_n estimation error using the MAD method with different numbers of neural background noise samples. Noise was randomly generated one hundred times from a noise model. Thus, MAD estimates σ_n one hundred times for each fixed number of noise samples.

To estimate σ_n in real time with efficient use of hardware resources, we introduce a noise Std calculator that consumes a small amount of power and area at the cost of speed, since speed is not a critical limitation in neural signal processing [86]. In this work, the noise Std calculator was optimized in terms of estimation error and time first, and was

proven to be tolerant to the firing rate later.

The noise Std calculator is based on the statistical theory that the probability of Gaussian noise exceeding its σ_n is known to be precisely 0.159. It outputs an estimated denoted as σ_n^{est}. Thus, if a neural signal is compared with σ_n^{est}, a 1-bit digital waveform can be generated with 1 representing each neural sample amplitude that is greater than σ_n^{est}. The duty cycle of the digital waveform is 0.159 if σ_n^{est} is equal to σ_n. Otherwise, the difference between the duty cycle and the value of 0.159 can be used to update σ_n^{est}.

A block diagram of the noise Std calculator to realize this approach is shown in Figure 3.9. A 1-bit digital comparator first compares each neural sample to σ_n^{est} and generates a '1' if the amplitude of neural signal is greater than σ_n^{est} or a '0' if otherwise. A counter then records the number of '1' values in the comparator output bit steam for *M* clock cycles. The output of the counter is then compared with the value of 0.159× *M*, and the difference from the subtractor is fed into a digital integrator to update σ_n^{est} every *M* clock cycles. The loop keeps updating σ_n^{est} every *M* clock cycles until the estimate has converged. The convergence detector is used to judge the convergence.

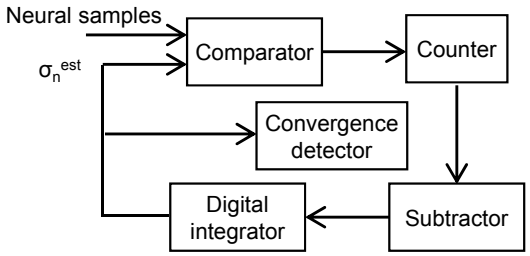
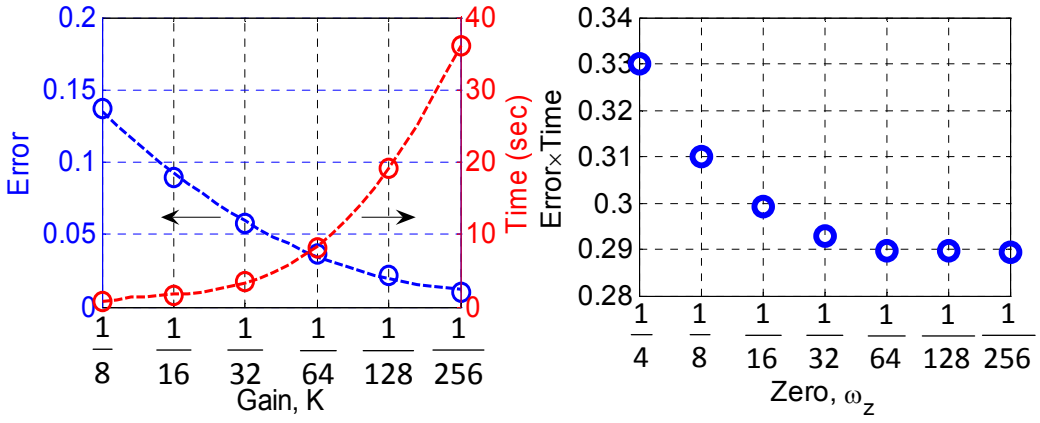


Figure 3.9. Block diagram of the noise Std calculator.

To design the digital integrator, a first-order filter was used to achieve high loop stability with low circuit complexity. The z-domain transfer function of this filter can be expressed as

$$G(z) = K\frac{1-\omega_z z^{-1}}{1-z^{-1}} \tag{3-22}$$

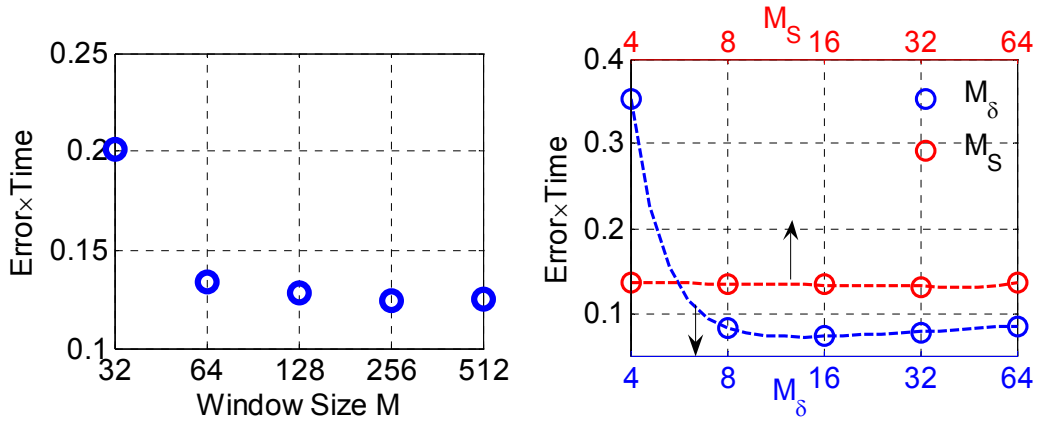
where K is the gain and ω_z is the zero of the integrator. For a time domain circuit implementation, if the filter coefficients K and ω_z could be realized as binary values, it would greatly reduce circuit complexity. To test the impact of using binary coefficients on the estimation error and time, different binary values for filter coefficients were studied. Figure 3.10(a) shows the error and the convergence time for different binary values of K. A large value of K converges quickly but at the expense of high error. To provide a balance between the error and the convergence time, K was chosen to be 1/64Figure 3.10(b) plots the error-time product for different binary values of ω_z. It can be seen that ω_z beyond 1/64 contributes little improvement in performance. To determine the window size M for the counter, Figure 3.10(c) shows the error-time product for different sizes of M. The optimal value of M was set as 256 from Figure 3.10(c), which also provides the best convergence time.



(a) (b)

Figure 3.10. The σ_n estimation error and time for the Std calculator design parameters (a) Gain, (b) zero, (c) M and (d) M_δ and M_S.

Figure 3.10 (cont'd)



(c) (d)

The convergence detector determines when the estimate converges. It performs the following functions step by step:

1) Compare the current estimate $\sigma_n^{est}(m)$ with the estimate produced in the previous cycle $\sigma_n^{est}(m-1)$. Assign a comparison result $\delta(m)$ of +1 when $\sigma_n^{est}(m)>\sigma_n^{est}(m-1)$, -1 when $\sigma_n^{est}(m)<\sigma_n^{est}(m-1)$, and 0 for $\sigma_n^{est}(m)=\sigma_n^{est}(m-1)$.
2) Accumulate M_δ samples of successive comparison results, denoted as $S_\delta = \sum_{m=1}^{M_\delta}\delta(m)$.
3) Check M_S successive samples of S_δ. If all $|S_\delta|\leq 1$, σ_n has converged.

Figure 3.10(d) shows the error-time product for M_δ and M_S. From Figure 3.10(d), M_δ shows best performance when it is equal to 16. Because the performance of M_S is little affected by different values, it was set as 4 to provide the smallest convergence time.

To validate these design choices, the σ_n estimation error of the noise Std calculator was analyzed for different firing rates of spikes. The results plotted in Figure 3.11 show the relative error of the noise Std calculator is proportional to the firing rate. For comparison, Figure 3.11 also shows the estimation error using the MAD method, which

is reported to be robust to the spike firing rate [91]. Because the noise Std calculator counts the number of neural samples greater than σ_n^{est} instead of using the amplitude of neural samples for σ_n estimation like MAD does, the noise Std calculator provides less error than MAD. Thus, we can see that the spike firing rate has less effect on the estimated σ_n for the noise Std calculator.

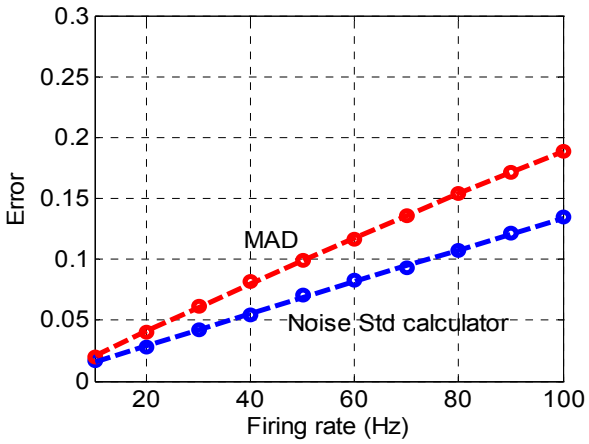


Figure 3.11. σ_n estimation error for different firing rates using MAD and the Std calculator.

3.3.4 Automatic NEO Thresholding for Spike Detection

Bringing all of the functional components described above together allows us to form the complete NEO-based automatic thresholding spike detector shown in Figure 3.12. To mitigate the sensitivity of NEO algorithm to high frequency noise, neural signals are first smoothed with an Exponential filter in (3-16) with α equal to 1/4. The NEO calculator and the second Exponential filter are used to compute $E[\Psi(x(n))]$. In the automatic NEO thresholding (ANT) block, the Noise Std calculator and the Zero-crossing Ω_{rms} calculator evaluate σ_n and Ω_{rms}, respectively, and these outputs feed the Threshold calculator to estimate the threshold value using (3-11). The threshold value Th_Ψ is stored in a Threshold register where it is compared with $E[\Psi(x(n))]$ to provide thresholding for

spike event detection. For the multichannel application, the ANT block can sequentially estimate threshold values for multiple channels. Thus, the threshold value for each channel can be updated every few seconds.

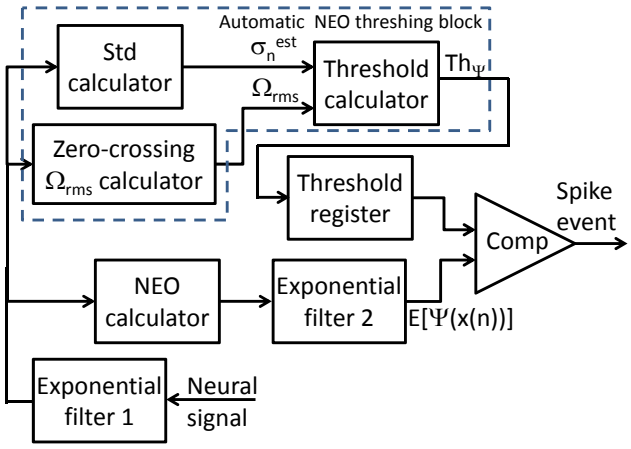


Figure 3.12. Block diagram of NEO-based automatic thresholding spike detector.

3.4 Simulation Results

3.4.1 Datasets

To evaluate the performance of the ANT method for the NEO-based spike detection, synthetic neural signals constructed from the real recordings were used. Synthetic neural signals are used rather than experimentally recorded neural signals because the synthetic signals provide spikes with ground truth, and the SNR and the firing rate of synthetic signals can be varied. Real neural signals recorded from the hippocampus of anesthetized rats were obtained from a public database (crcns.org/data-sets/hc/hc-1/) [44] that has been used to evaluate spike detection and spike sorting performance in several studies [109-111]. The datasets consist of simultaneous intracellular and extracellular recordings. The intracellular spikes fired by the same neuron have high SNR and can be used as indicators of extracellular spikes. Three real datasets (d533101.dat, d14521.001.dat and d12821.001.dat) were selected for the construction of synthetic datasets. One of the

datasets was used as a training dataset to determine the parameters of C_0 and N as discussed in Section 3.2. Because real datasets were acquired at different sampling rates, the extracellular signals were first resampled at 20kHz shown in Figure 3.13 (top). To remove the local field potential, the extracellular signal was bandpass filtered by a sixth order elliptical filter with the low cutoff frequency set to 300 Hz. Because the sampling rate f_s is suggested to be eight times greater than the signal for NEO algorithm [78], the high cutoff frequency was set as $f_s/8$. The bandpass filtered extracellular signal is shown in Figure 3.13 (middle). To obtain the spike template from each dataset, the extracellular spikes indicated from the intracellular spikes shown in Figure 3.13 (bottom) were extracted and aligned. The spike template was obtained by averaging the extracted spikes. The spike template was allocated 64 samples representing an approximately 3 ms spike length. The background noise was modeled as a colored Gaussian random process by an autoregressive model [106] with coefficients based on the spike free noise segments in each real dataset. Finally, the spike template was distributed into background noise using a Poisson firing model with firing rates ranging from 10 to 100 Hz and a refractory period of 3ms. Synthetic neural signals were tested at different SNRs. The SNR was varied from 3 to 6. The SNR in this thesis work is defined as [77]

$$\text{SNR} = \frac{\text{peak to peak amplitude of spike}}{2\times\text{standard deviation of noise}} \tag{3-23}$$

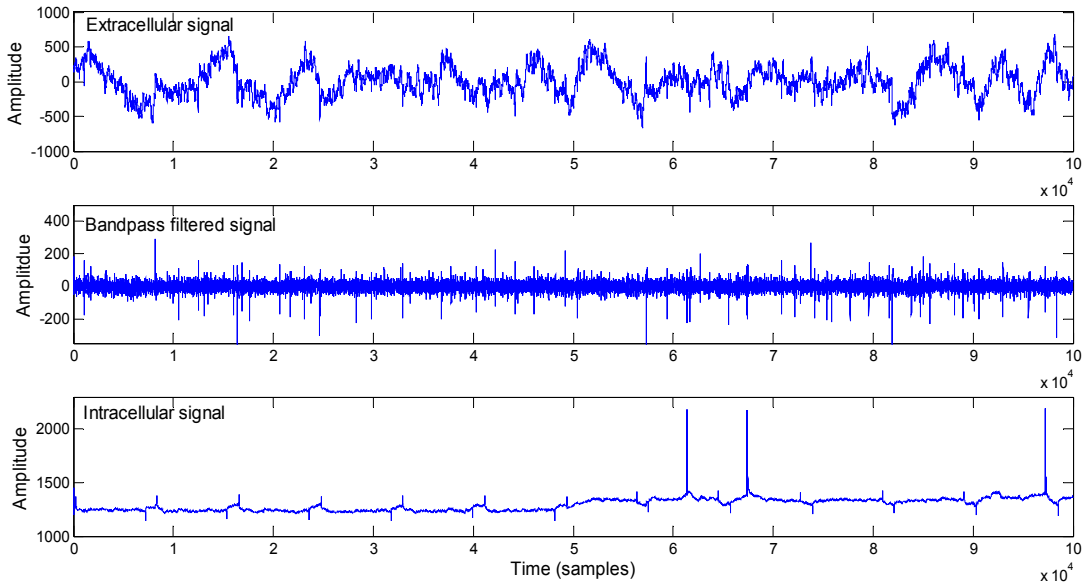


Figure 3.13. Real neural signals from a public database. Top: extracellular signal. Middle: Bandpass filtered extracellular signal from 300Hz to $f_s/8$. Bottom: intracellular signal showing spike locations.

3.4.2 Detection Performance

The spike detection performance was evaluated by measuring detection accuracy defined as

$$Accuracy = \frac{TP}{TP+FN+FP} \tag{3-24}$$

where *TP* is true positive representing the number of correctly detected spikes, *FN* is false negative representing the number of missed spikes and *FP* is false positive representing the number of false spikes due to detecting noise as spikes.

The performance of the ANT-based spike detector was compared with the performance of three different methods: The conventional thresholding (CT) method sets the threshold to a scaling factor *C* of the mean of $E[\Psi(x(n))]$. The scalar values of *C* were chosen to be 3.5, 4, 4.5 and 5 because these values were found to provide the best detection performance. The manual thresholding (MT) method chooses a threshold

manually, so the threshold value resulting in the best detection accuracy was chosen. To further compare the performance of σ_n estimation between the MAD and the Noise Std calculator, the MAD based ANT method (ANT_{MAD}) was also evaluated.

Figure 3.14 shows the detection accuracy of these methods at different SNRs averaged across the firing rate from 10 to 100 Hz. The accuracy of the ANT-based spike detector is above 90% on average and its variance is within 3.2% when the SNR is greater than 4. The ANT-based spike detector is within 2.5% of the MT method when the SNR is greater than 4 and its performance decreases compared to the MT method when the SNR is below 4. The performance of the ANT-based spike detector is slightly (1%) less than the ANT_{MAD} method when the SNR is greater than 4.5, but its performance becomes 5% better compared to the ANT_{MAD} method at low SNR. The reason for this is that the MAD method estimates σ_n with higher value than the Noise Std calculator, and thus the ANT threshold is less than the ANT_{MAD} threshold. When the SNR is high, the ANT method detects some noise as spikes, but, when the SNR is low, the ANT_{MAD} method missed more spikes than the ANT method. The CT method does not provide better performance than the ANT method no matter which scalar value *C* was chosen. The CT method with *C* equal to 4 shows the best performance on average across different SNRs compared to other scalar values. On average, the best CT performance is up to 5% worse than the ANT method. In its worst case, the ANT method is still comparable to the best CT performance when the SNR is greater than 3.5.

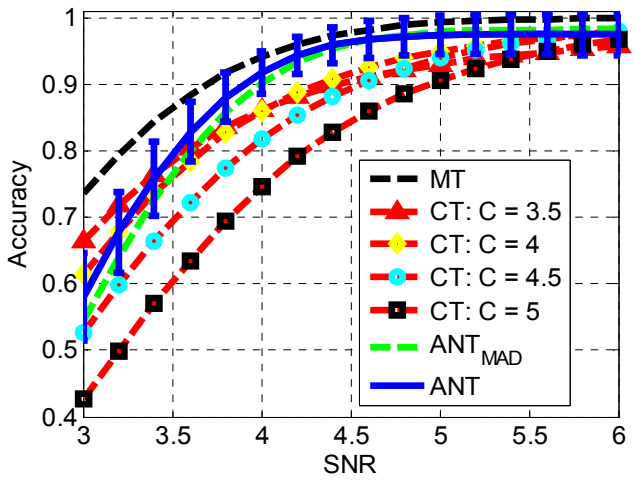


Figure 3.14. Spike detection accuracy against SNR using ANT, MT, CT and ANT_{MAD} methods averaged across the firing rate from 10 to 100 Hz.

To show the robustness of the ANT-based spike detector against the firing rate, Figure 3.15 compares the ANT method to the ANT_{MAD} method and the CT method with different scalar values, all when the SNR is four. The ANT method provides the best performance against the firing rate compared to other methods. The detection accuracy of ANT is almost constant for the firing rate between 30 to 100. It decreases slightly when the firing rate is below 20 because the number of *TP* is reduced but the number of *FP* is still the same when the firing rate is low. The variance of ANT performance is within 3% when the firing rate is below 70 and increases slightly to 4% when it is above 70. The performance of ANT_{MAD} method decreases at high firing rates where it is up to 5% worse than the ANT method. The CT method cannot provide robust performance for different values of *C*. When *C* was set for good performance at high firing rate, the accuracy is at least 10% less than ANT when the firing rate is below 20, and when C was set for good performance at low firing rate, the accuracy is at least 10% less than ANT when the firing rate is above 60. Overall, these results show that the ANT method can determine the threshold in a manner that is more robust to the firing rate than any other known method.

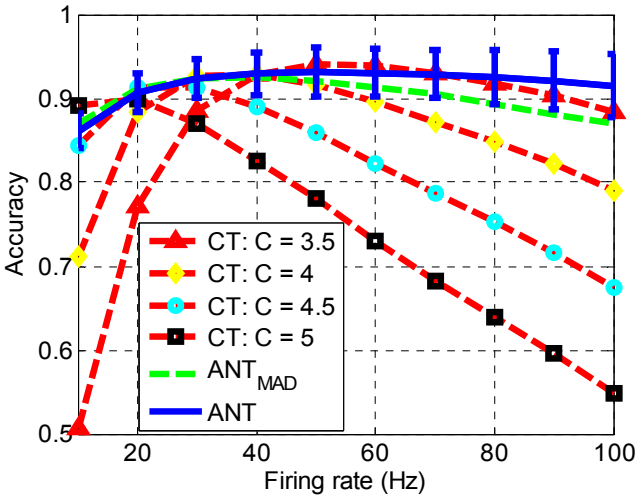
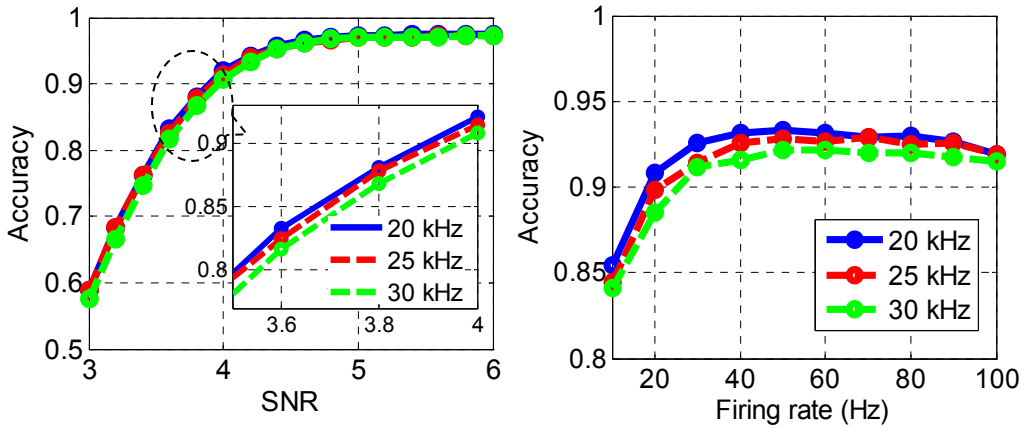


Figure 3.15. Spike detection accuracy of the ANT, CT and ANT_{MAD} methods at different firing rates when the SNR is 4.

Because neural recording systems can sample at higher frequency than 20 kHz, synthetic neural signals with sampling frequencies of 25 kHz and 30 kHz were constructed and tested. The parameter C_0 was determined to be 10 and 10.5 for 25 kHz and 30 kHz, respectively. The parameter N and its corresponding parameter α remain the same as 20 kHz. Figure 3.16 shows the performance of the ANT method for different sampling rates vs. SNR and firing rate. The ANT method performance only decreases slightly when the sampling rate increases. This occurs because, when the sampling rate increases, the high cutoff frequency $f_s/8$ of the bandpass filter also increases. Thus, the threshold value becomes larger to reflect the raise of background noise level, which results in missing more spikes.



(a) (b)

Figure 3.16. Spike detection accuracy of the ANT method for synthetic neural signals with different sampling rates at (a) different SNRs and (b) different firing rates.

3.4.3 Hardware Resource of the ANT Method

The ANT thresholding block shown in Figure 3.12 was designed in Verilog HDL. The data width for this implementation was chosen as 10 bit and the clock frequency was set to 20 kHz. The design of Verilog codes were then mapped to a 130nm CMOS technology using Synopsis software. Its power and area were estimated by Synopsys (post synthesis) to be 50 nW and 0.021 mm^2. Although the CT method consumes negligible power and area, its detection performance is intolerant to the firing rate as shown in Figure 3.15. To provide robust performance against the firing rate, the scalar C has to be recalibrated periodically. To the best of our knowledge, an automatic recalibration process for the high channel count has not been developed. Compared to the ANT_{MAD} method, the ANT method provides more accurate estimation of background noise level and lower hardware resource. The Noise Std calculator only occupies around 6300 μm^2 while the *median* operator needed for the ANT_{MAD} method would occupy 0.21 mm^2 in the same technology assuming sorting 100 neural data [112].

For a multichannel system, the ANT-based spike detector can be used with multiple channels sequentially by scaling the clock frequency, which would also scale power consumption proportionally. The ANT method estimates a threshold value in approximately five second. Thus, assuming the neural background noise will vary every five minutes [113], the ANT block can support up to 64 channels to sequentially update threshold values for each channel roughly every five minutes. In this manner, the power and area consumption for 1000 channels would be less than 1 μW and 0.4 mm^2, respectively. Overall, the ANT method provides the best trade-off between detection performance and hardware cost. In addition, its power and area consumption makes it scalable for high-channel-count application.

3.5 Conclusion

In this chapter, we presented a new method to automatically estimate the threshold value for NEO-based spike detection in real time. The threshold value is demonstrated to be associated with the standard deviation and the RMS frequency of neural background noise. Hardware efficient methodologies were designed to calculate these signal parameters such that the threshold value is adaptive to the noise level and robust to the firing rate. The automatic and robust NEO thresholding method allows for high spike detection performance online without user interventions. The low hardware cost of the new method demonstrates its feasibility for use in high-channel-count implantable neural recording microsystems for future BMI applications.

Chapter 4 Hardware Efficient Frequency Band Separability based Neural Spike Feature Extraction

When a spike is detected as described in Chapter 3, the next step is extracting useful information from the spike in a process called feature extraction. The results of feature extraction are spike features that represent a spike with significantly fewer data points than the raw recorded neural signal. The similarity of spike features is measured by the distance in the feature space. To improve the performance of spike sorting, the distance of spikes fired by the same neuron should be as small as possible in the spike feature space.

Thus, feature extraction has a significant impact on both the sorting accuracy and the hardware complexity. Recent methods such as Integral transform (IT) [114], zero crossing features (ZCF) [94], first and second derivative extrema (FSDE) [97] and discrete derivative extrema ($DD|_2$-Extrema) [96] represent each spike with no more than four features that can be extracted with simple computation . However, because the background noise of neural signals varies significantly every few minutes [113], the effect of this noise variation on features extracted using these methods, and hence spike sorting performance, has not been well explored. In this chapter, a frequency band separability method is introduced for spike feature extraction to analyze the spike and noise information within different frequency bands. Based on this method, and taking into account the hardware design complexity, a weighted haar DWT implementation is described to extract features that maximize the separability between spike classes. The new feature extraction method was tested at different signal-to-noise ratios using synthesized datasets consisting of considerable and various spike shapes extracted from the real neural recordings. The results show that the new method has 3-10% better spike

sorting performance than other hardware-efficient methods while consuming comparable hardware resources.

4.1 Training Datasets

In order to evaluate spike sorting performance of any feature extraction method, neural data with true spike times and classes are needed. The common approach is to use synthetic datasets constructed from real neural recordings [95, 106]. The datasets used in chapter 3 only provides limited spike shapes. To analyze various spike shapes, real neural signals recorded from the hippocampus of rats were used to construct synthetic neural datasets (crcns.org/data-sets/hc/hc-2/) [115]. The real datasets include the broadband raw data from simultaneous multichannel recordings along with the spike times and classes determined by the authors' offline spike sorting method. The recording duration was over 17 minutes at a sampling rate of 20 kHz. A sixth order elliptical filter with bandwidth from 300 to 3000 Hz was used to remove the local field potential from the raw data of each recording channel. Using the provided spike times and classes, spike templates were generated from the bandpass filtered data for spike classes with firing rates greater than 0.5 Hz. Figure 4.1 shows the spike template waveforms obtained from six different channels. Channels including two, three, and four spike classes were selected to study the sorting accuracy among different number of spike classes. In each channel, data segments with amplitude less than a certain value were extracted to create noise models. These spike templates and noise models were used as training datasets to analyze spike feature extraction statistics, as described in the next section. To synthesize neural signals, the background noise was modeled as a colored Gaussian random process by an autoregressive-moving-average model [87, 106]. To ensure the relationship between

spikes and noise match real recording conditions, spike templates and the noise model from the same channel were used to construct each dataset. Finally, a Poisson firing model with a firing rate of 50 Hz was used in each dataset with SNR varying from 3 to 6, where SNR is defined the same as in chapter 3.

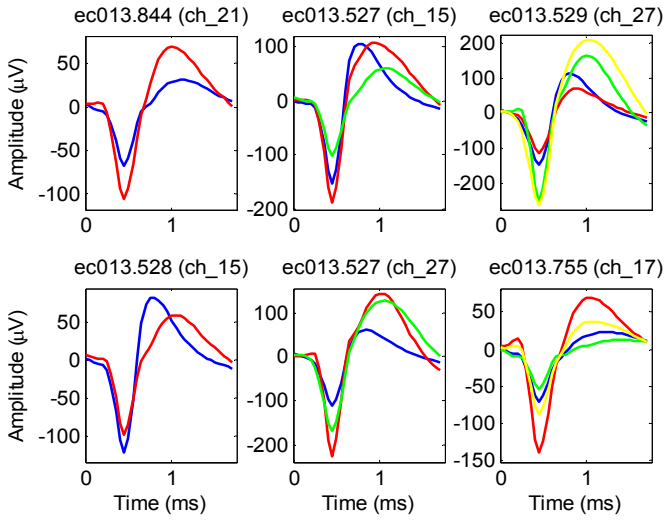


Figure 4.1. Spike templates extracted from six different neural recording channels. Left column contains two spike classes, middle three, right four. These spike templates are used as training datasets to analyze parameters for feature extraction. The title of each subplot describes the source file from which spike templates are extracted. For example, ec013.844 (ch_21) means the data file name is ec013.844 and the channel number is 21.

4.2 Spike Features

Studies have shown that neural spikes decay as the recording distance from a neuron increases and that high frequency components attenuate with distance more than low frequency components [116, 117]. Thus, spikes recorded from different neurons exhibit different frequency component magnitudes because of their relative position to the recording electrode [43]. As a result, the energy and amplitude of spikes are sensitive to their relative positions to the electrode and can be used to discriminate spikes recorded on a single channel. As shown in Figure 4.1, spikes obviously exhibit different negative and positive peaks. Consequently, early research used spike peaks as features [92]. However,

the peaks extracted from raw spikes are vulnerable to noise. To mitigate this noise effect, raw spikes can be filtered into a frequency band where spike energy dominates over noise energy. It has been shown that the spike spectrum has small magnitudes at low frequencies and falls off monotonically at high frequencies [118]. Therefore, highpass and lowpass filters can be deployed to de-emphasize neural signal energy in low and high frequency bands, respectively, where noise can dominate spikes. Figure 4.2 shows the relationship between the filtered spike peak and the filtered spike energy extracted from high and low frequency bands with different cutoff frequencies. It can be observed that the filtered spike peak is highly correlated to the spike energy. Thus, the peaks of filtered spike extracted from a given frequency band can be used as features to reflect the spikes' energy in that frequency band. In recent research, IT and ZCF extracted low frequency spike peaks while FSDE and $DD|_2$-Extrema extracted high frequency spike peaks.

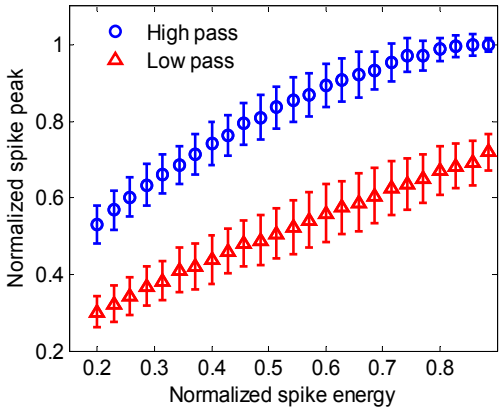


Figure 4.2. The relationship between the filtered spike peak and the filtered spike energy. Spikes are filtered using either highpass or lowpass filters with the cutoff frequency between 300 and 3000 Hz. The errorbar reflects the variance of all the spike templates in Figure 4.2.

4.3 Frequency Band Separability Analysis

In order to achieve high sorting accuracy, spike features should have two properties: 1) they must be informative such that they maximize the Euclidean between-class distance

in the feature space; 2) they must be immune to noise such that they minimize the within-class variance in the feature space. The between-class distance represents how well noise-free spikes fired by different neurons can be separated from each other, while the within-class variance can represent how well spikes fired by the same neuron cluster together in the feature space. It has been shown that classes are well separated when the ratio of between-class distance to within-class variance is maximized for linear discriminant analysis [119]. Thus, the feature extraction method should take both properties into consideration.

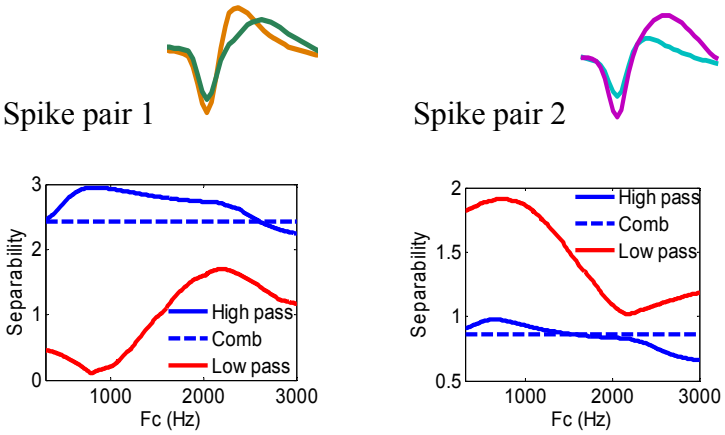
As presented above, filtered spike peaks extracted from a frequency band are informative. For features to be insensitive to noise, they need to be extracted from a specific frequency band such that the ratio of between-class distance (BCD) to within-class variance (WCV) is maximized compared to the same ratio for raw spike peaks. Thus, we propose a *Separability* metric defined as

$$\begin{aligned} Separability &= \left(\frac{BCD}{WCV}\right)_{filtered} / \left(\frac{BCD}{WCV}\right)_{non-filtered} \\ \left(\frac{BCD}{WCV}\right)_{non-filtered} &= \frac{Peak(S_1, S_2)}{Std(noise)} \\ \left(\frac{BCD}{WCV}\right)_{filtered} &= \frac{Peak(S_1 \otimes h(n), S_2 \otimes h(n))}{Std(noise \otimes h(n))} \end{aligned} \tag{4-1}$$

where S_1 and S_2 represent two spike templates, $h(n)$ is the finite impulse response of lowpass or highpass filter to emphasize either low or high frequency band information, *Std* means standard deviation, $\otimes$ is convolution and $Peak(S_1, S_2)$ calculates peak difference between two spikes. $\frac{BCD}{WCV}$ can be considered as in analogy to the SNR used to describe the quality of signals. To improve the quality of a signal, the output SNR should be larger than its input SNR after signal processing. Similarly, the *Separability* metric can be used to quantify the improvement of a feature extraction method compared to a naïve

method of extracting features directly from raw spikes without any signal processing.

By virtue of the *Separability* metric, one can determine whether any two spikes can be better separated from the low frequency or high frequency band. A lowpass filter can be used to separate spikes in the low frequency band while a highpass, or a comb filter generated from the highpass filter [-1 1], can be used to separate spikes in the high frequency band. The DD$|_2$-Extrema method is based on this comb filter with specific multiple passbands. Figure 4.3 shows an example of the *Separability* metric for two spikes pairs. It can be observed that spike pair 1 is better separated using a highpass filter or a comb filter while spike pair 2 is better separated using a lowpass filter.

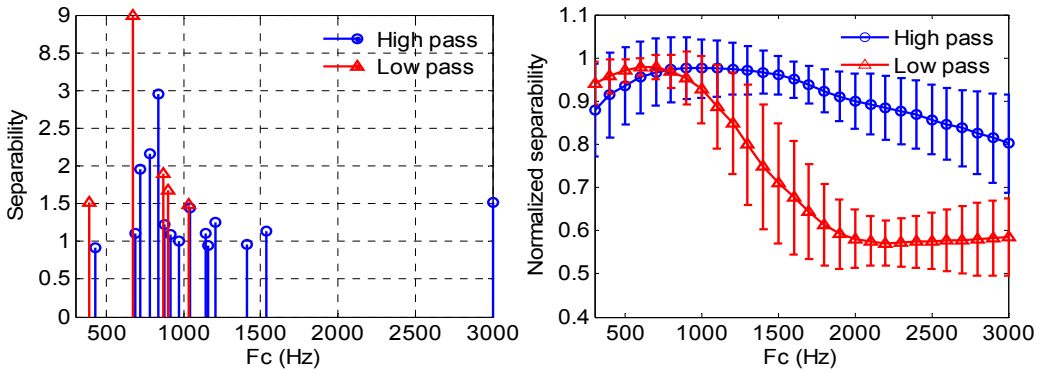


(a. Spike pair 1) (b. Spike pair 2)

Figure 4.3. An example of *Separability* analysis vs. cutoff frequency (Fc) for two spike pairs. Results illustrate spikes can be better separated in (a) the high frequency band using either a highpass or a comb filter or (b) the low frequency band using a lowpass filter. The comb filter represents the DD$|_2$-Extrema method.

The *Separability* performance of the highpass and lowpass filters is associated with the filter's cutoff frequency (F_c). To determine the best cutoff frequency for lowpass and highpass filters, the *Separability* vs. cutoff frequency was calculated for both filters using every possible spike template pair from the channels shown in Figure 4.1. Figure 4.4(a) shows the cutoff frequency of maximum *Separability* using the filter (highpass or

that best discriminates each spike template pair. 25% of spike pairs were found to be better separated using lowpass filter while 75% were better separated using highpass filter. *Separability* greater than 1 represents an improvement of feature extraction compared to the naïve method of extracting features directly from raw spikes. Thus, Figure 4.4(a) indicates that nearly 80% of spike pairs can be better discriminated by selecting the best cutoff frequency of the appropriate filter for each spike pair. The normalized *Separability* average and variation across possible pairings is plotted in Figure 4.4(b), where the *Separability* of each spike pair was normalized to its maximum value (across *Fc*) to eliminate the bias contributed by different maximum *Separability* values. We can see that the choice of F_c for the highpass filter has small effect on the *Separability* but is critical for the lowpass filter. The *Separability* is maximum around 700 Hz for the lowpass filter and 1000 Hz for the highpass filter.



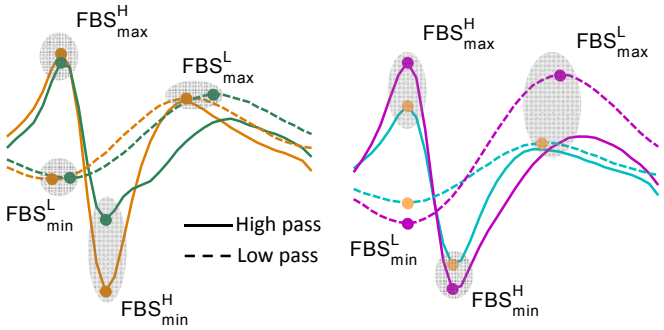
(a) (b)

Figure 4.4. (a) Cutoff frequency of maximum *Separability* for each spike template pair. (b) The averaged normalized *Separability* calculated based on the training datasets to determine the best cutoff frequency for highpass and lowpass filters.

4.4 Frequency Band Separability Features

As presented in section 4.3, the relative position of a neuron to a recording electrode affects spike energy and is reflected in the peaks of frequency-filtered spikes. The

frequency band analysis shown in Figure 4.4(a) validates the design decision to select both low and high frequency bands for feature extraction. The analysis shown in Figure 4.4(b) permits identifying the frequency bands in which the differences between filtered spike peaks are best distinguished. Thus, the positive (max) and negative (min) filtered spike peaks from low and high frequency bands provide a new feature set that we will define as *frequency band separability* (FBS) features. Specifically, FBS features are defined as peaks from the low frequency band (denoted as FBS^{L}_{max} and FBS^{L}_{min}) and peaks from the high frequency band (denoted as FBS^{H}_{max} and FBS^{H}_{min}). Figure 4.5 illustrates FBS features for two spike pairs shown in Figure 4.3 after highpass and lowpass filtering. Spike pair 1 is best separated using the FBS^{H}_{min} feature while the spike pair 2 is best separated using the FBS^{L}_{max} feature. This observation is consistent with the analysis shown in Figure 4.3.



(a. Spike pair 1) (b. Spike pair 2)

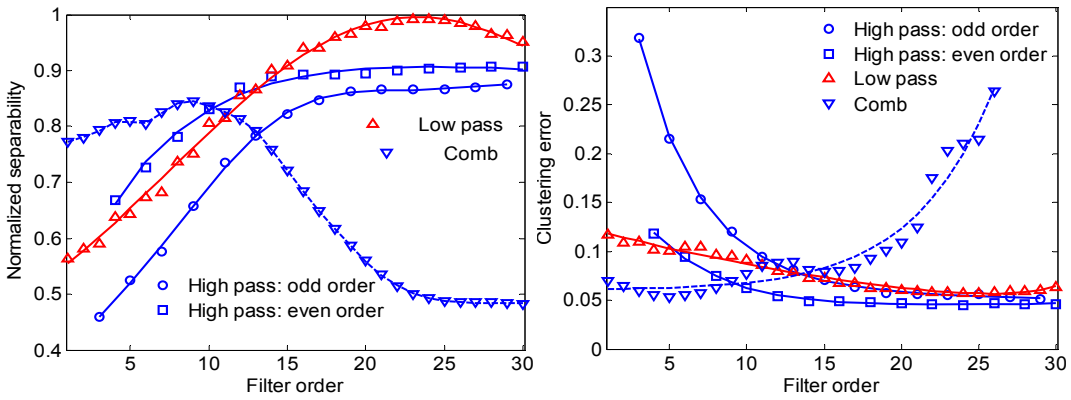
Figure 4.5. An illustration of FBS features for the Fig. 3 spike pairs after applying lowpass and highpass filters. The solid lines represent highpass filtered spikes and the dashed lines represent lowpass filtered spikes. The dots are the extracted FBS features.

4.5 Hardware Efficient Implementation of FBS Features with Haar DWT

4.5.1 Filter Design Complexity Analysis

Filter order is an important factor in filter design, and the impact of this choice on

Separability was studied. Because filter order can also impact sorting performance through clustering error, this relationship was analyzed for K-means clustering with neural signal SNR set to five. The results in Figure 4.6 show both lowpass and highpass filters provide better performance as the filter order increases. The comb filter performance decreases for filter order greater than 10. Thus, to extract FBS features from the high frequency band, the comb filter is preferred over a highpass filter due to hardware-efficient design. To extract FBS features from the low frequency band, the order of the lowpass filter should be larger than 20, but this would require an amount of memory that is unacceptable for a neural implant. Thus, a hardware efficient alternative implementation must be explored.



(a) (b)

Figure 4.6. (a) The normalized *Separability* and (b) K-means clustering error with SNR = 5 at different filter orders for highpass and lowpass and comb filters.

The *Separability* analysis suggests that the lowpass filter cutoff frequency should be around 700 Hz. To extract features from such a low frequency band without excessive memory usage, the DWT method can be utilized to provide fine frequency resolution in the low frequency band. At each level, DWT convolves the signal from the previous level with a half-band lowpass filter. By downsampling every other sample after convolution,

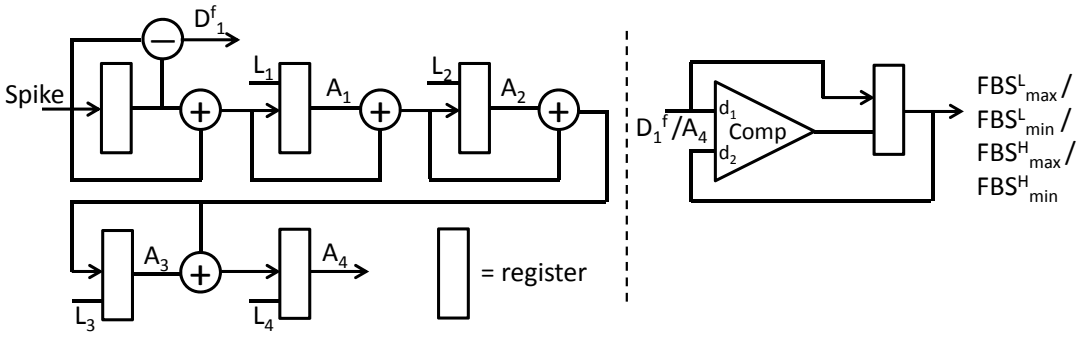
the same half-band lowpass filter can be used iteratively to scale down the low frequency bandwidth by two at each level. Because the desired cutoff frequency (700 Hz) is less than $\frac{1}{2^3}$ of the 10 kHz sampling rate, the 4th level DWT approximation coefficients can be used to extract low frequency band FBS features. This feature extraction procedure meets the lowpass filtering goals with hardware efficiency.

It is shown that the first order comb filter is well suited to extract FBS features from the high frequency band with low memory usage. Because the 1st level detail coefficients of the haar DWT method provide filtering results consistent with a first order comb filter, haar DWT can be leveraged to extract FBS features from both high and low frequency bands (denoted as FBS_{HT}): positive and negative peaks from the 1st level detail coefficients and from the 4th level approximation coefficients. At each level, only one register is required to calculate both approximation and detail coefficients. Thus, only four registers in total are needed for four level decompositions compared to more than 20 registers for a direct filter implementation. Therefore, FBS_{HT} enables a hardware-efficient design for spike feature extraction.

4.5.2 Hardware Design for FBS_{HT} Feature Extraction

To extract the four FBS_{HT} features (FBS^H_{max}, FBS^H_{min}, FBS^L_{max}, FBS^L_{min}) in hardware, two blocks are demanded: 1) a haar DWT block that magnifies spike information both in high and low frequency bands, 2) and a peak detector block that determines the feature. Figure 4.7(a) shows the structure of the haar DWT block, where A_i represents the ith level lowpass filtered approximation coefficient output and D^f_1 represents the 1st level highpass filtered detail coefficient output without decimation. The output D^f_1 is not decimated while having little effect on the FBS^H_{max} and FBS^H_{min}. The control signal L_i downsamples

A_i by two at the i^{th} level as illustrated in Figure 4.8. Figure 4.7(b) shows the structure of peak detector. This block is connected to either A_4 or D^f_1 output and can be configured to extract either positive or negative peaks. When it is used to extract positive peaks, the peak register is initially loaded with a minimum value and its value is updated when $d_1 > d_2$ at the comparator input. When the peak detector is used to extract negative peak, the register is initially loaded with a maximum value and its value is updated when $d_1 < d_2$. Totally, four peak detectors are required to extract all the FBS_{HT} features.



(a) (b)

Figure 4.7. (a) Structure of haar DWT. (b) Structure of peak detector.

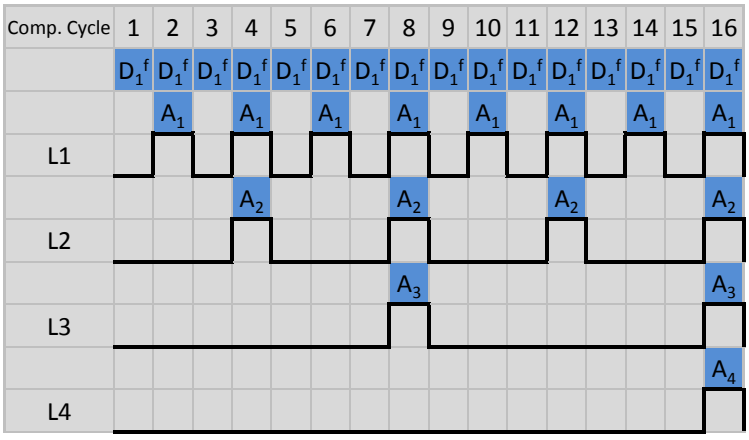


Figure 4.8. Computation cycles of haar DWT coefficients at each decomposition level.

4.6 Feature Scaling for Haar DWT based FBS Features

The FBS_{HT} features can be categorized into two feature sets, namely approximation (low frequency) and detail (high frequency) features. Figure 4.9 shows the normalized spectrum of a neural spike along with the frequency response for the haar DWT 1st level detail and the 4th level approximation. The spike spectrum dominates at the low frequency band, around 1000 Hz. It can be seen that the approximation level lowpass filter captures most of the spike energy in its passband while the detail level highpass filter preserves the high frequency band energy where the spike energy attenuates significantly. the dynamic range for these two feature sets will not be in the same scale; the approximation (low frequency) features exhibit larger range than the detail (high frequency) features. It has been shown that, for a Euclidean distance based clustering method such as K-means, results can be greatly affected by the differences in scale among features [120]. To utilize all of these features for clustering, it is important to scale both feature sets into the same dynamic range.

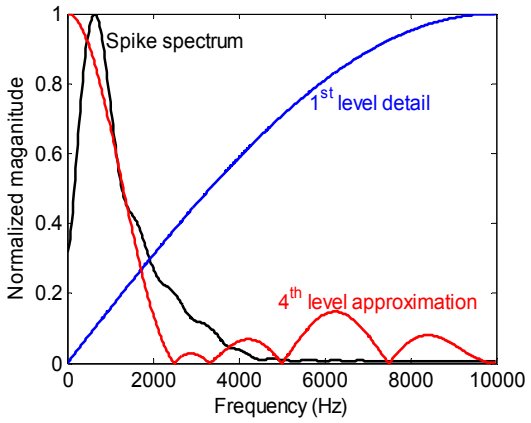


Figure 4.9. Averaged spike spectrum and the frequency response of the 1st level detail and the 4th level approximation haar DWT.

To estimate the variability of both features, the spike Euclidean distance between any two spike templates in each dataset were calculated in the approximation ($Dist_A$) and the

detail ($Dist_D$) feature spaces as:Equation

$$(Dist)_A = \| FBS_{HT}^{L}(S_1) - FBS_{HT}^{L}(S_2) \| + N_{fet} \times Std(noise_L)$$
$$(Dist)_D = \| FBS_{HT}^{H}(S_1) - FBS_{HT}^{H}(S_2) \| + N_{fet} \times Std(noise_H) \quad (4\text{-}2)$$

where FBS_{HT}^{L} and $FBS_{HT}^{H}$are features extracted from the approximation and detail levels, respectively, and $Std(noise_L)$ and $Std(noise_H)$ are standard deviations of noise in the the approximation and detail levels, respectively, and N_{fet} is the number of features in each feature set. Because the detail level contain less spike energy than the approximation level, FBS_{HT}^{H} needs to be weighted more than FBS_{HT}^{L}. The weighting factor for FBS_{HT}^{H}can be calculated asEquation

$$\overline{Weight} = \frac{Dist_A}{Dist_D} \quad (4\text{-}3)$$

where $\overline{Weight}$ represents the average weight among different datasets. From experimental analysis, $\overline{Weight}$ was set to be seven based on the training datasets at 20 kHz sampling rate.

4.7 Comparison to Haar DWT Features based on Lilliefors Test

The haar DWT method described in [91] selects ten wavelet coefficients as features based on Lilliefors test. The Lilliefors test selects coefficients based on the discrepancy between the cumulative distribution function of a coefficient and a normal distribution with the same mean and variance of that coefficient. Lilliefors test is suitable for offline training and has been shown to require a huge amount of computation and memory [95]. Lilliefors-based haar DWT feature extraction requires spikes to be aligned to a reference point before feature extraction. Thus, the spike alignment has a critical impact on the wavelet coefficients selected as features and hence the sorting performance.

The presented weighted FBS_{HT} method based on *Separability* analysis only extracts peaks from wavelet coefficients at specific decomposition levels and therefore does not demand any offline training. Furthermore, the extraction of peaks has no preceding requirement for spikes aligned to a reference point. As a result, the FBS_{HT} features are insensitive to spike misalignment. Additionally, the weighted FBS_{HT} method weights wavelet coefficients from different decomposition levels by analyzing the spike energy at different frequency bands. As will be shown in Section IV, this weighting step significantly influences the spike clustering performance.

In comparison, the computation complexity of FBS_{HT} features is much smaller than the Lilliefors test based DWT features. The number of FBS_{HT} features is six less than the Lilliefors test based DWT features, which reduces hardware resources for spike classification in the next step.

4.8 Results and Discussion

4.8.1 Testing Datasets

To evaluate the performance of the weighted FBS_{HT} (W-FBS_{HT}) method on various spike shapes, neural recordings from multiple channels were processed to extract spike templates as shown in Figure 4.10. The spike extraction procedures were the same as described in Section 4.1. 36 datasets were generated, out of which 12 contained two spike classes, 12 contained three and 12 contained four. Finally, the six training datasets in Figure 4.1 and the 36 testing datasets were all used to generate synthesized neural signals with the firing rate of 50 Hz and SNR from 3 to 6.

4.8.2 Clustering Performance

K-means clustering was used to separate spike features and demonstrate the

effectiveness of the *W-FBS*$_{HT}$ features. Other feature extraction methods were also evaluated on the same datasets for comparison including principal component analysis (PCA), FSDE and DD|$_2$-Extrema. The first four principal components were utilized such that the number of features was consistent in all methods.

Figure 4.11(a) shows the clustering performance on spikes with two classes. On average, *W-FBS*$_{HT}$ shows 3% to 5% better performance than DD|$_2$-Extrema and around better performance than FSDE. The variance of the clustering error over difference datasets for *W-FBS*$_{HT}$ is also shown. The variance increases from 2% to 5% as SNR decreases. It can be seen that, even in the worst case, *W-FBS*$_{HT}$ is still comparable to DD|$_2$-Extrema and is much better than FSDE. Figure 4.11(b) and Figure 4.11(c) show examples of spike classes are better separated in low frequency and high frequency bands, respectively. Spikes were projected into two dimensional approximation and detail feature

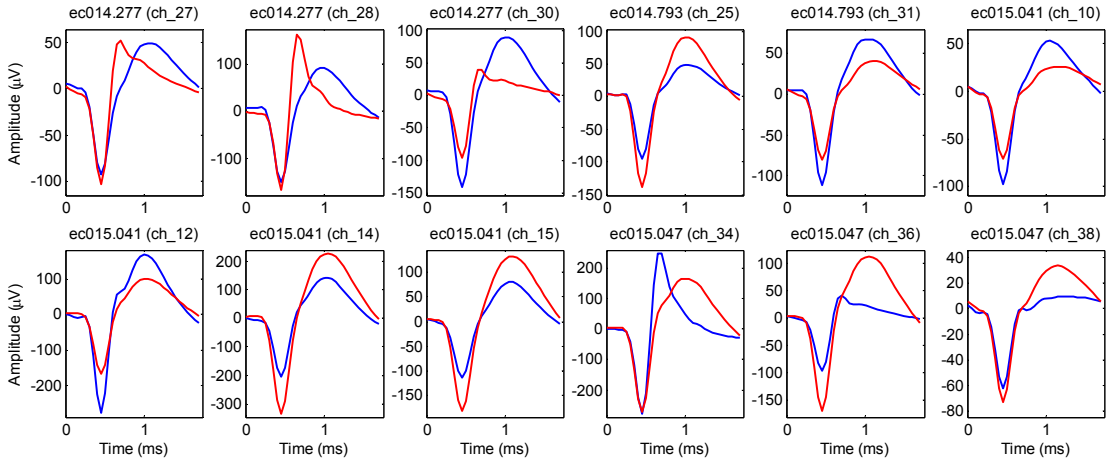
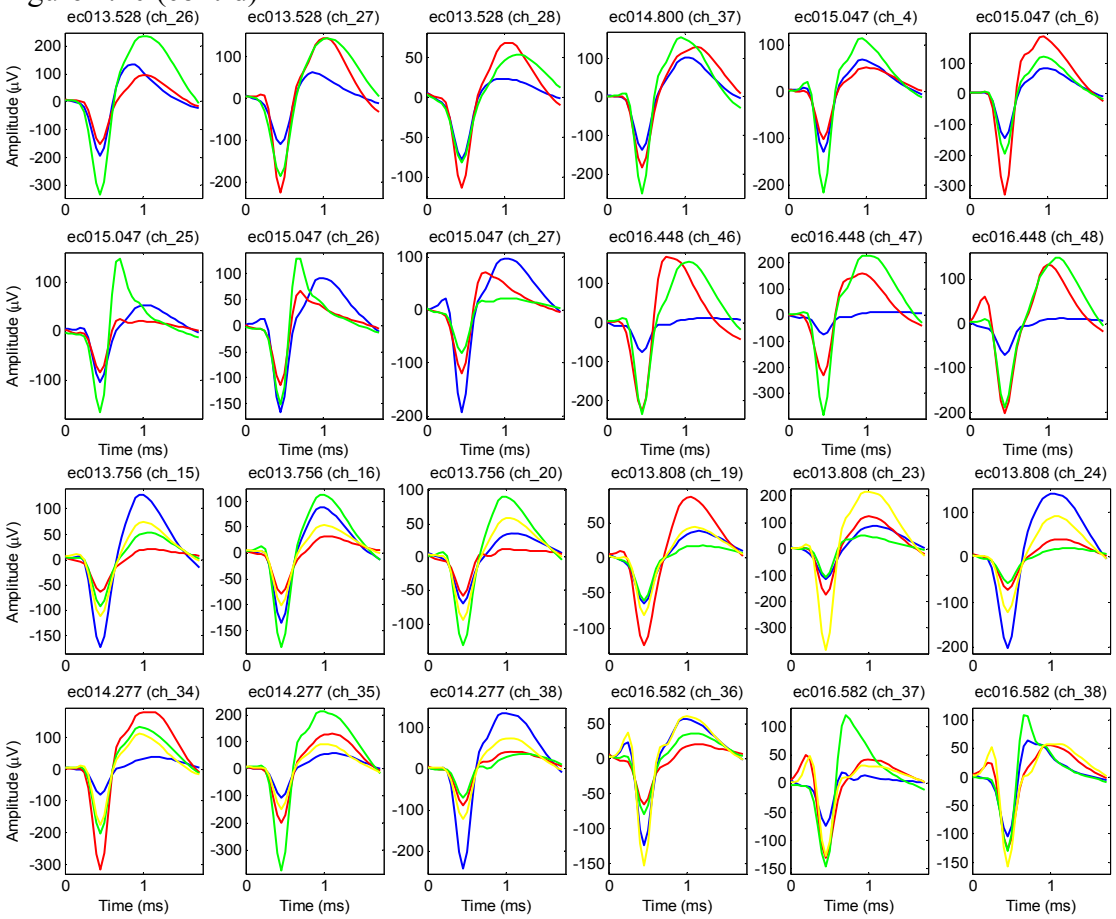
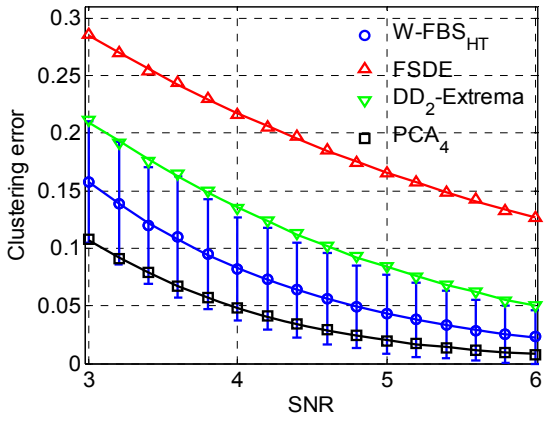


Figure 4.10. Spike templates extracted from 36 different neural recording channels used as testing datasets. The first two rows contain two spike classes, the middle two rows three spike classes, the last two rows four spike classes.

Figure 4.10 (cont'd)

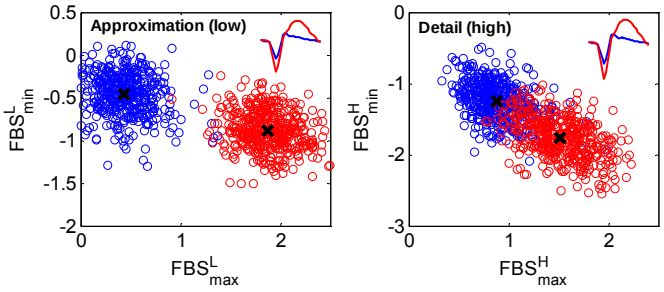




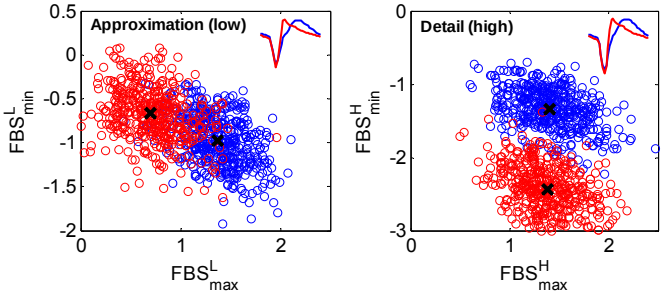
(a)

Figure 4.11. (a) Clustering errors at different SNRs with two spike classes for *W-FBS$_{HT}$*, FSDE, DD|$_2$-Extrema and PCA methods. (b) An example of spikes that can be better separated using the approximation features. (c) An example of spikes that can be better separated using the detail features.

Figure 4.11 (cont'd)



(b)



(c)

spaces separately. It is apparent that *W-FBS*$_{HT}$ can better distinguish spikes by using both high and low frequency bands.

Figure 4.12(a) evaluates the clustering performance in the case of three spike classes. *W-FBS*$_{HT}$ provides 5% and 10% better performance than DD|$_2$-Extrema and FSDE, respectively. *W-FBS*$_{HT}$ also shows worst case performance comparable to the DD|$_2$-Extrema average. Figure 4.12(b) plots an example of the distribution of three spike classes in the feature space. Intuitively, the red spike class can be distinguished in the approximation feature space while the blue and green spikes can be separated in the detail feature space. In fact, these three spikes are well separated in the entire four-dimensional space. This examples shows that, when three spikes are presented for clustering, features from low and high frequency bands can both be used for distinction. Thus, *W-FBS*$_{HT}$

features provide better separabiltiy than DD|$_2$-Extrema and FSDE features that emphasize high frequency information.

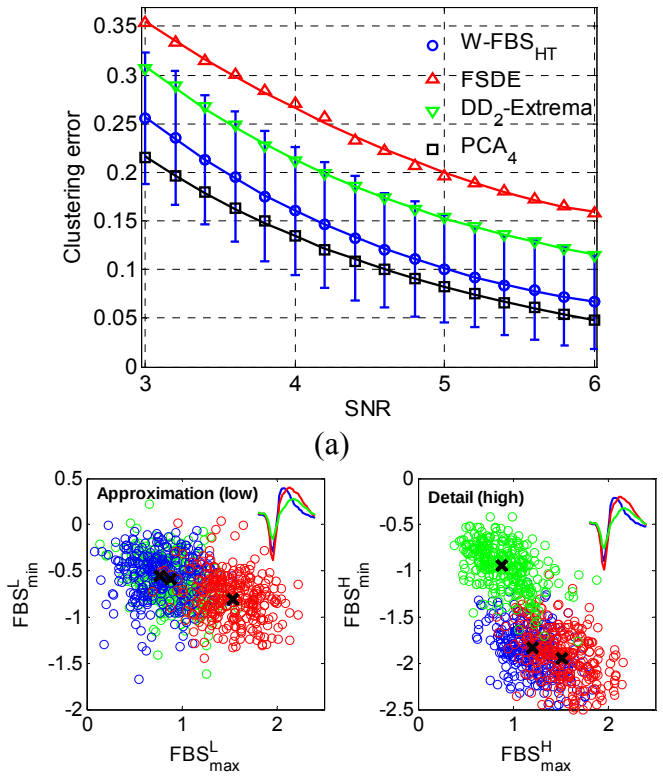
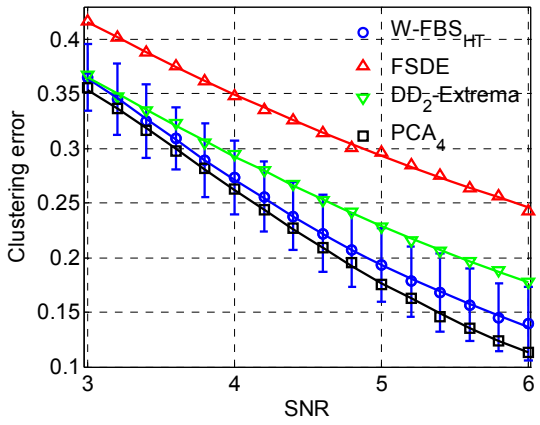
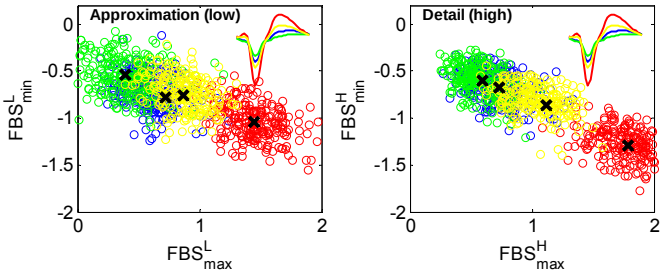


Figure 4.12. (a) Clustering errors at different SNRs with three spike classes for *W-FBS$_{HT}$*, FSDE, DD|$_2$-Extrema and PCA methods. (b) An example of three spike classes projected in the *W-FBS$_{HT}$* feature space.

Figure 4.13(a) shows the clustering performance when four spike classes are presented. Again, *W-FBS$_{HT}$* significantly outperforms DD|$_2$-Extrema and FSDE except for low SNR where *W-FBS$_{HT}$* degrades to the level of DD|$_2$-Extrema and even the gold standard PCA method exhibits a similar poor performance. As shown in Figure 4.13(b), *W-FBS$_{HT}$* cannot contribute to enough allow accurate separation among spikes in the feature space.



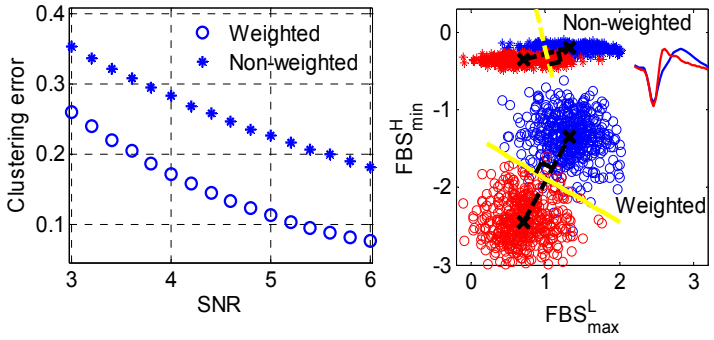
(a)



(b)

Figure 4.13. (a) Clustering errors at different SNRs with four spike classes for *W-FBS$_{HT}$*, FSDE, DD|$_2$-Extrema and PCA methods. (b) An example of four spike classes projected in the *W-FBS$_{HT}$* feature space.

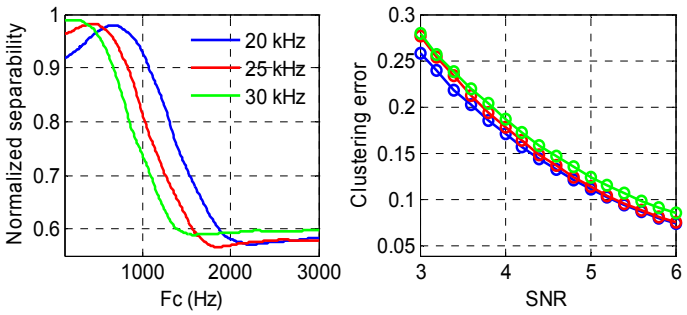
To illustrate the performance difference between weighted features and non-weighted features, Figure 4.14(a) plots the clustering performance for these two techniques. On average, weighted features shows 10% performance improvement compared to non-weighted. Figure 4.14(b) illustrates the advantage of *W-FBS$_{HT}$* graphically by projecting the two spike classes from Figure 4.11(c) into a feature space represented by one approximation and one detail features. Without scaling the dynamic range of the detail feature to match the approximation feature, the decision boundary is primarily determined by the approximation *FBS$_{HT}$* feature with its large dynamic range. Thus, scaling both feature sets to the same dynamic range provides significant performance improvement.



(a) (b)

Figure 4.14. . (a) Clustering errors for weighted and non-weighted FBS_{HT}. (b) An example of two spike classes projected in the weighted FBS_{HT} and non-weighted FBS_{HT} feature space using one approximation feature and one detail feature. The 'x' symbol represents the cluster centroid. The yellow solid line represents the decision boundary for weighted features. The yellow dashed line represents the decision boundary for non-weighted features.

Because neural recording systems can sample at a frequency higher than 20 kHz, synthetic neural signals with the sampling rate of 25 kHz and 30 kHz were constructed and analyzed. As the sampling rate increases, the decomposition level to extract approximation features must be changed accordingly. Figure 4.15(a) shows the *Separability* analysis for low cutoff frequency at different sampling rates. The maximum separability is around 400Hz and 300Hz for 25 kHz and 30 kHz, respectively. Correspondingly, the decomposition level should be five and six, respectively. However, because DWT downsamples spikes by two at each level, it is impossible to decompose spikes into the 6^{th} level. As a result, spikes were decomposed into the 5^{th} level at 30 kHz. Figure 4.15(b) shows the clustering performance at different sampling rates. At 25 kHz, *W-*FBS_{HT}'s performance is close to the one at 20 kHz at high SNR and decreases by 2% at most as SNR decreases. At 30 kHz, its performance is 2% worse than the one at 20 kHz on average because one more decomposition level is preferred to reach the ideal cutoff frequency.



(a) (b)

Figure 4.15. (a) Normalized separability for the lowpass filter at 20 kHz, 25 kHz and 30 kHz. (b) Clustering errors for datasets with sampling rate of 20 kHz, 25 kHz and 30 kHz.

4.8.3 Analysis of Hardware resource

Implantable application demands limit computation and memory resources. Computation complexity is usually expressed by the number of additions required to process each spike. Assuming each spike consists of N_{spk} samples, FBS_{HT} requires $\frac{N_{spk}}{2}$ computations for the 1st level detail features and $\frac{N_{spk}}{2}+\frac{N_{spk}}{4}+\frac{N_{spk}}{8}+\frac{N_{spk}}{16}$ computations for the 4th level approximation features. Totally, the computation complexity of FBS_{HT} is slightly less than $\frac{3}{2}N_{spk}$. The computation complexity of FSDE and DD|$_2$-Extrema are $2N_{spk}-3$ and $2N_{spk}-10$, respectively. Thus, all of these feature extraction methods are comparable to each other and computationally efficient for hardware implementation. Because feature extraction operates at the spike firing rate, its computation load is often not as important as that of processes operated at the sampling rate, like spike detection.

Memory usage is very important and can have a major impact of the size of a neural recording implant. FBS_{HT} requires five memory elements for each channel, one each decomposition level and additional one for input data. FSDE only requires two memory elements and DD|$_2$-Extrema demands seven memory elements. Although FSDE needs the

least memory, its clustering performance is the worst, as shown in Figure 4.11-Figure 4.13. $DD|_2$-Extrema requires the largest amount of memory and its performance is worse than FBS_{HT}. In summary, compared to the best known hardware-efficient feature extraction methods, FBS_{HT} shows the best spike clustering performance while maintaining a similar level of hardware resources suitable implantable applications.

4.9 Conclusion

In this chapter, a new feature extraction method called frequency band separability was presented. A *Separability* metric was created to investigate information content of low and high frequency bands of spikes and noise. Based on an analysis of *Separability,* and considering the need for resource-efficient hardware implementation, the weighted haar DWT method was derived to extract features from the desired frequency bands, namely positive and negative peaks from the 1st level detail and the 4th level approximation transform coefficients. To provide robust clustering performance, detail features were weighted to scale them the same dynamic range as the approximation features. The clustering performance of the presented features was tested using synthesized datasets consisting of various spike shapes based on real neural recordings from multiple channels. The results show that $W\text{-}FBS_{HT}$ features perform better than other hardware-efficient feature extraction methods while consuming a comparable level of hardware resources.

$W\text{-}FBS_{HT}$ features provide high spike sorting accuracy while only consuming a small amount of hardware resources. Because each spike is only represented by four feature samples, the hardware requirement for the spike classification block in the next chapter

will be significantly reduced. As a result, *W-FBS*$_{HT}$ features provide the capability of designing a hardware-efficient neural signal processor for high-channel-count application.

Chapter 5 Hardware Efficient Decision Tree based Neural Spike Classification

5.1 Introduction

In the previous chapter, a new spike feature set was developed with better sorting performance than other computationally efficient feature extraction while consuming comparable hardware resources. The new feature set (frequency separability features (FBS)) will be used as input for spike classification in this chapter. Spike clustering & classification is the last step in spike sorting to reduce the data rate of neural signals. Figure 5.1 illustrates the function of this step. Clustering determines the number of spiking neurons in each channel and the boundaries delineating different spike clusters. Classification assigns each spike feature vector a class label representing a spiking neuron. The clustering procedure can be implemented on-chip at the cost of significant memory cost to store transient parameters [103]. To save the power and area, clustering was implemented sequentially for each channel [103]. Alternatively, a hybrid strategy was proposed where clustering is processed off-chip and relevant parameters are sent back to the implant for real time classification [105, 121]. In this chapter, we adopt this hybrid approach and focus on the work to minimize the number of parameters required for spike classification. A new spike classification method is introduced, which outperforms the existing gold standard spike classification method in terms of both power and area consumption while maintaining comparable classification accuracy.

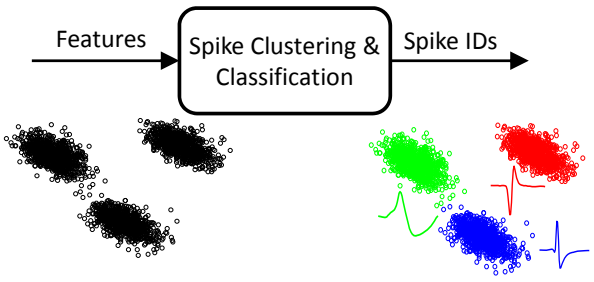


Figure 5.1. Illustration of the input and output signals for spike clustering & classification block.

5.2 Hardware Resource Analysis on Decision tree based Spike Classification

Spike classification usually compares the distance between an input spike feature vector and spike cluster means. Different metrics have been used to measure the distance, such as ℓ_2 norm (Euclidean) distance [102], ℓ_1 norm distance [103, 105] and Mahalanobis distance [94]. The only metric that has been applied for on-chip/implantable implementations is the ℓ_1 norm distance d_j between the input spike features *s(n)* and the cluster means $c_j(n)$:

$$d_j = \sum_{n=1}^{N_f} |s(n) - c_j(n)| \tag{5-1}$$

where N_f is the number of features representing a spike. The spike will be classified to the cluster j that minimizes this distance. The number of additions required to compute (1) can be calculated as

$$C_{L_1} = (2N_f - 1)N_c \tag{5-2}$$

and the memory size in bits per channel is

$$M_{L_1} = N_c N_f B_{Sa} \tag{5-3}$$

where N_c is the number of clusters and B_{Sa} is the number of bits used to represent a spike feature. ℓ_1 norm based spike classification satisfies the power and area requirements for a low channel count system. However, (5-2) and (5-3) reveal that all features of a spike is employed for computation N_c times to classify the spike. For high-channel-count systems, this method requires a large amount of memory to store the mean of each spike class for every channel. Additionally, accessing the memory multiple times for each calculation of the ℓ_1 distance increases the system power consumption.

It has been suggested that spike features with multimodality of distribution provide better separation than those with large variance [91]. Some statistical methods such as Lilliefors Test, Hartigan's Dip Test are used for analysis to select features that have multimodal distribution across spikes [122]. This indicates that spike classification can be estimated based on individual features or a subset of features instead of using all features for each detected spike. Figure 5.2 shows a 2-D projection of three spike classes using PCA and their cluster means. Each spike is represented with two features in this case. Supposing (5-1) is used for classification, each input spike will be measured against all three cluster centers at the cost of 12 additions. However, it can be observed that spike classes can be easily distinguished by using two axis-parallel lines. The first principal component can be compared with a constant value, and the outcome determines whether the second component is needed or not for comparison with another constant value. Therefore, at most two additions would be required for computation, which would reduce the computation complexity to less than 20% of (5-1).

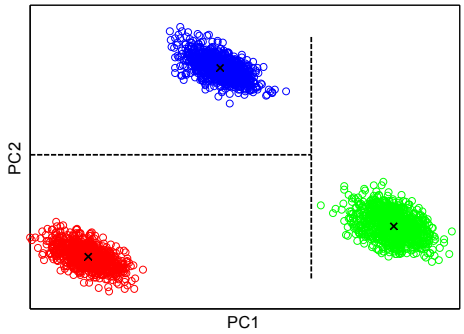


Figure 5.2. An example 2D projection of three classes of spikes using PCA feature extraction.

5.3 Development of Decision Tree Model

5.3.1 Tree Model

The classification model illustrated above can be implemented by a sequence of queries based on the features of a spike. The sequence of queries can be organized in a binary decision tree (DT) as shown in Figure 5.3. Each spike begins to be classified at the root node and terminates at a leaf node, where a class label is produced as output. Generally, each node can be considered as a hyperplane in N_f dimensional space. The output from a node is a binary value identifying if a feature vector lies above or below the hyperplane and determining which child node will be visited. The node or hyperplane function can be expressed as

$$f = \sum_{n=1}^{N_f} a(n)s(n) + W \qquad (5\text{-}4)$$

where *a(n)* is the normal vector and *W* is the weight bias for the hyperplane. The DT structure based on (5-4) is called an oblique DT [123]. Figure 5.1 is a special situation where only one of *a(n)* is equal to one and the rest are zero, which is called axis-parallel DT. Usually, for spikes with N_c classes, N_c−1 hyperplanes are needed to separate all the classes. In this case, the depth of the tree is $\lceil \log_2(N_c) \rceil$, which determines the maximum

number of nodes an input will visit. The hyperplane coefficients at each node are trained offline and coefficients of the hyperplane are sent back for spike classification. The algorithm to build a decision tree is described in the next section.

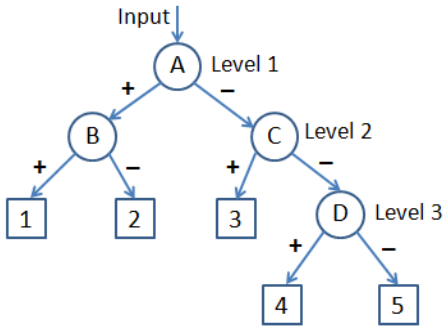


Figure 5.3. An example decision tree for five classes. The circles represent nodes (corresponding to comparisons between hyperplanes and feature vectors) while the squares represent leaves (corresponding to spike classes).

5.3.2 Quantization of Decision Tree Models

The coefficients of the hyperplane at each node are obtained by training spike features based on the most common algorithm defined as OC1 [123]. At each node, OC1 splits the training dataset into two subsets, one above the hyperplane and the other below it. The criterion to determine a hyperplane at a node is to minimize the total impurities of both subsets. Different types of impurity measures have been proposed for DT. A common used metric is the information gain which is defined as

$$I = -\frac{N_L}{N_t}\sum_{j=1}^{N_c} P(\omega_j|X_{tL})log2(P(\omega_j|X_{tL})) - \frac{N_R}{N_t}\sum_{j=1}^{N_c} P(\omega_j|X_{tR})log2(P(\omega_j|X_{tR})) \quad (5\text{-}5)$$

where N_t, N_L, N_R are the number of total samples, samples split into left and samples split into right respectively, and $P(\omega_j|X_{tL})$ and $P(\omega_j|X_{tR})$ represent the probability of j^{th} class ω_j in the left $X_{tL}$and right X_{tR} splitting subset of samples. The splitting criterion defines the tree structure such that an input will be assigned a class label either at the last or the

last but one depth.

The hardware realization complexity of the hyperplane function (5-4) is data dependent. If spike classes are well separated in the space as shown in Figure 5.1, (5-4) will be reduced to only one addition at each node. Otherwise, (5-4) demands multiplication if more than one of the coefficients *a(n)* is nonzero. To ease the computation complexity, *a(n)* can be quantized into fewer number of bits, much less than the word length of *s(n)*.

The traditional OC1 algorithm generates coefficients with any real number values. To quantize the coefficients, two additional steps were added into the OC1 algorithm. The first step quantizes coefficients with real number value into given B_a bits. After quantization, only one coefficient *a(n)* is exactly equal to 1, and all the coefficients are represented by B_a bits. The second step re-estimates the weight bias based on quantized coefficients. Figure 5.4 illustrates the steps of the modified OC1 algorithm for quantization of coefficients.

1. Input: spike feature set D and its class labels.
2. Output: a hyperplane $H = [a(1), a(2), \ldots, a(N_f), W]$.
3. Find the best axis-parallel hyperplane H_a and its impurity I_a.
4. Initialize $H = H_a$.
5. *Perturb* each coefficient of H in sequence and then calculate its impurity I after perturbation.
6. Repeat M times:
7. Choose a random vector $R = [r(1), r(2), \ldots, r(N_f), W_r]$ and let $H_1 = H + \alpha R$.

Figure 5.4. Description of the modified OC1 algorithm for quantization of coefficients at a single node.

Figure 5.4 (cont'd)

8. Find the optimal α by *perturbation* and calculate the impurity I_1.
9. If $I_1 < I$, then replace H by H_1, *perturb* each coefficient of H in sequence and calculate its impurity I.
10. Normalize each coefficient of H by the $a(n)$ with the maximum absolute value and quantize $a(n)$ into B_a bits including a sign bit.
11. *Perturb* the weight bias W in H and calculate its impurity I.
12. If $I_a < I$, then $H = H_a$.

where *perturb* or *perturbation* function is defined as follows:

1) Calculate $U_i = a(n) - \frac{\sum_{n=1}^{N_f} a(n)s(n)+W}{s_i(n)}$ for the i[th] spike feature.
2) Sort all the values U_i.
3) For each $a_i(n)$ in the middle point between U_i and U_{i+1}, calculate the impurity of splitting all U_i.
4) The $a_i(n)$ that provides the minimum impurity is chosen as $a(n)$.

13. To ensure the perturbation, input features need to be normalized such that each feature is greater than zero.

To determine the best number of bits for quantization of coefficients, simulated datasets described in Chapter 4 were used for analysis. Because (5-4) is data dependent, different feature extraction methods may result in different tree complexity. Both traditional PCA with four PC components and FBS spike features were used to determine quantization resolution. Classification performance across different quantization

resolutions is shown in Figure 5.5, where the DT classification error ($Error_{DT}$) is compared to the ℓ_2 norm distance based classification error ($Error_{\ell_2}$). B_a equal to one represents the simplest axis-parallel DT. On average, the improvement of quantization from two to four bits is negligible. The average improvement from one bit to two bits is around 2%. The error variation for one bit resolution is around 3% and 1% for more than two bits. Quantization of three and four bits provides little improvement as shown in Figure 5.5. Hence, the best quantization was selected to be two bits.

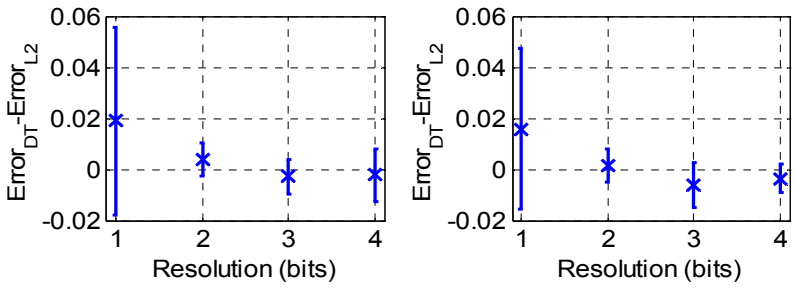


Figure 5.5. Comparison of spike classification error across different resolutions of *a(n)* using (a) PCA and (b) FBS feature extraction.

5.4 Hardware Architecture

Hardware implementation of (5-4) consists of a memory block, a computation core (CC), a controller and a class decoder as shown in Figure 5.6. The CC performs calculation of (5-4) at each node based on the input spike feature pattern and the node coefficients loaded from the memory block. The controller takes the binary output value from the CC and determines the address of the memory to access the coefficients for the child node. It also manipulates the data flow in the CC and informs the class decoder to compute the spike ID when the computation terminates. The class decoder stores the output of each node visited until the computation terminates at the leaf node. The

sequence of the node output from the root to leaf is unique and can be used as a spike ID. Usually, the number of neurons that can be recorded in each channel is less than six [103, 124]. Thus, the spike ID includes three bits.

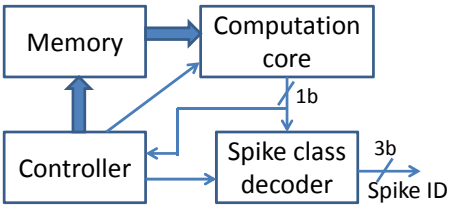


Figure 5.6. Block diagram of DT based spike classification circuit.

Each cell in the memory block stores two data fields representing the computation coefficients for each node, as shown in Figure 5.7. The weight bias W is represented by B_{Sa} bits, same as the number of bits to represent a spike feature. Because $a(n)$ is quantized into B_a bits, $N_f \times B_a$ number of bits are required to store all $a(n)$ coefficients. In this design, because $a(n)$ is quantized into two bits, '00', '01', '10', '11' are used to represent the value of 0, 0.5, -0.5 and 1, respectively. Thus, the total number of bits for each channel is

$$M_{DT} = \left(B_{sa} + B_a \times N_f\right)(N_c - 1) \qquad (5\text{-}6)$$

Memory is the most area dominant component in a spike sorting DSP. Figure 5.8 compares the memory size between (5-3) and (5-6) with B_{sa} set to ten and N_c set to six. The amount of coefficient memory is proportional to the number of features. Computationally efficient feature extraction methods use three or four features to represent a spike [96, 97]. In these cases, this DT based method can reduce the memory usage at least by 55% as shown in Figure 5.8.

Types	W	a(n)
# of bits	B_{sa}	$N_f \times Ba$

Figure 5.7. The data fields comprising a node in the decision tree, listed with their resolutions.

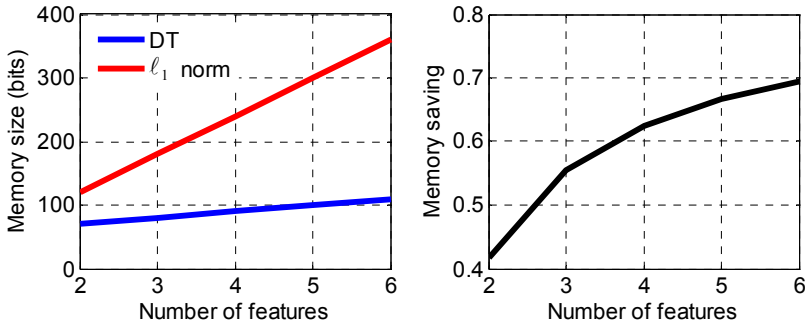


Figure 5.8. Comparison of memory size per channel between the ℓ_1 norm and DT classification methods as a function of the number of features.

Figure 5.9 shows the circuit implementation of the CC block and the spike class decoder. The weight bias is loaded into the CC in the first computation cycle. Then, each *a(n)* will be fed in sequentially on each clock cycle. If *a(n)* is zero, the CC takes no action and remains at the current state. If *a(n)* is one, the input feature *s(n)* will be directly connected to the accumulator by the multiplexer. Otherwise, *s(n)* will be multiplied by *a(n)* and the result will be forwarded to the accumulator. Because *a(n)* is quantized into one magnitude bit and one sign bit, the multiplication can be simplified to a shift operation. Completing one node computation takes a total of N_f+2 clock cycles as shown in Figure 5.10. The coefficients for comparing a feature vector with a single hyperplane are loaded from memory in the first clock cycle. Then a computation occurs on each of the next N_f clock cycles. During the computation cycles, each spike feature *s(n)* and its corresponding coefficient *a(n)* at current node are selected in sequence into the CC block for computation. In the last cycle the result is used to decide which memory will be

selected for computation of next node. The spike class decoder stores the binary output from the CC block in this last cycle for each node computation in sequence. All the binary outputs are concatenated together to form the spike ID at the end of the tree computation. The computation complexity is given by

$$C_{DT} = N_f \lceil log_2 N_c \rceil \tag{5-7}$$

The computation complexity of ℓ_1 norm method increases linearly as N_c increase as shown in (5-2) while the DT method does not change the complexity as long as N_c varies between 2^m+1 and 2^m. In the case of four features used for spike classification in this design, the computation complexity of the DT method can save around 65% of computation compared to the ℓ_1 norm method.

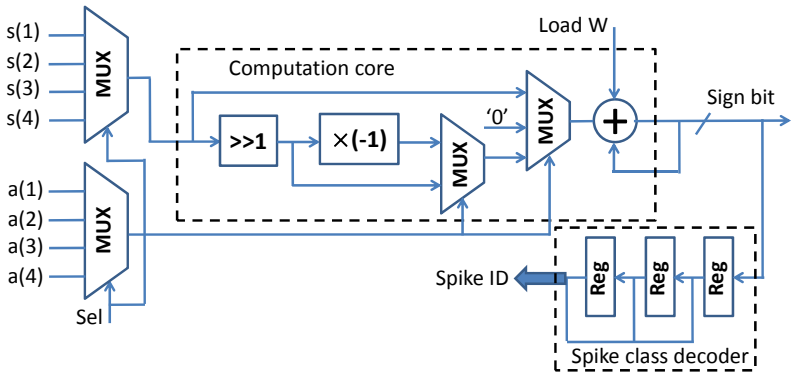


Figure 5.9. Structures of the computation core and the spike class decoder.

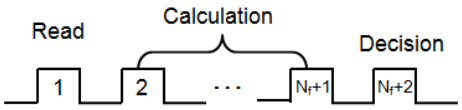


Figure 5.10. The operation phases for a complete node computation.

5.5 Simulation Results

Using the simulated datasets described in Chapter 4, DT based spike classification

performance was compared against ℓ_1 norm classification using both PCA with $N_f = 4$ and FBS for feature extraction shown in Figure 5.11. The Kmeans clustering method was first performed on spike features to obtain the centroids of spike classes for ℓ_1 norm classification. The Kmeans method also generates spike labels for each spike which were used for the DT method to train two-bit quantized hyperplanes. Results show that DT classification performance decreases when SNR decreases because spike features are less separable at low SNRs. The DT classification performance for PCA features are 3% on average better than FBS features. It shows the similar results to the Kmeans clustering results shown in Chapter 4 and it can be concluded that DT classification performance are mainly dependent on the goodness of spike features if hyperplanes are properly trained. Compared to ℓ_1 norm, classification errors for DT is within 1% on average over all SNRs no matter using PCA or FBS features. Thus, DT provides the comparable classification performance to ℓ_1 norm while consuming less computation and memory resources than ℓ_1 norm.

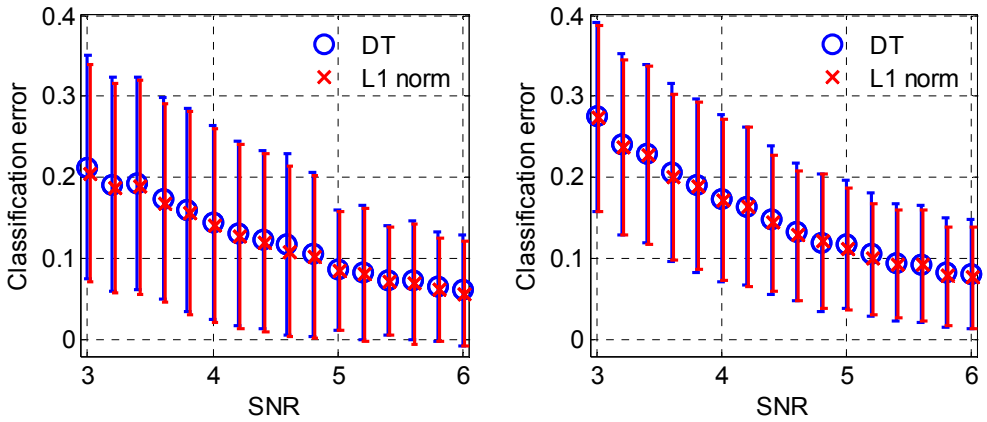


Figure 5.11. Comparison of spike classification error between ℓ_1 norm and DT methods using (a) PCA and (b) FBS features.

Hardware implementation of the DT method as described in Section 5.4 was realized in Verilog hardware description language (HDL) with $N_c = 6$. The design was synthesized and mapped into a 130nm standard cell library. Because spike classification occurs at the spike firing rate, its power consumption is negligible compared to the one of spike detector which is operated at the sampling rate. The area consumption is about 10400 μm^2 out of which the memory occupies around 45%. For multichannel application, [103] chose interleaved architecture as a trade-off between area and power consumption per channel. However, because the power consumption of spike classification is negligible, area minimization is more important than power. In the next chapter, system integration of all spike sorting blocks, including spike detector, feature extractor and spike classifier and their scalabilities toward multichannel application will be analyzed

5.6 Conclusion

This chapter presented a DT based spike classification method and hardware design for neural recording implants that achieves spike classification accuracy comparable to ℓ_1 distance classification. This new method stores fewer coefficients than needed for ℓ_1 norm, which reduces memory size and hence the spike classification block area. The computation complexity was also minimized by quantization of hyperplane coefficients without appreciable loss of performance. A DT based spike classification architecture was implemented in Verilog HDL and mapped into 130nm standard library. It consumes 10400 μm^2 area per channel. The memory-friendly property makes the DT based spike classification suitable for high-channel-count spike classification.

Having automatic thresholding spike detector developed in Chapter 3, noisy insensitive feature extractor developed in Chapter 4 and memory-efficient spike classifier

developed in this chapter, a neural signal processor (NSP) will be designed to integrate all individual hardware blocks together in the next chapter. A single channel NSP will be first designed to cooperate each block to process a neural signal in real time. Then the scalability towards high-channel-count applications for the NSP will be analyzed in terms of power and area tradeoff to specify the scalability of each block. Finally, the multichannel NSP will be implemented and its function will be validated to demonstrate its capability for high-channel-count application.

Chapter 6 Spike Sorting based Neural Signal Processor

6.1 Introduction

In Chapter 3 to 5, hardware blocks for each spike sorting step have been developed. To implement a power-area efficient neural signal processor for high-channel-count BMI applications, scalable integration of all spike sorting blocks needs to be analyzed. This chapter presents a single-channel architecture to combine each block together first. Then design methodologies are developed to extend the single-channel architecture to a multichannel architecture based on the scalability analysis of each block. Finally, a 32-channel NSP module that is highly scalable is implemented in Verilog HDL and its function is tested on FPGA to validate its real-time capability.

6.2 Single-channel Design of Spike Sorting

Real-time spike sorting requires two phases: 1) a training phase to determine the threshold value for spike detection and coefficients for spike classification model and 2) a real-time sorting phase to assign a spike ID to each detected spike. To provide hardware efficiency, it has been commonly adopted that coefficients for spike classification model are trained offline [105, 121, 125]. Figure 6.1 describes sequential procedures for the single-channel training phase. During the training phase, the threshold value is estimated first online. After that, spike detection and feature extraction are performed online to obtain spike features. Spike features are then transmitted out for offline training where spike clustering is executed first and coefficients of the classification model are trained after clustering. When the offline training is finished, relevant parameters are sent back to a neural signal processor for real-time spike sorting. In this section, we first introduce an automatic spike clustering algorithm to determine the number of clusters. A block

diagram of a single-channel implementation is presented for real-time spike sorting later.

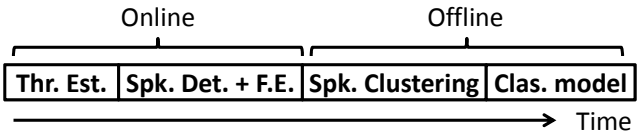


Figure 6.1. Illustration of sequential procedures for single-channel training phase.

6.2.1 Offline Training for Automatic Spike Clustering

The main role of spike clustering is to automatically estimate the number of clusters and their centroids from spike features. The centroids and all the spike features are then used as input to obtain coefficients of a spike classification model.

The gold standard Kmeans clustering method requires the prior knowledge of the cluster number and is not suitable for automatic spike clustering. Superparamagnetic clustering introduced in [91] sweeps a parameter called temperature from low to high. As the temperature increases, more clusters will be created with different cluster sizes. This method needs to set a threshold such that the number of clusters is determined when a new cluster is less than the threshold. Osort clustering has been used to automatically determine the number of clusters [102]. However, it is only suitable for spike template based clustering where a threshold is estimated from the standard deviation of neural signals. For spike feature based clustering, the threshold for Osort still needs to be set manually. Thus, an automatic spike clustering method without any parameter settings is required to estimate the number of clusters.

The *gap statistic* technique was developed to estimate the number of clusters in a set of data [126] and is deployed in this thesis for automatic spike clustering. Figure 6.2 illustrates the general idea of the *gap statistic* technique. A spike feature set is first

clustered into k clusters using Kmeans cluster algorithm. The cluster number k is swept from 1 to K. For each Kmeans clustering with k clusters, denoting r^{th} cluster dataset as C_r, a within-cluster dispersion W_k is calculated by

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r \tag{6-1}$$

where D_r is the pairwise distance in C_r contacting n_r points and is calculated as

$$D_r = \sum_{x_i \in C_r} \sum_{x_j \in C_r} |x_i - x_j|^2 \tag{6-2}$$

From the spike feature set, a reference dataset is generated by uniformly and randomly sampling either the dataset bounding box as shown in Figure 6.3(a) or a box aligned with the principle components of the dataset as shown in Figure 6.3(b). The within-cluster dispersion of the reference dataset is then calculated and denoted as W^*_k. The reference dataset is usually generated multiple times and W^*_k is calculated each time. Hence, W^*_k can be considered as a random variable. The *gap statistic* is defined as

$$Gap_k = E[\log(W_k^*)] - \log(W_k) \tag{6-3}$$

Finally, a decision metric M_k defined as

$$M_k = Gap_k - (Gap_{k+1} - s_k) \tag{6-4}$$

is used to select the optimal cluster number K_{opt} which is the smallest k such that M_k is greater than zero. Figure 6.4 shows M_k as the cluster number k varies and the result of automatic spike clustering for the dataset shown in Figure 6.3. The *gap statistic* determines the optimal number of clusters is three which is the same as intuitive observation.

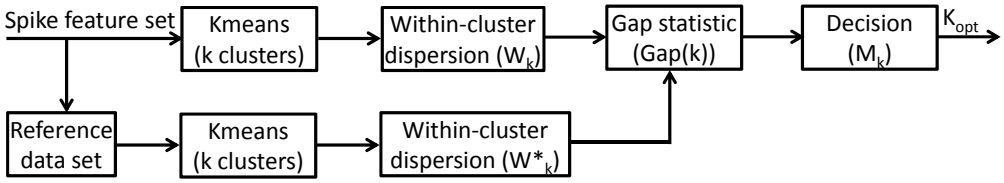
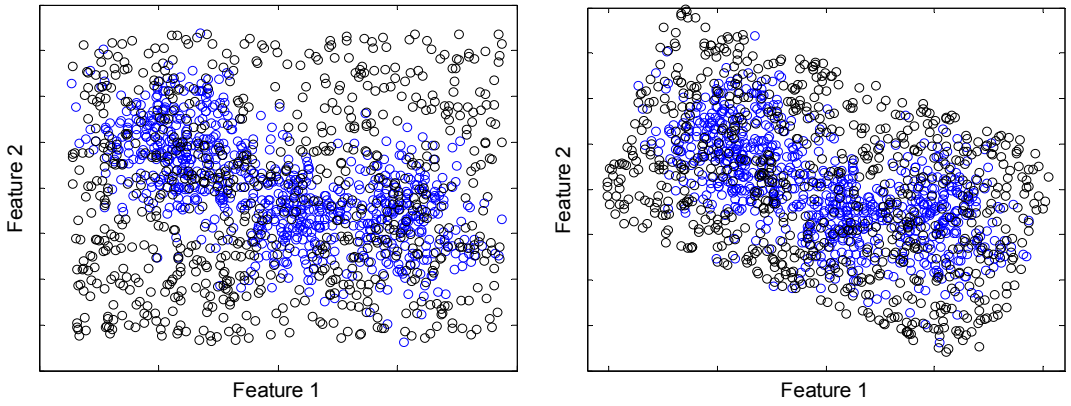
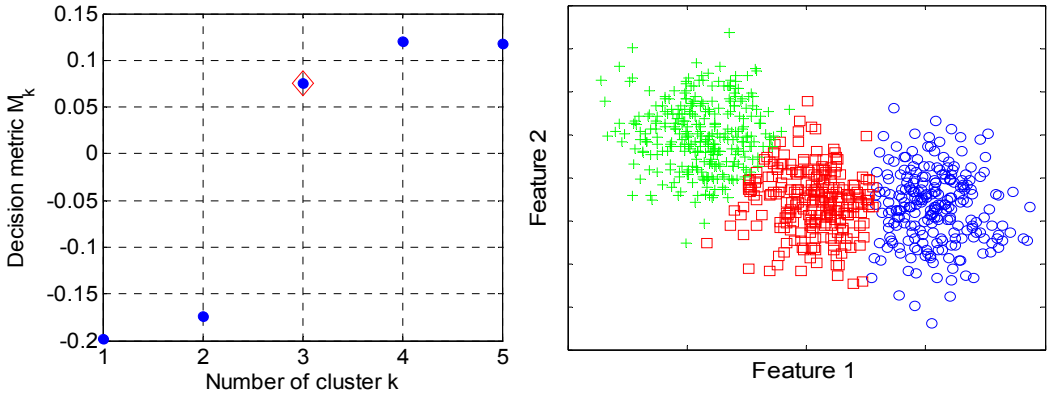


Figure 6.2. The data processing flow for the *gap statistic* technique.



(a) (b)

Figure 6.3. Reference dataset generated by uniformly sampling either (a) the dataset bounding box or (b) a box aligned with the principle components of the dataset.



(a) (b)

Figure 6.4. (a) The decision metric as a function of number of cluster. (b) Automatic spike clustering result using *gap statistic* Kmeans clustering.

6.2.2 Single-channel Implementation

Having spike detector, feature extractor and spike classifier developed in Chapter 3 to

5, additional hardware resources are required to provide high performance of spike sorting in real time. Traditionally, a spike alignment block is needed to provide a whole spike waveform to a feature extractor, because the traditional feature extraction methods request point wise operations on spikes. The whole spike waveform is stored and aligned to a common reference point in this alignment block. The reference point for alignment plays a critical role in spike sorting performance. However, the spike alignment block increases the computational load and the memory demand of the neural signal processor. In our filtering based feature extraction method, point wise operations on spikes are not required and therefore the alignment block is not necessary. Because a spike is detected when it crosses a threshold, several spike samples are missed before threshold crossing. Therefore, to extract useful features from missing spike samples, a pre-threshold crossing buffer is required. To study the effect of the buffer size on spike classification performance, the dataset described in Figure 4.1 were used and the result is shown in Figure 6.5. It can be seen that the classification performance provides little improvement (less than 0.5%) for the buffer size beyond six. Considering the fact that the NEO pre-processor delays the neural data by two samples, the best buffer size was chosen to be eight in this design.

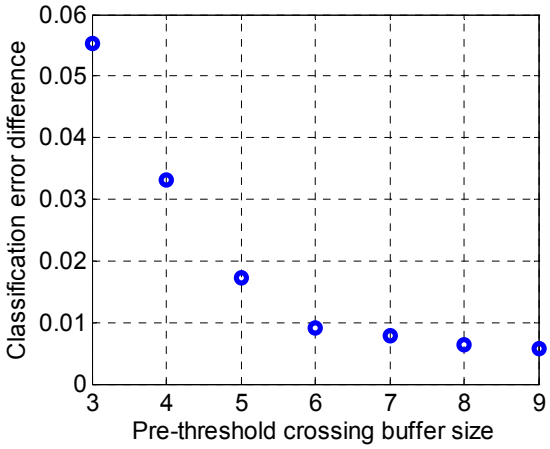
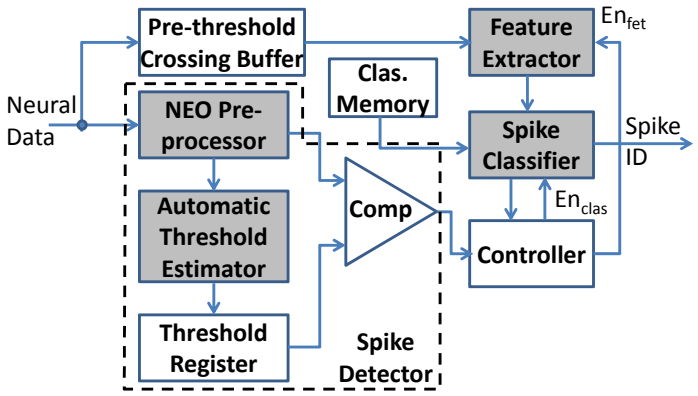


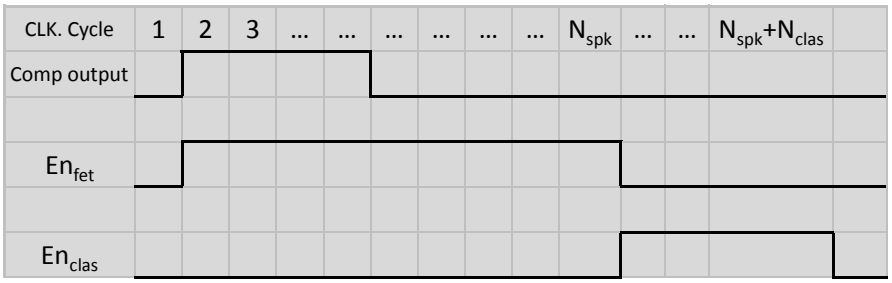
Figure 6.5. The classification error with different buffer sizes compared to the classification error using whole spike waveforms.

Because the feature extractor and the spike classifier are only activated when a spike is detected, a controller is needed to generate control signals for both of them. Figure 6.6(a) shows the block diagram for single-channel spike sorting and Figure 6.6(b) illustrates the control signals to manage different blocks. When a NEO coefficient is above a threshold value, the controller generates a enable signal (En_{fet}) to activate the feature extractor for a period equal to the length of a spike window (N_{spk}). After features are extracted, the controller triggers the spike classifier by setting an enable signal (En_{clas}) high for N_{clas} clock cycles.

To process multichannel neural signals, single-channel architecture can be linearly scaled at the expense of a huge amount of hardware consumption. Implantable applications desire efficiently sharing blocks including NEO preprocessor, threshold estimator, feature extractor and spike classifier for multichannel signal processing. These blocks contribute nearly 90% of area to the signal-channel design. Thus, it is imperative to provide a scalability analysis for each block in terms of tradeoffs among power, area and speed.



(a)



(b)

Figure 6.6. (a) Block diagram for single-channel spike sorting. (b) Control signals from the controller to manage the activities of the feature extractor and the spike classifier.

6.3 Multichannel Design of Spike Sorting

6.3.1 Scalability Analysis

A. NEO Preprocessor

The scheme of NEO preprocessor is shown in Figure 6.7. The computational cells including multipliers and adders occupy around 85% of the NEO preprocessor area. Interleaving is an effective technique to save area by reusing the computational cells across channels with increased clock frequency. When the NEO preprocessor is interleaved to support N_{intlv} channels, each delay element Z^{-1} in Figure 6.7 is replaced with N_{intlv} cascaded delay elements. As a result, the computational resources for each channel are reduced by a factor of N_{intlv}. In the meantime, the data rate of input data is

raised by N_{intlv}, which increases the power consumption of the NEO preprocessor. To determine the best value of N_{intlv}, the power-area product per channel were studied for different values of N_{intlv} as shown in Figure 6.8. It can be seen that area per channel reduces as N_{intlv} is increased and eventually saturates to the area of single-channel delay elements. Power consumption is proportional to N_{intlv} because of faster clock frequency. The four and eight interleaved designs provide similar power-area product performance. In our design, N_{intlv} was chosen to be eight in favor of 35% area reduction compared to N_{intlv} equal to four. In this case, the area consumption was reduced by 75% compared to the one without interleaving.

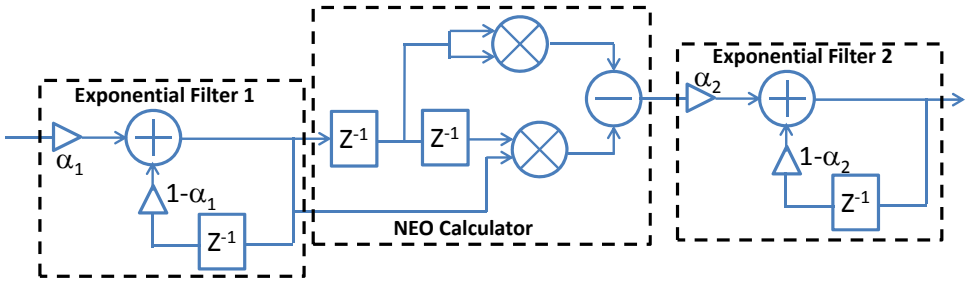


Figure 6.7. The scheme of NEO preprocessor unit for one channel.

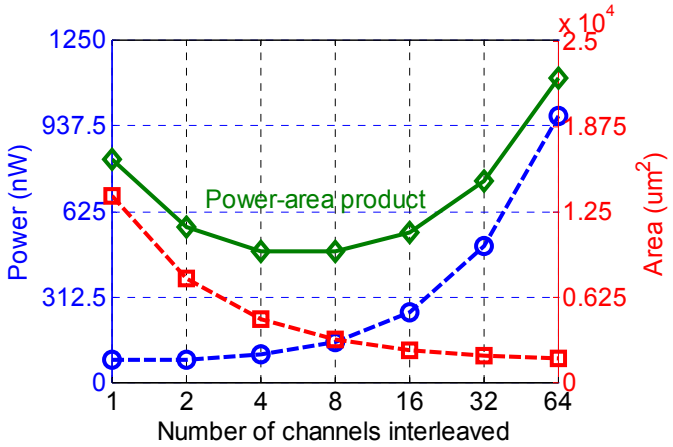


Figure 6.8. Power and area tradeoff over the number of channels interleaved.

B. Threshold Estimator

The strategy to scale the threshold estimator depends on the noise variation. The threshold estimator should be able to keep track of the noise variation in time such that spike detection performance is not seriously affected. It has been shown that the noise level can vary at timescale of minutes (five minutes) [113] while the threshold estimator in our design cost around five seconds for computation. Thus, for multichannel application, the threshold estimator is determined to cycle through 32 channels and sequentially update threshold values for each channel roughly every three minutes. As a result, both power and area consumptions are reduced by around 95% compared to the choice without scaling.

C. Feature Extractor & Spike Classifier

Because the feature extractor and the spike classifier are only active when a spike is detected, both of them operate at the frequency of the spike firing rate instead of the sampling rate such as the NEO preprocessor. Thus, they can be scaled in the same way together for multichannel signal processing. It is obvious that both blocks are idle most of time in the single-channel application. The idle time can be utilized to process detected spikes from other channels as long as spikes from different channels are not detected at the same time. This concludes that the scalability of the feature extractor and the spike classifier is determined by the maximum number of channels (N_{max}) in which spikes are fired simultaneously when processing N_{ch}-channel data. To study the relationship between N_{max} and N_{ch}, we synthesized N_{ch}-channel spike trains using Poisson firing model with the firing rate equal to 100 Hz and analyzed the statistic of N_{max}. Here, we consider spikes are fired simultaneously if the difference of spike occurrence time among channels is less than spike duration time (assuming 2 ms). Figure 6.9 shows the

relationship between N_{max} and N_{ch}, where N_{max} is calculated under the condition that the probability of simultaneously fired spikes across N_{ch} channels greater than N_{max} is less than 0.1%. It can be seen that the scalability defined as $\frac{N_{ch}}{N_{max}}$ increases if more channels are processed at the same time. This seems to imply that more channels need to be involved for hardware-efficient multichannel processing. However, in practical implementation, there must be a multi-input-multi-output (MIMO) multiplexer between the N_{ch}-channel data buffers and the N_{max} feature extractor & spike classifier (FESC) blocks to connect channels where spikes are detected to those idle FESC blocks as shown in Figure 6.10. Thus, the scaling effect on the area cost of this MIMO multiplexer should also be considered. Figure 6.11 shows the area per channel of the FESC block and the MIMO multiplexer when N_{ch}-channel neural data are processed by N_{max} FESC blocks. It can be seen that the area of the FESC block decreases while the MIMO multiplexer increases as N_{ch} becomes larger. The best choice for N_{ch} is 32 where the total area of these two blocks is the smallest and only 13 FESC blocks are needed from Figure 6.9. This design choice reduces the area by almost 35% compared to using one FESC block for each channel.

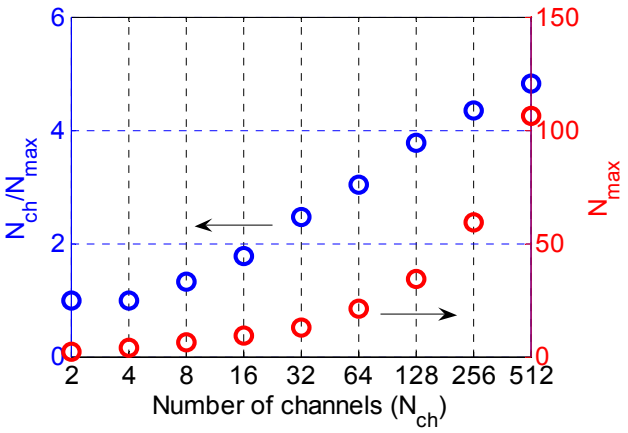


Figure 6.9. The relationship between N_{max} and N_{ch}. The scalability defined as $\frac{N_{ch}}{N_{max}}$ describes the number of channels can be shared by the same feature extractor and spike classifier blocks when N_{ch}-channel neural data are processed.

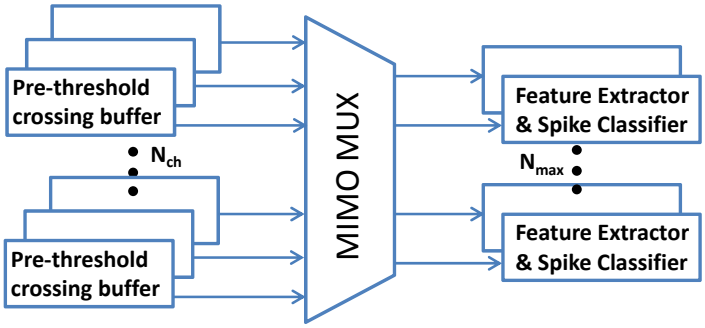


Figure 6.10. Illustration of connection between N_{ch}-channel data buffers and the N_{max} feature extractor & spike classifier blocks.

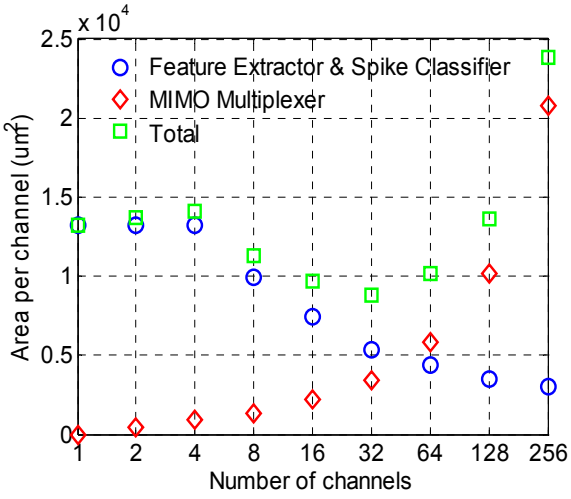


Figure 6.11. The area per channel of FESC blocks and the MIMO multiplexer when N_{ch}-channel neural data are processed by N_{max} FESC blocks.

6.3.2 Multichannel Implementation

The scalability analysis induces the design of a spike sorting NSP module that can process 32 neural channels simultaneously. This module includes four NEO preprocessors each processing 8-channel interleaved data, one threshold estimator calculating threshold values sequentially for all channels and 13 FESC blocks classifying detected spikes from 32 channels. Other blocks like the pre-threshold crossing buffer, the threshold register and the classification memory are linearly extended. A global controller is required to organize system operations and manipulate data flows among different blocks. Figure 6.12 shows the architecture of the 32-channel NSP module. Assuming neural data sampling rate is 20 kHz, the NSP module operates at the clock frequency of 160 kHz. When a spike is detected and classified, this NSP module outputs a channel index from the global controller and a spike ID from the spike classifier.

The global controller plays an important role in functioning multichannel signal processing. It initiates the training phase to evaluate threshold values sequentially for each channel and then enable spike detection and feature extraction for all the channels. When coefficients of all spike classification models are received from the offline training, the global controller activates spike classifiers for real-time multichannel spike sorting. The global controller routes threshold values from threshold registers through the MUX in the spike detector for spike detection. When spikes are detected, it determines channel indexes and FESC blocks that are in idle state at the same time. Thus, pre-threshold crossing buffers from those spike-detected channels are connected to idle feature extractors through the MIMO multiplexer. After features are extracted and fed into spike classifiers, the global controller directs coefficients from classification memory blocks to

spike classifiers through the same MIMO multiplexer. Notice that the feature extractor is synchronized at the data sampling rate (20 kHz) while the spike classifier is operated at the clock frequency (160 kHz). Thus, the spike classifier almost immediately outputs a spike ID after features are extracted.

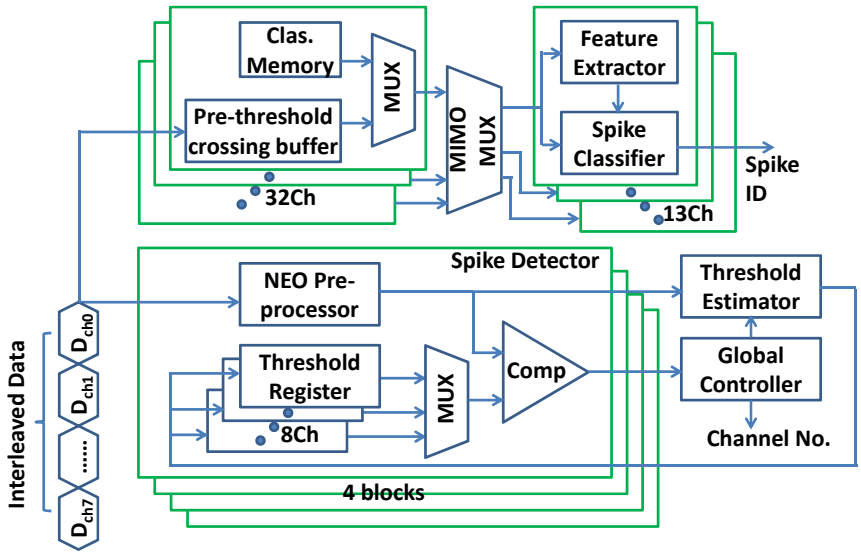


Figure 6.12. Architecture of 32-channel spike sorting NSP module.

6.4 Results

6.4.1 Hardware Performance

The 32-channel spike sorting NSP was implemented in Verilog HDL, and the spike sorting function was verified in an Altera Cyclone III FPGA to demonstrate the real-time performance of the new spike sorting scheme and enable future integration with a complete neuroprocessing platform. Figure 6.13 depicts the FPGA test setup used to verify the function of our NSP module. First, the soft processor receives input data from the PC through a UART interface and stores it into the SDRAM. Then the soft processor sends input data one by one into the NSP coprocessor through the input first-input-first-

output (FIFO) buffer. The NSP sends back output data one by one through the output FIFO to the soft processor which stores the output data into the SDRAM. Finally, the soft processor transmits all output data back to the PC for verification.

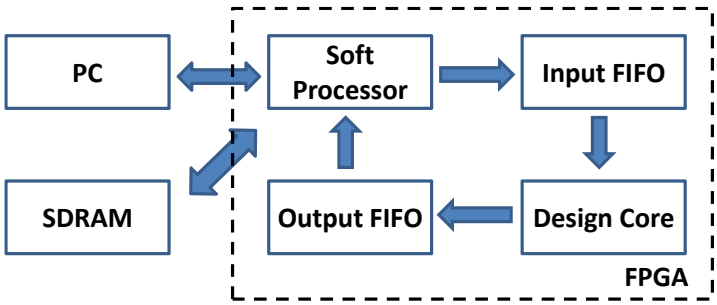


Figure 6.13. FPGA based test setup for NSP verification.

To estimate power and area performance, the design was synthesized and mapped on a 0.13µm CMOS process. For accurate estimation of the module's power consumption, a simulated four 8-channel interleaved signals were fed into the post-synthesis gate-level NSP module using Synopsys Verilog Compiler Simulator (VCS). VCS verifies the function of the gate-level design and calculates the switching activity of each internal node. The switching activity information was then provided to the power estimation tool using Synopsys PrimeTime PX. Table 6-1 lists the power and area consumption for each block based on post synthesis simulation. Our 32-channel spike sorting NSP implementation approximately occupies 0.73mm^2 area and consumes 24.0µW with clock frequency of 160 kHz for a 20 kHz sampling rate and 10 bit data width. The NEO preprocessor dominates almost 70% of power consumption because it operates at the clock frequency for every data sample over all channels. The pre-threshold crossing buffer consumes the second most significant power because it updates all buffer values at data sampling rate. The FESC block only consumes less than 7% of power as it is only active when spikes are detected. The classification memory dominates nearly 30% of area

consumption while the FESC block and the MIMO multiplexer together occupy around 33% of area. Without using quantized decision tree based spike classification and analyzing scalability for the FESC block, the total area would increase roughly by 20% and 10%, respectively. On average, the spike sorting NSP module consumes only around 0.75 μW power and 0.023 mm^2 area per channel.

Table 6-1 Power and area performance of hardware cells from post-synthesis simulation

Cells	Power		Area	
	μW	%	μm^2	%
NEO Preprocessor	**16.6**	**69.1%**	99320	13.6%
Feature Extractor & Spike Classifier	1.6	6.7%	130738	17.9%
Threshold Estimator	0.1	0.4%	23060	3.2%
Global Controller	0.6	2.5%	28109	3.8%
MIMO Multiplexer	0.3	1.3%	114961	15.7%
Pre-threshold Crossing Buffer	3	12.5%	84603	11.6%
Threshold Register	0.1	0.4%	16753	2.3%
Classification Memory	0.9	3.8%	**214226**	**29.3%**
Others (MUX, Comparator)	0.8	3.3%	18380	2.5%
Total	***24.0***		***730150***	

6.4.2 Spike Sorting Performance

To evaluate spike sorting performance, simulated datasets including various spike shapes described in Chapter 4 were generated and fed into our NSP module. Specifically, 40-second 10-bit neural signals with SNR from 3 to 7 and the firing rate from 10 to 100 were synthesized from each dataset. Figure 6.14 shows the overall spike detection performance of the spike detector. The spike detection performance similarly to Chapter 3 is defined as

$$Det_{acc} = \frac{TP}{TP+FN+FP} \tag{6-5}$$

where *TP* is true positive representing the number of correctly detected spikes, *FN* is false negative representing the number of missed spikes and *FP* is false positive representing the number of false spikes due to detecting noise as spikes. The NEO based automatic thresholding spike detector achieves nearly 90% accuracy for SNR greater than 6. Because NEO amplifies spikes of high frequency energy, the spike detector still misses some of spikes dominated in low frequency spectrum at high SNR. It provides detection accuracy on average within 5% of the manual thresholding method for SNR greater than 5. When the SNR is low, the performance degrades up to within 12% of manual thresholding. Figure 6.15 plots the detection performance as a function of the firing rate at different SNRs to show the robustness of the automatic thresholding spike detector against the firing rate. When the firing rate is above 30, the variance of the detection performance is within 2% for SNR greater than 6 and the variance increases up to 7% for SNR below 4. The performance decreases when the firing rate is below 30 even at high SNRs because the number of *TP* is reduced but the number of *FP* is still the same.

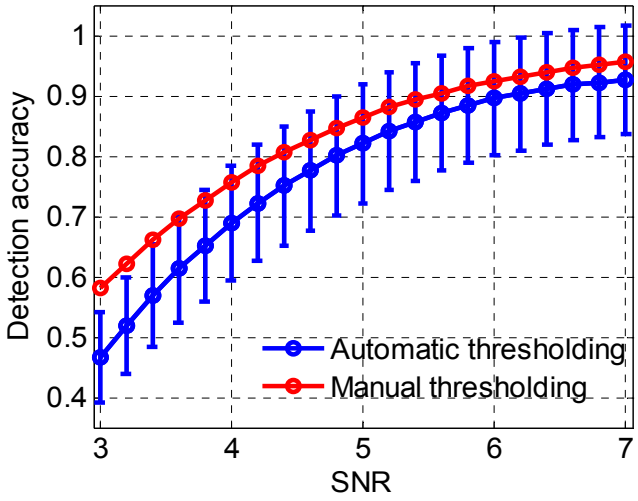


Figure 6.14. Spike detection accuracy against SNR using the automatic thresholding spike detector and the manual thresholding method.

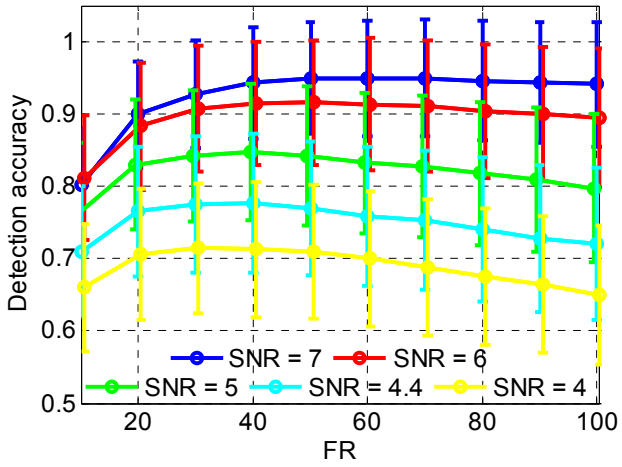


Figure 6.15. Spike detection accuracy of the automatic thresholding spike detector against the firing rate.

Figure 6.16 shows the spike sorting performance of the NSP module which is defined as

$$Sorting_{acc} = \frac{TP_{clas}}{TP+FN+FP} \tag{6-6}$$

where TP_{clas} is the number of correctly detected and classified spikes. The variance of spike sorting performance is higher at high SNRs than at low SNRs, because the NSP module provides dynamic performances for different datasets when SNR is high and

consistently poor performances for all datasets when SNR is low. The performance of our NSP module is compared to the PCA based spike sorting method where the feature extraction method in our module is replaced by PCA with first four PC components (PCA_4). Our spike sorting performance is within 4% less than the PCA based spike sorting method. However, PCA requires the storage of the whole spike waveforms and extensive computations to obtain PCA features. Thus, our NSP module provides comparable spike sorting performance while consuming negligible hardware resources compared to PCA.

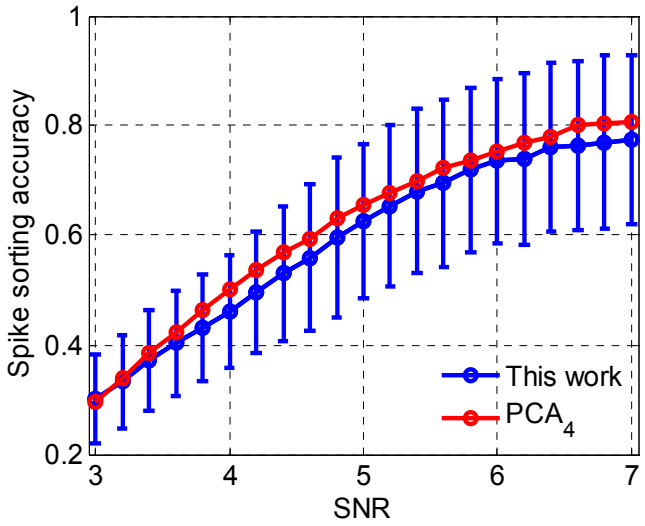


Figure 6.16. Spike sorting performance of our NSP module and the one using PCA features.

6.4.3 Comparison to other work

At the moment of this thesis dissertation, according to our best knowledge, only one work has implemented multichannel on-chip spike sorting NSP [103] which is based on an unsupervised spike clustering algorithm called Osort [102]. Figure 6.17 shows the classification accuracy of our spike sorting method and Osort. To eliminate the impact of spike detection, all spikes are assumed to be detected and aligned to the same reference

point. Our work provides on average 15% higher classification accuracy than Osort and the performance of our work is comparable to Osort even in the worst case. Osort shows more variability in performance because it needs to estimate a clustering threshold parameter from neural signals and the parameter estimation method is sensitive to the spike firing rate. Table 6-2 lists the specifications of our work and the Osort based spike sorting NSP. This work provides 6X power reduction and 3X area reduction compared to the Osort NSP. The main reason for our superior hardware performance is that Osort is a spike-template based spike sorting method which has to store cluster means each with dimensionality equal to the spike length. As a result, it is reported that the Osort based NSP requires the memory size of 50 kb for 16-channel (3125 b/channel) implementation and the memory consumes 66% of power even with power optimization. In our work, we only extract four features from each spike waveform. Furthermore, by using quantized decision tree based spike classification, each channel only demands the memory size of 140 b. For implantable applications, the size of the electronic system is limited by the size of the microelectrode array. Current microelectrode technologies have demonstrated the implantable feasibility with the size of 4 mm × 4 mm. Under this area constraint, our NSP module can be extended to process the number of channels as high as 690 while the Osort based NSP can only process 225 channels. If considering the CMOS process difference, our NSP module is scalable to process 2760 channels under the 65-nm CMOS process. Thus, the spike sorting NSP developed in our work is more suitable for high-channel-count BMI applications.

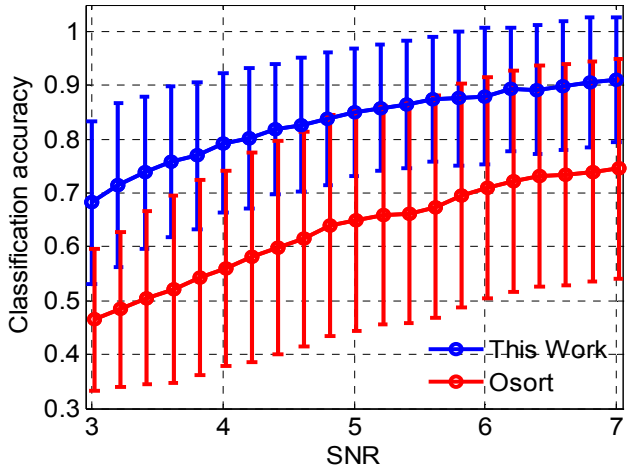


Figure 6.17. Classification accuracy of this work and Osort at different SNRs.

Table 6-2 Performance summary of reported spike sorting NSPs

	Process (nm)	Voltage (V)	Power (μW/channel)	Area (mm^2/channel)
[103]	65	0.27	4.68	0.070
This work	130	1.2	**0.75**	**0.023**

6.5 Conclusion

This chapter presented an integration method to combine each block developed from Chapter 3 to 5 together as a spike sorting NSP module. The scalability of each block was analyzed to share them efficiently for multichannel application. The multichannel spike sorting NSP was designed in Verilog HDL and tested in an FPGA for real-time implementation. The performance of our NSP module achieves 15% higher spike sorting accuracy on average, 6X power and 3X area reduction compared to the current existing work. Results of this work demonstrate that our NSP module is suitable for high-channel-count neural recording microsystems in next-generation BMI applications.

Chapter 7 Summary and Future Work

7.1 Summary

This thesis research developed a spike sorting based neural signal processor (NSP) that overcomes the challenges of preserving useful neural spike information in a compact energy-efficient hardware platform for next-generation high-channel-count BMI applications. Thorough analysis of neural spike and background noise properties was conducted to develop spike sorting algorithms that can maintain high quality neural spike information. Specifically, this work performed statistical analysis on background noise, and spectral analysis of the relationship between neural spikes and background noise. Based on the statistical analysis, an automatic spike detection method was invented to enable highly accurate spike detection across various spike firing rates and noise levels. Based on this spectral analysis, a new noise-insensitive feature set was created allowing distinctive separation among diverse spike shapes. Utilizing the new feature set, a decision tree based method was designed to classify spikes with low area consumption. All algorithms were designed to be computationally efficient and scalable to high-channel-count applications. Finally, by integrating spike sorting algorithms in a power-area efficient manner, a new NSP hardware architecture was designed to process neural signals with scalability to high-channel-count recording systems. The NSP was successfully implemented in Verilog HDL and tested on an FPGA to verify real-time processing capability.

7.2 Contributions

The work in this thesis provides the following significant contributions:

a. Developed the first-known ***hardware-efficient and adaptive signal detection thresholding method*** *for an* ***automatic*** *non-linear energy operator (NEO) based spike*

detector that enables highly accurate spike detection over a wide range of SNRs and spike firing rates.

To set the threshold value adaptively to the noise level, statistical analysis was performed to study the probability density (pdf) function of neural background noise, According to the pdf, the threshold value is demonstrated to be correlated with the standard deviation (Std) and the root mean square (RMS) frequency of background noise. Hardware efficient methodologies were designed to estimate the Std and the RMS frequency such that the threshold value is robust to the firing rate from 10 to 100 Hz. This work implemented the first hardware-efficient spike detector with both noise-adaptive and spike-firing-rate insensitive threshold estimation.

b. Developed a ***hardware-efficient noise-insensitive feature set*** *that provides* ***highest sorting accuracy*** *compared with other existing computationally-efficient feature extraction methods.*

A *Separability* metric was created to analyze frequency spectrum information of neural spikes and background noise. The spectral analysis allows extracting noise-insensitive spike features from best frequency bands. Considering the need for resource-efficient hardware implementation, the weighted haar discrete wavelet transform method was derived to extract features from the desired frequency bands. The new feature extraction method was tested at different signal-to-noise ratios using synthesized datasets consisting of considerable and various spike shapes. The new feature set provides 3-10% better spike sorting performance than other hardware-efficient feature sets while consuming comparable hardware resources.

c. Developed ***the first-known area-efficient quantized oblique decision tree based spike***

classifier *that provides 50% memory usage reduction while achieving the same classification accuracy compared to the traditional classification method.*

A quantized oblique decision tree (DT) classification method was developed to classify spikes where the coefficients of the DT model were quantized into only two bits to reduce both power and area consumption. The new spike classifier stores 50% fewer coefficients than the traditional ℓ_1 norm based classifier, which reduces memory size and hence the area. The DT method achieves classification accuracy within 1% on average compared to the ℓ_1 norm method. The architecture of the spike classifier was implemented in Verilog and found to consume only 10400 μm^2 area per channel.

d. Developed ***the most power-area efficient spike sorting based neural signal processor*** *for high-channel-count application that achieves 15% higher spike sorting accuracy on average, 6X power and 3X area reduction than the current existing work.*

First, a single channel neural signal processor was implemented in Verilog HDL to manage communications among spike sorting blocks. The scalability analysis was then performed on each spike sorting block in terms of power and area tradeoffs to efficiently share hardware resources for multichannel processing. A global controller block was designed to manipulate data flows for multichannel blocks. Finally, a multichannel neural signal processor was implemented and achieves 15% higher accuracy, 6X power and 3X area reduction than the current existing work.

7.3 Future work

This research has developed high accuracy hardware-efficient spike sorting algorithms and implemented a compact energy-efficient spike sorting based neural signal processor for high-channel-count applications. Future work on spike sorting algorithm optimization and decoding analysis would further improve the spike sorting accuracy and hardware

efficiency.

a. Analyze the relationship between the spike detection and feature extraction in neural signal spectrum and develop an algorithm to unify two steps together.

The NEO based spike detection considers neural spikes as instantaneous increase in energy and thus emphasizes the high frequency energy of neural spikes. From the spectral analysis for spike features, some spike types are more noise-insensitive in low frequency band than high frequency band. This indicates that spike detection can also be performed in low frequency band to achieve higher detection accuracy. Thus, one future work will study the optimal low and high frequency bands in terms of overall performance of detection and feature extraction. A hardware-efficient architecture will be designed to combine these two steps together, which would further reduce the hardware complexity of the neural signal processor.

b. Adding new functions to spike sorting algorithms for eliminating interference from artifacts and multi-unit spikes.

Real extracellular recordings include both large amplitude artifacts and small amplitude multi-unit spikes. The artifacts usually originate from some other sources rather than the brain of interest, such as body movements, momentary electrode impedance change and so on. The multi-unit spikes are produced by distant neurons which can be detected but cannot be separated into different clusters due to their relatively small amplitudes. Without artifact removal and multi-unit spikes separation, it is impossible to process real neural signals. Thus, it is critical for spike sorting algorithms to be capable of eliminating these two types of interference for robust performance.

c. Designs optimal spike sorting algorithms in terms of decoding performance.

In real extracellular recordings, there is no ground truth about timestamps of spikes and source neurons that fire spikes. The ultimate goal of spike sorting is to provide high quality information of neuron activities such that neural signals can be accurately decoded into machine commands. Thus, the best way to quantify spike sorting performance for real neural signals is to study the effect of spike sorting results on the decoding performance. In spike detection, it is important to study the impacts of the correctly and falsely detected spikes on decoding performance. This will help to determine the best threshold value. In spike clustering and classification, it is useful to analyze the impact of the number of spike clusters on the decoding performance. The analysis will help to determine the number of spike clusters necessary for decoding and hence improve the hardware efficiency of spike classification.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] "http://www.medicalnewstoday.com/articles/146819.php."

[2] T. N. Taylor, *et al.*, "Lifetime Cost of Stroke in the United States," *Stroke,* vol. 27, pp. 1459-1466, Sep. 1 1996.

[3] K. D. Anderson, "Targeting Recovery: Priorities of the Spinal Cord-Injured Population," *Journal of Neurotrauma,* vol. 21, pp. 1371-1383, Oct. 01 2004.

[4] A. Farin, C. Y. Liu, I. A. Langmoen, and M. L. J. Apuzzo, "Biological Restoration Of Central Nervous System Architecture And Function: Part 3—Stem Cell‐ And Cell‐Based Applications And Realities In The Biological Management Of Central Nervous System Disorders: Traumatic, Vascular, And Epilepsy Disorders," *Neurosurgery,* vol. 65, pp. 831-859, 2009.

[5] B. Graimann, B. Allison, and G. Pfurtscheller, "Brain–Computer Interfaces: A Gentle Introduction," in *Brain-Computer Interfaces*, B. Graimann, G. Pfurtscheller, and B. Allison, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 1-27.

[6] E. V. Evarts, *Relation of pyramidal tract activity to force exerted during voluntary movement* vol. 31, 1968.

[7] J. R. Wolpaw and D. J. McFarland, "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 101, pp. 17849-17854, December 21 2004.

[8] G. Schalk, *et al.*, "Two-dimensional movement control using electrocorticographic signals in humans," *Journal of neural engineering,* vol. 5, p. 75, 2008.

[9] J. M. Carmena, *et al.*, "Learning to Control a Brain–Machine Interface for Reaching and Grasping by Primates," *PLoS Biol,* vol. 1, p. e42, 2003.

[10] J. Wessberg, *et al.*, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature,* vol. 408, pp. 361-365, 2000.

[11] L. R. Hochberg, *et al.*, "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm," *Nature,* vol. 485, pp. 372-375, 2012.

[12] J. L. Collinger, *et al.*, "High-performance neuroprosthetic control by an individual with tetraplegia," *The Lancet,* vol. 381, pp. 557-564, 2013.

[13] T. A. Lebedev MA, Hanson TL, Li Z, O'Doherty JE, Winans JA, Ifft PJ, Zhuang KZ, Fitzsimmons NA, Schwarz DA, Fuller AM, An JH, Nicolelis MA, "Future

developments in brain-machine interface research," *Clinics,* 2011.

[14] R. Michael, O. Iyad, H. C. Stephen, and D. W. Patrick, "A single-chip signal processing and telemetry engine for an implantable 96-channel neural data acquisition system," *J. Neural Eng.,* vol. 4, p. 309, 2007.

[15] H. Miranda*, et al.*, "HermesD: A High-Rate Long-Range Wireless Transmission System for Simultaneous Multichannel Neural Recording Applications," *IEEE Trans. Biomed. Circuits Syst.,* vol. 4, pp. 181-191, 2010.

[16] A. B. David, Y. Ming, A. Juan, and N. Arto, "An implantable wireless neural interface for recording cortical circuit dynamics in moving primates," *J. Neural Eng.,* vol. 10, p. 026010, 2013.

[17] H. Bin, "Neural Engineering," *Springer; 2nd ed,* 2013.

[18] E. V. Evarts, "Relation of pyramidal tract activity to force exerted during voluntary movement," *J. Neurophysiol.,* vol. 31, pp. 14-27, 1968-01-01 00:00:00 1968.

[19] D. R. Humphrey, E. M. Schmidt, and W. D. Thompson, "Predicting Measures of Motor Performance from Multiple Cortical Spike Trains," *Science,* vol. 170, pp. 758-762, Nov. 1970.

[20] A. Georgopoulos, J. Kalaska, R. Caminiti, and J. Massey, "On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex," *The Journal of Neuroscience,* vol. 2, pp. 1527-1537, November 1 1982.

[21] B. S. Oken*, et al.*, "Brain–Computer Interface With Language Model–Electroencephalography Fusion for Locked-In Syndrome," *Neurorehabilitation and Neural Repair,* vol. 28, pp. 387-394, May 1 2014.

[22] L. Karl*, et al.*, "Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface," *Journal of neural engineering,* vol. 10, p. 046003, 2013.

[23] N. Rowland, J. Breshears, and E. Chang, "Neurosurgery and the dawning age of Brain-Machine Interfaces," *Surgical Neurology International,* vol. 4, pp. 11-14, 2013.

[24] A. B. Schwartz, X. T. Cui, Douglas J. Weber, and D. W. Moran, "Brain-Controlled Interfaces: Movement Restoration with Neural Prosthetics," *Neuron,* vol. 52, pp. 205-220, 2006.

[25] N. Birbaumer, "Brain–computer-interface research: Coming of age," *Clinical Neurophysiology,* vol. 117, pp. 479-483, 2006.

[26] G. Baranauskas, "What limits the performance of current invasive Brain Machine Interfaces?," *Frontiers in Systems Neuroscience,* vol. 8, Apr. 29 2014.

[27] G. Schalk and E. C. Leuthardt, "Brain-Computer Interfaces Using Electrocorticographic Signals," *Biomedical Engineering, IEEE Reviews in,* vol. 4, pp. 140-154, 2011.

[28] C. L. Eric*, et al.*, "A brain–computer interface using electrocorticographic signals in humans," *Journal of neural engineering,* vol. 1, p. 63, 2004.

[29] W. Wang*, et al.*, "An Electrocorticographic Brain Interface in an Individual with Tetraplegia," *PLoS ONE,* vol. 8, p. e55344, 2013.

[30] S. I. Ryu and K. V. Shenoy, "Human cortical prostheses: lost in translation?," *Neurosurgical FOCUS,* vol. 27, p. E5, July 01 2009.

[31] J. D. Simeral*, et al.*, "Neural control of cursor trajectory and click by a human with tetraplegia 1000 days after implant of an intracortical microelectrode array," *Journal of neural engineering,* vol. 8, p. 025027, 2011.

[32] G. Santhanam*, et al.*, "A high-performance brain–computer interface," *Nature,* vol. 442, pp. 195-198, 2006.

[33] M. Velliste*, et al.*, "Cortical control of a prosthetic arm for self-feeding," *Nature,* vol. 453, pp. 1098-1101, 2008.

[34] C. T. Moritz, S. I. Perlmutter, and E. E. Fetz, "Direct control of paralysed muscles by cortical neurons," *Nature,* vol. 456, pp. 639-642, 2008.

[35] B. Wodlinger*, et al.*, "Ten-dimensional anthropomorphic arm control in a human brain−machine interface: difficulties, solutions, and limitations," *J. Neural Eng.,* vol. 12, p. 016011, 2015.

[36] V. Gilja*, et al.*, "Clinical translation of a high-performance neural prosthesis," *Nat Med,* vol. 21, pp. 1142-1145, 2015.

[37] M. A. Lebedev*, et al.*, "Future developments in brain-machine interface research," *Clinics,* vol. 66, pp. 25-32, 2011.

[38] M. N. Shadlen and W. T. Newsome, "Noise, neural codes and cortical organization," *Current Opinion in Neurobiology,* vol. 4, pp. 569-579, 1994.

[39] B. Cessac, H. Paugam-Moisy, and T. Viéville, "Overview of facts and issues about neural coding by spikes," *Journal of Physiology-Paris,* vol. 104, pp. 5-18, 2010.

[40] E. Chorev*, et al.*, "Electrophysiological recordings from behaving animals—going beyond spikes," *Current Opinion in Neurobiology,* vol. 19, pp. 513-519, 2009.

[41] J. Csicsvari, M. Penttonen, J. Hetke, and K. Wise, *Extracellular recording and analysis of neuronal activity: from single cells to ensembles*, 1998.

[42] A. Belitski*, et al.*, "Low-Frequency Local Field Potentials and Spikes in Primary Visual Cortex Convey Independent Visual Information," *The Journal of Neuroscience,* vol. 28, pp. 5696-5709, May 28 2008.

[43] C. Gold, D. A. Henze, C. Koch, and G. Buzsáki, *On the Origin of the Extracellular Action Potential Waveform: A Modeling Study* vol. 95, 2006.

[44] D. A. Henze*, et al.*, *Intracellular Features Predicted by Extracellular Recordings in the Hippocampus In Vivo* vol. 84, 2000.

[45] E. E. Fetz, "Operant Conditioning of Cortical Unit Activity," *Science,* vol. 163, pp. 955-958, Feb. 28 1969.

[46] E. Schmidt, "Single neuron recording from motor cortex as a possible source of signals for control of external devices," *Annals of Biomedical Engineering,* vol. 8, pp. 339-349, July 01 1980.

[47] A. Schwartz, R. Kettner, and A. Georgopoulos, "Primate motor cortex and free arm movements to visual targets in three- dimensional space. I. Relations between single cell discharge and direction of movement," *The Journal of Neuroscience,* vol. 8, pp. 2913-2927, Aug. 01 1988.

[48] K. Najafi and K. D. Wise, "An implantable multielectrode array with on-chip signal processing," *Solid-State Circuits, IEEE Journal of,* vol. 21, pp. 1035-1044, 1986.

[49] M. S. J. Steyaert and W. M. C. Sansen, "A micropower low-noise monolithic instrumentation amplifier for medical purposes," *Solid-State Circuits, IEEE Journal of,* vol. 22, pp. 1163-1168, 1987.

[50] M. G. Dorman, M. A. Prisbe, and J. D. Meindl, "A monolithic signal processor for a neurophysiological telemetry system," *Solid-State Circuits, IEEE Journal of,* vol. 20, pp. 1185-1193, 1985.

[51] J. Ji and K. D. Wise, "An implantable CMOS circuit interface for multiplexed microelectrode recording arrays," *Solid-State Circuits, IEEE Journal of,* vol. 27, pp. 433-443, 1992.

[52] H. J. Song, D. R. Allee, and K. T. Speed, "Single chip system for bio-data acquisition, digitization and telemetry," in *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on*, 1997, pp. 1848-1851 vol.3.

[53] T. Akin, K. Najafi, and R. M. Bradley, "A wireless implantable multichannel digital neural recording system for a micromachined sieve electrode," *Solid-State*

Circuits, IEEE Journal of, vol. 33, pp. 109-118, 1998.

[54] R. H. O. H.Yu, K. D.Wise, and K. Najafi, "A wireless microsystem for multichannel neural recording microprobes," *Proc. Solid-State Sensor Actuator Microsystems Workshop,* pp. 107-110, 2004.

[55] P. Mohseni, K. Najafi, S. J. Eliades, and W. Xiaoqin, "Wireless multichannel biopotential recording using an integrated FM telemetry circuit," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on,* vol. 13, pp. 263-271, 2005.

[56] R. R. Harrison*, et al.*, "Wireless Neural Recording With Single Low-Power Integrated Circuit," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on,* vol. 17, pp. 322-329, 2009.

[57] A. M. Sodagar*, et al.*, "An Implantable 64-Channel Wireless Microsystem for Single-Unit Neural Recording," *Solid-State Circuits, IEEE Journal of,* vol. 44, pp. 2591-2604, 2009.

[58] G. Buzsaki, "Large-scale recording of neuronal ensembles," *Nat Neurosci,* vol. 7, pp. 446-451, 2004.

[59] R. J. Chandler*, et al.*, "A system-level view of optimizing high-channel-count wireless biosignal telemetry," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, 2009, pp. 5525-5530.

[60] R. R. Harrison, "The Design of Integrated Circuits to Observe Brain Activity," *Proceedings of the IEEE,* vol. 96, pp. 1203-1216, 2008.

[61] R. R. Harrison*, et al.*, "A Low-Power Integrated Circuit for a Wireless 100-Electrode Neural Recording System," *IEEE J. Solid-State Circuits,* vol. 42, pp. 123-133, 2007.

[62] A. M. Sodagar, K. D. Wise, and K. Najafi, "A fully-integrated mixed-signal neural processing module for implantable multi-channel cortical recording," in *Biomedical Circuits and Systems Conference*, 2006, pp. 37-40.

[63] R. H. Olsson, III and K. D. Wise, "A three-dimensional neural recording microsystem with implantable data compression circuitry," *IEEE J. Solid-State,* vol. 40, pp. 2796-2804, 2005.

[64] Q. R. Quiroga, "spike sorting," *Scholarpedia,* 2007.

[65] B. Gosselin*, et al.*, "A Mixed-Signal Multichip Neural Recording Interface With Bandwidth Reduction," *IEEE Trans. Biomed. Circuits Syst.,* vol. 3, pp. 129-141, 2009.

[66] M. A. Shaeri, A. M. Sodagar, and H. Abrishami-Moghaddam, "A 64-channel neural signal processor/ compressor based on Haar wavelet transform," in *Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2011, pp. 6409-6412.

[67] K. G. Oweiss*, et al.*, "A Scalable Wavelet Transform VLSI Architecture for Real-Time Signal Processing in High-Density Intra-Cortical Implants," *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 54, pp. 1266-1278, 2007.

[68] A. M. Kamboh, M. Raetz, K. G. Oweiss, and A. Mason, "Area-Power Efficient VLSI Implementation of Multichannel DWT for Data Compression in Implantable Neuroprosthetics," *IEEE Trans. Biomed. Circuits Syst.,* vol. 1, pp. 128-135, 2007.

[69] A. M. Kamboh, K. G. Oweiss, and A. J. Mason, "Resource constrained VLSI architecture for implantable neural data compression systems," in *IEEE International Symposium on Circuits and Systems*, 2009, pp. 1481-1484.

[70] H. Hosseini-Nejad, A. Jannesari, and A. M. Sodagar, "Data Compression in Brain-Machine/Computer Interfaces Based on the Walsh-Hadamard Transform," *IEEE Trans. Biomed. Circuits Syst.,* vol. 8, pp. 129-137, 2014.

[71] R. Chandra and L. M. Optican, "Detection, classification, and superposition resolution of action potentials in multiunit single-channel recordings by an on-line real-time neural network," *IEEE Trans. Biomed. Eng.,* vol. 44, pp. 403-412, 1997.

[72] S. N. Gozani and J. P. Miller, "Optimal discrimination and classification of neuronal action potential waveforms from multiunit, multichannel recordings using software-based linear filters," *IEEE Trans. Biomed. Eng.,* vol. 41, pp. 358-372, 1994.

[73] D. A. Schwarz*, et al.*, "Chronic, wireless recordings of large-scale brain activity in freely moving rhesus monkeys," *Nat Meth,* vol. 11, pp. 670-676, 2014.

[74] M. S. Chae*, et al.*, "A 128-Channel 6 mW Wireless Neural Recording IC With Spike Feature Extraction and UWB Transmitter," *IEEE Trans. Neural Syst. and Rehab. Eng.,* vol. 17, pp. 312-321, 2009.

[75] V. Karkare, S. Gibson, and D. Markovic, "A 130uW, 64-Channel Neural Spike-Sorting DSP Chip," *IEEE J. Solid-State,* vol. 46, pp. 1214-1222, 2011.

[76] R. Q. Quiroga, "Spike sorting," *Scholarpedia,* 2007.

[77] K. A. Ludwig*, et al.*, "Chronic neural recordings using silicon microelectrode arrays electrochemically deposited with a poly(3,4-ethylenedioxythiophene) (PEDOT) film," *J. Neural Eng.,* vol. 3, pp. 59-70, Mar 2006.

[78] J. F. Kaiser, "On a simple algorithm to calculate the `energy' of a signal," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process*, 1990, pp. 381-384.

[79] K. K. 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 Trans. Biomed. Eng.,* vol. 47, pp. 1406-1411, 2000.

[80] J. H. Choi, H. K. Jung, and T. Kim, "A new action potential detector using the MTEO and its effects on spike sorting systems at low signal-to-noise ratios," *IEEE Trans. Biomed. Eng.,* vol. 53, pp. 738-746, 2006.

[81] S. Mukhopadhyay and G. C. Ray, "A new interpretation of nonlinear energy operator and its efficacy in spike detection," *IEEE Trans. Biomed. Eng.,* vol. 45, pp. 180-187, 1998.

[82] B. Gosselin and M. Sawan, "An Ultra Low-Power CMOS Automatic Action Potential Detector," *IEEE Trans. Neural Syst. and Rehab. Eng.,* vol. 17, pp. 346-353, 2009.

[83] E. Koutsos, S. E. Paraskevopoulou, and T. G. Constandinou, "A 1.5 uW NEO-based spike detector with adaptive-threshold for calibration-free multichannel neural interfaces," in *IEEE Int. Symp. Circuits and Systems*, 2013, pp. 1922-1925.

[84] I. Obeid and P. D. Wolf, "Evaluation of spike-detection algorithms for a brain-machine interface application," *IEEE Trans. Biomed. Eng.,* vol. 51, pp. 905-911, 2004.

[85] R. J. Brychta*, et al.*, "Wavelet Methods for Spike Detection in Mouse Renal Sympathetic Nerve Activity," *IEEE Trans. Biomed. Eng.,* vol. 54, pp. 82-93, 2007.

[86] Y. Yang, C. S. Boling, A. M. Kamboh, and A. J. Mason, "Adaptive Threshold Neural Spike Detector Using Stationary Wavelet Transform in CMOS," *IEEE Trans. Neural Syst. and Rehab. Eng.,* vol. 23, pp. 946-955, 2015.

[87] K. H. Kim and S. J. Kim, "A wavelet-based method for action potential detection from extracellular neural signal recording with low signal-to-noise ratio," *IEEE Trans. Biomed. Eng.,* vol. 50, pp. 999-1011, 2003.

[88] Y. Yang, A. Kamboh, and A. J. Mason, "Adaptive threshold spike detection using stationary wavelet transform for neural recording implants," in *IEEE Biomed. Circuits and Syst. Conf. ,* , 2010, pp. 9-12.

[89] Y. Zhi*, et al.*, "A new EC–PC threshold estimation method for in vivo neural spike detection," *J. Neural Eng.,* vol. 9, p. 046017, 2012.

[90] Y. Zhou*, et al.*, "On the robustness of EC–PC spike detection method for online neural recording," *J. Neuroscience Methods,* vol. 235, pp. 316-330, 2014.

[91] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Computation,*

vol. 16, pp. 1661-1687, Aug 2004.

[92] 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, pp. 53-78, Nov 1998.

[93] A. Zviagintsev, Y. Perelman, and R. Ginosar, "Algorithms and architectures for low power spike detection and alignment," *J. Neural Eng.,* vol. 3, p. 35, 2006.

[94] A. M. Kamboh and A. J. Mason, "Computationally Efficient Neural Feature Extraction for Spike Sorting in Implantable High-Density Recording Systems," *IEEE Trans. Neural Syst. and Rehab. Eng.,* vol. 21, pp. 1-9, 2013.

[95] S. Gibson, J. W. Judy, and D. Markovic, "Technology-Aware Algorithm Design for Neural Spike Detection, Feature Extraction, and Dimensionality Reduction," *IEEE Trans. Neural Syst. and Rehab. Eng.,* vol. 18, pp. 469-478, 2010.

[96] M. Zamani and A. Demosthenous, "Feature Extraction Using Extrema Sampling of Discrete Derivatives for Spike Sorting in Implantable Upper-Limb Neural Prostheses," *IEEE Trans. Neural Syst. and Rehab. Eng.,* vol. PP, pp. 1-1, 2014.

[97] S. E. Paraskevopoulou*, et al.*, "Feature extraction using first and second derivative extrema (FSDE) for real-time and hardware-efficient spike sorting," *J. Neuroscience Methods,* vol. 215, pp. 29-37, 2013.

[98] M. Salganicoff, M. Sarna, L. Sax, and G. L. Gerstein, "Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. I. Algorithms and implementation," *J. Neuroscience Methods,* vol. 25, pp. 181-187, 1988.

[99] M. Sahani, "Latent Variable Models for Neural Data Analysis," *Ph.D. Dissertation,* 1999.

[100] D. G. S. Kadir, and K. Harri, "High-dimensional cluster analysis with the Masked EM Algorithm," *arXiv.org,* 2013.

[101] S. Shoham, M. R. Fellows, and R. A. Normann, "Robust, automatic spike sorting using mixtures of multivariate t-distributions," *J. Neuroscience Methods,* vol. 127, pp. 111-122, 2003.

[102] U. Rutishauser, E. M. Schuman, and A. N. Mamelak, "Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, in vivo," *J. Neuroscience Methods,* vol. 154, pp. 204-224, 2006.

[103] V. Karkare, S. Gibson, and D. Markovic, "A 75uW, 16-Channel Neural Spike-Sorting Processor With Unsupervised Clustering," *IEEE J. Solid-State,* vol. 48, pp. 2230-2238, 2013.

[104] C. Tung-Chien, L. Wentai, and C. Liang-Gee, "128-channel spike sorting processor with a parallel-folding structure in 90nm process," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, 2009, pp. 1253-1256.

[105] A. Mendez, A. Belghith, and M. Sawan, "A DSP for Sensing the Bladder Volume Through Afferent Neural Pathways," *IEEE Trans. Biomed. Circuits Syst.,* vol. 8, pp. 552-564, 2014.

[106] H. Semmaoui, J. Drolet, A. Lakhssassi, and M. Sawan, "Setting Adaptive Spike Detection Threshold for Smoothed TEO Based on Robust Statistics Theory," *IEEE Trans. Biomed. Eng.,* vol. 59, pp. 474-482, 2012.

[107] D. Dimitriadis, A. Potamianos, and P. Maragos, "A Comparison of the Squared Energy and Teager-Kaiser Operators for Short-Term Energy Estimation in Additive Noise," *IEEE Trans. Signal Process.,* vol. 57, pp. 2569-2581, 2009.

[108] A. E. Barnes, "Instantaneous spectral bandwidth and dominant frequency with applications to seismic reflection data," *Geophysics,* vol. 58, pp. 419-428, 1993.

[109] J. Gasthaus, F. Wood, D. Gorur, and Y. W. Teh, "Dependent Dirichlet Process Spike Sorting," in *Advances in Neural Information Processing Systems 21*, 2008, pp. 497-504.

[110] S. Shahid, J. Walker, and L. S. Smith, "A New Spike Detection Algorithm for Extracellular Neural Recordings," *IEEE Trans. Biomed. Eng.,* vol. 57, pp. 853-866, 2010.

[111] A. Calabrese and L. Paninski, "Kalman filter mixture model for spike sorting of non-stationary data," *J. Neuroscience Methods,* vol. 196, pp. 159-169, 2011.

[112] H. Yamasaki and T. Shibata, "A high-speed median filter VLSI using floating-gate-MOS-based low-power majority voting circuits," in *Proc. 31st European Solid-State Circuits Conf.*, 2005, pp. 125-128.

[113] M. D. Linderman*, et al.*, "Neural Recording Stability of Chronic Electrode Arrays in Freely Behaving Primates," in *Proc. 28th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2006, pp. 4387-4391.

[114] A. Zviagintsev, Y. Perelman, and R. Ginosar, "Low-Power Architectures for Spike Sorting," in *2nd International IEEE EMBS Conference on*, 2005, pp. 162-165.

[115] K. Mizuseki, A. Sirota, E. Pastalkova, and G. Buzsáki, "Theta Oscillations Provide Temporal Windows for Local Circuit Computation in the Entorhinal-Hippocampal Loop," *Neuron,* vol. 64, pp. 267-280, 2009.

[116] C. Bédard, H. Kröger, and A. Destexhe, "Modeling Extracellular Field Potentials

and the Frequency-Filtering Properties of Extracellular Space," *Biophysical Journal,* vol. 86, pp. 1829-1842, 2004.

[117] K. H. Pettersen and G. T. Einevoll, "Amplitude Variability and Extracellular Low-Pass Filtering of Neuronal Spikes," *Biophysical Journal,* vol. 94, pp. 784-802, 2008.

[118] M. S. Fee, P. P. Mitra, and D. Kleinfeld, "Variability of extracellular spike waveforms of cortical neurons," *J. Neurophysiol,* vol. 76, pp. 3823-3833, 1996.

[119] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*: Wiley-Interscience, 2000.

[120] M. C. P. de Souto*, et al.*, "Comparative study on normalization procedures for cluster analysis of gene expression datasets," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2008, pp. 2792-2798.

[121] J. Navajas*, et al.*, "Minimum requirements for accurate and efficient real-time on-chip spike sorting," *J. Neuroscience Methods,* vol. 230, pp. 51-64, 2014.

[122] S. Gibson, J. W. Judy, and D. Markovic, "Technology-Aware Algorithm Design for Neural Spike Detection, Feature Extraction, and Dimensionality Reduction," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on,* vol. 18, pp. 469-478, 2010.

[123] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *J. Artif. Int. Res.,* vol. 2, pp. 1-32, 1994.

[124] C. Pedreira, J. Martinez, M. J. Ison, and R. Quian Quiroga, "How many neurons can we see with current spike sorting algorithms?," *J. Neuroscience Methods,* vol. 211, pp. 58-65, 2012.

[125] J. Dragas, D. Jäckel, A. Hierlemann, and F. Franke, "Complexity Optimization and High-Throughput Low-Latency Hardware Implementation of a Multi-Electrode Spike-Sorting Algorithm," *IEEE Trans. Neural Syst. and Rehab. Eng.,* vol. 23, pp. 149-158, 2015.

[126] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology),* vol. 63, pp. 411-423, 2001.