Computationally Efficient Adaptive Spike Processor With Real-Time Decoding of Neural Signals for Implantable Applications

Majid Zamani

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy
Of
University College London

Department of Electrical Engineering
University College London

Supervisor:
Prof. Andreas Demosthenous

February 2017

Declaration

I herewith certify that all material in this dissertation which is not my own work has been properly acknowledged.

Majid Zamani

Copyright

Abstract

Recent advances in the field of neuroscience have suggested that new generation brain computer interfaces demand a critical step in biomedical signal processing requiring *online/on-chip spike sorting*. Spike sorting is the process of grouping signals from an individual neuron by grouping action potentials (spikes) into a specific cluster based on the similarity of their shapes. The extraction of single-unit activity by sensors at a distance from specific neurons is necessary for a wide range of clinical applications such as disorder treatments, muscular stimulation (e.g., epidural spinal cord stimulation for treatment acceleration), cochlear implant and neural prostheses. A brain machine interface, for example, can potentially substitute the missing motor pathway/sensory information between the motor cortex and an artificial limb.

With the aim of developing an energy-efficient spike sorting chip for hardware implantable systems, this thesis introduces a new feature extraction method based on extrema analysis (positive and negative peaks) of spike shapes and their discrete derivatives. The proposed method runs in real-time and does not require any offline training. Compared to other methods it offers a better tradeoff between accuracy and computational complexity using online sorting. It additionally eliminates multiplications which are computationally expensive, power hungry and require appreciable silicon area.

A minimum power limit for implantable neural front-end interfaces is also derived. It involved: 1) system level optimization - the front-end specifications including the bandwidth, data converter resolution and sampling rate were defined by exploring the effect of the parameters on spike sorting via a standard spike bank; 2) block level optimization - The front-end power was minimized by using an opamp-less cyclic converter; and 3) estimating the power limit equation of the front-end. The new optimization methodology addresses the future demands of neural recording interfaces.

Finally the thesis presents the design, implementation and testing of the first generation of an adaptive spike sorting processor. It enhances the accuracy-power characteristics by employing self-calibration of processing features. The chip prototype was fabricated in a 180-nm CMOS technology. It achieves an overall clustering accuracy of 84.5% using a standard spike data bank and has a power consumption of 148-µW from 1.8-V supply voltage. The fabricated spike

processor has almost 10% higher clustering accuracy than the state-of-the-art. Measurements show good power-performance characteristics compared to the state-of-the-art online and offline clustering methods.

To my parents

Acknowledgments

I would like to thank Prof. Andreas Demosthenous for his constant guidance, support, patience, and contribution throughout this work. It has been an honor to work with him on my Ph.D. dissertation during which I learned a lot from him both academically and professionally. He encouraged me to develop my own ideas with his positive spirit and great patience to find the direction towards the right path.

A special thanks goes to my second thesis advisor Dr. Miguel Rodriguez for his inspiring discussions which helped me to understand to successfully conduct my thesis. I am especially thankful to Prof. Nick Donaldson from the Implanted Devices Group at University College London for providing the recorded in-vivo neural data.

I would like to thank Dr. Dai Jiang for his support regarding chip tape-out and layout of the spike processor. I am grateful to Peter Langlois who provided a lot of input to my papers and thesis corrections. This project was financially supported by a Ph.D. scholarship from University College London (UCL) Graduate School.

Last, but not least, my deep gratitude goes to my beloved family for their love and encouragement and their continuous faithful support during all phases of my life. Their timeless and countless supports were always motivating during my education. Indeed, they have a big share to this achievement and I would like to express my deep appreciation.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABS: Absolute Value

ADC: Analog to Digital Converter

ADDs: Adaptive Discrete Derivatives

ANN: Artificial Neural Networks

AS: Asynchronous Sampling

ASIC: Application-Specific Integrated Circuit

BMI: Brain Machine Interface

BW: Bandwidth

CBSC: Comparator-Base Switched-Capacitor

CMOS: Complementary Metal-Oxide-Semiconductor

CS: Compressed Sensing

DAC: Digital to Analog Converter

DBS: Deep Brain Stimulation

DDs: Discrete Derivatives

DDS-MDT: Discrete Derivatives and Maximum Difference Test

DDS-USAMP: Discrete Derivatives and Uniform Sampling

DFFs: D-type Flip-Flops

DLC: Dynamic Latch Comparator

DR: Dimensionality Reduction

DSP: Digital Signal Processing

DWT: Discrete Wavelet Transform

EAPs: Extracellular Action Potentials

ECG: Electrocardiography

ECoG: Electrocorticography

EEG: Electroencephalography

EM: Expectation Maximization

ENOB: Effective Number of Bits

FDV: First Derivative

FDVSDV: First and Second Derivatives

FE: Feature Extraction

FoM: Figure of Merit

FPGA: Field-Programmable Gate Array

FR: Firing Rate

FV: Feature Vector

GLF: Graph Laplacian Feature

HAL: Hybrid Assistive Limb

KS: Kolmogorov-Smirnov

ICA: Independent Component Analysis

IRN: Input Referred Noise

ISIH: Inter-Spike Interval Histogram

IT: Integral Transform

LFPs: Local Field Potentials

LNA: Low Noise Amplifier

LUT: Look-Up-Table

MAF: Moving Average Filtering

MDT: Maximum Difference Test

MEAs: Multi-Electrode Arrays

MUA: Multi-Unit Activity

NEF: Noise Efficiency Figure

NEO: Non-linear Energy Operator

NFI: Neural Front-end Interface

NMF: Noise Multiplying Factor

NOSPC: Number of Spikes per Cluster

NSR: Noise-to-Signal Ratio

OQ: Optimal Quantization

O-Sort: Online Sorting

PCA: Principal Components Analysis

PGA: Programmable Gain Amplifier

RA: Reverse-Adjustment

RF: Radio Frequency

SAR: Successive Approximation Registry

SD: Standard Deviation

SDV: Second Derivative

SNQR: Signal -to- Noise and Quantization Ratio

SNR: Signal-to-Noise Ratio

SPC: Superparamagnetic Clustering

SPSs: Synchronous Processing Systems

SThr: Sorting Threshold

SUA: Single-Unit Activity

SWTP: Stationary Wavelet Transform Product

TDC: Threshold Detection Comparator

TEO: Teager Energy Operator

TM: Template Matching

UART: Universal Asynchronous Receiver / Transmitter

USAMP: Uniform Sampling

ZCB: Zero Crossing Based

ZCF: Zero - Crossing Features

CHAPTER 1

Introduction

1.1 Motivation

Millions of people suffer from different neurodegenerative diseases around the world [1]. The consequence of such diseases is the devastating loss-of-function and resulting emotional problems such as depression for the patients and their families. In addition, treatment of such diseases requires significant financial support. For example, there were approximately 45 million cases of brain disorders in the UK, with a cost of £110 billion per annum [2]. The five most costly disorders were dementia, psychotic disorders, mood disorders, addiction and anxiety disorders. In addition, figures show that there are currently 127,000 people diagnosed with Parkinson's disease in the UK and this is predicted to rise to 162,000 by 2020, an increase of 28% [3].

One way of curing and managing diseases is the prescription of appropriate medication. It should be noted that this treatment method is not very effective for types of disease such as Alzheimer's. For example, medicines (e.g., acetylcholine) can ease the Alzheimer's symptoms and slow down the progress of the disease, but the effect lasts for a limited time with possible side effects such as diarrhea, vomiting, insomnia and fatigue. On the other hand, the application of therapeutic devices in the category of alternative treatment have offered more efficient and reliable treatment (e.g., in the case of Parkinson's disease) with promising results and fewer side effects [4].

Detailed understanding of neuron-related activities such as the generation of thoughts/ perceptions/actions remains an important challenge for improved alternative treatments using neuroprosthetic devices. The first step in any alternative treatment is the ability of interfacing and decoding the interactions between neurons. This ability has already significantly changed the development path of neuroprosthetic devices. Typically neuroprosthetic devices are categorized into: 1) decoding sensory information (stimulation patterns) which is used for rehabilitation (e.g., spinal cord injury) or to lessen symptoms (e.g., Parkinson's); and 2) devices for extraction of motor pathways for controlling assistive technologies such as a prosthetic hand. There are nearly 2 million people living with limb-loss in the United States; the main cause is diabetes [5].

Approximately 185,000 amputations occur in the United States each year and in 2009 [6], hospital costs associated with amputation totaled more than $8.3 billion [7]. An increased projection of 2 million cases is estimated from 2005 to 2050 for those who require amputation surgery [5].

Studies in the field of neuroscience have suggested that the realization of neuroprosthetic devices for motor/sensory applications are significant within the concept of brain machine interfaces (BMIs). The BMI concept is an interpreter for analysis of the characteristics of active neurons in the acquired data such as the number of neurons, their firing rates and the degree of correlation between the identified neurons. These features help to build application-specific experimental models (e.g., for neurodegenerative disease treatments). Nowadays reliable acquisition of high channel count recording is possible thanks to astonishing advancements in microtechnology such as interfacing probes and CMOS integrated circuits [8-10]. The quality of neural data monitoring depends on the level of invasiveness including: (1) electroencephalography (EEG); (2) electrocortiography (ECoG); and (3) multi-electrode arrays (MEAs) with sub-micron resolution features which all will be extensively discussed in Chapter 2.

The science of integration in the field of recording technology in conjunction with nanostructure fabrication techniques (e.g., Michigan arrays) have significantly increased the number of recording sites. The monitoring of the brain functionality down to individual neuron level [referred to as extracellular action potentials (EAPs) or spikes] allows the study of the underlying network dynamics. Implantable neural recording systems follow a modified Moore's law as shown in Figure 1.1, where the number of recording channels since the 1960s has exponentially grown. The number of neurons recorded doubles every 7 years [11].

On the other hand, the vast amount of data recorded from the distributed recording sites introduce fundamental limitations in either processing data using highly complex processing methods or transmission of raw data to an external processing unit. In implantable devices there is a limited power budget which limits the type of processing that can be included. Hence high channel count processing of data using complex methods is extremely difficult. In addition, transmitting large quantities of data to external units results in a transmitter bandwidth bottleneck. To alleviate the issues related to high channel count monitoring schemes, reduction of the data prior to transmission is necessary. This can be achieved by *on-chip spike sorting*. Spike sorting is the process of identifying individual neurons from the multiple signals sensed by an electrode tip. This process

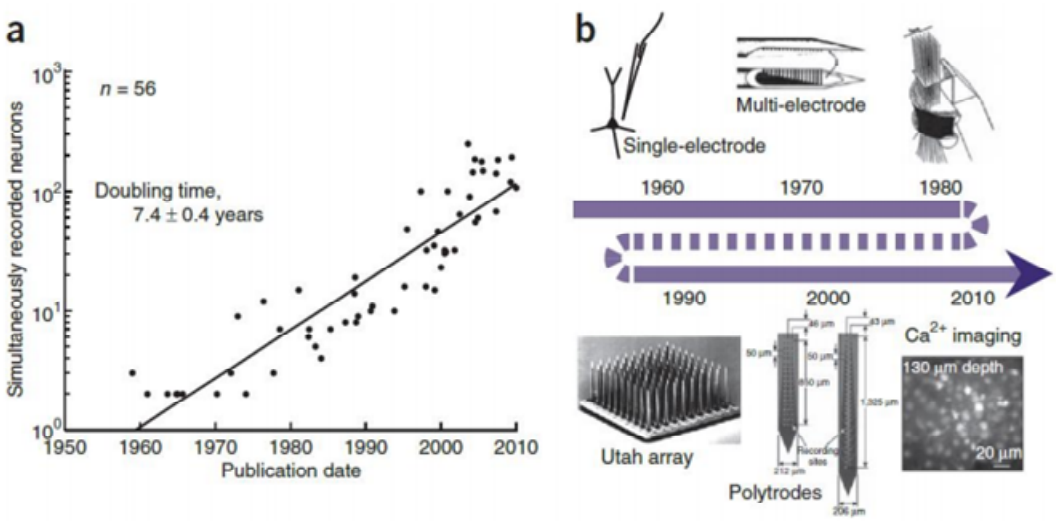

Figure1.1: Examining 56 studies of neural recording systems published over the last five decades. (a) Number of simultaneously recorded neurons; (b) Timeline of recording technologies. Adopted from [11].

aims to extract some distinguishable features of each neuron and to subsequently classify each neuron to its source of origin. The research in this thesis is primarily devoted to providing advanced processing using low power techniques suitable for implantable on-chip spike sorting. In addition to overcoming these issues, retaining the useful information regarding active neurons can be utilized for application-specific BMIs. The spike sorting concept has its inherent signal processing complexities; hence employing traditional power hungry algorithms such as principle component analysis (PCA) are not suitable.

1.2 Research Objectives

Within this context, there is still a gap to fill in the realization of real-time, on-chip and highly accurate spike sorting. Processing specifications are crucial in order to fulfill closed-loop and open-loop application requirements. For example, in a closed-loop application (e.g., spinal cord paralysis rehabilitation), it is important to develop a highly efficient on-chip sorting method for generating stimulation commands with minimum delay.

The research in this thesis aims to introduce an on-chip spike sorting framework from theory to hardware design and implementation which is scalable with the number of recording sites and has

high efficiency in terms of resource utilization to fulfill implantable applications in the future generation of BMIs. The research has the following objectives:

- To develop a feature extraction method which outperforms the available established state-of-the-art. In addition to power efficiency, another concern relies on the fact that the simplicity in spike waveform transformation should not result in information fidelity loss which would directly affects the clustering performance. Other questions considered are about scalability and reconfigurabilty (adaptivity) of feature extraction. An online method with low complexity is of particular interest when a large number of channels are monitored. This thesis develops a feature extraction methodology which provides better power-area-accuracy compared to the state-of-the-art.
- To identify the neural front-end interface (NFI) theoretical minimum power limit taking account of all the analog processing chain key stages (amplification, filtering and digitization) together with the spike sorting process. The NFI specifications have direct effect on the performance of the spike sorting unit. This study has three important potentials including: 1) proper resource utilization in NFI design parameters without compromising the spike sorting performance; 2) pushing the power envelope towards the predicted NFI power limit using novel digitization methods; and 3) introducing a desired power region for future NFI design.
- To propose a new processing approach in the context of adaptivity for on-chip sorting utilization. Spike sorting is a computationally demanding signal processing technique due to its inherent challenges, thus developing a novel hardware implantable processing scheme with performance self-tuning and inherent power suppression capabilities is required. To this extent, investigation of design methods and signal processor structures with reference to “adaptivity” or “reconfigurability” to complement the conventional synchronous digital signal processors (DSPs) is important. An adaptive processor is capable of shaping its parameters (behaviors) according to the property of the input neural data stream. Embedding the adaptive framework in the classical synchronous processing systems makes the processor operation “signal-dependent” which offers various benefits from accuracy tuning to asynchronous power management.

1.3 Outline of the Thesis

The thesis proposes an advanced and effective algorithm for improving spike sorting performance in its major stages, namely, feature extraction and clustering. The aim is to have the least amount of supervision and complexity whilst obtaining high clustering performance. The feature extraction and clustering of neural waveforms and the proposed approach to improve the results at each stage are discussed. The thesis has the following chapters:

- Chapter 1 has discussed the motivation, identifying the solution and laying out the objectives of the research.
- Chapter 2 provides the necessary background information on the subject of BMI. Some physiological background regarding the nature of neural signals is discussed. The spike sorting is defined and explained as a key component within the context of BMIs which gives benefit for many important applications. The chapter continues with a general discussion on spike sorting and highlights the state-of-the-art. Study of the published material demonstrates a shift from traditional offline spike sorting to online sorting, and even performing in a mathematically mapped domain (e.g., compressed domain) to cluster the recorded neural data. To understand the detailed operation of the sorting process, relatively high level descriptions of the spike sorting algorithms for each processing unit are presented. The inherent associated challenges in the whole processing chain confirm the fact that the spike sorting is not a trivial task.
- In Chapter 3, a new feature extraction algorithm suitable for implantable implementation is described. The reasoning behind the use of this method is discussed at the beginning of this chapter and will be highlighted throughout the chapter. The proposed method is based on extrema analysis (positive and negative peaks) of spike shapes and their discrete derivatives. It runs in real-time and does not require any offline training. The proposed method was simulated and compared with other feature extraction algorithms using a standard neural database and well-established performance metrics. Compared to other methods it offers a better tradeoff between accuracy and computational complexity using an online sorting clustering method.
- Chapter 4 aims at developing an optimization methodology for high channel count (e.g., 1,000) recording and processing of neural data. A NFI including a low noise amplifier, a programmable gain amplifier and a data converter are considered to derive the theoretical

minimum power limit of the recording system. This chapter shows the possibility of further power reduction through utilization of more advanced techniques (i.e., CBSC [12]) for data converter implementation. Since the focus of this research is towards on-chip spike sorting, the effect of NFI key parameters on spike sorting performance is quantified. This is performed via sweeping the considered parameters (e.g., amplifier bandwidth and bit resolution) to evaluate the sorting performance. This suggests a design must not only investigate the sensitivity of NFI parameters on sorting performance but also optimize the use of hardware resources. Finally, the derived theoretical NFI power bound using two types of analog-to-digital converter are analyzed without compromising the spike sorting performance. In addition, the desired power region for the design of ultra-low power NFIs is defined and possible developments are briefly discussed.

- In Chapter 5 an adaptive processing methodology is introduced to enhance the accuracy-power characteristics by employing self-calibration of design features. In the feature extraction block, an adaptive version of discrete derivatives (ADDs), is used for selective spike waveform decomposition. The ADDs are used to extract the features (positive and negative peaks) from the desired frequency bands of the neural data. In the clustering unit, a multi-aspect optimization methodology (power, area and accuracy) is proposed to satisfy strict hardware implantable criteria. A register bank memory structure provides up to twice the dynamic power reduction in the training phase compared to the conventional implementation. This work is the first demonstration of a spike sorting processor with feature extraction. The number of locations for saving the spike features is reduced almost by 8X. This reduces the number of locations for buffering the cluster means. The clustering efficiency is improved using noise-tolerant $\ell 1$-norm distance calculation and a modified version of O-Sort [13] for sorting threshold calibration in an iterative validation phase. The chip prototype was fabricated in a 180-nm CMOS technology. It achieves an overall clustering accuracy of 84.5% using a standard spike data bank and has a power consumption of 148-µW from 1.8-V supply voltage. The new spike processor has almost 10% higher clustering accuracy than the state-of-the-art.
- Chapter 6 concludes the thesis with a special emphasis on its most important contributions. Additionally, a discussion of possible future work and improvements is provided.

1.4 List of Publications

The research conducted during this work has led to the following published papers:

1) **M. Zamani** and A. Demosthenous, "An Efficient, Unsupervised Clustering Method using Extrema Sampling of Discrete Derivatives and Online Sorting for Real time Spike Sorting in Multi-Channel Neural Recording Systems," ***IEEE Transactions on Neural Systems and Rehabilitation Engineering****., vol. 22, no.41, pp. 716–726, July. 2014.*

2) **M. Zamani** and A. Demosthenous, "Dimensionality Reduction Using Asynchronous Sampling of First Derivative Features for Real- Time and Computationally Efficient Neural Spike Sorting," *IEEE International Conference on Electronics, Circuits, and Systems (**ICECS**), Abu Dhabi, UAE, December 2013.*

3) Y. Scolich, **M. Zamani,** A. Demosthenous , and M. R. D. Rodriguez, "Hardware Efficient Features with Application to On-chip Neural Spike Sorting," *Accepted in **International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)**, Milano, Italy, August 2015.*

4) **M. Zamani** and A. Demosthenous, "Design Optimization of Analog Front-ends for Neural Recording Interfaces," *International Symposium on Circuits and Systems (**ISCAS**), Lisbon, Portugal, May 2015.*

5) **M. Zamani**, E. Clemens and A. Demosthenous, "Power Dissipation Limits of CBSC-Based Pipelined Analog-to-Digital Converters," *IEEE International Conference on Very Large Scale Integration (**VLSI-SoC**), Istanbul, Turkey, October 2013, pp. 352-357.*

6) **M. Zamani**, E. Clemens and A. Demosthenous, "CBSC-Based Pipelined ADC Design Based on Power Bound Analysis," *IEEE International Conference on Electronics, Circuits, and Systems (**ICECS**), Seville, Spain, December 2012, pp. 436-439.*

CHAPTER 2

Background and State-of-the-Art

2.1. Brain-Machine Interfaces (BMIs)

The history of recording dates back to 1791 when Luigi Galvani discovered that the living tissue of frog muscles exhibits contraction by applying stimulating current [14]. Since that time, it has inspired interfacing machines to human brain to either read mind or controlling it. BMI concept is becoming pervasive in the development of "smart" biomedical devices over the past decades, with the significant advances in the field of micro-electronics and signal processing.

BMI provides an efficient interfacing for interpretation of neurons functionality in order to investigate and model application specific processing set-up (e.g., see Figure 2.1). The BMI concept is the result of a combination of neuroscience and engineering principles mainly aimed at restoring lost functionalities [15].

The BMI concept can be employed in the development of therapeutic technologies such as stimulation to control the tremors of Parkinson's disease [16], decoding signals to drive neural prostheses [17] control systems, cognitive prosthetic research to replace lost brain functionality [18], defining practical translational pathway for rehabilitation [19] and behavioral analysis of some neurons when they respond to a specific drug in pharmacology science [20]. In clinical research, analysis of single-unit activity is typically required to study the neuron response to a specific stimulus.

The development of therapeutic and assistive devices has already changed the quality of life of thousands of patients. Examples of commercial devices in the market include Medtronics' deep brain stimulators (DBS) [21], prosthetic limbs by Advanced Arm Dynamics [22] and exoskeletons such as hybrid assistive limb (HAL) [23].

BMIs are classed in two categories. The first category is related to the processing and analysis of the recorded signal in order to extract the sensory stimuli patterns mimicking a neurological function. In sensory systems, for example, sound can be processed for auditory prostheses for individuals with profound deafness [24]. In addition, brain stimulation can be used for the control

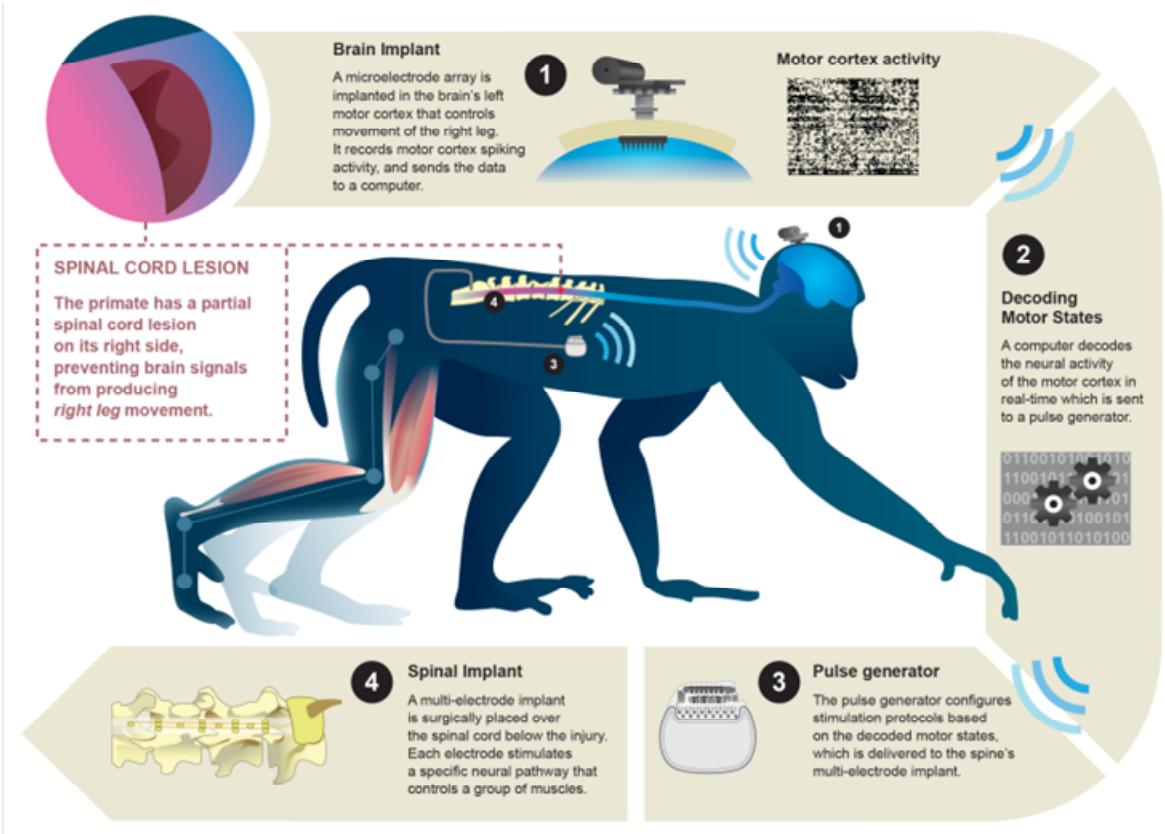

Figure 2.1: BMI demonstration for rehabilitation of a paralyzed monkey [19]. The stimulus parameters are sent to the implanted site for adjusting the stimulation parameters using a wireless data transmission system.

and suppression of motor disorders such as in Parkinson's diseases. This process is real time and the relevant regions of brain are stimulated. The second category of BMIs performs real time data processing of monitored brain activity to extract the motor commands in order to build a communication bridge to deliver the lost motor commands to external assistive devices for people with damaged sensory/motor functions (such as Deka Arm [25]).

2.1.1. Levels of Interfacing

Figure 2.2 shows the different existing neural recording approaches classified as a function of their invasiveness level in the brain. The illustrated methods are employed by BMIs through tapping into the sensory and motor pathways. In invasive approaches, the quality of the recorded signal is better and it provides clearer distinction between neurons resulting in better functionality. The level of interfacing depends on the type of application and the level of the risk involved. The first method which has been extensively used by BMIs is the electroencephalogram (EEG), which

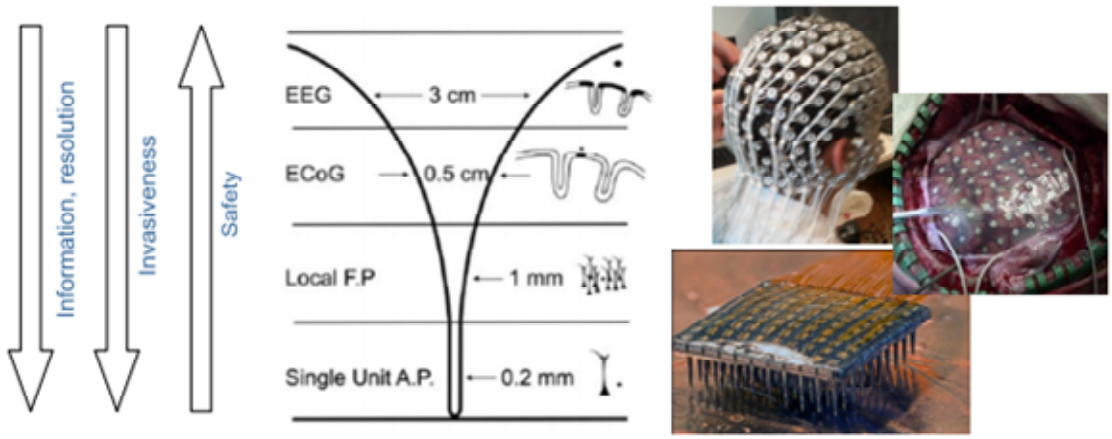

Figure 2.2: Various neural interface modalities with different levels of spatio-temporal resolution and invasiveness. Adapted from [26-28].

reflects the averaged activity of millions of neurons. This type of recording is performed by means of groups of electrodes placed at the surface of the scalp (top-right photo of Figure 2.2). The use of EEG has provided numerous successful BMIs which include spelling devices [29], controlling computer cursors [30] and driving a wheelchair [31]. As these electrodes are located about 2-cm above the cortex, EEG is a limited method in terms of spatial and temporal resolution. The quality of recording is severely affected by issues such as overlapping electrical activity of different cortical areas and attenuation of signals through brain tissue, bone and layers of the scalp.

A more invasive approach such as electrocorticography (ECoG) [32] can be utilized to increase the recording resolution. In this method the electrodes are placed on the brain surface to record neural activity. Compared to EEG, this method is more localized and it increases the precision of

Table 2.1: Various neural interface modalities with different levels of spatio-temporal resolution and invasiveness. Adapted from [33-39].

Signal	**Amplitude (peak to peak)**	**Bandwidth**	**Temporal Resolution**	**Spatial Resolution**	**Invasiveness**
EEG	1 - 100 µV	≤50Hz	~50ms	>cm^2	Surface (Scalp)
ECoG	1 - 500 µV	≤200Hz	~5ms	~ cm^2	Surface (Brain)
LFP	500 µV - 5mV	≤300Hz	~3ms	~ mm^2	Intra-cortical
SUA	50 µV - 500 µV	250Hz-10kHz	≤0.2ms	Sub- mm^2	Intra-cortical

recording. In ECG, there exists a barrier between the recording sites and the cerebral cortex surface which is avoided in the ECoG, thus improving the quality of the acquired signal both temporally and spatially [40, 41]. This results in successful demonstrations of BMI-related applications such as cursor control [32]. In [33] it is shown that the ECoG-based BMIs are more effective than their EEG-based counterparts [33].

Although EEG and ECoG have been demonstrated in different applications, research has moved towards the use of micro-electrode arrays (e.g. Utah micro-electrode array [42]) implanted into brain. This type of recording provides the highest the highest spatial and temporal recording resolution, so it has gained considerable interest amongst neuroscientists since they can monitor the neural activity at a higher resolution. Such high resolution of direct interfacing has enabled the realization of the type of BMIs discussed in Section 2.1 (e.g., prosthetics with high degrees of freedom [43]).The work presented in this thesis is in the context of monitoring neural activity via micro-electrodes. Table 2.1 presents the various neural interface modalities with different levels of spatio-temporal resolution and invasiveness.

2.1.2. Nature of Neurons and Action Potentials

Neurons generate and transmit impulses in the form of electrical signals [44], [45]. They communicate with other neurons via synapses. The role of a neuron is to receive, process and transmit information by electrical or chemical signals. The structure of a typical neuron is shown in Figure 2.3(a).

Due to the negative charge distribution along the inner and positive charge along the outer surface of a neuron, there is a steady potential difference between the internal and external environments connected by open/closed ion channels. When the ion channels are inactive (resting state), the concentration of Potassium (K^+) is high inside the cell and the concentration of Sodium (Na^+) and Chloride (CI^-) are relatively high outside the cell. The charge distribution difference in the resting mode results in a membrane potential around (-70mV) [46].A stimulus provokes an electrical response (postsynaptic potential) which generates a signal, called action potential in axon hillock. This excitatory synaptic input results in an electrical charge which travels across the cell membrane in the form of ion flow through voltage-gated channels. The response of a cell to a stimulus is as follows. If the ion movements result in a reduced charge difference across the

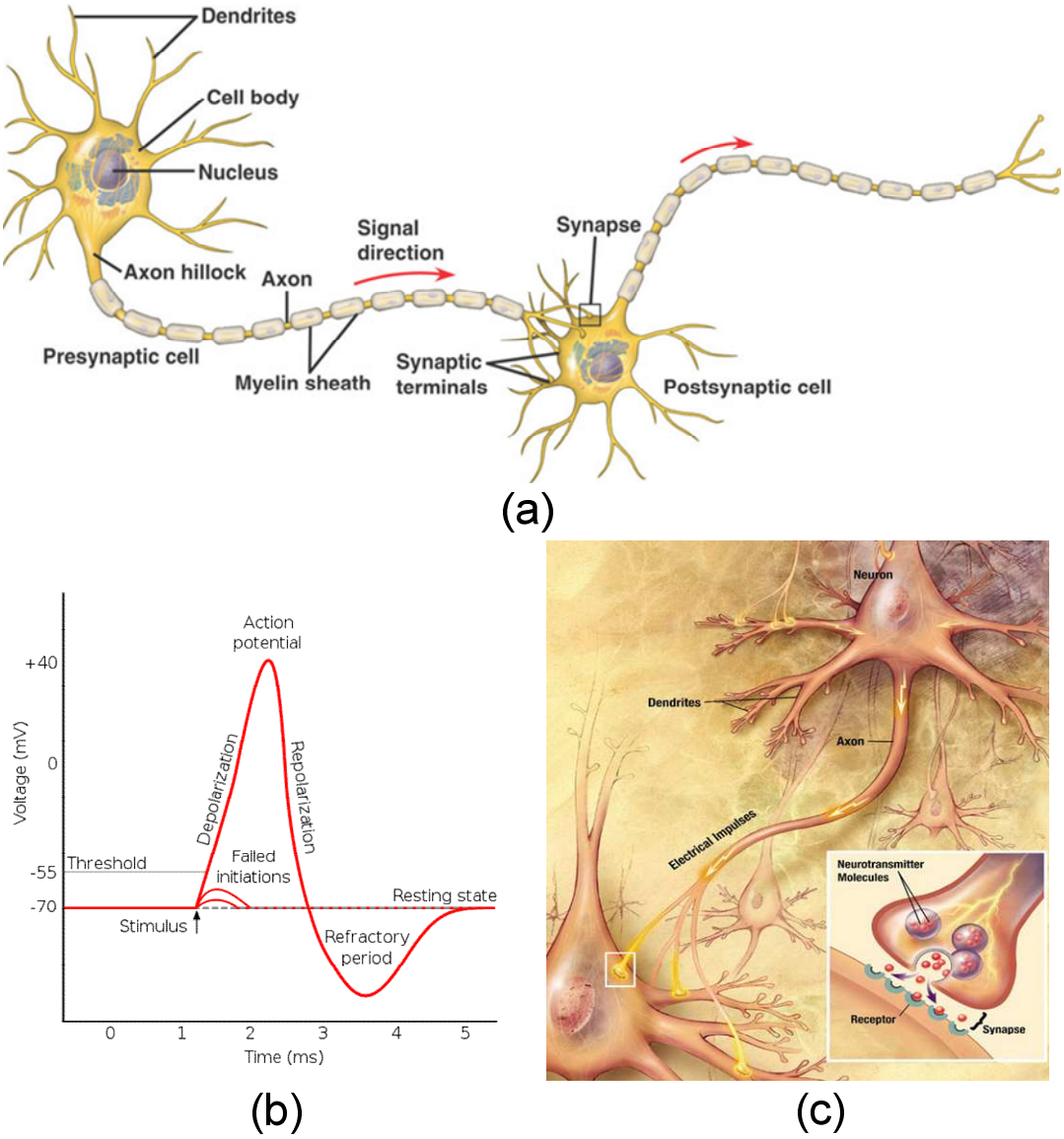

Figure 2.3: (a) Parts of a neuron: body, dendrites and the axon; (b) typical action potential; and (c) communication between neurons [47].

membrane it is depolarized, and if the ion movements result in an increased charge difference it is hyperpolarized. During this depolarization process, the voltage-gated channel (Na^+ channel) is opened and Na^+ flows into the cell which causes the rising voltage of the action potential. This process will continue up to a peak depolarization limit (40mV). The Na^+ channel is then closed and the K^+ channel is activated to trigger the repolarization phase. The K^+ ion flow results in the falling voltage of the action potential. After firing, an action potential enters the refractory period

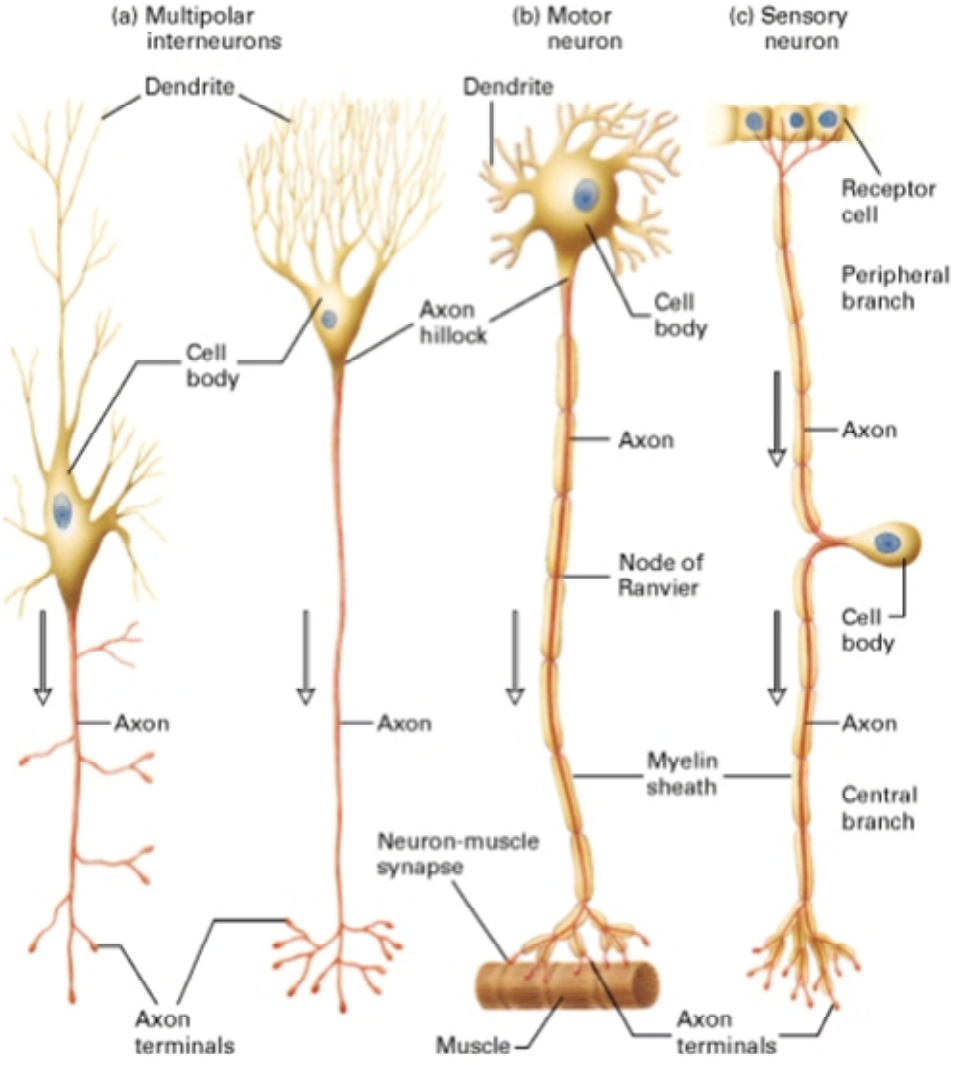

Figure 2.4: Structure for different types of neuron [47].

in which the membrane cannot respond to any stimulus. A neuron is non-excitable for approximately up to 2 ms after the action potentials [48]. The procedure is depicted in Figure 2.3(b). This phase will continue until the neuron rest phase or the charge balance condition is finished. The released action potential is carried away from the body of a neuron by an axon. The end of each axon is composed of synapses which conduct the signals between two neurons. There is no anatomical connection between the presynaptic and postsynaptic cells and the action potential is emitted between two sites through a small space which is called the synaptic cleft [Figure 2.3(c)]. Although many types of neuron exist in a number shapes (depending on their location and functionality) [48], the typical structures are shown in Figure 2.4.

2.2 Recorded Signal

Normally the recorded signal is composed of high-frequency extracellular action potentials (EAPs) and low frequency local field potentials (LFPs) (Figure 2.5). The frequency range of EAPs is between 300 Hz and 6000 Hz and for the LFPs is below 300 Hz due to the superimposition of the summed dendritic and synaptic activities (LFP peak-to-peak amplitude can be as much as 5 mV). Since the signal of interest is the EAP, the recorded data stream is amplified, band-pass filtered (300 Hz and 6000 Hz) with the specified cut-off frequencies and finally digitized before feeding it to the processing chain. As discussed in Section 2.1.1, the main interest is to utilize micro-electrode arrays in order to obtain high temporal and spatial resolution neuronal activity. This type of recording is necessary for development of the next generation of BMIs featuring higher accuracy than existing systems.

The typical reported detected amplitude of EAPs/spikes[1] is in the range of 60µV to 1 mV depending on electrode position. The distance for observing high precision EAPs is in the region of 50 µm and 140 µm from the electrode tip [49], beyond this range the monitored activity of the EAPs is comparable to the underlying background noise and is not detectable.

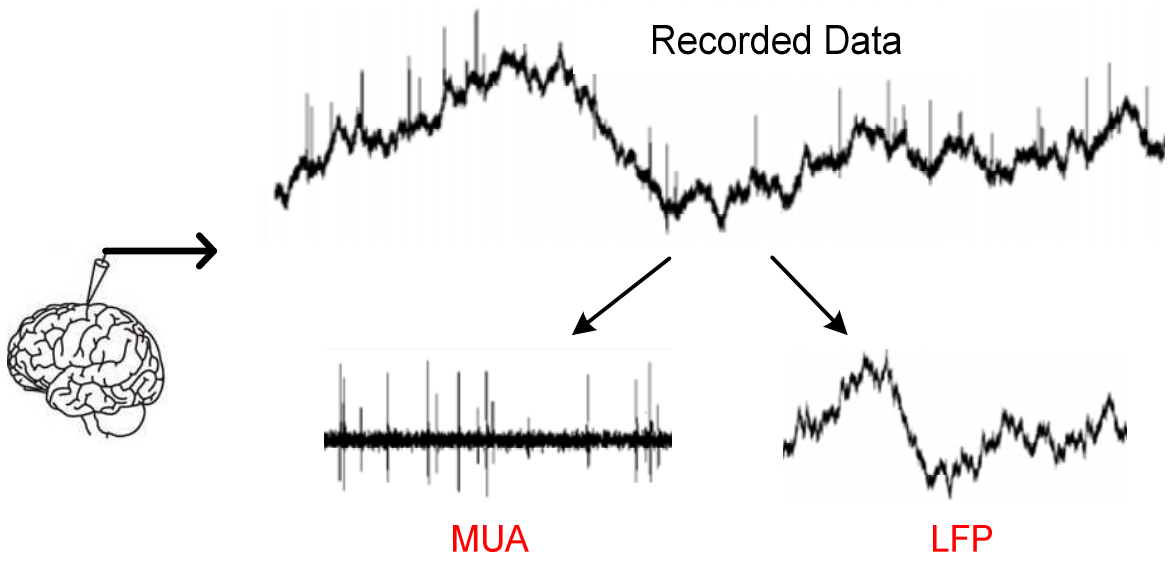

Figure 2.5: Contribution of LFP and multi-unit activity (MUA) which is the sum of the EAPs.

[1] Spike is a general term which is used in spike sorting community for evaluation of the sorting methods.

2.3 Spike Sorting

In the following discussions EAPs are referred to as ‘spikes’ as commonly used in the literature. Spike sorting is the process of assigning detected spikes to their originating neurons. The spike sorting chain is depicted in Figure 2.6. The steps in the sorting procedure include: 1) detection, 2) alignment, 3) feature extraction, 4) dimension reduction, and 5) clustering. The spike sorting process is based on the concept of pattern recognition/machine learning which is extensively discussed in [50].

The entire concept of spike sorting is developed based on the assumption that each spike has distinct morphology/features which enables the distinction between different neurons. Each spike shape varies depending on some contributing factors such as neuron's tree topology, ion channel distribution and topological placement of the micro-electrodes (i.e., orientation and proximity) as shown in Figure 2.7 [51].

Typically the digitized signal contains activity from 5 to10 neurons [51], so the first step of sorting is to detect the spiking events from the original data. The extracted spikes are aligned to an event in time and then projected into a new space called ‘feature space’. In the feature space, some distinguishable features are retained which are fed into the clustering stage. Finally, spikes are mapped to different clusters associated with the originating neurons.

2.3.1. Constraints in Hardware for Implantable Devices

Introduction of real-time and online spike sorting processors requires the realization of highly accurate processing algorithms which can be used within the context of the implementation constraints of implantable devices (e.g., area and power). The benefit of developing such systems is already discussed in categorizing the BMIs and related discussed applications (section 2.1). For

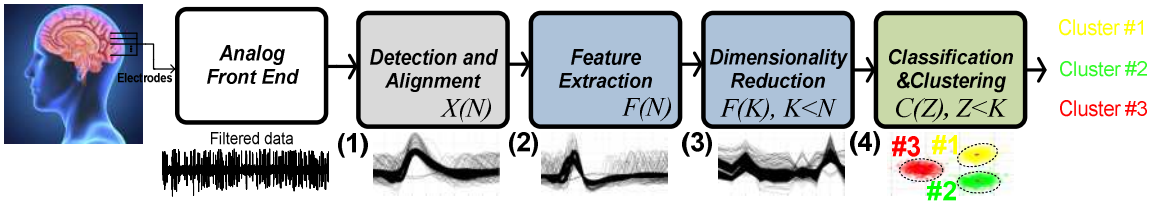

Figure 2.6: Spike sorting chain for determining single unit activity. Data dimensionality is reduced over processing units ($Z < K < N$).

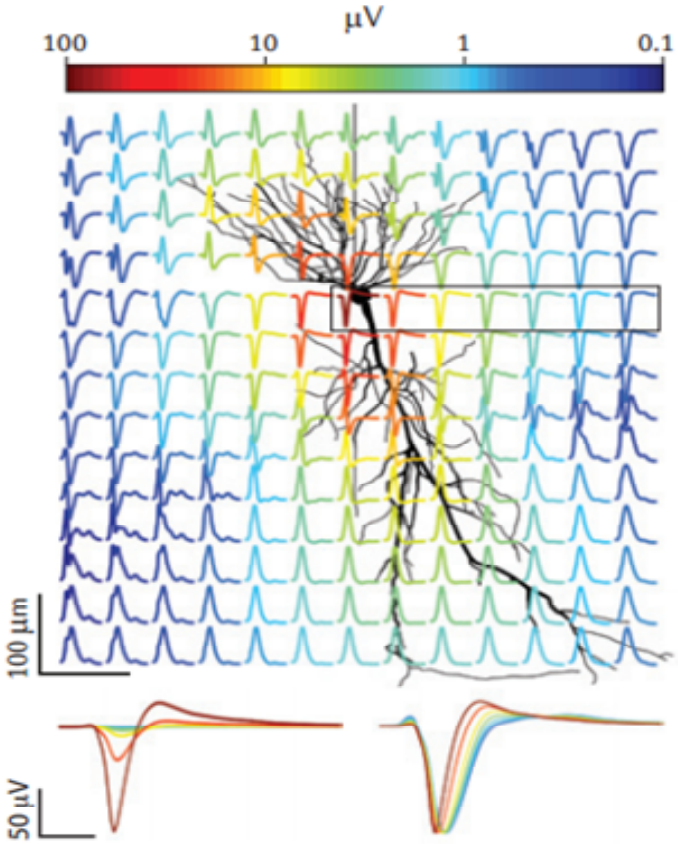

Figure 2.7: Variation in the observed extracellular action potential profile from a pyramidal cell with spatial position. Variations depend on some cell proprieties such as the cell type and the cell geometry, as well as the distance of the electrode from cell and position of the recording electrode relative to the cell. Adapted from [51].

example in the first category of BMIs, spike sorting is used as a powerful neuroscientific tool for identifying the behavior of specific neuron responses to a stimulus and then generating a fine-tuned pattern in a closed-loop manner for rehabilitation. In implantable devices chip area, battery life, power consumption and thermal dissipation density are important factors which need to be taken into account including the design of on-chip spike sorting processors. For example, the reported limit on thermal dissipation density to avoid tissue damage is 800 μW/mm [52]. Implantable devices consist of different blocks including a recording site, on-chip processing unit and radio frequency (RF) unit for data transmission. Output data-rate reduction is another important design constraint when designing implantable transceivers. Existing implantable transceivers are not able to transmit very large amounts of recorded data. For example, in a 64 channels recording with an analog to digital converter having a sampling frequency of f_S=30kS/s

and resolution of 7 bits, the input data entering the signal processing unit is at a rate of 19.2Mb/s. Reported transceivers cannot support such a rate [53].
A guideline for data transmission in medical implant communication systems is between 0.5 and 1 nJ/b [54], [55]. Thus, the transmitter requires a total power up to 19.2 mW to transmit the raw data. Depending on the type of implant including its location and packaging, this level of power dissipation may be hazardous for surrounding tissue, especially for very small size implants. To reduce the data transmission requirements for high channel count implantable recording systems, on-chip signal processing is essential.

2.3.2. State-of-the-Art Spike Sorting

There have been numerous methods proposed to solve the challenges of spike sorting over the past years. Spike sorting methods are usually characterized based on their off-chip/on-chip application. There are two main approaches for spike sorting. Firstly measurement and extraction of morphological features of spikes in order to identify the number of clusters. In the feature based method, some distinguishable features are extracted in order to identify the clusters in the recorded data. Since (ideally) a small number of distinguishable features are kept for each spike, the dimensionality is reduced based on the input/output ratio. Assume that the dimensionality reduction ratio is N/K, where N is the number of sample points per spike and the K is the number of chosen features. Secondly the use of spike templates to distinguish the clusters (template based approach).

Three different categories are considered here to review the spike sorting solutions in the literature, including: 1) traditional implementation of spike sorting which requires off-chip/offline processing; 2) development of on-chip spike sorting methods as an alternative to satisfy the hardware implantable devices constraints and the efficiency requirements of different applications; and 3) the use of data compression techniques for either transmitting the data to an external unit for decompression and spike sorting tasks or designing a specific type of processing method for spike clustering.

The following section presents various spike sorting solutions proposed in literature and discusses the state-of-the-art.

2.3.2.1 Off-chip Spike Sorting

Many highly complex methods and statistical analysis tools can be considered in off-chip spike sorting methods since power consumption and the computational complexity are not major constraints. There are some widely used automatic spike sorting algorithms such as KlustaKwik [56], Plexon [57], spike2 [58] and Waveclus (whose interface is shown in Figure 2.8) [59]. In these spike-sorting programs, a signal processing chain performs extraction of spike events and allocates them to the originating neurons. They all involve highly complex mathematical operations which are inimical to hardware implantable device constraints. As a result, data must be transmitted from the subject to a computer or to an off-chip processing unit for detailed clustering analysis (Figure 2.9). These spike sorting platforms cannot provide a satisfactory solution for on-chip spike sorting due to complexity and scalability issues. In addition to the methods above, there are other offline approaches which require large amounts of power for extraction of the most important features buried in the neural signals. One of the major categories of offline methods is related to the conversion of correlated variables into linearly uncorrelated variables using orthogonal transformation such as principal component analysis (PCA) [60] and independent component

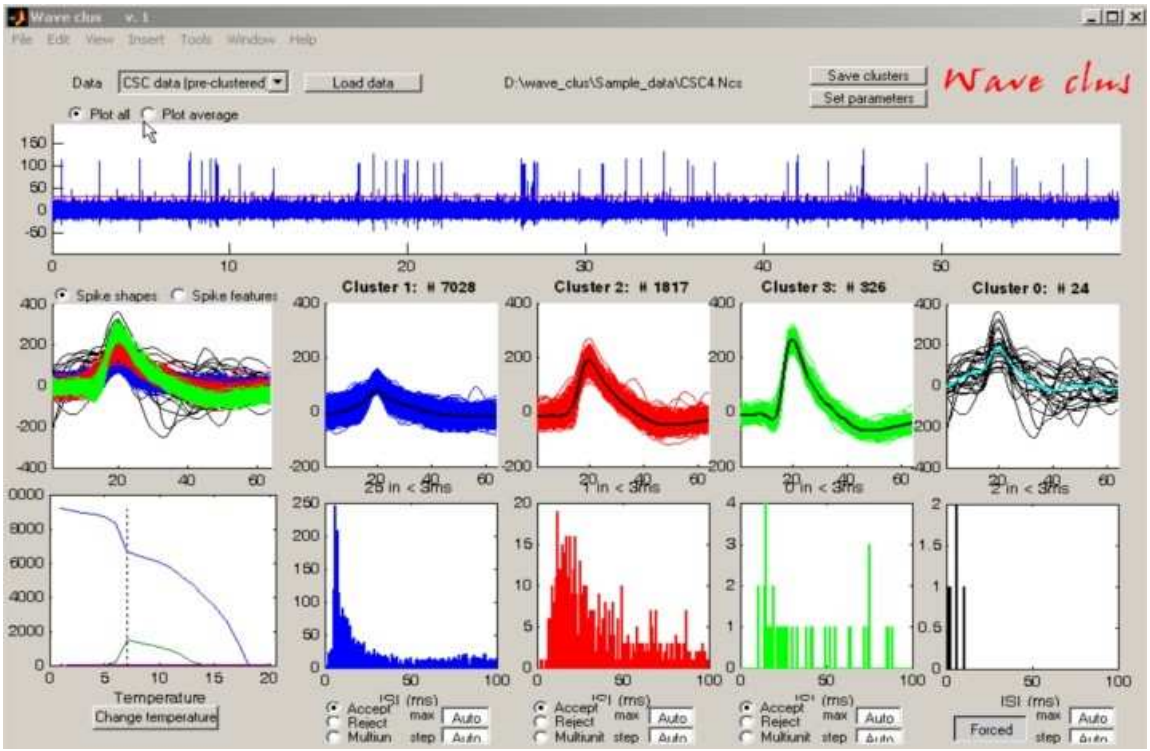

Figure 2.8: Waveclus graphical user interface.

analysis (ICA) [61]. Apart from these methods, there are many feature sets which are extracted via Graph Laplacian features (GLFs) [62] and decomposed features of signals over different frequency bands based on the discrete wavelet transform (DWT) [59]. Reference [63] presents an algorithm in which a feature design framework is used to provide a hardware implantable version of PCA and GLF. This approach provides a simple way of designing features which are compatible with on-chip processing constraints. There are also statistical feature extraction methods such as expectation maximization based competitive decomposition algorithms [64] and probabilistic approaches such as Gaussian mixture models as described in [65].

2.3.2.2 On-chip Spike Sorting

The new trend in bio-acquisition systems aims at integrating on-chip signal processing as shown in Figure 2.10. Significant research has been done on on-chip spike sorting to meet a tradeoff between hardware resources utilization and sorting accuracy. The focus of published materials in the field of on-chip spike processing realization is categorized into three implementation strategies. There have been some studies on analog processing methods for multi-unit activity extraction and analysis. The focus of other studies are either based on using off-the-shelf hardware platforms such as FPGAs or designing custom digital implementations for implantation purposes.

In [66], the authors designed a fully integrated system for the detection and characterization of action potentials. In this paper, the integrated circuit consists of an analog implementation of a non-linear energy operator (NEO) and peak detectors to extract the maximum and minimum of the detected spikes. The circuit was implemented in 0.13 µm CMOS technology; the chip area

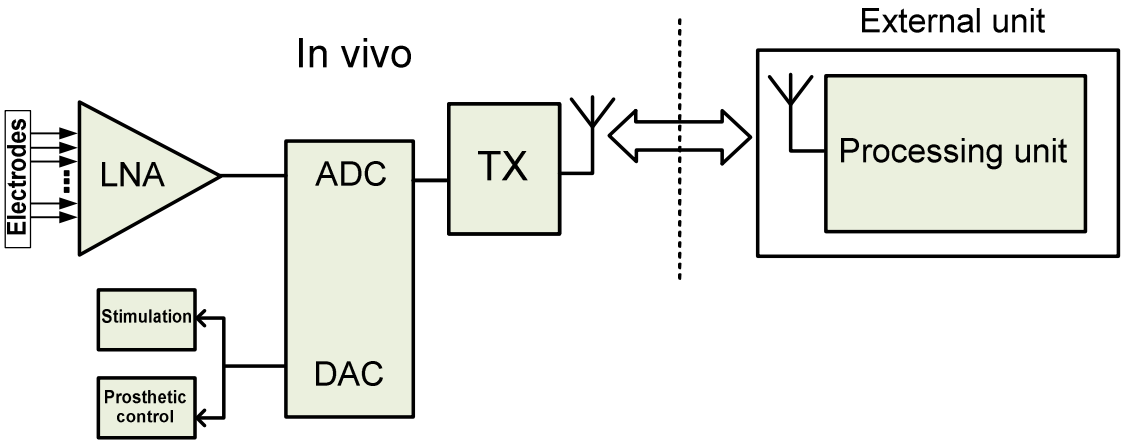

Figure 2.9: Block diagram of the conventional neural processing interface.

occupies 0.17 mm^2 and consumes 1 µW from a 1 V supply. In [67], the authors introduced a new feature extraction method, the integral transform (IT) and discussed its analog implementation. They reported 98% off-chip classification accuracy while requiring only 2.5% of the commonly used PCA computational complexity. Another paper which addresses the analogue implementation of on-chip spike sorting is discussed in [68]. The design uses spike features such as the peak and trough amplitudes and peak width for clustering. It was implemented in 0.35 µm CMOS technology with a power consumption of 70 µW per channel with over 90% off-chip classification accuracy.

In [69], the authors describe the application of spike sorting for sensing the bladder volume. Weighted Euclidean distance in conjunction with Levenberg-Marquardt variance estimator is used as a feature extractor. The design was implemented in an Actel FPGA Igloo AGL1000v2 platform. It consumes 485 µW and the number of classes for classification needs to be defined. Another spike sorting process which uses FPGA is discussed in [70]. It uses Hebbian Eigen filter based PCA algorithm. The process seems to be efficient, although it requires a high amount of resources for realization. The system requires neural signals to be stored, since it runs statistical analysis of the whole data. Further to power consumption and complexity, the system requires significant time for data analysis which suggests that the system is not real-rime.

A multi-channel spike-sorting DSP which performs detection and feature extraction has been demonstrated in [71]. It uses a parallel-folding structure to reduce the hardware resources, but the results show that the device is not compatible with hardware implantable constraints. A pulse-based feature extractor using derivatives of the spikes (action potentials) has been proposed in [72]. This method uses information encoding of spike derivatives in pulse trains. The chip results show that there is a difference between the simulation and the measured outputs. This suggests that the hardware mismatch due to process variations causes an appreciable change to the results. An asynchronous spike sorting DSP has been introduced in [73]. The asynchronous self-timed methodology has inherent latency adjustment due to process variations. Furthermore, the asynchronous scheme suppresses the leakage power and standby power in the presence of unwanted variations. The crucial challenge in the asynchronous methodology is to design complex circuits for the handshaking circuitry overhead and also the amount of process latency especially in clustering circuit implementation.

In [74], the authors demonstrate a digital implementation of a spike processor which performs the detection, alignment and feature extraction for 64 channels. The processor building blocks were chosen based on the complexity-performance analysis. The fabricated chip in 90 nm CMOS consumes 130 µW from a 0.55 V power supply. Reference [75] presents an online version of IT which is called Zero-Crossing-Features (ZCF). ZCF is based on different neurons having spikes with different area after the zero-crossing points. These features are compared to PCA, and found to be equally good performing while using 5% of the resources. In [76], the authors introduce a hardware-efficient multi-channel spike sorting processor. The processing chain includes the absolute value detector and unsupervised clustering (i.e., a template matching feature extraction method is used to sort the spikes). Since the sorting accuracy is decreased with increasing noise level, the clustering is modified using the ℓ_1-norm metric to take differentiation of spikes in assignment and training units. The 16 channel spike sorting chip was fabricated in a 65 nm CMOS technology and has a power dissipation of 75 µW with a power supply of 270mV. The online processing of data results in 240X data reduction and the clustering performance of 75%. In [77], the authors report a multi-channel neural recording integrated circuit with a spike processor. A NEO based spike detector was implemented for identifying spikes. A digital frequency-shaping filter removes the low frequency noise which results in better identification of similar neurons from different origins. The peaks of the original spike waveforms and maximum and minimum values of its first derivatives are used to classify the spikes. The power consumption per channel is 100 µW. Most of the developed on-chip spike sorting methods either do not include the implementation of the necessary spike sorting stages or have below average clustering accuracy.

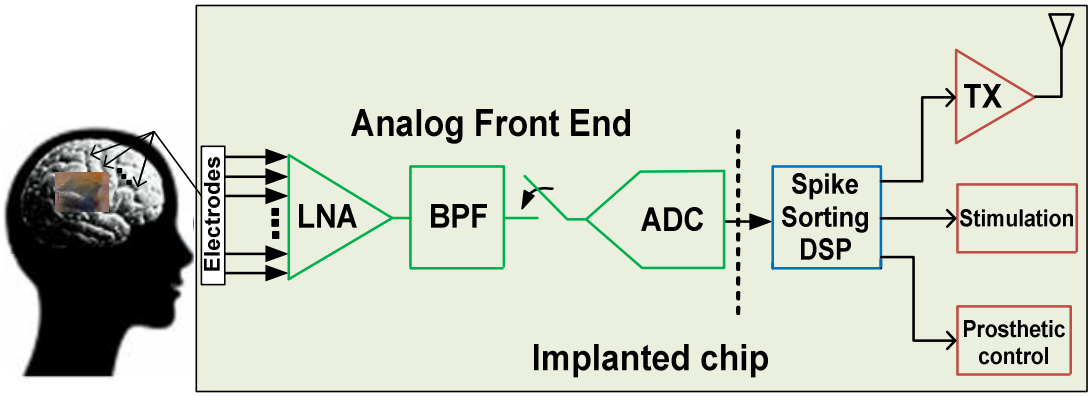

Figure 2.10: General architecture of an implantable chip from recording (front-end neural interface) to processing and transmission/stimulation/decoding (back-end).

The primary objective of this thesis is to introduce and implement an on-chip spike processor which outperforms the existing designs.

2.3.2.3 Compression-based Spike Sorting Methods

A challenge in on-chip processing systems is to deal with the large amounts of generated data from high channel count recording. In the compression based methods, dimensionality reduced data can either be transmitted to an external unit for decompression (reconstruction) and spike sorting or spike sorting can be performed directly to the compressed data by defining a suitable communication protocol between the compressed data and spike processor [78] (see Figure 2.11). Compression of spike processing has not gained popularity amongst the researchers due to the inefficiency of the developed methods. For example, one of the existing strategies for data compression is wavelet signal decomposition [79]. The wavelet technique in [79] reduces the bandwidth but requires 120 mW which is impractically large in implantable systems. In [80], the use of event detection has been proposed in which the data is typically limited to only the time and amplitude of neural spikes and therefore the encoded data often contains limited information. For

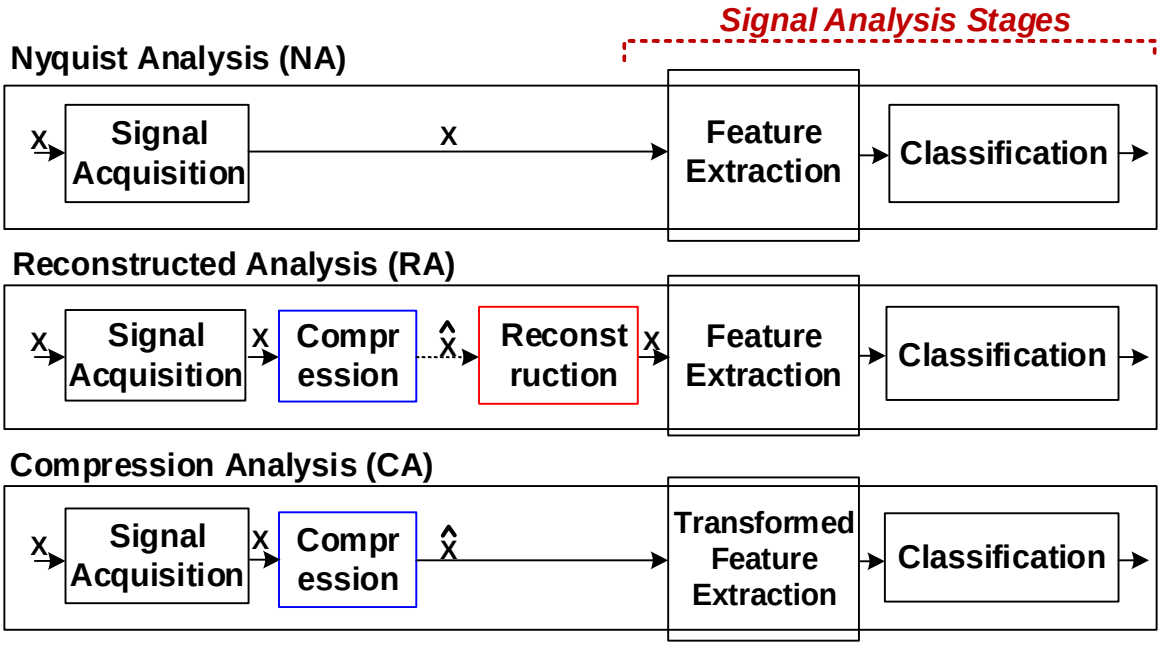

Figure 2.11: Demonstration of processing stages used in each of the three approaches. (a) Nyquist Analysis (NA), (b) Reconstructed Analysis (RA) and Compressed Analysis (CA). Reconstruction is bypassed in CA to provide significant savings in computational energy. Adapted from [85].

the compressed sensing approach in [81], the compressed data is reconstructed to apply the spike processing for mapping the neuronal activities to their origins. Other techniques for data compression are asynchronous sampling [82] and logarithmic compression [83]. Recently researchers started working on the realization of processing methods in compressed domains [84]. In reference [85] communication between thousands of neurons (1K+ channel recording/processing) as part of brain activity demands a shift in the processing paradigm, and provides a preliminary representation of signal decomposition (DWT) in the compressed domain.

2.4 Spike Sorting Testing Methodology

2.4.1 Test Data

This section provides a detailed description of the datasets used throughout this thesis. To compare the performance of the proposed method with other work the Waveclus spike bank[2] was used. Each dataset contains the true spike time and true spike classes for detection and clustering accuracy calculation as shown in Figure 2.12.

The database contains different average spike waveforms recorded from human neocortex and basal ganglia. To emulate the background noise activity, spike waveforms randomly chosen from the data library have been added to the generated datasets. There are advantages in using this database. Firstly, each dataset provides true spike classes, which is useful for accuracy calculations, and the ground truth can be established. Secondly, the diversity of the data enables evaluation of spike sorting algorithms from the same source. Datasets with different degrees of difficulty (i.e., similarity of spike shape) and noise levels are provided. The four datasets are *C_Easy1_noise*, *C_Easy2_noise*, *C_Difficult1_noise* and *C_Difficult2_noise*, where *noise* denotes the noise level in terms of standard deviation, namely, 0.05, 0.1, 0.15 and 0.2. '*Easy*' and '*Difficult*' is the similarity index between the spike shapes in each dataset. Thirdly, the characteristics of the datasets are similar to real practical recordings. Figure 2.13 shows the three different types of spike shape present in each of the four test datasets and the calculated Bray-Curtis similarity indices [86] between all the spike shapes in each dataset. The Bray-Curtis similarity index $S_{x,y}$ is:

[2]Available online: http://www2.le.ac.uk/centres/csn/spike-sorting.

$$S_{x,y} = 1 - \frac{\sum_{i=1}^{n} |x(i) - y(i)|}{\sum_{i=1}^{n} |x(i)| + |y(i)|} \tag{2.1}$$

where x and y are the two spike waveforms being compared and n is the number of sample points. $S_{x,y}$ is in the range (0-1), with 1 corresponding to identical signals. The sorting difficulty becomes more demanding with increasing the similarity between the spike shapes (width and amplitude fluctuations). Figure 2.14(a) shows color-coded spikes corresponding to different neurons from dataset 2 (*C_Easy2_0.05*), Figure 2.14(b) shows the two dimensional (2-D) projection of spike clusters, and Figure 2.14(c) shows a short recording segment with colored markers (with prior knowledge from simulation). Each color corresponds to a single-unit activity. As the difficulty of a dataset increases, sorting becomes more challenging. Furthermore, as the noise level is increased, filtering might be required to reduce the effect of noise on the efficacy of the sorting procedure.

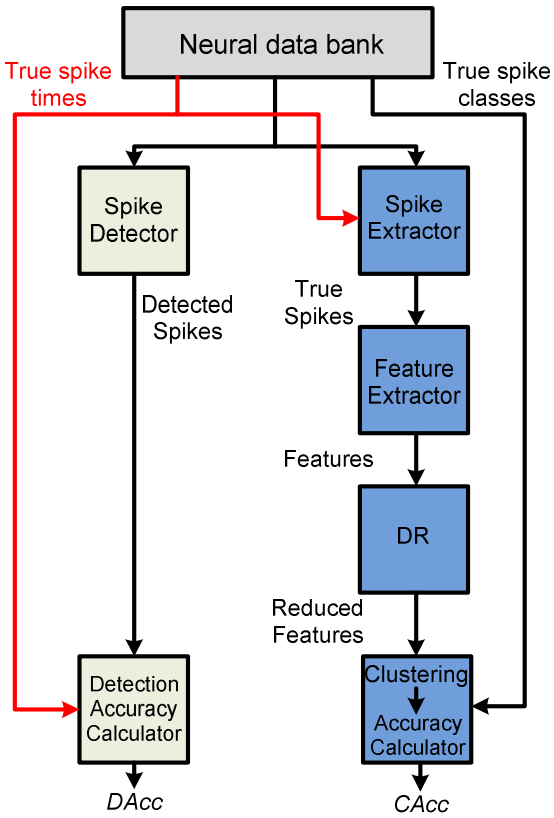

Figure 2.12: Signal processing chain used to evaluate detection and clustering accuracies.

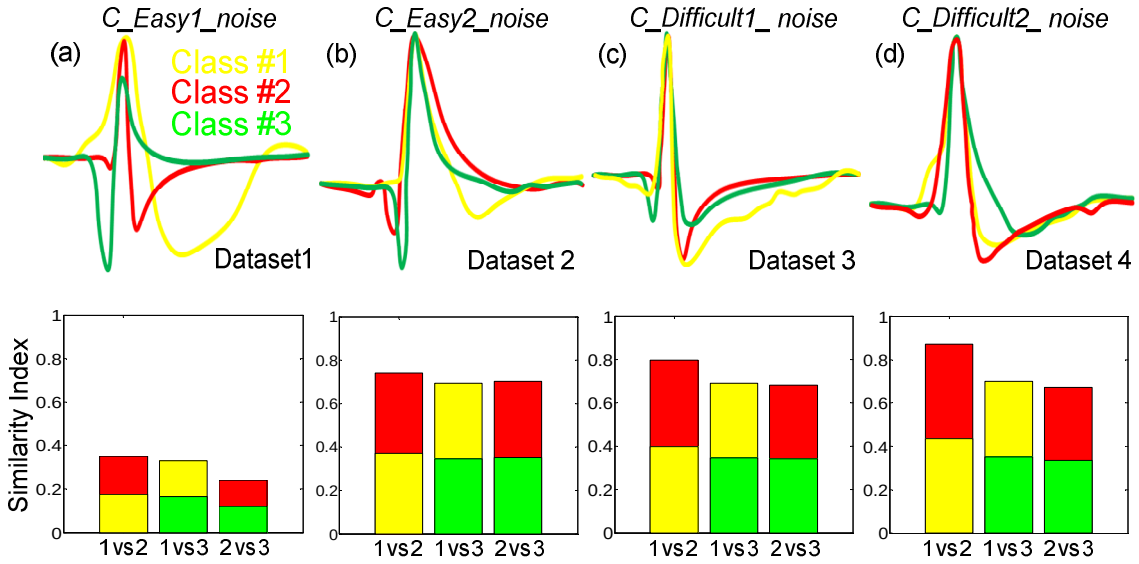

Figure 2.13: Spike bank mean waveforms (peak-aligned) used for testing with corresponding Bray–Curtis similarity index (shown at the bottom). (a) *C_Easy1_noise*, (b) *C_Easy2_noise*, (c) *C_Difficult1_noise* and (d) *C_Difficult2_noise*.

2.4.2 Accuracy Calculations

To evaluate the performance of each building block of a spike processing chain, various metrics can be defined to determine the overall system performance. The criteria considered are detection accuracy (DA_{CC}) and classification accuracy (CA_{CC}). As discussed in Section 2.4.1, a neural simulator which provides datasets containing true spike times and classes is used for accuracy calculation. Since the true spike times are known, the detection accuracy is defined as:

$$DA_{CC} = \frac{TPS}{TPS+FPS+MS} \times 100\% \tag{2.2}$$

where true positive spike (*TPS*) shows truly detected spikes, false positive spike (*FPS*) is the false alarm due to noise or overlapping spikes and *MS* is the number of missed spikes. This equation is designed to ensure that missed spikes and false alarms carry the same weight in accuracy calculations.

CA_{CC} is the next quantitative metric for comparing the accuracy performance of different algorithms. In each dataset the true spike classes and number of allocated spikes to each cluster determines the CA_{CC}. It is given by:

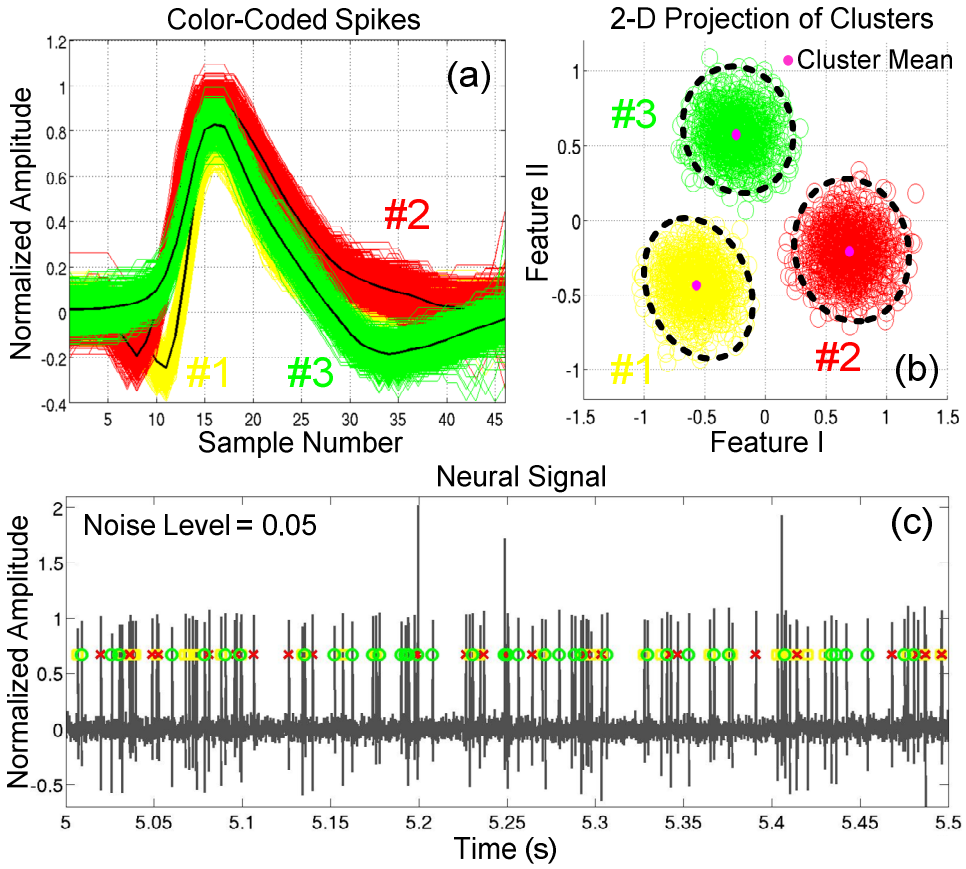

Figure 2.14: Test dataset 2 (*C_Easy2_0.05*) showing: (a) Color-coded spikes corresponding to different neurons (#1 yellow, #2 red, #3 green). (b) 2-D projection of spike clusters. (c) Segment of simulated dataset 2.

$$CA_{CC}=\frac{TPCC}{NTS}\times 100\% \tag{2.3}$$

where *TPCC* is the number of true positive spikes (*TPS*) that correctly clustered and *NTS* is the number of total spikes.

2.5 Explanations of the Spike Sorting Operators

In this section the description of the blocks in each step of spike sorting are presented. The following algorithms are explained:

- For spike detection:
  - Absolute value (ABS)
  - Nonlinear energy operator (NEO)

    - Stationary-wavelet-transform product (SWTP)
- For alignment, representing the temporal tuning of extracted spikes at a temporal reference.
- For feature extraction:
    - Principal component analysis (PCA)
    - Discrete wavelet transform (DWT)
    - Feature Set
    - Template Matching (TM)
    - Zero Crossing Features (ZCF)
- For dimensionality reduction:
    - Asynchronous sampling
- For Clustering:
    - *k*-Means Clustering
    - Artificial Neural Networks (ANN)
    - Superparamagnetic Clustering (SPC)
    - Online Sorting (O-Sort) Clustering

2.5.1 Spike Detection

Most spike detection methods consist of two steps: 1) calculation of the detection threshold which identifies the presence of a spike when exceeded; and 2) its use for spike activity identification. In the first step there are different methods such as absolute value, nonlinear energy operator and stationary wavelet transform product. In the second step, when neural or preprocessed signals exceed the threshold, a window of 3 ms (1 ms before the threshold crossing and 2 ms after) enveloping the spike is extracted for further processing. In some sampling modes the window length to cover the duration of a spike is different, but for normal sampling operation it is unlikely that the spike is longer than 3ms. Spike detection can be implemented either in analog when the data digitization requires higher resolution, or in digital structure. A comprehensive study has been published in [87] for various detection methods. Normally these methods automatically generate the threshold above which a spike event is identified.

2.5.1.1 Determination of the Threshold Using the Nonlinear Energy Operator

One unsupervised method to determine the threshold values is the Teager Energy Operator (TEO), also called (NEO) [88]. This class of detector calculates the energy variation of the raw signal to interpret the spike events in time. For the discrete signal, the NEO equation is defined as:

$$\psi\left[x(n)\right] = x^2(n) - x(n+1).\,x(n-1) \tag{2.4}$$

where *x(n)* is the input digitized signal and $\psi[(n)]$ is the calculated signal energy value at sampling time *n*. From the NEO equation, this operator highlights the variations which are large in power and frequency. The essence of spike activity is instantaneous amplitude-energy variation, so the NEO operator recognizes these localized variations. The detection threshold can be expressed as:

$$Thr = \alpha \frac{1}{N_S} \sum_{n=1}^{N_S} \psi\left[x(n)\right] \tag{2.5}$$

where N_s is the number of samples in the signal *x(n)* and α is threshold scaling factor. A short segment of a real neural signal including the extracted spikes and corresponding *Thr* are shown in Figure 2.15.

2.5.1.2 Determination of the Threshold Using the Absolute Value

Another method for spike detection is called absolute value [89] where the threshold level is defined by using an estimate of the standard deviation of the noise (σ_N). An intuitive value for this

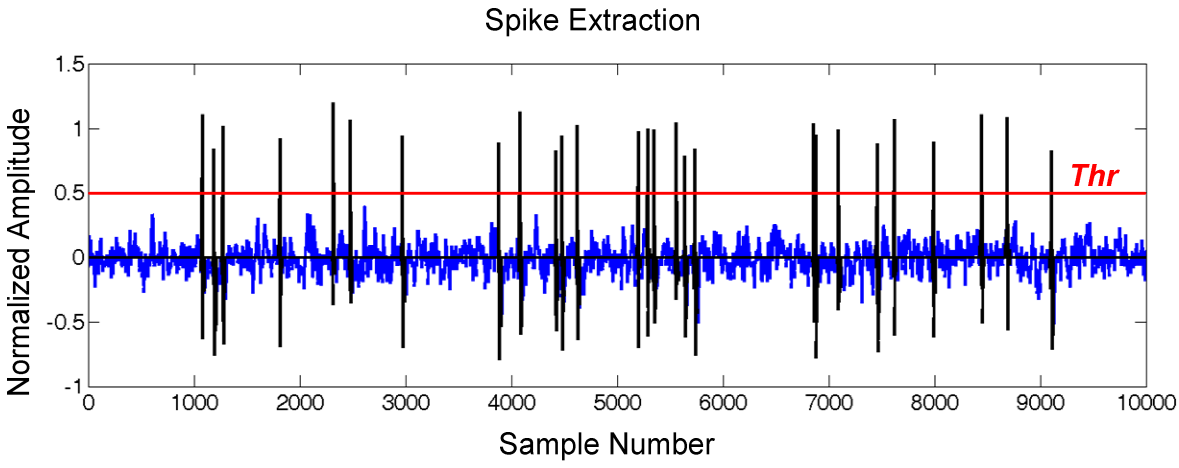

Figure 2.15: A short segment of a real neural signal including the extracted spikes. The extracted spikes (black color) are superimposed to the original waveform (blue color).

threshold would be a multiple of the standard deviation of the noise. However, with increasing spike activity, the threshold level increases and detection error will increase, so the equation below is suggested as a good estimation of noise standard deviation (σ_N):

$$\sigma_N = median\left(\frac{|x(n)|}{0.6745}\right) \tag{2.6}$$

where $x(n)$ is a sample of the original signal x at time n. The detection threshold is defined as:

$$Thr = 4\sigma_N \tag{2.7}$$

2.5.1.3 Stationary Wavelet Transform Product (SWTP)

One of the most effective methods for spike detection uses matched filtering. The correlation of the incoming signal to the filter bank shows the probability of the existence of a spike. One efficient method for template matching implementation uses SWT [90]. The SWT is calculated over some dynamic scales ($W(2^j,n)$, $j = 1,\ldots,5$) and the scale $2^{j_{max}}$ with the largest sum of absolute values (j_{max}) is:

$$j_{\max} = \text{argmax}\left(\sum_{n=1}^{N}\left|W\left(2^j,n\right)\right|\right), \quad j \in \{1,\ldots,5\} \tag{2.8}$$

The point product between the SWT at this scale and the SWTs at the two previous scales is calculated as:

$$p(n) = \prod_{j=j_{\max}-2}^{j_{\max}}\left|W\left(2^j,n\right)\right| \tag{2.9}$$

A Bartlett window $W(n)$ is used for smoothing the convolution in order to attenuate the effect of the spurious variations to calculate the detection threshold which is expressed as:

$$Thr = C\frac{1}{N}\sum_{n=1}^{N}\omega(n) * P(n) \tag{2.10}$$

where N is the number of samples in the signal and C is a constant value which is chosen empirically.

2.5.2 Alignment

Aligning the detected spikes is an important step in spike sorting before feature extraction. Typically, feature extraction is the projection (mapping) of the spike waveforms to a reduced dimensionality space. The efficiency of feature point positions can be severely compromised without an accurate temporal spike alignment to a reference point. When the signal crosses the calculated threshold in the detection unit a 3 ms window is applied to extract the spike as shown in Figure 2.16. The extracted spikes are aligned to the threshold crossing point. There is an uncertainty in threshold level due to sampling jitter and biological noise, and such uncertainty in threshold crossing (misalignment) can affect the clustering accuracy. To improve the clustering performance, all the spikes can be aligned to an event in time such as the positive peak [91], maximum slope [92], maximum energy [93] or maximum peak (mixed-peak) [13] which could be either positive or negative. The alignment process begins with upsampling of spikes, aligning them to an event in time and finally downsampling to avoid high computational complexity in the sorting stage (Figure 2.16).

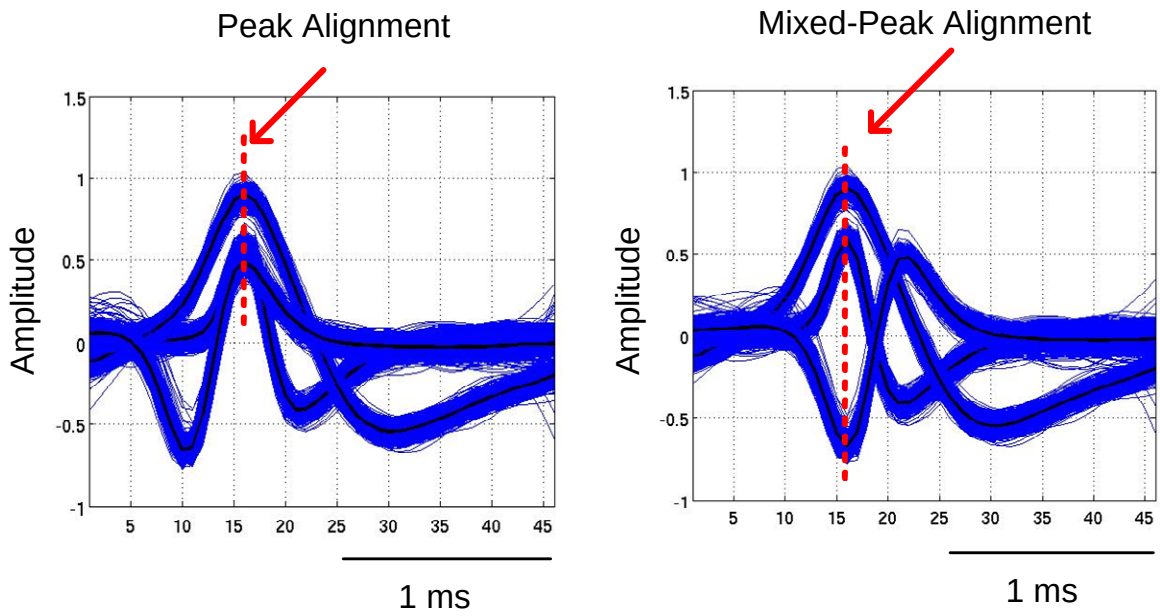

Figure 2.16: Examples of two different alignment methods. Left: peak-alignment method, Right: mixed-peak alignment method.

2.5.3 Feature Extraction

In order to extract meaningful information from *n* dimensional space, feature extraction is used to transform the aligned spikes to a low-dimensional space which highlights the spike waveform differences. Typically, this leads to dimensionality reduction from *N* to *K* (*K*<*N*), where *K* is the number of selected features (see Figure 2.6). In this section analytical feature extraction methods regardless of level of computational complexity and calibration time are investigated. They include PCA, DWT, derivative-based feature extraction methods, and explanation of other existing feature extraction methods.

2.5.3.1 Principle Component Analysis (PCA)

PCA has been the most commonly used algorithm for feature extraction [94] because it yields an efficient coding of spike waveforms (only the first 2-3 principal components need to be retained). The idea behind the PCA is to change the orthogonal basis vectors in a way that the maximum variance is achieved. The principle components are obtained via the *i* largest eigenvectors from the covariance matrix. So each spike is expressed as a series of scores [91]:

$$c_i = \sum_{n=1}^{N} PC_i(n)s(n) \tag{2.11}$$

where *s(n)* is a spike, *N* is the number of samples in a spike and PC_i illustrates the i^{th} *PC*. Retaining more *PCs* represent an accurate estimation of largest variations in each spike (Figure 2.17). It is assumed that the *i* largest eigenvectors results in an efficient separation between the detected spikes in a lower dimension. However, PCA requires offline training and calculation of the covariance matrix of the data demands high computational cost and hardware resources.

2.5.3.2 Discrete Wavelet Transform (DWT)

DWT [89] is a time-frequency representation that uses multi-resolution transformation. These are obtained using variable window sizes at various decomposition levels. This approach gives a detailed representation of the signal in the time-frequency domain. The DWT function *(f)* represents the linear combination of convolutions between the spike waveform and wavelet basis function, and is given by [95]:

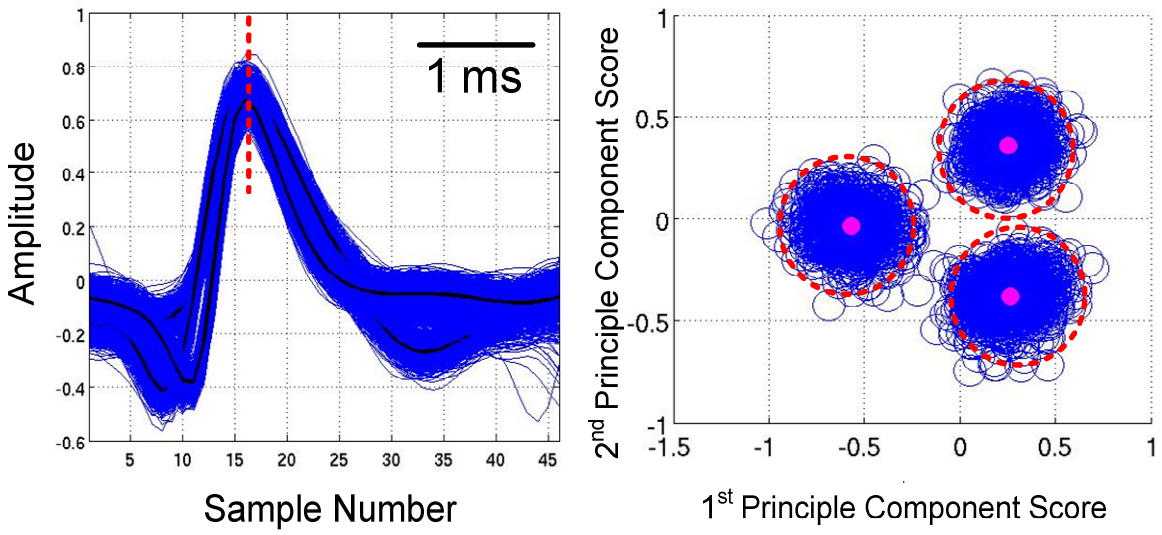

Figure 2.17: Example of feature extraction using PCA. Left: alignment to maximum amplitude, Right: PC coefficients in feature space. The neural data simulator explained in section 2.4.1 is used for PCA analysis.

$$f(t) = \sum_{j,k} s(n)\,\psi_{j,k}(t) \tag{2.12}$$

where *s(n)* is a spike and ψ is the wavelet basis function. The wavelet basis function derived from the mother wavelet Ψ, is defined as follows:

$$\psi_{j,k}(t) = \frac{1}{2^{j/2}}\Psi\left(2^{j}t - k\right) \tag{2.13}$$

Index *j* (called the scaling index) changes the behavior of $\psi_{j,k}$ in frequency space, while index *k* (called the translation index) shifts the wavelet along the time axis. The convolution between the wavelet basis function and spike *s(n)* results in multi-resolution time and frequency decomposition of the original spike waveform. In [89], four-level decomposition using Haar wavelets is proposed as an efficient way to convert the extracted spikes to the wavelet coefficients. Furthermore, the Haar DWT is an attractive approach since it can be implemented using filter banks. In order to determine the wavelet coefficients the operator representation of filters can be used [96]. An example to cover the entire signal bandwidth or band (BW), at each decomposition step a high-

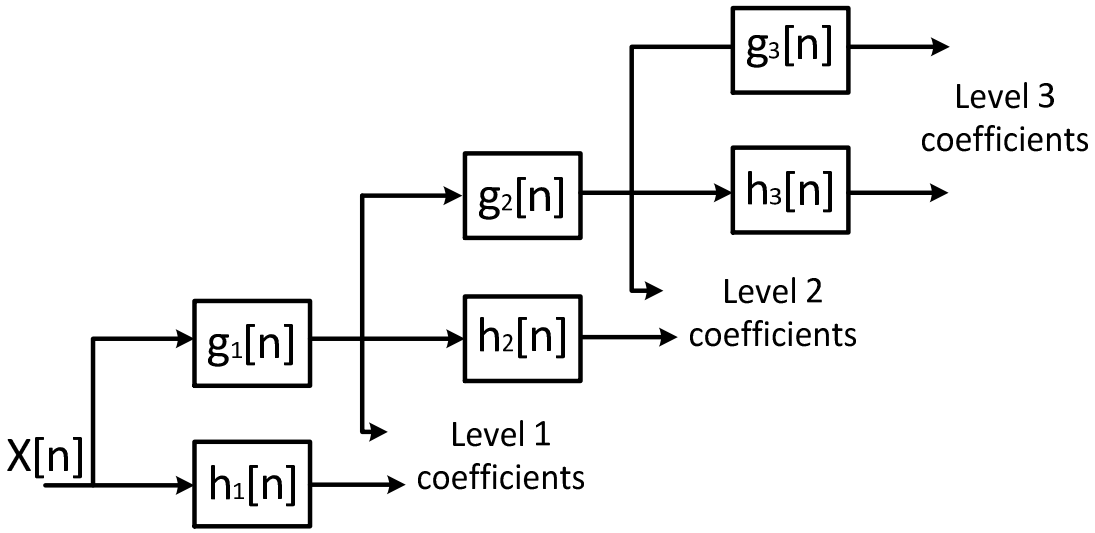

Figure 2.18: Wavelet decomposition by filter bank (from [96]). 3-level Haar wavelet decomposition of a spike waveform using MATLAB DWT tool (dwtool) SP.

pass (HP) and a low-pass (LP) filter must be used (Figure 2.18). The advantage of DWT is that each decomposition step requires half the computations of the previous step. When the decomposition process is performed, the number of wavelet coefficients is smaller than the original sample number.

2.5.3.3 Feature Set

Feature set refers to the spike shape characteristics such as spike amplitude, peak-to-peak value (spike height), maximum gradients (positive or negative), peak position and spike width. One or more of these features or can be considered as a feature vector. These types of features are not reliable in spike sorting and they have low classification accuracy especially in the spike datasets with high similarity index between the templates. A comprehensive review of spike shape features is provided in [97] (Table 2.2). Some of the spike features are highlighted with green color in Figure 2.19.

2.5.3.4 Template Matching (TM)

In TM, the spike shapes (*s(n)* in Figure 2.19) or the features of different types of neurons are collected and trained in order to identify the templates of the clusters. This process requires a training period and different distance similarity functions (Table 2.3 [98]) can be used to compare

Table 2.2: Overview of neuronal signal features [97].

Feature	Advantages	Disadvantages
Negative amplitude Positive amplitude	Easy to interpret Stresses specific shape characteristics	Vulnerable to signal offset; Only suitable in particular cases; vulnerable to signal offset
Left/right spike angle	Stresses specific shape characteristics	Only suitable in particular cases; vulnerable to noise distortions
Neg./pos. signal energy	Robust features in terms of noise distortions or low sample rates	Naturally correlates with amplitude and angle features; vulnerable to signal offsets
Core spike duration	Stresses specific shape characteristics	Only suitable in particular cases; vulnerable to noise distortions
NEO coefficient	Higher resolution than negative amplitude in particular cases	Naturally correlates with negative amplitude
Principal components	Usually accurate description of datasets with a set of uncorrelated parameters	Possible inaccuracies if more than two spike shapes are present in the dataset
Distribution/Shannon Wavelet coefficients	Potent Wavelet scale features with no significant correlation	High computational complexity compared to other methods; multi-resolution analysis limited by sample rate

the incoming spikes/spike features to set up the finalized cluster templates. The training is performed periodically to track the changes in the recorded data.

2.5.3.5 Zero Crossing Features (ZCF)

One of the most distinctive features of spike shapes are the integral of positive and negative lobes which normally contain the information about the amplitude variations of the spike lobes, position of the positive and negative spike peaks, and width of the spike lobes [75]. ZCF is a modified version of the Integral Transform [67]. The ZCF equations ZC1 and ZC2 (see Figure 2.19) are mathematically expressed as:

$$ZC1 = \sum_{n=0}^{n_1} S(n), \quad ZC2 = \sum_{n=n_1}^{n_2} S(n) \tag{2.14}$$

Table 2.3: Distance and similarity measures used in template matching [98].

Measure	Form
Minkowski distance or L_p norm	$D\left(x_i, x_j\right)=\left(\sum_{l=1}^{d}\left\|x_{il}-x_{jl}\right\|^{p}\right)^{1/p}$
Euclidean distance or L_2 norm	$D\left(x_i, x_j\right)=\left(\sum_{l=1}^{d}\left\|x_{il}-x_{jl}\right\|^{2}\right)^{1/2}$
City-block distance or L_1 norm	$D\left(x_i, x_j\right)=\sum_{l=1}^{d}\left\|x_{il}-x_{jl}\right\|$
Squared Mahalanobis distance	$D\left(x_i, x_j\right)=\left(x_i-x_j\right)^{T}\operatorname{cov}\left(x_i, x_j\right)^{-1}\left(x_i-x_j\right)$
Pearson correlation	$D\left(x_i, x_j\right)=\left(1-r_{ij}\right)^{2*}$

*where $r_{ij}=\dfrac{\sum_{l=1}^{d}\left(x_{il}-\bar{x}_i\right)\left(x_{jl}-\bar{x}_j\right)}{\sqrt{\sum_{l=1}^{d}\left(x_{il}-\bar{x}_i\right)^2\sum_{l=1}^{d}\left(x_{jl}-\bar{x}_j\right)^2}}$

where n_2 is the number of samples in a spike and n_1 is the index of first zero crossing after the spike has been detected (Figure 2.19).

2.5.4 Dimensionality reduction

As discussed in section 2.5.3.1, there are some feature extraction methods with inherent feature encoding such as PCA. In some cases the dimensionality of the transformed spike waveform is as same as the extracted spike, or even the features domain transformation results in higher dimensionality. There are morphological features of a spike/feature waveform that can be chosen for the processing stage which are susceptible to noise as discussed in [97].

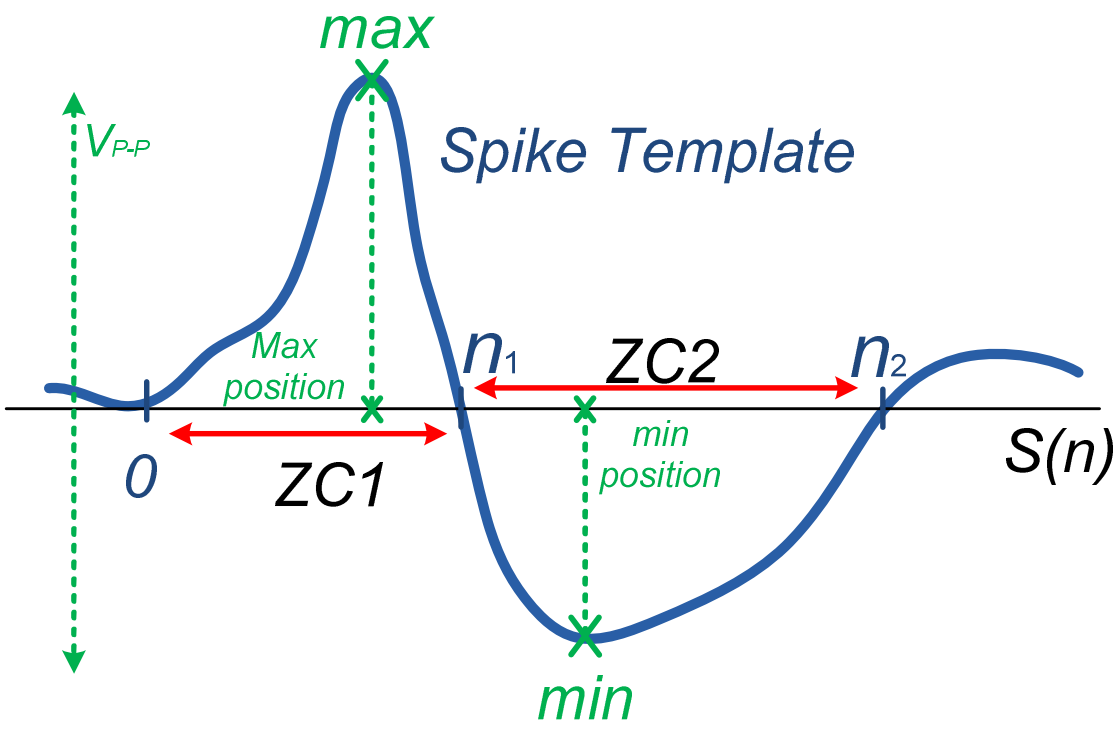

Figure 2.19: Spike template, *s(n)*, for TM clustering. Illustration of zero crossing features (*ZC1*, *ZC2*).

There are two advantages in selecting appropriate samples/features in spike sorting. Firstly, it reduces the dimensionality of data, which improves classification accuracy. Secondly, it reduces the required memory (hardware and area) and power consumption of clustering.

These advantages are achieved if the coefficients are selected properly. If true samples are not chosen, valuable information is lost and the reduced-dimensionality feature vector is a limiting factor for clustering. The simplest way that the dimensionality of features can be reduced is uniform sampling. There is necessarily no guarantee of obtaining an appropriate reduction performance when K ($K<N$) evenly spaced samples are chosen for clustering. This technique is simple and its computation cost is lower than other sophisticated approaches, but this type of sample selection might lead to non-segregation of clusters. Since the K samples are randomly selected relatively poor clustering accuracy is expected.

An algorithm was produced in [98] to choose the optimal samples (features) for sorting. The metric in [98] refines data based on mutual information of all the cluster combinations to retain the coefficients with multimodal distribution for clustering. Samples with bimodal distribution deviate

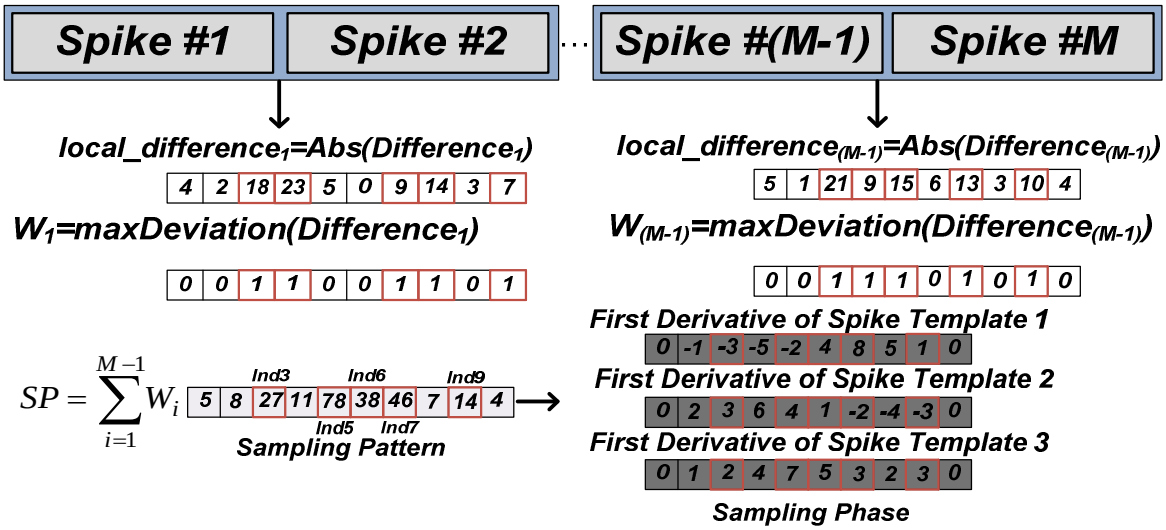

Figure 2.20: Asynchronous sampling of first derivative features. *M* spikes are considered for training period. Differences between two subsequent spikes are taken and weight *W*=1 allocated to coefficients with maximum deviation from normality. Sampling pattern vector (*SP*) is the summation of differences to find uncorrelated limits. Finally, sampling phase is done based on the sampling pattern from feature waveforms (*K*=5). The model was adopted from [100].

from unimodality or Gaussian distribution, so they exhibit multiple peaks and valleys as a sign of multimodality. The selection procedure can be done using offline training with the Lilliefors test [99]. However, the Lilliefors test is a statistical approach and cannot be used for implantable spike sorting hardware due to its complexity. In addition to the Lilliefors test, independent component analysis (ICA) [61] has been introduced as an efficient approach that minimizes the mutual information which leads to finding non-Gaussian directions or maximum deviation from normality.

As an alternative to the Lilliefors test and ICA (both need offline training), asynchronous sampling (AS) of original spike waveforms (or feature waveforms) is proposed in [100] as an efficient approach, not only in terms of computational complexity but also accuracy to reduce dimensionality ($N \rightarrow K$). The algorithm is divided into the sampling pattern generation phase and the sampling phase (the process is shown in Figure 2.20). In the former phase the localized shape differences between the aligned spike waveforms (feature waveforms) of spike templates are noted. In the sampling phase the marked areas are extracted for sorting. In the first stage the maximum-difference test [101] is used to generate the sampling pattern. *M* spike waveforms are chosen and the differences between two consecutive spike waveforms are taken. Then for each

Table 2.4: Dimensionality reduction procedure.

Sampling pattern generation phase 1. Take the difference between two consecutive spikes and write its absolute value in *local_difference.* 2. Allocate W=1 to the K largest coefficients in *local_difference.* 3. Sum all weight vectors W_1 to $W_{(M-1)}$ to computer *SP*. 4. Identify the K largest value indices in *SP*.
Sampling phase Extract the coefficients corresponding to the selected K indices from the original spike or feature waveforms. Sampling is performed asynchronously according to the location of the maximum differences (e.g., MD1, MD2 and MD3; amp(MD1) > amp(MD2) > amp(MD3)). An example is shown in Figure 2.6.

pair of spike waveforms $W = 1$ is allocated as a weight to the K samples with the highest difference in value and $W = 0$, otherwise. The process continues up to the M^{th} spike to extract the differences between all the spikes in the training period. The sampling pattern vector is written as follows:

$$SP = \sum_{i=1}^{M-1} W_i \tag{2.15}$$

where W_i is the weight vector of the i^{th} difference. *SP* represents the absolute value of the accumulated local differences between all the clusters at the end of the training period. Length reduction is obtained by identifying the K largest value indices in the sampling pattern. Finally, in the sampling phase the coefficients corresponding to the identified K indices in the previous phase are extracted from the original spike or feature waveforms. The main steps of the algorithm (training period) are listed in Table 2.4. In terms of complexity the proposed method is much less complex than the Lilliefors test, and thus is suitable for efficient hardware implementation.

Figure 2.21 shows clustered spike waveforms with various difficulty levels (C_Easy2_01 and C_Difficult2_01) with their corresponding sampling profiles. As seen in Figure 2.21(b) and 2.21(d) the marked local differences occur asynchronously between the spike templates.

Factors contributing to the sampling pattern generation include temporal alignment and upsampling, the latter alleviates the effect of sampling jitter and noise. This leaves a smooth pattern for sampling. For dimensionality reduction, two extraction models could be considered: i) select the K largest coefficients from *SP*, and ii) select the K largest coefficients around the major peaks (typically 2-3 main peaks could be chosen and around each peak 4-5 samples are selected). The

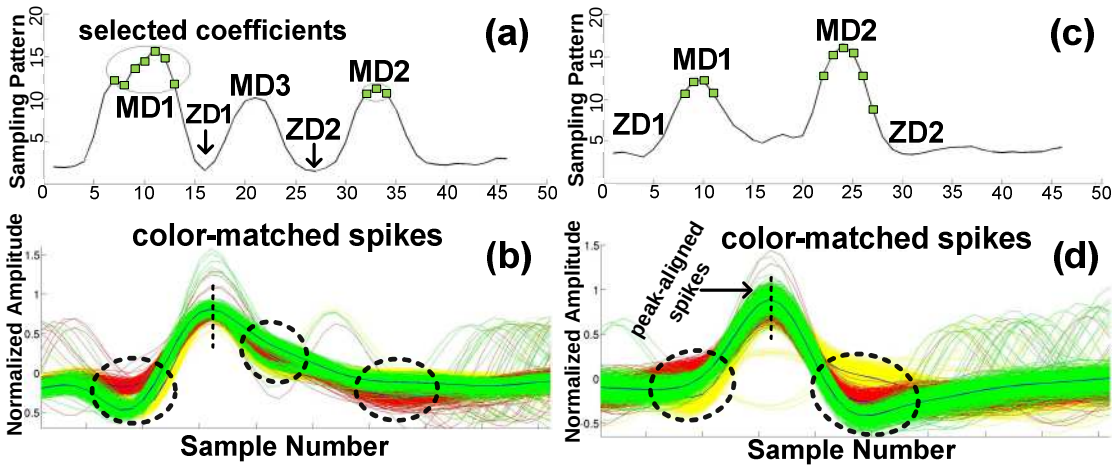

Figure 2.21: Sorting results with absolute value of sampling pattern for different datasets from original spike shapes. (a) and (b), C_Easy2_01. (c) and (d), C_Difficult2_01. Maximum difference (MD) and zero difference (ZD) areas are annotated on the sampling profiles. The *K* (=10 here) largest points are depicted with green squares. Adapted from [91].

second method is more computationally demanding but might result in performance improvement. In addition, major peak selection and upsampling could be used for sample selection (but has relatively high complexity). In the implementation used in this thesis the *K* largest coefficients from *SP* were selected without upsampling.

2.5.5 Clustering

The final sorting stage is the assignment of spikes to their originating neurons as clusters in the feature space. Online and unsupervised clustering is one of the most complex parts of sorting process. In primitive sorting methods, cluster boundaries were defined by hand in feature space [91] which might be resulted in human error during the separation process [102]. There exist different clustering methods for classification/clustering including: *K*-means [91], fuzzy c-means clustering [101], Bayesian clustering [103], expectation maximization (EM) [104], artificial neural networks (ANN) [105], valley seeking clustering [106], superparamagnetic clustering (SPC) [89] and online sorting (O-Sort) [13]. A comprehensive study of clustering methods is provided in [107].

2.5.5.1 *k*-Means Clustering

The *k*-means [50] algorithm is one of the most accurate clustering algorithms which can be utilized in spike sorting process. The *K*-means clustering method aims to partition the selected feature space into *k* clusters, in which each spike/feature template belongs to a cluster with the nearest mean. The distance metric is defined as:

$$\arg_S \min = \sum_{(i=1)}^{k} \sum_{(x_j \in S_i)}^{1} \left\| x_j - \mu_i \right\|^2 \tag{2.16}$$

where (x_1, x_2... x_n) is the set of spike features, *k* is the number of clusters and S_K are the different sets (i.e., spike classes). Although this method is very simple to implement, the major drawback is that knowledge of the number of clusters is required for the training phase. This method cannot be used for BMI applications since it is not an online, adaptive and unsupervised method. In some BMI applications, monitoring the number of active neurons must be automatic. This limitation can

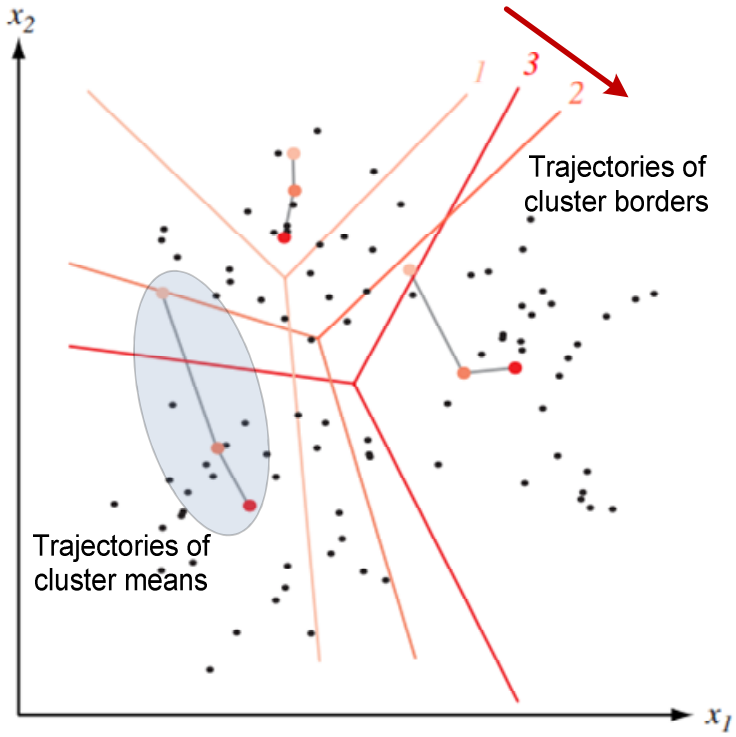

Figure 2.22: Trajectories for the means of the *k*-means clustering procedure applied to two-dimensional data [50].

be addressed by estimating the number of neurons as proposed in [108] which can reliably estimate the number of clusters but there are some invalid cases for *K*-means initialization.

2.5.5.2 Artificial Neural Networks (ANN)

ANN [105] is based on the elementary principles of the human neural operation system. A large variety of networks have been constructed to imitate the human brain for realization of systems with high order of complexity. All networks are composed of artificial neurons, and connections between them, which determine the behavior of the network (Figure 2.23a)). The input layer is characterized by input signals (*X*1…*Xn*).The number of neurons in the hidden layer is chosen empirically by the user. Finally, the output layer comprises neurons for the *k* classes (class1…classK). Each connection in the neuronal network is characterized by a weight factor which is modified by successive iterations during the training process. The state of each neuron is

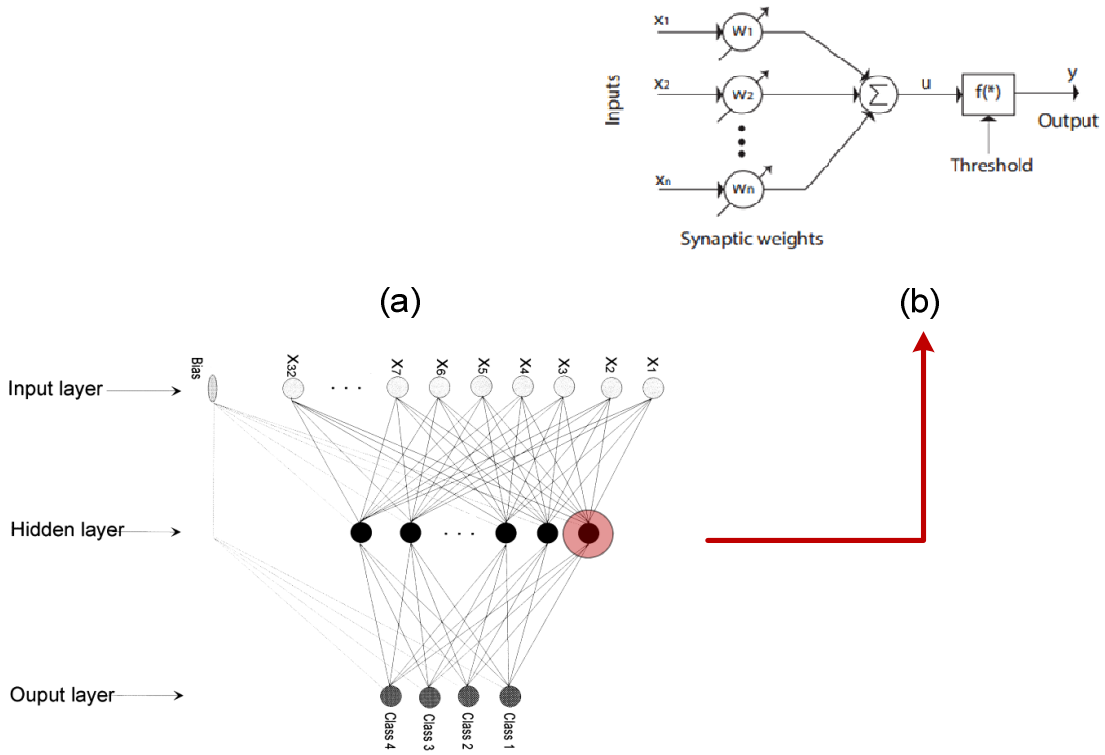

Figure 2.23: (a) Structure of a neural network used in the classification. (b) Basic artificial neuron model, where f (*) is the activation function. The synaptic weights are updated in the training phase by supervised or unsupervised algorithms. (Adapted from [105]).

configured by the input variables and the state of the other existing neurons in hidden and output layers are determined subsequently (Figure 2.23(b)).

2.5.5.3 SPC

SPC is an unsupervised clustering algorithm that has been used in spike sorting application. The first step of SPC clustering is interaction strength calculation between each data point and its K-nearest neighbors. The algorithm is initiated by finding the interaction strength between the selected features of each spike *i* by a point x_i in an m-dimensional feature pace and the *K*-nearest neighbors:

$$J_{ij} = \begin{cases} \frac{1}{K_{nn}} \exp\left(-\frac{\left\| x_i - x_j \right\|^2}{2a^2} \right) & \text{if } x_i \text{ is a nearest neighbor of } x_j \\ 0 & \text{else} \end{cases} \tag{2.17}$$

where *a* is the average nearest-neighbors distance and K_{nn} is the number of nearest neighbors. The interaction strength is decreased exponentially with increasing Euclidean distance which is determined based on the similarity of x_i and its neighbors. In the second step, random states are assigned to each point x_i from 1 to q, where q is a constant representing the number of possible spins. For each temperature (e.g. *T*=0:0.2:0.2), Monte Carlo simulation is performed. During each iteration a frozen bond (state) between nearest neighbor points x_i and x_j is considered if they satisfy the probability equation which is expressed as:

$$p_{ij} = 1 - \exp\left(\frac{J_{ij}}{T} \delta_{si,sj} \right) \tag{2.18}$$

where $\delta_{si,sj}$ is equal to 1 for the points in the frozen bond. This procedure is repeated for another point several times in order to get representative statistics. Having performed the Monte Carlo runs, the connectedness probability of a point is calculated and if it is greater than the specified threshold (θ), the point belongs to the cluster.

2.5.5.4 Online Sorting (O-Sort) Clustering

This algorithm provides real-time mapping of spikes to single neuron activity for closed-loop applications. The operation of O-Sort is as follows: 1) Initialization: Assign the first data point to its own cluster. 2) Calculate the distance between the next data point and each cluster centroid. The distance metric could use, for example, the Euclidean norm or the ℓ1-norm. 3) If the smallest distance is less than the merging threshold T_M, assign the point to the nearest cluster and re-compute that cluster's mean. Otherwise, start a new cluster. 4) Check the distances between each cluster and every other cluster. If any distance is below the sorting threshold T_S, merge those two clusters and recompute its mean. Steps 2-4 are then repeated indefinitely. In the simplified version of the algorithm proposed in [13] (and used later in the thesis) $T_M = T_S = T$. The threshold T is defined as $T = S(\sigma_r)^2$, where σ_r is the average standard deviation of the data computed continuously with a long (~1 min) sliding window, and S is the number of datapoints of a single waveform.

O-Sort is simple in operation with good complexity-accuracy tradeoff and satisfies online sorting constraints (memory and power). The algorithm is adaptive, thus nonstationarity of data in time is applied to the cluster position and number of clusters. A disadvantage of O-Sort is that it may split clusters into sub-clusters leading to a reduction in clustering performance. The created sub-clusters are not matched with any source and are considered as noise clusters. An example illustrating the

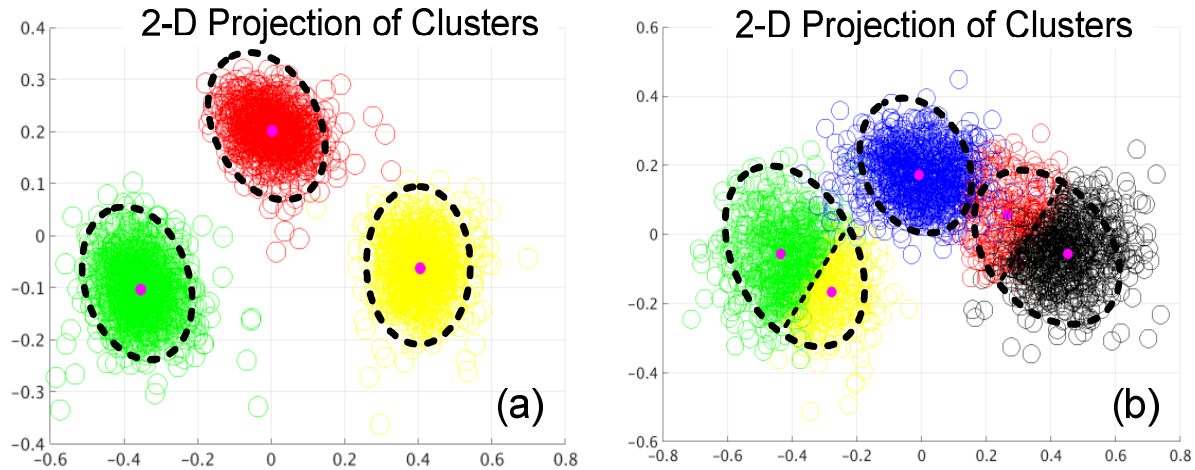

Figure 2.24: Clustering results using two different clustering methods, including (a) *K*-means and (b) O-Sort. The overclusterig issue is illustrated using different colors and borders are depicted with dashed line. The used data in this demonstration discussed previously in section 2.4.1.

clustering results using *K*-means and O-Sort is shown in Figure 2.24. O-Sort tends to overcluster, dividing a cluster to some sub-clusters especially when the noise level is increased.

2.6 Conclusion

In this chapter the process of spike sorting which is an important concept for many scientific and clinical applications has been examined. For each constituent building block of the sorting chain, different methods have been detailed identifying advantages and disadvantages. There is a trade-off between the accuracy of processing methods and computational complexity. Nevertheless careful design is needed to obtain a clustering accuracy of over 90% for on-chip spike processors with real time operation capability. Clustering performance will be investigated by introducing highly optimal and tunable spike-sorting methods in the Chapters 3 and 5.

CHAPTER 3

New Feature Extraction Method Using Extrema Sampling of Discrete Derivatives for Spike Sorting

3.1 Introduction

As described in previous chapters, the spike-sorting concept has been a considerable interest in recent years. There exist two approaches to realize highly efficient on-chip spike sorting: template-based and feature-based spike sorting. In feature-based spike sorting methods the main trade-off is about introducing an additional step to reduce both dimensionality of extracted spike waveforms and minimizing the buffering requirements in hardware implementation. Feature extraction is usually performed by an initial mathematical transformation which comprises selection of a subset of critical features which are the best representatives of each spike waveform.

This chapter introduces a new energy efficient feature extraction method based on extrema analysis (positive and negative peaks) of spike shapes and their discrete derivatives [109] with different sampling intervals. It runs in real-time and does not require any offline training. It is shown that compared to other methods using online sorting it offers a better trade-off between accuracy and computational complexity. Unlike other systems, the spike sorting procedure requires no multiplication operation which is computationally expensive, power hungry and requires appreciable silicon area. These are important features in implantable devices particularly when there is a high channel count.

The key characteristics of the proposed feature extraction method are:

1) Simplicity and effectiveness.
2) Low complexity (power consumption) for compatibility to the hardware implantable devices restriction as discussed in Chapter 2.
3) Simplicity of expansion for multi-channel processing.
4) Capability of reconfigurable (adaptive) hardware implementation.

This chapter is organized as follows.

Section 3.2 describes prior-art and the new method. It demonstrates that the subset of features obtained by extrema sampling of the decomposed spike waveform provide the majority of separation between the identified active neurons and has high immunity against spike similarity and noise level. The computational requirements and spike sorting accuracy of the proposed method are quantified and compared against commonly used feature extraction methods. The new method achieves a relatively high spike sorting accuracy with very low computational requirements, hence is suitable for low power hardware implementation. Section 3.3 presents the results and discussion of comparative performance analysis. These include simulation results for clustering accuracy using synthetic data, clustering using synthetic and recorded in-vivo neural data, and complexity analysis. A metric based on the projection test is proposed for quantifying the discrimination degree of clusters. It is used to compare the sorting quality of the proposed method against other work. In addition, the overall complexity for sorting is optimized using the $\ell 1$-norm distance calculation. Finally Section 3.4 presents concluding remarks.

3.2 Algorithms

By way of example, Figure 3.1 shows the signal processing and control chain including spike sorting for a prosthetic hand. Spike sorting is the process of grouping the recorded spikes into clusters based on the similarity of their shapes. The process can be divided into the following steps:

1) S*pike detection and alignment*, separating spikes from noise and aligning the spikes to a common point;
2) *Feature extraction*, extracting features of the spike shapes which gives a *dimensionality reduction*, i.e., going from a space of dimension N (with N the number of data points per spike) to a dimensional space of a few features;
3) C*lustering*, grouping spikes with similar features into clusters, corresponding to the different neurons.

In this chapter, the focus is on feature extraction and clustering.

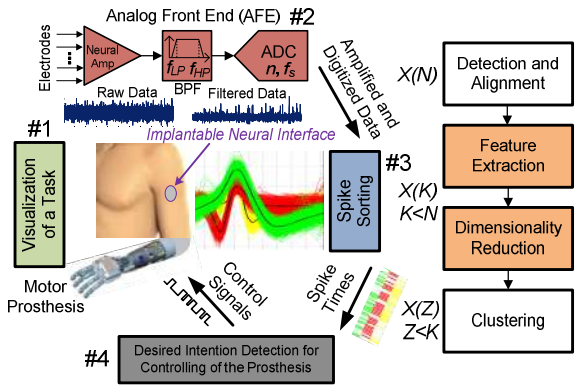

Figure 3.1: Schematic description for a BMI that relies on real-time recording and processing of neural activity to control a robotic prosthetic arm. The implanted electrodes in the recording site are used to monitor the activity of large populations of single neurons simultaneously. Spike sorting is used to classify the recorded spikes (active neurons) to their source of origin. The combined activity of classified spikes is transformed by a decoding (mathematical) algorithm into arm control signals (arm trajectory signals) that can be used to control the movements of the robotic prosthetic arm. The closed-loop control system is stablished by providing the subject with both visual and tactile feedback signals [110].

3.2.1 Off-Chip Feature Extraction Methods

Principal component analysis (PCA) [94] has been the most commonly used algorithm for spike sorting because it yields an efficient coding of spikes (only the first 2-3 principal components need be retained). However, PCA requires offline training which is not compatible with online real-time spike sorting, and calculating the covariance matrix of the data demands high computational cost and hardware resources. In addition there is no guarantee of optimal separation of clusters [89].

Another common technique is the discrete wavelet transform (DWT). This is a multi-resolution algorithm that provides good time resolution at high frequencies and good frequency resolution at low frequencies. But the convolution of the wavelet function and original signal requires many multiplications and additions per spike, resulting in a high computational cost. Both PCA and DWT have generally been used for off-chip spike sorting.

3.2.2 Proposed Method for On-Chip Feature Extraction

A simplified model of the DWT was presented in [109]. In it, discrete derivatives (DDs) are computed by calculating the slope at each sample point over a number of different time scales:

$$\mathrm{DD}_{\delta}(n) = s(n) - s(n-\delta) \tag{3.1}$$

where *s* is the spike waveform, *n* is the sample point and δ is the scaling factor(time delay). The equation shows subtraction between the samples *n* and ($n - \delta$). Normally *P* DDs can be calculated per spike with different scaling factors to give multi-resolution spike decomposition, which corresponds to different frequency bands. This yields $P\times$ dimensionality expansion of feature space compared to the number of samples of an aligned spike (i.e., $N \rightarrow P \times N$). For example, the size of the feature space will be $N = (P = 3) \times (N = 45)$ samples per spike for DDs of three values of δ. Feature space dimensionality directly impacts on the computational complexity of spike sorting. As an illustration, the DDs of two typical spike waveforms with delay values of δ equal to 1, 3 and 7are shown in Figure 3.2.

Feature extraction based on extrema sampling (positive and negative peaks of DDs) is proposed here as an efficient approach not only in terms of computational simplicity but also accuracy. Retention of a subset of features significantly reduces the dimensionality from $P \times N$ to *K*, where *K* is the number of selected features for the clustering stage ($K < N$). For example, using $\delta = 1, 3, 7$ the following features are identified in Figure 3.2: a) positive peaks $DD|_{\delta = 1,3,7(max)}$; b) negative peaks $DD|_{\delta = 1,3,7(min)}$; and c) peak-to-peak amplitude ($V_{p\text{-}p}$)of each DD. Different combinations are introduced in this section by sweeping the decomposition window length (δ) in order to explore the frequency sub-bands (from δ=1 to δ=7) which accommodate the most informative (multimodal) features [100]. To compare with other work (in terms of classification accuracy and computational complexity) the following nine permutations of feature sets are considered:

- Combination 1: $DD|_{\delta = 1,3,7(max)}$ and $DD|_{\delta = 1,3,7(min)}$.
- Combination 2: $V_{p\text{-}p}$ of $DD|_{\delta = 1,3,7}$(i.e., $DD|_{\delta = 1,3,7(max)} - DD|_{\delta = 1,3,7(min)}$).
- Combination 3: $DD|_{\delta = 3,5(max)}$ and $DD|_{\delta = 3,5(min)}$.
- Combination 4: $DD|_{\delta = 3,5(max)}$ and $DD|_{\delta = 3,5(min)}$, together with $V_{p\text{-}p}$ of $DD|_{\delta = 3,5}$. $DD|_{\delta = 5}$ is not annotated in Fig. 3 for brevity.
- Combination 5:$DD|_{\delta = 3,7(max)}$ and $DD|_{\delta= 3,7(min)}$.
- Combination 6: Features in Combination5 together with $V_{p\text{-}p}$ of $DD|_{\delta= 3,7}$.
- Combination 7: $DD|_{\delta = 3,7(max)}$ and $DD|_{\delta= 3,7(min)}$ together with original spike positive and negative peaks.
- Combination 8: $DD|_{\delta = 3,7(max)}$ and $DD|_{\delta= 3,7(min)}$ together with original spike height.

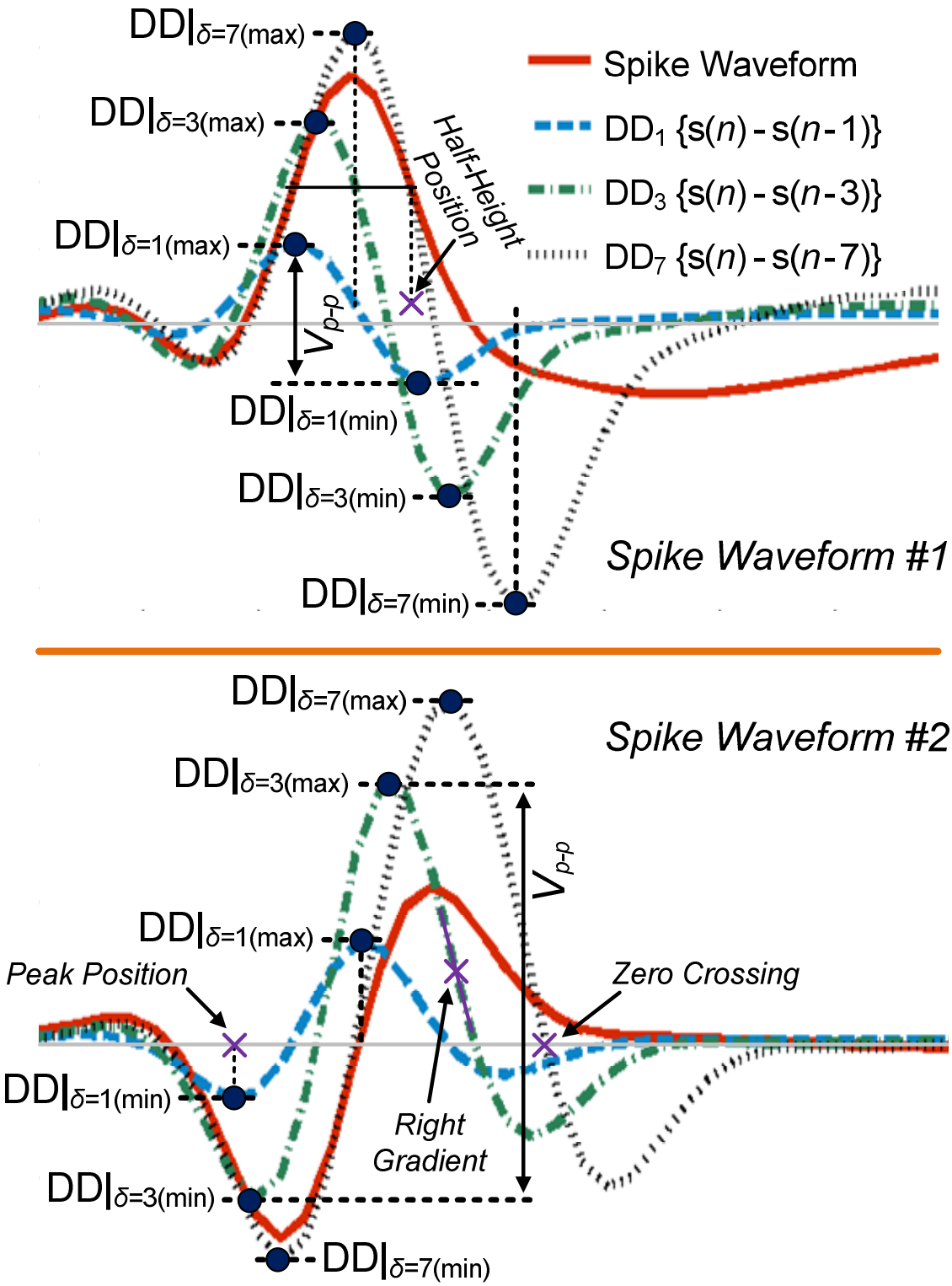

Figure 3.2: Two spike waveforms (spike shapes) and their discrete derivatives. The positive peaks, negative peaks and peak-to-peak amplitudes are annotated. Other features such as spike gradients and peak position are also depicted.

- Combination9:DD$|_{\delta=7(\max)}$ and DD$|_{\delta=7(\min)}$ together with original spike positive and negative peaks.

In addition to the above permutations, various values of δ and geometric characteristics such as positive (or negative) signal energy, half-height position, right (or left) spike gradients, peak position and zero crossing points could be considered to define other possibilities. Some of the mentioned features are annotated in Figure 3.2.

3.2.3 Comparison with Other Feature Extraction Methods

1) *Waveclus* [89]: In this algorithm, the combination of the wavelet transform with super-paramagnetic clustering (unsupervised clustering method with offline training) is used for unsupervised and online spike sorting. Feature extraction is calculated based on four-level multi-resolution decomposition using Haar wavelets which results in 64 wavelet coefficients for each spike. Then the Kolmogorov-Smirnov test for normality is used to select the first 10 coefficients with the largest deviation from normality for the sorting stage.

2) *Discrete Derivatives and Maximum Difference Test (DDs-MDT)* [101]: In this approach, the maximum difference test (MDT) is applied to each scaling factor of DDs to extract the multimodal coefficients. The MDT is a simplified model of the Lilliefors test for selecting the uncorrelated directions (minimum mutual information) for blind signal separation. Samples with bimodal distribution have deviation from unimodality or Gaussian distribution, thus they exhibit multiple peaks and valleys as a sign of multimodality.

3) *First and Second Derivatives* [111]: Here the first and second derivatives of a spike represent its geometrical characteristics. The first derivative (FDV) interprets the gradient variations of a spike shape. It is defined as:

$$FDV\ (n) = s(n) - s(n-1) \tag{3.2}$$

The second derivative (SDV) highlights low frequency coefficients by computing the difference of the samples n and $(n-1)$ of the FDV. That is,

$$SDV(n) = FDV(n) - FDV(n-1) \tag{3.3}$$

In the FDV and SDV extrema (maximum and minimum peaks) are used to distinguish the clusters. This method is referred to as FDVSDV for the rest of the paper in the accuracy and complexity discussions.

4) *DDs and Uniform Sampling (DDs-USAMP)* [101]: In this method, after computing the DDs of the spike with three different values for the delay (δ = 1, 3, 7), uniform sampling is performed to select the subset of features. Seven coefficients are selected for each DD level and 21 per spike since the median accuracy was in the SNR range of 15-20 dB. Uniform sampling is a blind approach to decrease the dimensionality from N to K (in this case from 3 × 48 to 21) but this type of sample selection could lead to non-segregation of clusters.

5) *Spike Shape* [112]: In this method all the samples of detected and peak-aligned spikes (without upsampling) are used for calculating the similarity measure between the mean of the spikes in clustering.

3.2.4 Sorting (O-Sort Clustering)

For clustering, O-Sort has been selected. It is the only online, automatic and unsupervised algorithm that is suitable for hardware implementation [13]. This algorithm provides real-time mapping of spikes to single neuron activity for closed-loop applications. The operation of O-Sort is as follows:

1) Initialize by assigning the first data point to its own cluster.
2) Calculate the distance between the next data point and each cluster centroid. The distance metric could use, for example, the Euclidean norm or the ℓ1-norm.
3) If the smallest distance is less than the merging threshold T_M, assign the point to the nearest cluster and re-compute that cluster's mean. Otherwise, start a new cluster.
4) Check the distances between each cluster and every other cluster. If any distance is below the sorting threshold T_S, merge those two clusters and recompute its mean. Steps 2-4 are then repeated indefinitely. In the simplified version of the algorithm (proposed in [13] and used herein), $T_M = T_S = T$. The threshold T is defined as $T = S(\sigma_r)^2$, where σ_r is the average standard deviation of the data computed continuously with a long (~1 min) sliding window, and S is the number of datapoints of a single waveform.

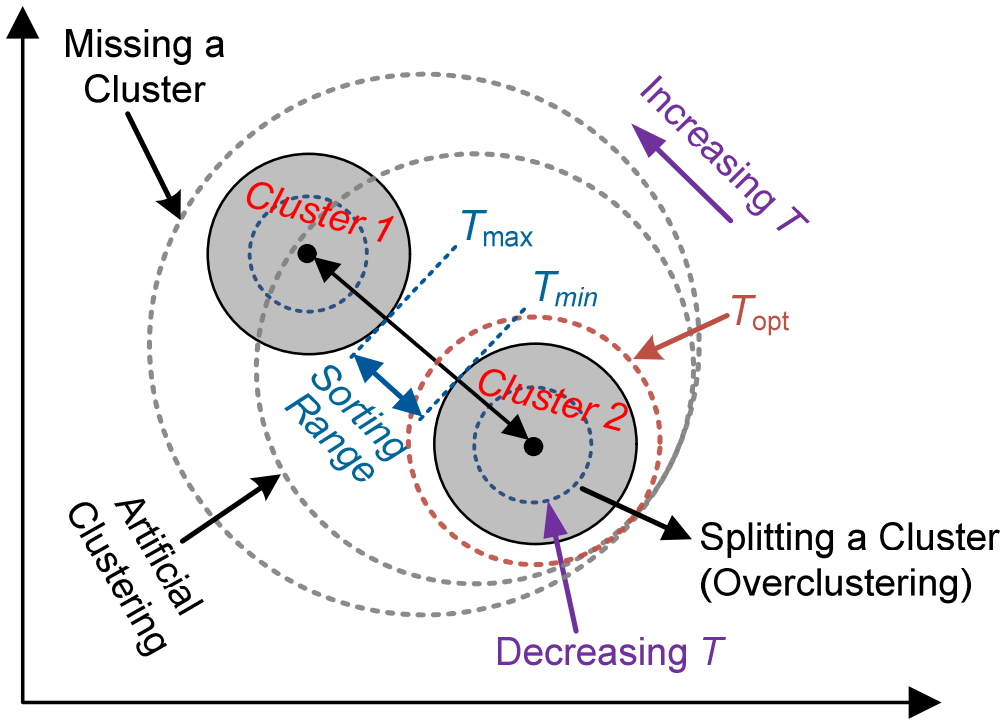

Figure 3.3: 2-D representation of feature space for two clusters from dataset 4 with the highest similarity. The effect of increasing (or decreasing) the threshold T is depicted. Different threshold levels (T_{opt}, T_{max} and T_{min}) and sorting range are indicated. For $T > T_{max}$ the risk of missing a cluster and artificial clustering is high. For $T < T_{min}$ the main cluster would artificially be split into two or more sub-clusters.

O-Sort is simple in operation with good complexity-accuracy tradeoff and satisfies online sorting constraints (memory and power). This algorithm is adaptive thus nonstationarity of data in time is applied to the cluster position and number of clusters. A disadvantage of O-Sort is that it may split clusters into sub-clusters leading to a reduction in clustering performance. The created sub-clusters are not matched with any source and are considered as noise clusters.

3.3 Results and Discussions

3.3.1 Determination of the Optimal Threshold

The procedure adopted to determine the optimal threshold T is as follows:

1) The range of threshold values is determined via the dataset with the highest similarity index between the spike shapes. The maximum limit for threshold (T_{max}) is defined using dataset 4 (which has the highest similarity measure) when no clusters are missed and there is no artificial

clustering (Figure 3.3). Artificial clustering erroneously allocates spikes to a cluster. The threshold value can be decreased to a minimum limit (T_{min}) below which overclustering occurs. Overclustering is the splitting of a single cluster into multiple clusters. T_{min} is determined when clusters in dataset 4 start to overcluster. For $T_{min} < T < T_{max}$ no clusters will be missed and no overclustering will occur.

2) To determine the sorting accuracy, the optimal threshold (T_{opt}) for each method (published and proposed) is found by sweeping the threshold value through the range $T_{max} - T_{min}$ using the spike data bank in section 2.4.1.

Using this procedure determines T_{opt} for each method when investigating its effectiveness using different metrics. Finding optimal (T_{opt}) threshold is resulted in 5-8% improvement in clustering performance.

3.3.2 Classification Accuracy

In this subsection the nine combinations of feature set proposed in Section 3.2.2 are evaluated in terms of classification accuracy to determine the be one in conjunction with O-Sort. This is then compared with the feature extraction methods outlined in Section 3.2.3. Classification accuracy is defined as:

$$CA_{CC} = \frac{TPCC}{NTS} \times 100\% \tag{3.4}$$

where TPCC is the number of truly detected and correctly classified spikes and NTS is the number of truly detected spikes. NTS = DTS – (FPS + MS), where DTS is the number of detected spikes, FPS is the number of false alarm spikes due to noise or overlapping spikes, and MS is the number of missed spikes.

The average sorting results are summarized in Table 3.1. The methods were evaluated across all datasets and noise levels. Combination 5 with dimensionality (K) of 4 achieves the highest CA_{cc} whereas Combination 8 (with K = 9) achieves the lowest CA_{cc}. The methods were also examined with overclustering ratio criteria. It was observed that Combination 5 achieves the lowest overclustering whereas Combination 2 in conjunction with O-Sort generally tends to divide a cluster into sub-clusters. The set of features used in Combination 5 is therefore selected and will from hereon be referred to as the $DD|_2$-Extrema method.

For comparison, the CA_{cc} of the methods listed in Section 3.2.3 was investigated, in particular, Waveclus (K = 10), DDs-MDT (K = 21), DDs-USAMP (K = 21), Spike Shape (K = 45) and PCA3 (projection of first three principal components). The results (averaged across all noise levels) are shown in Figure 3.4. The CA_{cc} of DDs-MDT drops significantly in datasets 2 and 3 due to the intense overclustering effect. The number of coefficients (K) representing deviation from normality is 21, i.e., 7 from each scaling factor (δ = 1, 3, 7). The results show that $DD|_2$-Extrema and DDs-MDT perform equally across datasets 1 and 4[3]. The former has better similarity tolerance in datasets 2 and 3 which means significant reduction in complexity or required memory. Spike Shape works satisfactorily for datasets 1 and 2 only. It significantly increases the computational complexity without improvement in performance. In this method the simplest metric to sort the spikes is the distance (e.g., Euclidean distance) between the unclassified spikes and the stored templates. As discussed in [113] the classification accuracy of Spike Shape declines when the spike waveforms have similar patterns (increasing the similarity index).

Table 3.1:
Classification accuracy comparison of the examined feature set combinations

Average Classification Accuracy					
Combination	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Mean
Combination 1 ($^*K$ = 6)	94.8%	92%	88%	85.8%	90.2%
Combination 2 (K = 3)	76.2%	78.6%	73.4%	68.6%	74.2%
Combination 3 (K = 4)	91%	86%	77.2%	79.8%	83.6%
Combination 4 (K = 6)	91.4%	86.8%	83. 6%	81.4%	85.8%
Combination 5 (K = 4)	95.8%	93.4%	87.8%	89.6%	91.6%
Combination 6 (K = 6)	91.2%	89.6%	80.4%	79.8%	85.2%
Combination 7 (K = 6)	89.6%	84.6%	79.2%	74.8%	82.%
Combination 8 (K = 5)	75.6%	77.4%	70.4%	68.8%	73%
Combination 9 (K = 4)	80.4%	76.4%	75.4%	71.2%	75.8%

$^*K$ = number of features

[3] The used datasets for spike sorting methods evaluation are discussed in Chapter2, section 2.4.1.

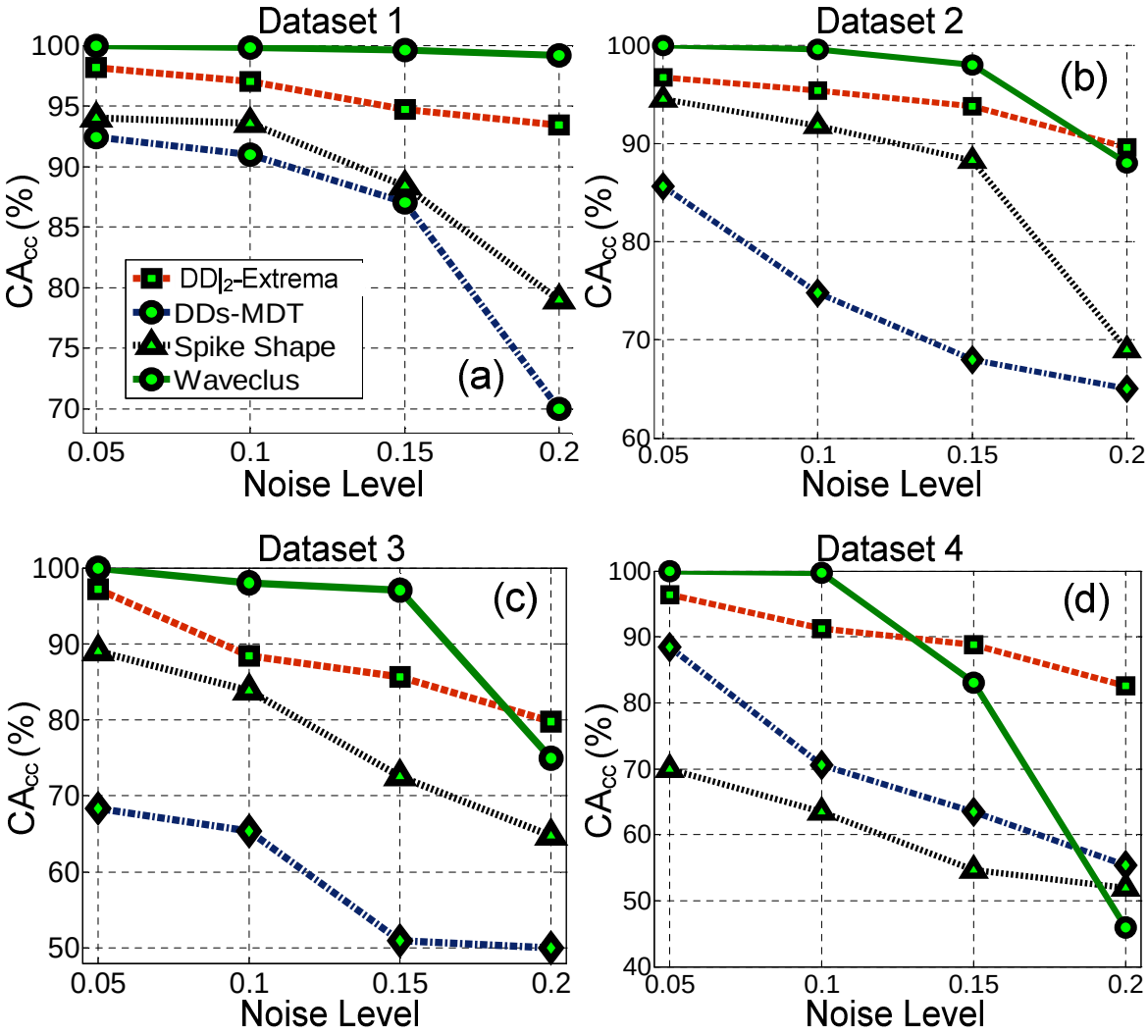

Figure 3.4: Comparison of classification accuracy between the DD|$_2$-Extrema method and other methods as a function of noise level for the four datasets. The result of Waveclus in [89] is used for comparison. The optimum threshold (T_{opt}) is calculated for each feature extraction method.

PCA uses maximum variance, correlated coefficients, unlike the Waveclus and DDs-MDT. In the test for *C_Difficult1_noise* and *C_Difficult2*_noise the sorting algorithm did not distinguish one of the three clusters with PCA feature vectors. Although PCA is computationally complex, there is no guarantee of efficient results. Selecting the coefficients with the largest variance does not necessitate deviation from normality and it may compromise the sorting performance. Feature space probability density function of two different clusters using PCA overlap (correlated directions) was investigated in [114]. PCA and DDs-USAMP did not perform well in the tests.

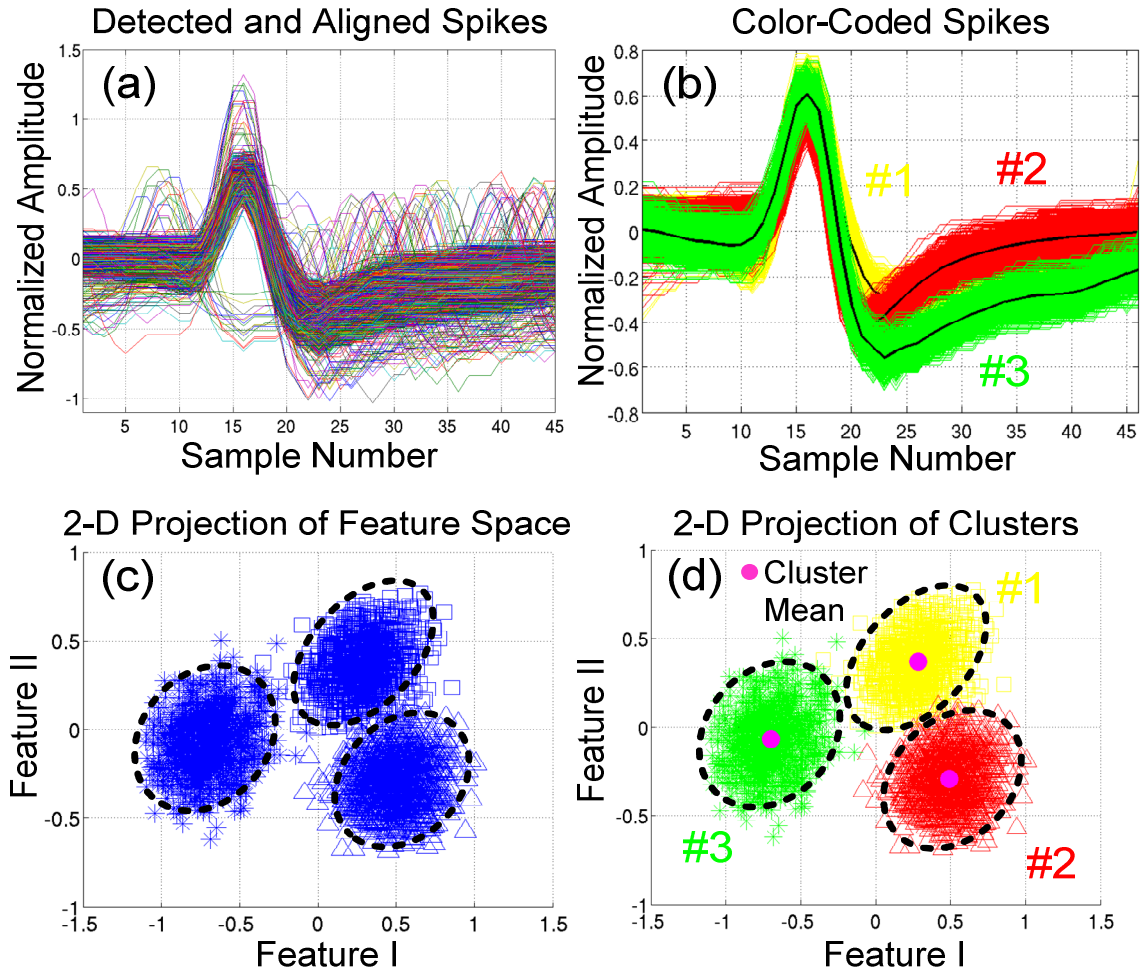

Figure 3.5: Test dataset 3 (*C_ Difficult1_0.05*) showing: (a) Detected and peak-aligned spikes. (b) Color-coded spikes corresponding to different neurons(#1 yellow,#2 red,#3 green). (c) 2-D projection of feature space as seen by the spike classifier. (d) 2-D projection of spike clusters. The markers "□", "Δ", and "*" refer to the features for the members of the first, second, and third spike templates, respectively.

Their average CA_{cc} with the O-Sort classifier is 66.4% and 61%, respectively, and therefore they are not plotted in Figure 3.4.

In conclusion, the best performance for accuracy-dimensionality is DD|$_2$-Extrema (K = 4). It exhibits superior similarity and noise immunity.

3.3.3 Clustering Results with Synthetic Data

The evaluation of spike sorting with DD|$_2$-Extrema is shown in Figure 3.5 using dataset 3 (*C_ Difficult1_0.05*). Figure 3.5(a) shows detected and aligned spikes, Figure 3.5(b) shows color-coded

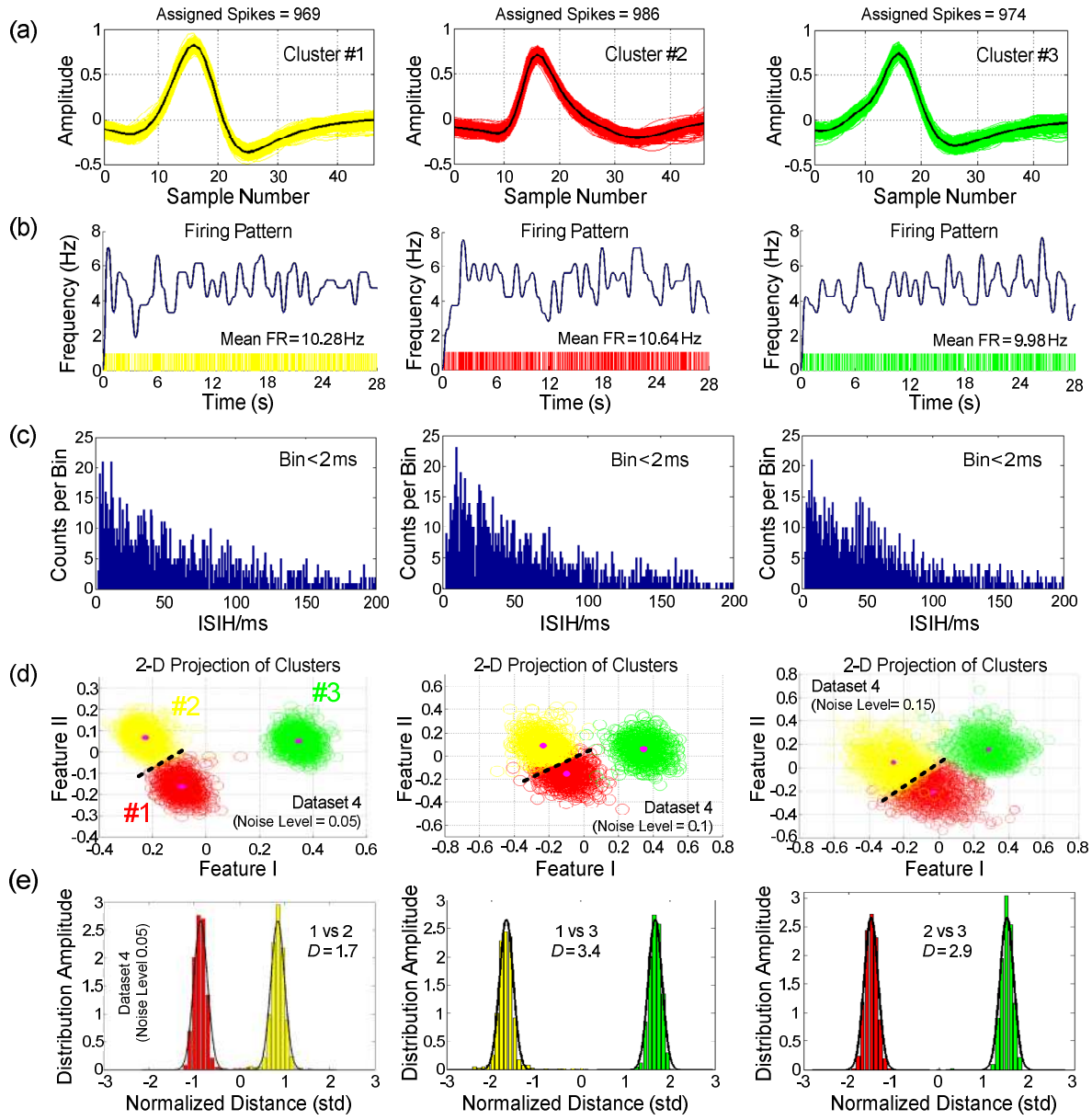

Figure 3.6: Sorting results of *C_Difficult2_005*. (a) Color-coded clusters with number of assigned spikes in each cluster (#1 yellow, #2 red, #3 green. Note: The colors are not matched with Fig. 4). The amplitude is of arbitrary units. (b) Corresponding firing pattern which depicts firing rate of each neuron (0-28 s). Approximated firing rate determined by the Gaussian window function. The mean firing rate (FR) is annotated in each plot. (c) Inter-spike interval histogram (ISIH) of each cluster. (d) 2-D projection of clusters for *C_Difficult2_0.05*, *C_Difficult2_0.01* and *C_Difficult2_0.15*. (Spikes have been colored according to the ground truth). (e) Illustrates projection test using probability density functions for the three combinations of cluster (*C_Difficult2_0.05*).For each combination of neurons the distance between the two distributions is described by how many standard deviations they are apart (*D* value in each plot).

spikes in each cluster, Fig.5(c) shows mapping of the spikes on the feature space as seen by the spike classifier, and Figure 3.5(d) shows color-coded mapping to distinguish between the neurons. The clusters are distinguished with the knowledge of true identities from the neural simulator. For clarity, noise events and overlapping spikes were not plotted.The typical way of illustrating the isolated units is superimposing detected spikes with different colors.

The detailed sorting results using dataset 4 are shown in Figure 3.6. The two statistical tests, namely the inter-spike interval histogram (ISIH) test and the projection test as discussed in [115], were used to quantitatively assess the sorting quality. A total of 3040 raw waveforms were detected, 2929 (96.4%) of which were assigned to one of the three well-separated single units (969, 986, and 974 for each cluster, respectively). Figure 3.6(a) shows the normalized raw waveforms and the mean waveform for each of the three clusters. Each neuron is color-coded across the whole figure (#1 yellow, #2red, #3 green). Figure 3.6(b) shows the firing pattern of each neuron across a time window of 0-28 s. The mean firing rate (FR) [116] of neurons #1, #2 and #3 is 10.28 Hz, 10.64 Hz and 9.98 Hz, respectively. Each diagram in Figure 3.6(b) also has the raster plot (or spike times) corresponding to the temporal firing of each neuron. Figure 6(c) shows the ISIH for each neuron. The ISIH window should not be less than the action potential refractory period (< 2 ms after alignment). Single-unit activity is stated when the spike waveform is clearly distinguishable with no ISIH less than the refractory period. Figure 3.6(d) shows from left to right the 2-D projection of clusters with increasing noise level (0.05, 0.1 and 0.15). Increasing the noise level adversely affects the projection and may result in some degree of overlap, but it is observed that the borders of clusters are clear even in the most difficult dataset with noise level of 0.15. Figure 3.6(e) shows the results of the projection test using probability density functions for three combinations of cluster in *C_Difficult2_005*. The projection test [13], [117] is a one-dimensional representation between two known means with a distance indicator (*D*) which assesses the quality of clustering. This test shows whether or not spikes from multiple neurons are artificially assigned to a particular cluster by the sorting algorithm. Furthermore, it detects the invalid merging of two clusters. In Figure 3.6(e) the normalized distance between probability density functions for the three combinations of cluster are sufficiently large to permit correct assignment of spikes to unique neurons.

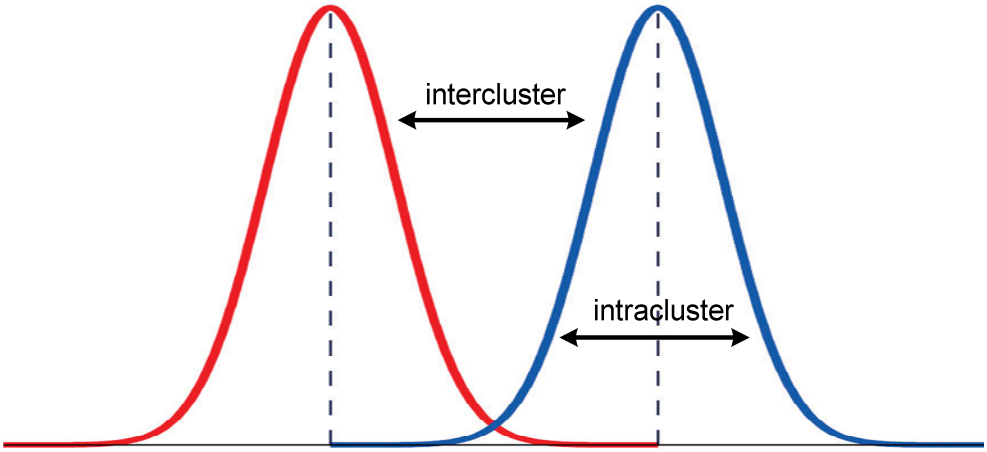

Figure 3.7: Representation of intracluster and intracluster for DisDeg calculation.

Inspired by [114] a quantitative metric based on the projection test is proposed for assessing the level of distinctness of the generated clusters. The metric for quantifying the discrimination degree of clusters is the ratio of intercluster distance to intracluster distance, defined as

$$\text{DisDeg} = \frac{\text{inter}}{\text{intra}} = \frac{\min_{\substack{i=1,2,\ldots,y \\ j=1,2,\ldots,y;\ i\neq j}} \left(\sum_{i=1}^{y} \sum_{j=1}^{y} PD_{ij} \right)}{T} \tag{3.5}$$

where PD_{ij} is the projection test distance between clusters i and j. This ratio is a metric for cluster quality measurements and is useful for optimization. The higher the value of DisDeg the better the cluster separation quality (Figure 3.7).

Figure 3.8 compares the separation confidence level of the Graph-Laplacian feature (GLF) [114], for DD$|_2$-Extrema and DDs-MDT. DisDeg verifies the efficiency of sorting compared to other techniques. Increasing the similarity in each dataset from *Easy1* to *Difficult2* and noise level adversely affects the cluster separation. The results of the statistical tests verify the efficiency of the proposed method.

3.3.4 Clustering Results with Recorded In-Vivo Neural Data

Collected neural signals from the peripheral median nerve in pig (obtained with a multi-electrode cuff in-vivo) were used to test the sorting performance of the DD$|_2$-Extrema. 6564 spike waveforms were detected from 24 single neurons in four different channels: eight in channel 1, five in channel 2, five in channel 3 and six in channel 4. The analysis of the sorting results is shown

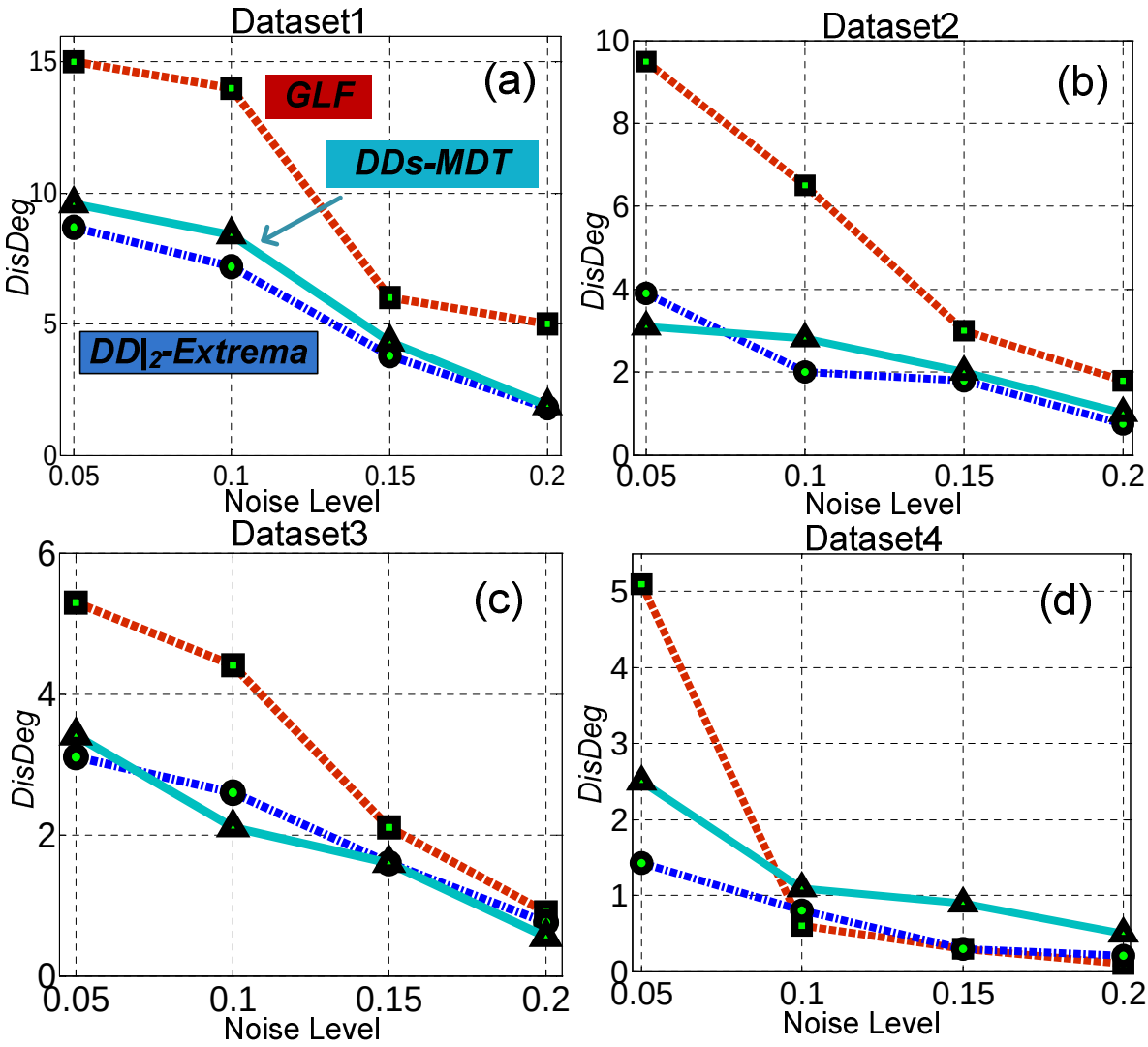

Figure 3.8: Discrimination degree of GLF, DDs-MDT and DD|$_2$-Extrema. For simplicity T=1. DisDeg of GLF was calculated using the quality metric in [114].

in Figure 3.8 for three of the channels. Figure 3.9(a) shows the mean spike waveforms of channels l-3; each color corresponds to a unique neuron. Figure 3.9(b) shows the sorting results of three similar neurons from channel 1. The ISIH test shown in Figure 3.9(c) verifies the accurate segregation between the chosen neurons. In Figure 3.9(d), the projection test quantifies the distance between every pair of clusters in channel 2. The normalized distance between each pair (standard deviation criterion) is large enough to conclude the efficient separation of clusters. In total 5973 (91%) of all detected spikes were assigned to the identified neurons.

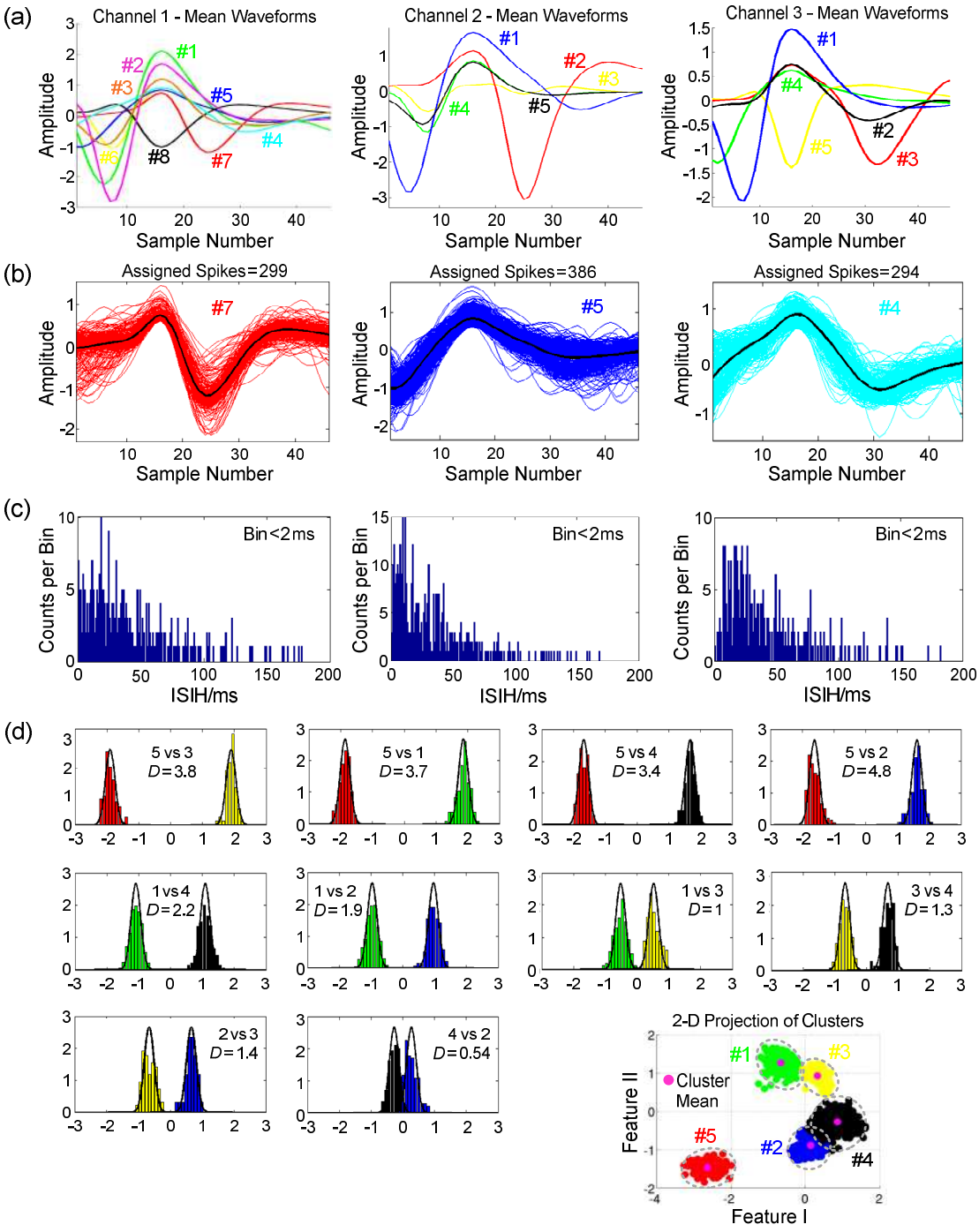

Figure 3.9: Sorting results of the recorded in-vivo neural data. (a) Illustration of found mean waveforms in channels 1-3. The amplitude is of arbitrary units.(b) Color-coded clusters with number of assigned spikes in each cluster form channel 1 (#7 red, #5 blue, #4 cyan). (c) ISIH of the neurons form channel 1. (d) Results of projection test using estimated probability density functions for all possible combinations of channel 2. 2-D projection of clusters is included for visual clarity. The vertical axis shows the distribution amplitude and the horizontal axis is the distance between the two distributions.

3.3.5 Complexity Analysis

In order to assess the hardware requirements of different methods the computation complexity metric was used. It is defined as [101], [118]:

$$\mathrm{ComputComp} = N_{\mathrm{add(sub)}} + 10N_{\mathrm{mult(div)}} \tag{3.6}$$

where N_{add} is the number of additions (or subtractions), and N_{mult} is the number of multiplications (or divisions) required. A 10-bit multiplier uses almost 10X the hardware resources of a 10-bit adder. Table 3.2 compares the proposed method and seven other published methods in terms of estimated computational complexity, clustering algorithm used, average classification accuracy (CA_{cc}) and number of features (dimesionality). Compared to DDs-MDT, DD$|_2$-Extrema has 3.6× lower complexity with 19.8% higher average accuracy. FDVSDV has 5.76× and 1.62× higher complexity compared withDD$|_2$-Extrema and DDs-MDT, respectively. It was reported in [110] that FDVSDV has 6.97% classification error with varying noise levels across all four synthetic (Section 3.3) datasets and $M(2N-3)$ complexity with the *k*-means classifier.

Since in FDVSDV extrema are selected using the attenuated projections of a spike shape, the O-Sort sorting threshold needs to be reduced to distinguish the clusters. The threshold level for FDVSDV was found using $T = S(\alpha\cdot\sigma)^2$ where ($\alpha = 0.25$) is a factor derived from simulation. There are disadvantages with the calculated threshold. They include: 1) the possible creation of an impractically small threshold value (e.g., 0.001) which is then very sensitive to noise variations, and 2) determining the sorting threshold involves an extra multiplication (evaluating T for

Table 3.2:
Computational complexity comparison of various feature extraction and dimensionality reduction methods.

Method	Feature Extraction Additions	Feature Extraction Multiplications	Dimensionality Reduction	Clustering Algorithm	*Average CA_{cc}	Number of Features (K)	ADC Simulator
DD$|_2$-Extrema	$M(2N-10)$	-	Max / Min / $V_{p\text{-}p}$ of DD$|_{2(\delta\,=\,3,7)}$	O-Sort	91.6%	4	✓
PCA	$M(N^2+1)$	$M(N^2+N)$	-	O-Sort	66.4%	3 first PCs	✓
FDVSDV	$M(2N-3)$	-	Max / Minof FDV and SDV	O-Sort	73.6%	4	✓
[a]DDs-MDT	$M(3N-11)$	-	MDT→$(M-1)(3N)$	O-Sort	71.8%	21	✓
[a]DDs-USAMP	$M(3N-11)$	-	Uniform Sampling	O-Sort	61%	21	✓
Spike Shape	-	-	-	O-Sort	68.4%	45	✓
[c]ZCF[75]	$M(N)$	-	-	*k*-means	>94.0%	2ZCFs = 3	×
**DWT	$M(4N)$	$M(8N-10)$	Kolmogorov-Smirnov	SPC	[b]92.6%	10	×

M = number of spikes; N = sample number per spike
*Averaged across 4 datasets with varying degrees of noise level
***DWT* (four-level Haar wavelet) and Kolmogorov-Smirnov used in Waveclus
a - DDs-MDT with scaling factors $\delta = 1, 3, 7$
b - The results in [89] are used for comparison
c – ZFC performance was evaluated with a different spike bank

recalculating the average standard deviation when using a sliding window). FDVSDV was implemented with O-Sort to provide a fair comparison with the method proposed in this paper. The average accuracy of FDVSDV is 73.6%.DDs-MDT has an average accuracy of 71.8% with O-Sort. Six of the eight methods listed in Table 3.2 use the O-Sort clustering algorithm. It should be mentioned that the average accuracy of O-Sort is around 70% while both *k*-means and fuzzy *c*-means have over 90% discrimination accuracy. However, the only online and unsupervised algorithm in the context of implantable sorting hardware is O-Sort. In addition, *k*-means and fuzzy *c*-means clustering require prior knowledge of the number of clusters.

To test the accuracy of the classifiers, original spike shapes were applied (without upsampling) to establish the accuracy with unprocessed data. Waveclus is very efficient with an acceptable classification error (7.4%). However, its computational complexity is much higher than that of either DD_2-Extrema or DDs-MDT. $DD|_2$-Extrema without offline training requires less than 5% of the computational complexity of Waveclus (the computational complexity of Kolmogorov-Smirnov is not considered). Although $DD|_2$-Extrema and Waveclus have comparable similarity immunity, there is a sustained improvement in $DD|_2$-Extrema performance with increasing noise level (noise immunity).

The overall sorting complexity consists of creating clusters and merging phases. It is defined as:

$$SortComp = \underset{Add}{MK\ (3C-3) + K(3C-3)} + \underset{Mult}{20\ MKC} \tag{3.7}$$

where *M* is the number of feature vectors, *K* is the number of features representing each spike in feature space, and *C* is the number of clusters. The Euclidean distance calculation requires the multiplication operation in Eq.(3.7) which leads to high sorting complexity. $\ell 1$-norm distance metric makes O-Sort a particularly good choice for hardware implementation since it is much less dependent on the dimensionality of feature space. The $\ell 1$-norm distance is less susceptible to biological noise than the Euclidean distance [76], hence resulting in a better average CA_{CC}. Figure 3.10 shows the classification error versus computational complexity of feature extraction and sorting. The DD2-Exrema has better tradeoff between classification error and complexity. Figure 3.11 displays the average classification error versus dimensionality factor for various methods. The dimensionality factor is the ratio of feature space dimensions to the number of samples per spike. As can be seen, there are significant differences between $DD|_2$-Extrema (K = 4) and

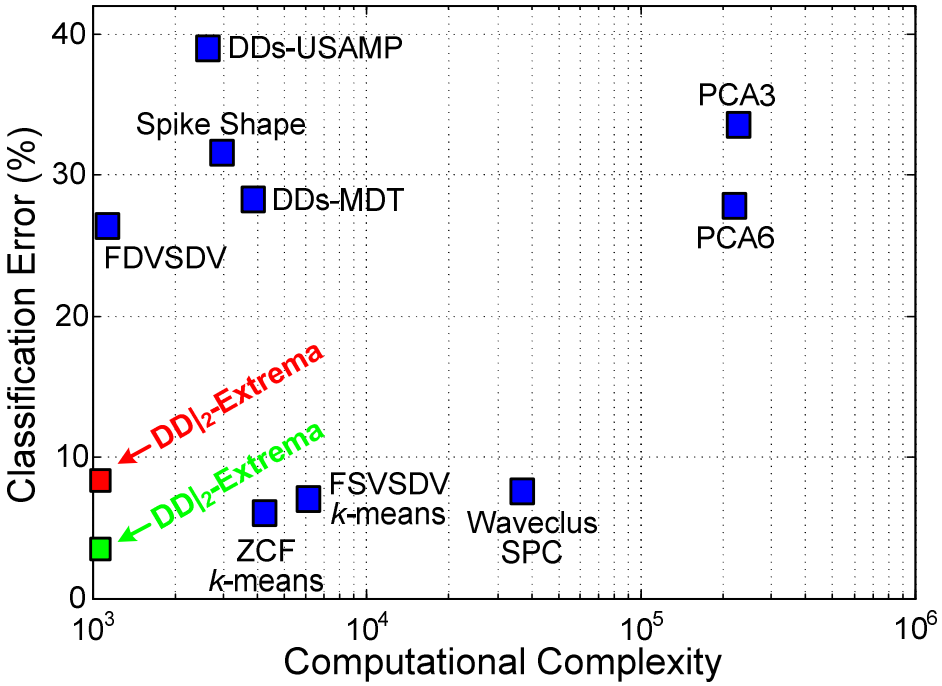

Figure 3.10: Classification error versus computational complexity for the different feature extraction and sorting methods considered herein. The red square (DD|$_2$-Extrema) shows the average accuracy amongst all datasets and all noise levels. The green square (DD|$_2$-Extrema) shows the average accuracy amongst all datasets using noise with a standard deviation of 0.05.

FDVSDV (K = 4), Spike Shape (K = 45), DDs-MDT (K = 21), DDs-USAMP (K = 21) and PCA (K = 3). The competing methods for DD|$_2$-Extrema are ZCF [75] (2ZCFs = 3) and Waveclus (K = 10) with 2.4% and 1% lower classification error, respectively. It can be clearly observed that DD|$_2$-Extrema outperforms all the other methods and provides the best tradeoff in complexity, accuracy and dimensionality.

3.3.6 Proposed Data Reduction Application Example

As noted in Section 3.2, one of targeted applications is the development of a custom integrated circuit for an implantable multi-electrode neural interface for upper-limb prostheses. The chip will amplify and reduce the data rate needed to represent the many spike signals. Each interface will typically have 20+ channels (microelectrodes) and there are four nerves in the upper arm which carry most of the motor axons. Prior to spike sorting the recorded data will be digitized by an analog-to-digital converter (ADC, see Figure 3.1). Assuming each channel is sampled at 30 kHz with 7-bit ADC resolution, the average data rate at the input of the digital spike sorting processor will be (4×20×30 kHz×7bits) 16.8 Mb/s (Figure 3.12). A typical neuron generates on average 40

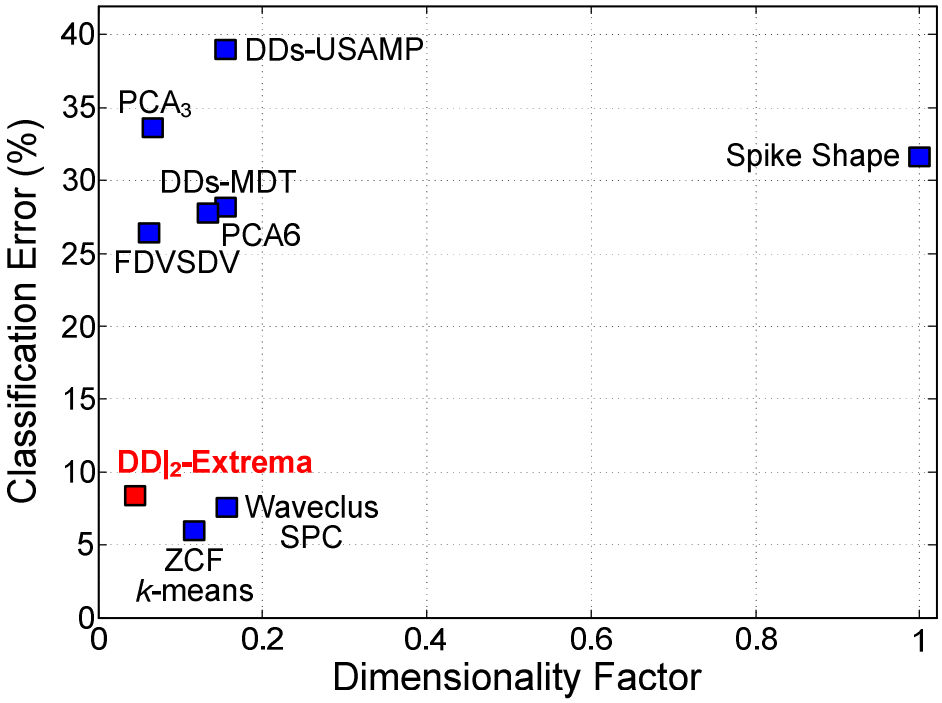

Figure 3.11: Classification error versus dimensionality factor for the different feature extraction methods.

spikes per second when active [119], and up to 25 active neurons are estimated in each channel. Therefore, after detection and alignment the data rate will become 1.26 Mb/s (100 neurons × 40 spikes/(neuron.second) × 45 samples/spike × 7 bits/sample). Using DD$|_2$-Extrema to encode the spikes into four features (28bits/spike) the extracted coefficients will yield a data rate of 112 kb/s or 0.007% of the original data rate. In the final step typically the cluster number and channel number will be processed via O-Sort, which will yield a final transmission data rate of 56 kb/s (i.e., 0.004% of the original data rate). Such a low data rate is feasible for wireless transmission using a single pair of coils for both power and data [120]. Reducing the data rate to only 0.004% of the original data rate should also be attractive for high channel count recording front-ends for other applications [121], [122].

3.4 Conclusion

This chapter has proposed and investigated a new feature extraction method based on spike waveforms and their discrete derivatives. Nine combinations of extrema features have been examined with the proposed DD$|_2$-Extrema method offering the highest average classification accuracy. Specifically, DD$|_2$-Extrema with $M(2N-10)$ complexity and dimensionality factor of 4,

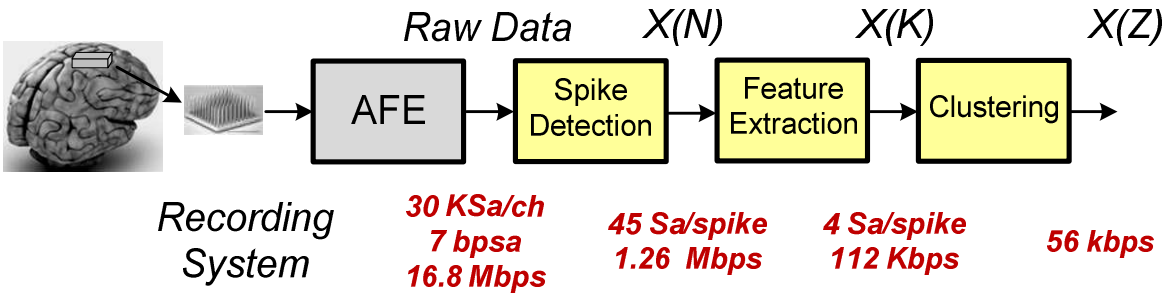

Figure 3.12: The spike-sorting process, annotated with an example of typical data rates. The data rate at the end of spike sorting is lower than that of the raw data. (Assumptions are annotated on the figure in red italics)

achieves 91.6% average classification accuracy. It requires about 1% of the computational complexity of PCA while providing higher accuracy. The combination of DD$|_2$-Extremawith O-Sort provides a considerable accuracy-complexity tradeoff. This should allow on-chip processing without any assumption of high level of fidelity for sorting multi-unit activity. The results confirm that the average classification error is less than 4% across all the datasets tested using noise with a standard deviation of 0.05.This theoretical limit can be used to determine the design parameters of the analog front-end including data converter resolution and sampling rate, filter type, bandwidth and order, and amplifier noise, bandwidth and gain. The overall complexity of spike sorting has been optimized using the ℓ1-norm distance calculation [76]. The clustering performance of the proposed method has been evaluated using both synthetic and recorded in-vivo neural data. DD$|_2$-Extrema outperforms various online and offline algorithms which have significantly more complexity. In addition, it offers a better trade-off between complexity, accuracy and dimensionality than all the other methods considered herein. DD$|_2$-Extrema could be used as an integral part of a future neural amplifying and spike sorting chip for a range of neural prostheses applications such as prosthetic hand (see Figure 3.1), cortical neural recording [123], and bladder control after spinal cord injury [124].

CHAPTER 4

Design Techniques and Power Limit Analysis of Neural Front-end Interfaces Targeting Ultralow Power Implantable Devices

4.1 Introduction

Neural front-end interface (NFI) is a critical pre-processing stage in all neural recording systems. The quality of neural data monitoring is directly affected by the NFI. As discussed in the introduction (Figure 1.1), the trend in signal acquisition is towards a large number of recording channels (e.g., 1K channels). Conventional design has high front-end specifications which result in power and area utilization in hardware which is not compatible with high channel count recording. A design framework is required to achieve sufficiently low power and area utilization. With reference to Figure 4.1 this chapter has the following aims:

1- Development of a NFI power optimization framework suitable for high channel count (1K+) recording. By optimizing each channel in terms of power and area, there is a capability for 1K+ recording and processing for future generation of implantable BMIs.
2- An optimization procedure which results in a power efficient NFI, allowing provision of more functions such as spike sorting for a given amount of power consumption per channel, and reducing the energy cost associated with the transmitter (e.g., nJ/bit).

The optimization framework [125] is illustrated in Figure 4.2. The power consumption of the NFI is assessed by examination of the constituent building blocks comprising a low noise amplifier (LNA), a programmable gain amplifier (PGA) and an analog-to-digital converter (ADC).

The minimum power required for the NFI is theoretically derived based on its constituent building blocks. In addition to deriving the NFI power bound, it is shown that the noise-power contribution can further minimize the NFI power bound by appropriate selection of data conversion design such as comparator-based switched-capacitor (CBSC) [12], [126-127]. Following the NFI power dissipation bound analysis a parametric optimization methodology is developed to define the optimal values of NFI practical parameters including signal bandwidth,

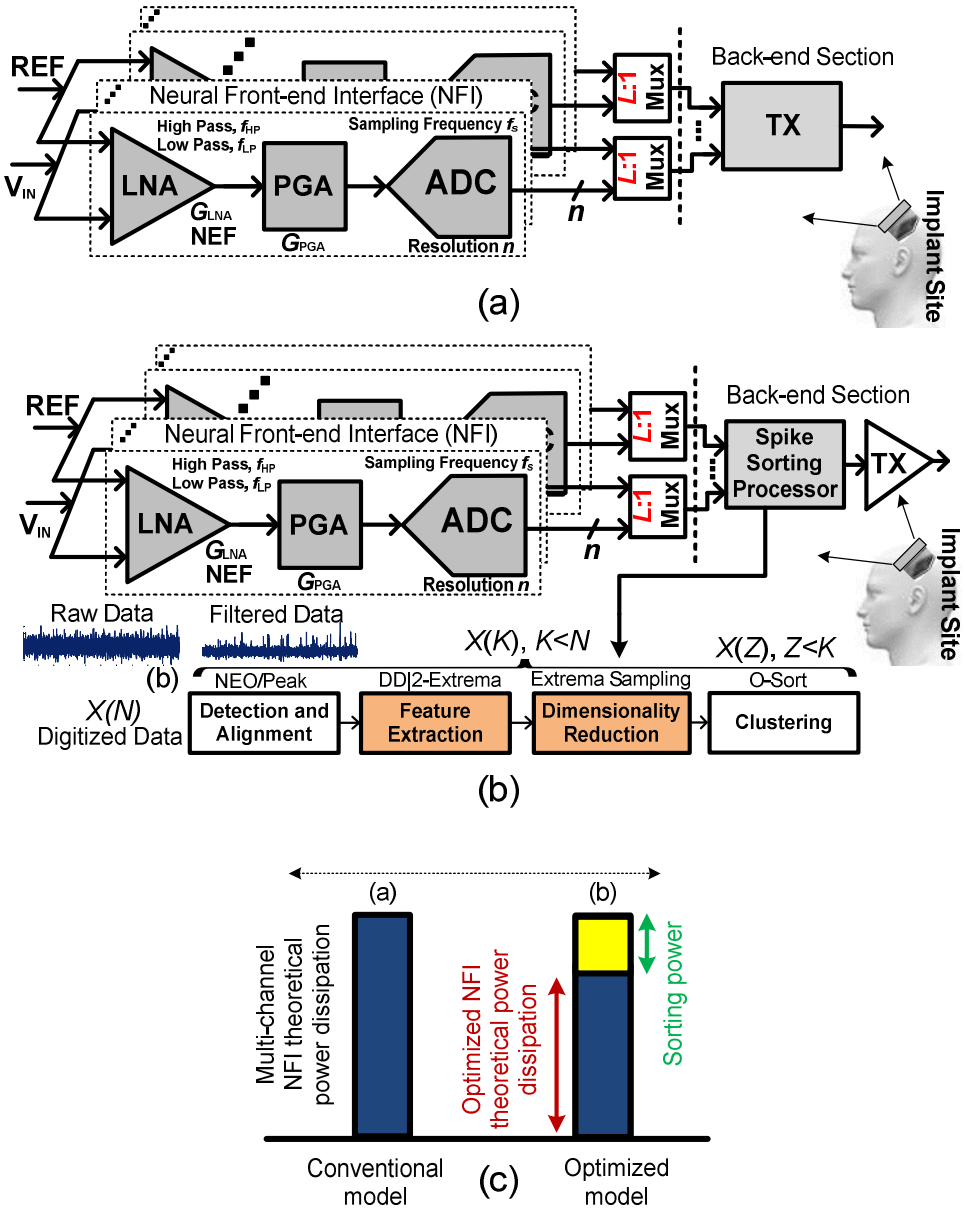

Figure 4.1: Multi-channel signal monitoring with (a) conventional recording and (b) the spike sorting chain is included in the neural interface. (c) The power reduction due to the proposed power optimization framework allows the integration of spike sorting into the implant for the same power consumption as in the traditional model. The amount of saved power budget for spike processor integration is 30% which is discussed in section 4.4.3.

sampling rate and resolution. The underlying suggestion is that the current NFI state-of-the-art designs are not optimally engineered. This optimization tool is developed by an FPGA-MATLAB interface and allows designers to assess spike sorting robustness to variations in the NFI practical parameters. The NFI behavioral model is simulated in MATLAB, which provides a flexible way

to investigate the effect of different topologies on neural data. A spike sorting processor was implemented on an FPGA to complete the assessment loop. This FPGA-MATLAB is a simple realization of the whole processing chain to achieve a good balance between resource efficiency and outright spike sorting performance. The NFI optimum parameter selection procedure is applied to standard datasets with different signal-to-noise ratios (SNRs) and various spike similarity levels. The analysis tests the robustness of spike processing accuracy.

It is shown that for a resolution of 7 bits (optimum for spike sorting) using a comparator-based switched-capacitor (CBSC) cyclic ADC reduces power consumption by approximately 30% compared to a successive approximation register (SAR) ADC.

The design steps are:

1. *Estimation of the NFI Theoretical Minimum Power:* The theoretical minimum power of the NFI is derived by considering its individual building blocks. This limit provides a minimum design target.
2. *Choice of ADC:* The commonly used SAR ADC is compared to the CBSC cyclic ADC. Other data converter architectures such as zero-crossing-based [128] and pulsed bucket brigade [129] are possible alternatives.
3. *Selection of NFI Key Parameters:* These include LNA bandwidth, ADC resolution and sampling rate, specified by exploring their effect on the accuracy of the spike sorting processor. An online, unsupervised spike sorting processor was implemented on an FPGA to verify the extracted NFI parameters. The derived parameters take into account the accuracy requirements for spike sorting and are compared with a number of NFIs in the literature.

To validate the study, the derived minimum power limits are compared with published designs obtained from the state of the art. Optimizing the NFI design prevents over-engineering and significantly reduces the power cost of recording to transmission.

The rest of the Chapter is organized as follows. In Section 4.2 the power models of the LNA, PGA and ADC (SAR and CBSC cyclic) are presented. In Section 4.3 an online FPGA implementation of the spike sorting processor is presented. In Section 4.4 the design optimization of the NFI is examined based on spike sorting requirements. By considering the ADC and proper selection of

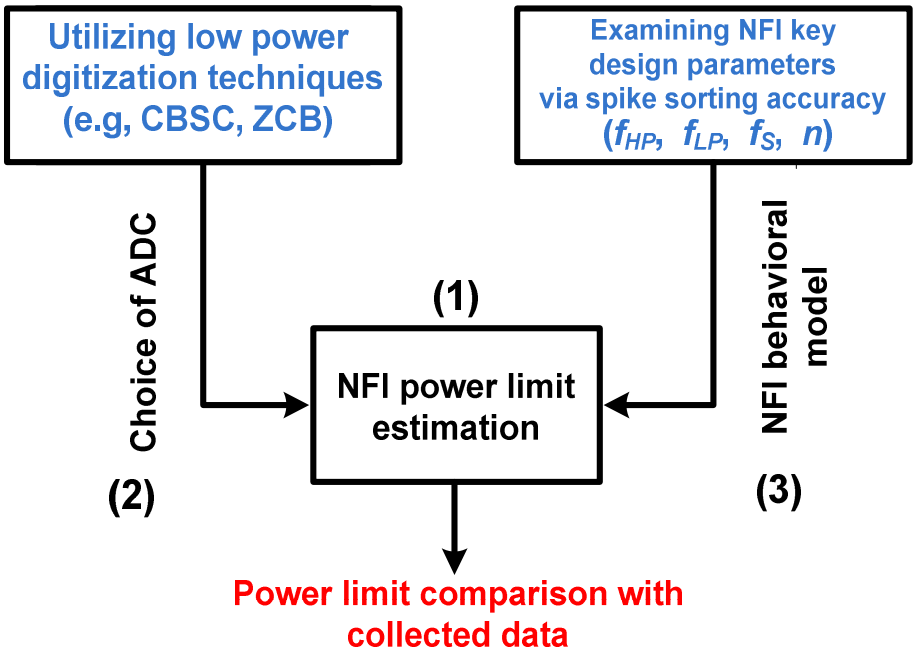

Figure 4.2: The optimization framework.

NFI parameters, NFI power bounds using two types of ADC are derived and compared. A parametric NFI design is also briefly discussed in this section. Conclusions are presented in Section 4.5.

4.2 NFI Power Model

The general architecture of a typical implantable bio-potential recording system is shown in Figure 4.1. The NFI comprises: a) LNA, b) PGA, and c) ADC. The LNA is used to record the neuronal activity (the amplitude of the extracellular potentials is in the range of 50 µV to 1 mV[4]) and in a specific frequency range (between 300Hz and 10kHz). Thus the recorded signals must be amplified and filtered before being processed. It also removes any dc level generated at the recording site across the electrode-tissue interface. The PGA provides further (programmable) amplification. It typically has a bandpass response to eliminate unwanted signal components (e.g., local field potentials and high frequency noise), to prevent aliasing and to provide offset adjustment. The ADC performs digitization for the subsequent processing operations (e.g., spike

[4]This range is based on various neural interface modalities with different levels of spatio-temporal resolution and invasiveness as discussed in Chapter 2.

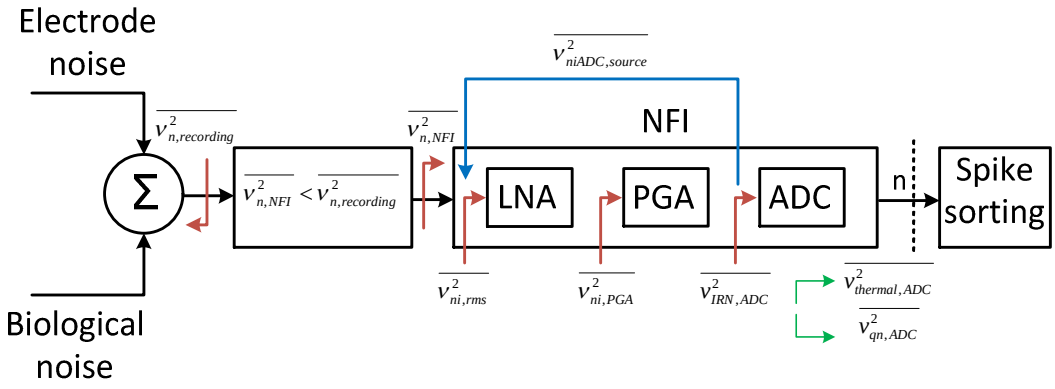

Figure 4.3: Illustration of noise sources in the NFI. The definition of each noise source is: $\overline{v^2_{n,recording}}$: neural recorded signal noise power, $\overline{v^2_{n,NFI}}$: Overall input referred NFI noise power, $\overline{v^2_{ni,rms}}$: LNA Input referred noise power, $\left(\overline{v^2_{ni,PGA}} \approx 0\right)$: PGA Input referred noise power, and $\overline{v^2_{IRN,ADC}}$: ADC Input referred noise power which is composed of thermal noise power ($\overline{v^2_{thermal,ADC}}$) and quantization noise power ($\overline{v^2_{qn,ADC}}$). $\overline{v^2_{niADC,source}}$: the LNA input referred thermal noise power of the ADC.

sorting). Since the focus of the next two sections concerns noise-power analysis, the noise sources from different blocks of NFI are illustrated and defined in Figure 4.3. The input referred noise of the PGA is assumed to be zero for simplicity in calculations when demonstrating the contribution of the data converter thermal noise on the NFI power limit.

4.2.1 LNA Power Analysis

A figure of merit for LNA design is introduced and discussed in [130]. The minimum power consumption of the LNA is dictated by the input referred noise voltage ($v_{ni,rms}$). It should be noted that the *1/f* noise is not the dominant noise source in the circuit due to large gate areas of the pMOS input devices and contributes about 0.1 µV/Hz (0.5 Hz to 50 KHz [131]) to the total noise voltage ($v_{ni,rms}$). Hence the power bound of the LNA is based only on thermal noise considerations. The noise efficiency factor (NEF) introduced in [130] identifies the trade-off between $v_{ni,rms}$ (integrated over bandwidth) and LNA power consumption:

$$NEF = v_{ni,rms} \cdot \sqrt{\frac{2\, I_{LNA}}{\pi . U_T . 4kT . \left(BW_{LNA} = f_{LP} - f_{HP}\right)}} \tag{4.1}$$

where I_{LNA} is the total LNA supply current, BW_{LNA} is the 3-dB bandwidth of the LNA and U_T is

the thermal voltage (kT/q), k is Boltzmann's constant and T is temperature in Kelvin. An LNA with programmable bandwidth is included in Eq.(4.1). The required LNA power consumption P_{LNA} is:

$$\begin{aligned} P_{LNA} &= V_{DD} \cdot I_{LNA} \\ &= V_{DD} \cdot \frac{(NEF)^2}{\overline{v_{ni,rms}^2}} \cdot \frac{\pi . U_T . 4kT . BW_{LNA}}{2} \end{aligned} \tag{4.2}$$

where V_{DD} is the power supply. The noise constraint on the LNA is influenced by the quantization noise of the ADC. The total amount of noise at the input of data converter should be less than the quantization noise power ($\overline{v_{qn,ADC}^2}$):

$$G_{LNA}{}^2 . G_{PGA}{}^2 . \overline{v_{ni,rms}^2} + \overline{v_{thermal,ADC}^2} \le \overline{v_{qn,ADC}^2} \tag{4.3}$$

G_{LNA} and G_{PGA} are the gains of the LNA and PGA, respectively, and $\overline{v_{thermal,ADC}^2}$ is the ADC thermal input referred noise power. $\overline{v_{thermal,ADC}^2}$ can be expressed as a percentage of $\overline{v_{ni,rms}^2}$, hence Eq.(4.3) is rewritten as below:

$$G_{LNA}{}^2 . G_{PGA}{}^2 . \overline{v_{ni,rms}^2} + G_{LNA}{}^2 . G_{PGA}{}^2 . \left(\alpha . \overline{v_{ni,rms}^2} \right) \le \overline{v_{qn,ADC}^2} \tag{4.4}$$

$$G_{LNA}{}^2 . G_{PGA}{}^2 . \overline{v_{ni,rms}^2} . (1+\alpha) \le \overline{v_{qn,ADC}^2} \tag{4.5}$$

$$G_{LNA}{}^2 . G_{PGA}{}^2 . \overline{v_{ni,rms}^2} \le \frac{\overline{v_{qn,ADC}^2}}{(1+\alpha)} \tag{4.6}$$

$\alpha = \dfrac{\overline{v_{thermal,ADC}^2}}{G_{LNA}{}^2 . G_{PGA}{}^2 . \overline{v_{ni,rms}^2}}$ is a noise multiplying factor (NMF) (its selection is discussed in Section 4.2.2). It is calculated by dividing the ADC thermal noise power contribution by the LNA noise power referred to the input of the ADC. $\overline{v_{qn,ADC}^2}$ is replaced by $V_{FS}^2 / 12.2^{2n}$:

$$G_{LNA}{}^2 . G_{PGA}{}^2 . \overline{v_{ni,rms}^2} \le \frac{V_{FS}^2}{(1+\alpha) . 12.2^{2n}} \tag{4.7}$$

where V_{FS} (e.g., 1V) is the full scale voltage considered for quantization and n is the ADC

resolution. α is set to ≥0.1 when the calculated value $\overline{v_{thermal,ADC}^2}$ is noticeable. This means that $\overline{v_{ni,rms}^2}$ must be designed to be extremely low in the LNA to compensate for the effect of α. When α is set to zero, the LNA design can be more relaxed which means no additional power is required to nullify the effect of α in Eq.(4.6). The simplified models of Eq. (4.7) for α≥0.1 and α=0 are:

$$G_{LNA}^2.G_{PGA}^2.\overline{v_{ni,rms}^2} \le \begin{cases} \dfrac{V_{FS}^2}{(1+\alpha).12.2^{2n}} & e.g.SAR \quad \alpha \ge 0.1 \\ \left(V_{FS}^2/12.2^{2n}\right) & e.g.CBSC\ \alpha = 0 \end{cases} \tag{4.8}$$

When the $\overline{v_{thermal,ADC}^2}$ is low (CBSC α=0), the LNA power consumption is dependent on the ADC quantization noise which is defined by V_{FS} and the converter resolution n. On the other hand, for α≥0.1, the LNA requires more rigorous requirements in terms of power consumption to reduce the LNA input-referred noise. Hence, the minimum LNA power consumption is:

$$P_{LNA} = G_{LNA}^2.G_{PGA}^2(NEF)^2.\frac{2.\pi(kT)^2.BW_{LNA}}{q}.\begin{cases} \dfrac{(1+\alpha).12.2^{2n}}{V_{FS}^2} & e.g.SAR \quad \alpha > 0.1 \\ \dfrac{12.2^{2n}}{V_{FS}^2} & e.g.CBSC\ \alpha = 0 \end{cases} \tag{4.9}$$

4.2.2 Assigning Noise Multiplying Factor (NMF= α) through Data Converter Noise Analysis

This section provides a study of the input referred-noise in the CBSC gain stage and SAR data converter. For the purpose of the CBSC gain stage noise analysis, a single pipeline stage (1.5-bit/stage) is considered. As shown in Figure 4.4 the input referred noise (IRN) of a CBSC gain stage consists of a contribution from the threshold detection comparator (TDC), switches (S_S, S_L) over their operation time, and charging capacitance (C_L). The overall thermal noise of a CBSC gain stage is:

$$\overline{v_{thermal,CBSC}^2} = \overline{v_{TDC,in}^2} + \overline{v_{switches,in}^2} + \overline{v_{in,C_L}^2} \tag{4.10}$$

where $\overline{v_{TDC,in}^2}$ is the TDC noise power, $\overline{v_{switches,in}^2}$ is the switches noise power and $\overline{v_{in,C_L}^2}$ is the capacitive load noise contribution. An example comparator (TDC) with added noise current sources is shown in Figure 4.5 [132] (TDC operation is discussed in section 4.2.4.2.2). The noise

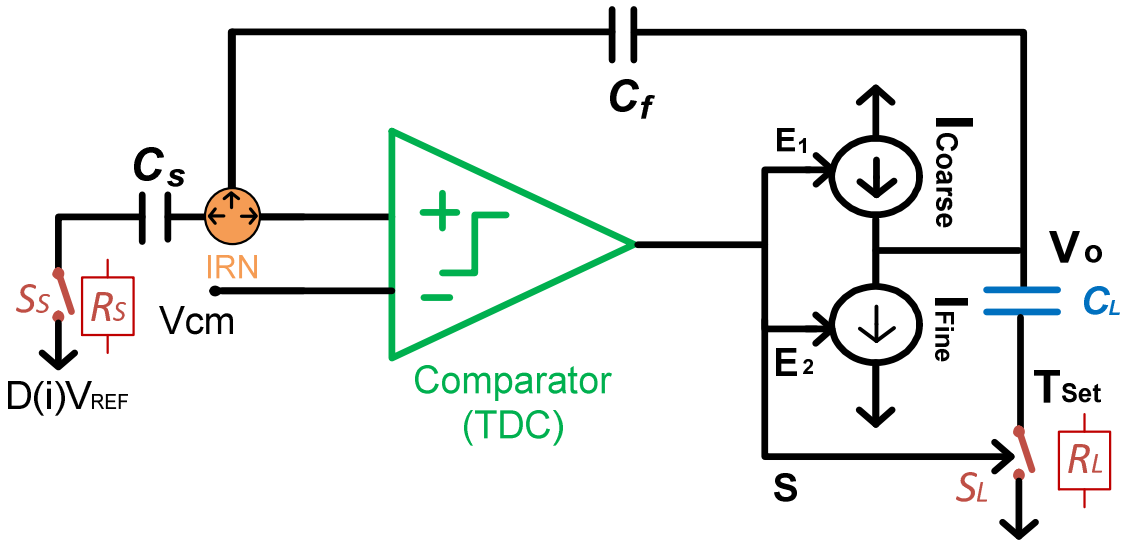

Figure 4.4: CBSC gain stage noise analysis. The effect of noise contribution from different elements are depicted via green (TDC noise), blue (load sampling capacitance C_L) and red (switches R_S and R_L). The overall input referred noise of gain stage is represented via (IRN). D(i)V_{REF} is the output of analog multiplexer in 1.5-bit/stage. Vcm is the common mode voltage.

current generated in M1 is mirrored in M4 (also M3). The total current noise at the output node of TDC is the product of the power spectral density and equivalent noise bandwidth is:

$$\overline{v_{T,out}^2} = \left(\overline{i_{M1,n}^2} + \overline{i_{M2,n}^2} + \overline{i_{M3,n}^2} + \overline{i_{M4,n}^2}\right).\left(r_{o2} \parallel r_{o4}\right)^2.\frac{1}{2t_{d,TDC}}$$
$$\left(16KT\,\gamma\, g_m\right).\left(\overbrace{r_{o2} \parallel r_{o4}}^{R}\right)^2.\frac{1}{2t_{d,TDC}} = \left(16KT\,\gamma\right).\left(\overbrace{R.g_m}^{A_{TDC}}\right).\frac{R}{2t_{d,TDC}} \tag{4.11}$$

where $\overline{i_{Mi,n}^2}$ is the current noise introduced by the i^{th} transistor, (R=r_{o2}∥r_{o4}) is the impedance observed at the output node and $1/2t_{d,TDC}$ is the CBSC noise bandwidth [133] where $t_{d,TDC}$ is the delay of comparator. The input-referred voltage noise is given by:

$$\overline{v_{TDC,in}^2} = \frac{\left(16KT\,\gamma\right).\left(\overbrace{R.g_m}^{A_{TDC}}\right).\frac{R}{2t_{d,TDC}}}{{A_{TDC}}^2} = \frac{8KT\gamma.R}{A_{TDC}.t_{d,TDC}} = \frac{8KT\gamma}{g_m.t_{d,TDC}} \tag{4.12}$$

where A_{TDC} is the total gain of the TDC. The second noise source in the CBSC gain stage is the contribution from the switches (S_S, S_L), shown in Figure 4.4. During the charge transfer phase, S_L

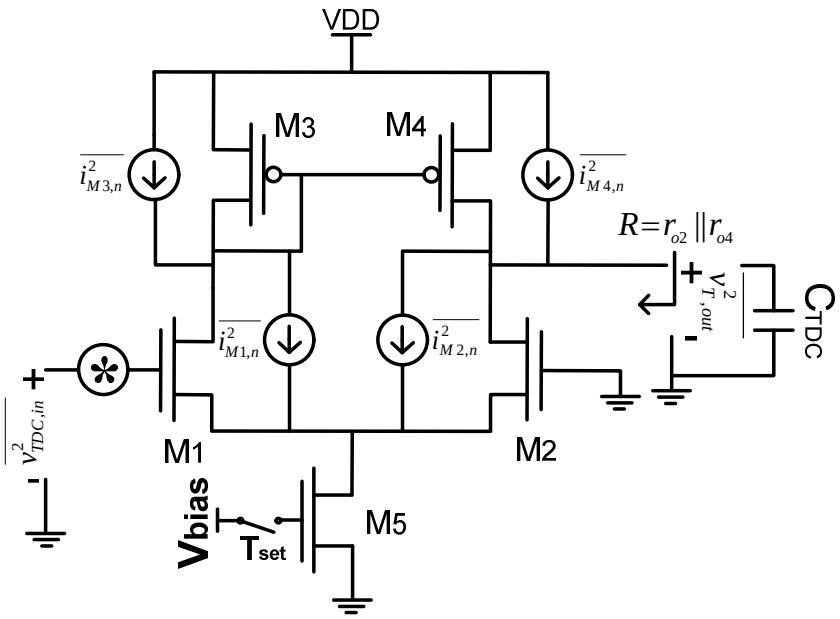

Figure 4.5: Comparator with noise sources. TDC is active when T_{Set} is on.

is on and in series with C_L and S_S is on and in series with C_S. Therefore the input-referred voltage noise due to the switches is expressed as:

$$\overline{v^2_{switches,in}} = 4KTR_{tot}.\frac{1}{2t_{d,TDC}} \tag{4.13}$$

where R_{tot} is the total resistance which is seen at the input node of the TDC (IRN node) which is equal to:

$$R_{tot} = R_L\left(\frac{C_{fL}}{C_{fL}+C_S}\right)^2 + R_S\left(\frac{C_S}{C_{fL}+C_S}\right)^2 \tag{4.14}$$

where C_{fL} is the series combination of the C_f and C_L. It should be noted that the switch noise is filtered via the noise bandwidth of TDC ($1/2t_{d,TDC}$).

The last noise source is the contribution of the noise sampled onto the load capacitance from the capacitor network. The total capacitance value at the output node is defined as:

$$C_{tot} = \frac{C_{series}.C_L}{C_{series}+C_L} \tag{4.15}$$

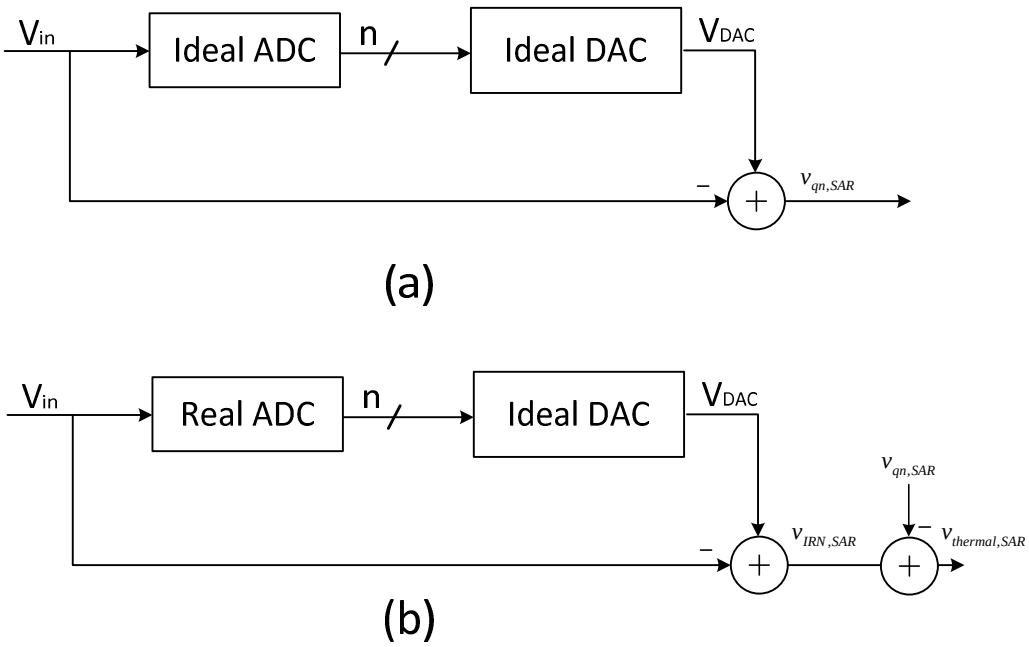

Figure 4.6: Model for ideal quantization noise and (b) Proposed model for thermal noise voltage in an SAR ADC.

where C_{series} is the series combination of the C_f and C_s. Therefore, the noise value of C_L is:

$$\overline{v_{C_L}^2} = \frac{KT}{C_{tot}}\left(\frac{C_{series}}{C_{series}+C_L}\right)^2 = \frac{KT}{C_L}\left(\frac{C_{series}}{C_{series}+C_L}\right) \tag{4.16}$$

The input referred noise of C_L is calculated by dividing it by the gain value ($G_{1.5\text{-bit}}$) of the 1.5bit gain stage ($C_S = C_f$):

$$\overline{v_{in,C_L}^2} = \frac{KT}{C_L}\left(\frac{C_{series}}{C_{series}+C_L}\right).\left(\frac{C_f}{C_f+C_S}=\frac{1}{2}\right)^2 \tag{4.17}$$

The nominal thermal input referred noise value is around <2 (differential architecture)×25μV_{rms} for f_s = 30kHz. In [134], detailed statistical analysis for calculating IRN of three types of SAR ADCs is proposed. The popular model for ideal quantization noise power ($\overline{v_{qn,SAR}^2}$) calculation is shown in Figure 4.6(a). In Figure 4.6(b), the difference between the output of digital to analog (DAC) converter V_{DAC} and V_{in}, $\overline{v_{IRN,SAR}^2}$, is then a measure of the combined effect of quantization

and thermal noise in the noisy system. When the ideal quantization noise ($\overline{v_{qn,SAR}^2}$) is subtracted from $\overline{v_{IRN,SAR}^2}$, thermal noise power $\overline{v_{thermal,\ SAR}^2}$ is derived. The input-referred noise of a SAR ADC $\overline{v_{IRN,SAR}^2}$ in conversion phase is:

$$\overline{v_{IRN,SAR}^2} = \overline{v_{thermal,\ SAR}^2} + \overline{v_{qn,\ SAR}^2} \tag{4.18}$$

where $\overline{v_{thermal,\ SAR}^2}$ and $\overline{v_{qn,\ SAR}^2}$ are thermal and quantization power terms. The typical value for $v_{thermal,SAR}$ is in the range of 0.7–2.2mV. This value can be justified through different constituent building blocks noise contributions in the bit-conversion phase (switching noise is not included). In the bit-resolve phase, the bits are resolved one-by-one by comparator and capacitor arrays. The thermal noise contribution of capacitor arrays is equal to kT/C_{tot} where C_{tot} is the total capacitance of the capacitor array (64μV for a C_{tot} =1 pF). The second unit, the digitization phase, is a dynamic comparator which intrinsically has 10X higher noise ($v_{n,comp}$ > 0.5–2mV ≈ $v_{thermal,SAR}$) [134-137]. This implies that the dynamic comparator noise power term ($\overline{v_{n,comp}^2}$), dominates by a factor of 100X over other terms. The noise study in this section, a CBSC 1.5-bit/stage[5] architecture suggests that it is more suitable for biomedical applications due to its noise behavior. In order to provide a simplified model for power analysis, noise multiplying factor (NMF≈ α), is added in the LNA power model to show the effect of converter input referred thermal noise voltage on this block. The next two points develop the power-noise analysis in this section:

(NMF= α) selection: To calculate α, first the ADC IRN is reflected to the input of analog section. It is assumed that an ideal LNA is used in designing of NFI, so the ADC IRN can be modeled as an independent noise source ($\overline{v_{niADC,source}^2}$) in the input of analog section:

$$\overline{v_{niADC,source}^2} \le \frac{\overline{v_{thermal,\ ADC}^2}}{G_{LNA}^2 . G_{PGA}^2} \tag{4.19}$$

[5] More efficient noise behavior is expected from ZCB gain stage [128].

α is the defined as $\alpha = \frac{\overline{v_{niADC,source}^2}}{\overline{v_{ni,rms}^2}} = \frac{\overline{v_{thermal,ADC}^2}}{G_{LNA}^2.G_{PGA}^2.\overline{v_{ni,rms}^2}}$. So the value of α depends on input referred noise power of LNA $\overline{v_{ni,rms}^2}$. For example, in an NFI with ($v_{ni,rms}$ [6]=4μV_{rms}), employing SAR ADC for digitization, α is in the range of (0.175-0.55). The NFI systems employing SAR ADC demand more stringent requirement for noise suppression in LNA compared with the CBSC architecture where $\alpha \approx 0$ due to the fact that the nominal input referred noise value of a CBSC 1.5-bit/stage is much lower compared to SAR data converter.

IRN of analog section: the NFI IRN ($\overline{v_{n,NFI}^2} = \overline{v_{ni,rms}^2} + \overline{v_{niADC,source}^2}$) should be lower than the background noise of the neural recorded signal ($\overline{v_{n,recording}^2}$) (see Figure 4.7). In addition to this limit, the overall amplified noise of NFI at the input of the ADC should not be higher than the quantization noise to avoid any information loss in the digitization process. The limits are:

$$\begin{cases} \left(\overline{v_{n,NFI}^2} < \overline{v_{n,recording}^2}\right) & A \\ \overline{v_{n,NFI}^2} \le \dfrac{\overline{v_{qn,ADC}^2}}{G_{LNA}^2.G_{PGA}^2} & B \end{cases} \tag{4.20}$$

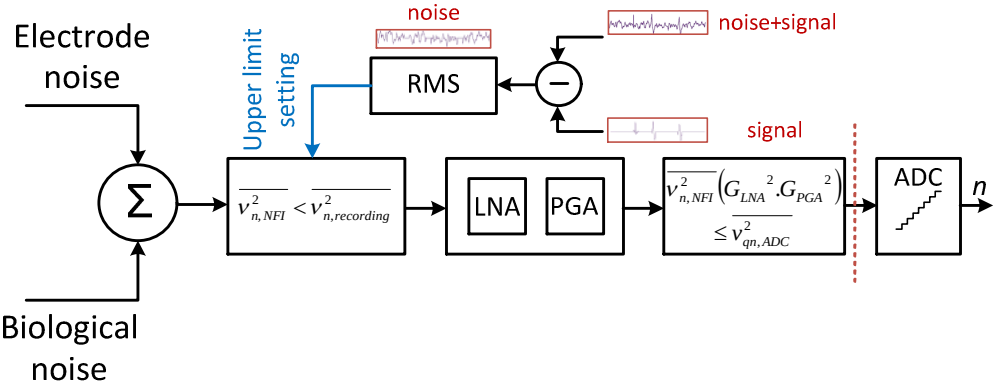

Figure 4.7: illustration of conditions in Eq.(4.21).

[6] This value can be calculated based on different circuit architectures such as [131].

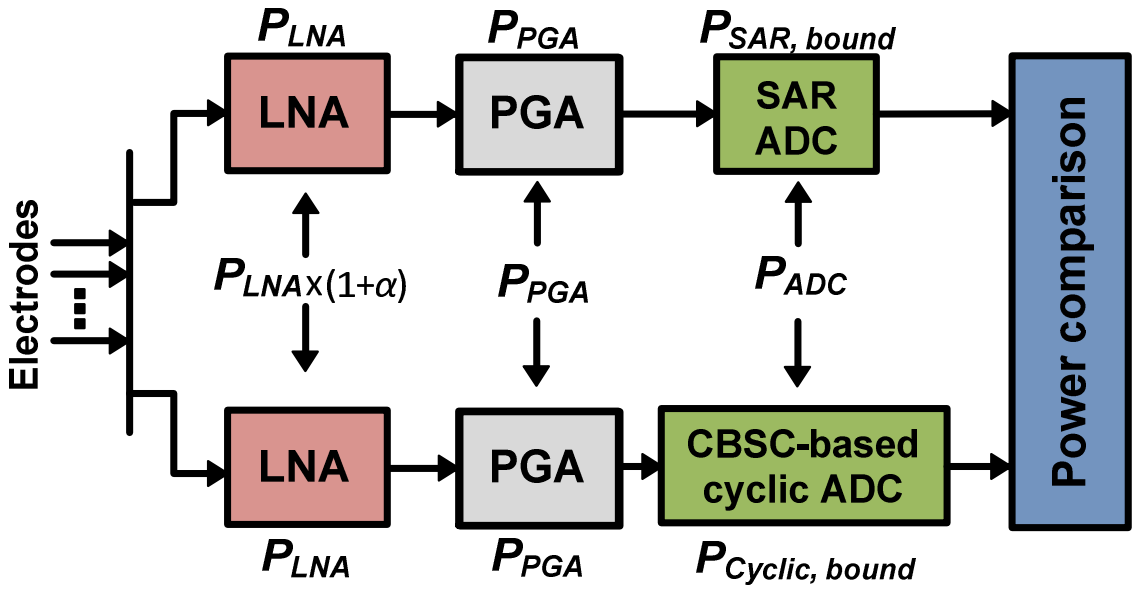

Figure 4.8: Block diagram for NFI power evaluation using different converters. The overall power consumption is expressed as $P_{NFI}=(P_{LNA}).(1+\alpha)+(P_{PGA})+(P_{ADC})$.

$$A \& B \longrightarrow C \quad \frac{\overline{v_{qn,ADC}^2}}{G_{LNA}^2.G_{PGA}^2} < \overline{v_{n,recording}^2} \tag{4.21}$$

The $V_{n,recording}$ of a LNA can be set at a reasonable upper limit which is 10μV_{rms} (it satisfies Eq.(4.21)). This limit is estimated by extracting the spikes from neural data and subtracting them from the original waveform. The remaining signal is pure noise and the noise root-mean square (RMS) of this signal is an accurate indication of noise level prior to LNA.

The power bound evaluation procedure of NFIs employing different types of data converters is in Figure 4.8.

4.2.3 Programmable Gain Amplifier (PGA)

Assume that there is a sample-and-hold (S/H) as part of a data converter which operates at Nyquist rate $2BW_{LNA}$ where BW_{LNA} is the LNA bandwidth. Further assume that the LNA has one dominant pole and the output of the LNA goes through a first linear settling. As illustrated in Figure 4.9, S/H tracking sampling time ($T_{sampling}$) and hold time (T_{hold}). The sum of the S/H processing time ($T_{S/H}=T_{sampling+Thold}$) must be less than the conversion time ($T_{S/H}<T_{ADC}$). In a conventional single amplifier with a single dominant pole, it is takes some time to settle to an acceptable residual error

As a result the time left for data conversion is significantly reduced and the conversion speed must operate at a higher speed. In order to deal with the residual issue, a low power system is proposed in [138] that differs from the conventional approach. In addition to the LNA, the system consists of a PGA that drives the S/H. The benefits of this system are:

- The optimum power can be obtained via proper design of PGA and ADC parameters.
- Fine tuning of the gain is available.

The PGA drives the ADC and must meet a slew rate constraint. Typically, the time constant of the PGA is:

$$\tau_{PGA} = R_{PGA} C_{L,PGA} = \frac{G_{PGA}}{g_{m,PGA}} C_{L,PGA} \tag{4.22}$$

where R_{PGA} is PGA output resistance, $C_{L,PGA}$ is the load capacitance seen at PGA output node and $g_{m,PGA}$ is PGA transconductance. The required bias current during the slew time (T_{Slew}) in the PGA is:

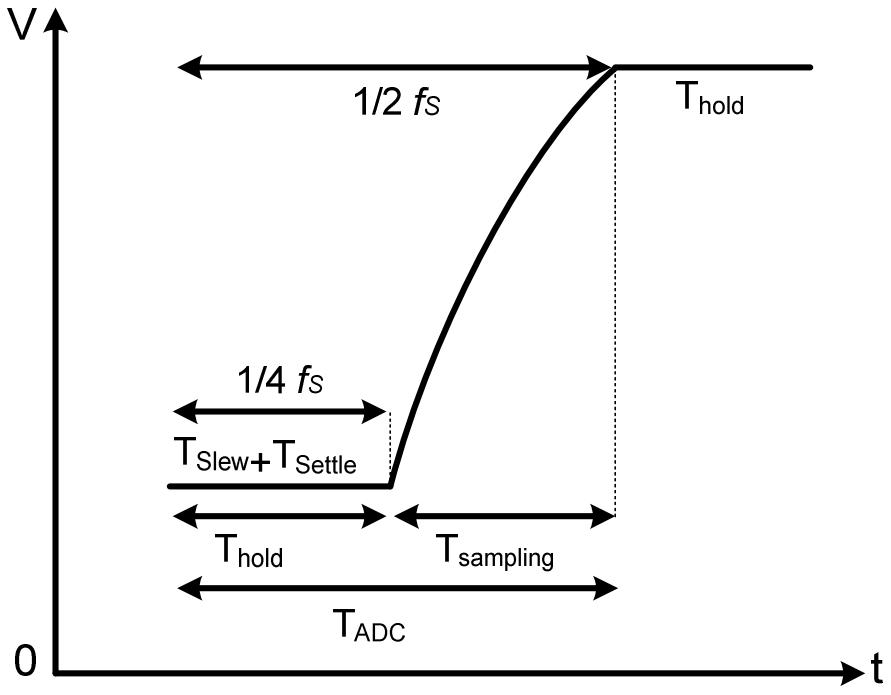

Figure 4.9: Illustration of PGA operation timing. PGA drives the S/H of data converter.

$$I_{PGA,\ Slew}=\frac{C_{L,PGA}.G_{PGA}.V_{FS}}{T_{Slew}} \tag{4.23}$$

where V_{FS} is the full scale range of the ADC. Typically, the time constant of the circuit must be small enough to make the circuit settle to within 2^{-n} for an allowable settling time T_{Settle} i.e.:

$$e^{-\left(T_{settle}/\tau_{PGA}\right)=2^{-n}} \tag{4.24}$$

which leads to $\tau_{PGA}=T_{Settle}/(n.\ln 2)$. For a given capacitance $C_{L,\ PGA}$, this result bounds $g_{m,PGA}$ and, therefore, yields a minimum supply current of the circuit based on linear settling $I_{PGA,Settle}= g_{m,\ PGA}.V_{eff}$:

$$\begin{cases} g_{m,PGA}=\dfrac{G_{PGA}.C_{L,PGA}.n.\ln 2}{T_{Settle}} \\ I_{PGA,Settle}=g_{m,\ PGA}.V_{eff} \end{cases} \tag{4.25}$$

The total time allowed for slewing (T_{Slew}) and linear settling (T_{Settle}) can be set at $T_{PGA} = 1/4f_s$ (Figure 4.9). The minimum PGA power consumption is:

$$P_{PGA} = P_{PGA\ ,Slew} + P_{PGA\ ,Settle} \tag{4.26}$$

$C_{L,PGA}$ can be equated to the capacitor network in the sampling path of S/H (C_u). C_u is determined by equating the sampling noise with the quantization noise [139]:

$$C_U = \frac{\beta.12.KT.2^{2.n}}{VFS^{\,2}} \tag{4.27}$$

where β is a constant factor between 2–5 [140], and V_{FS} is the full-scale voltage of the ADC. Since the theoretical limit for the unit capacitance C_u obtained by equating the sampling and quantization noise can be hardly reached in practice because other non-ideal effects such as mismatch, parasitic capacitances or charge injection from switches, the β coefficient is added to make the value of C_u more realistic. PGA and power consumption values then take into account some practical effects.

4.2.4 ADC Power Consumption

The theoretical minimum power consumption is examined in this section, firstly of a SAR ADC and secondly a power-scalable cyclic ADC using the CBSC approach.

4.2.4.1 SAR ADC:

The conversion process in a SAR ADC is performed by three basic building blocks (Figure 4.10):

a dynamic latch comparator (DLC), a capacitive DAC and a successive-approximation register (with its associated control logic). Assuming the power consumption of the noise-limited capacitive DAC is negligible, the power consumption of the SAR ADC is:

$$P_{SAR\ ,bound} \approx P_{DLC} + P_{register} + P_{Switch} \tag{4.28}$$

where P_{DLC} is the power consumption of the DLC, $P_{register}$ is the power consumption of the registers, and P_{Switch} is the switching power.

4.2.4.1.1 Dynamic Latch Comparator (DLC): This section explains the DLC power consumption in regeneration (reconfiguration phase) and reset phases. For simplicity of the analysis during regeneration phase, the DLC is modeled as two back-to-back inverters as shown in Figure 4.11. The output voltage equation (V_O=V_X-V_Y) of comparator [141] is:

$$V_O = A_V V_{I,diff} \exp\left(\frac{t_{d,DLC}}{\tau_{DLC}}\right) \tag{4.29}$$

where A_V acts as a gain factor from the inputs to the initial imbalance of the inverters, $V_{I,diff}$ is the input voltage difference to the DLC, $t_{d,DLC}$ is the DLC decision time, and the time constant τ_{DLC} =$C_{DLC}/g_{m,DLC}$ where C_{DLC} is the capacitive load of the DLC and $g_{m,DLC}$ is the total transconductance of the DLC simplified model.

In the binary search algorithm in a n bit DAC, n steps are needed to complete one conversion as the DAC output gradually approaches the input voltage. The $V_{I,diff}$ for the i^{th}-step can be expressed

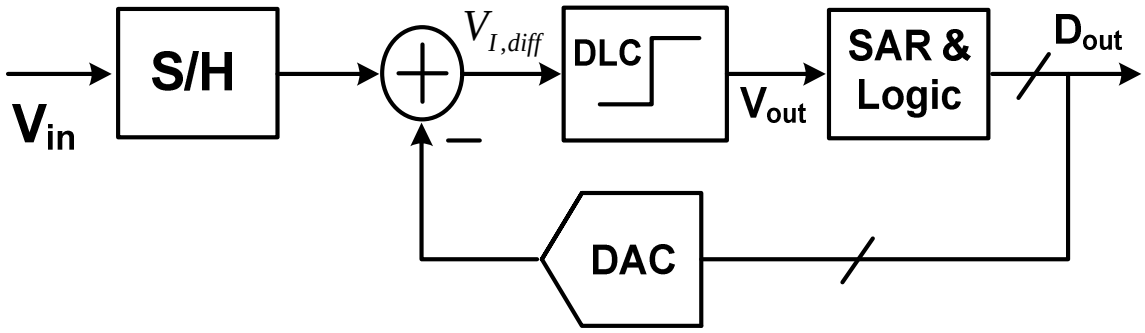

Figure 4.10: Charge-redistribution SAR ADC.

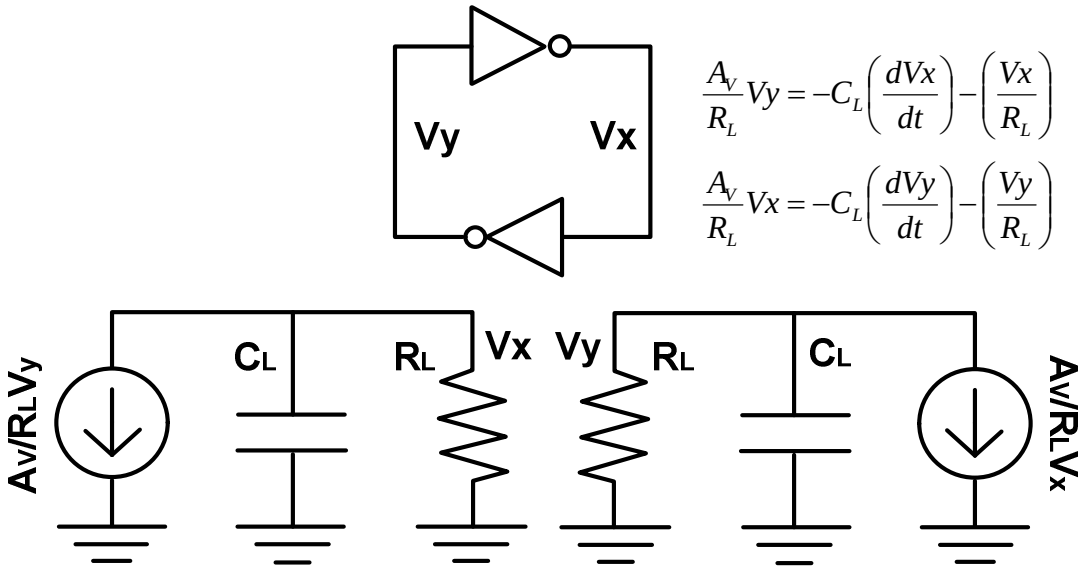

Figure 4.11: Simple model for dynamic latch comparator.

as:

$$V_{I,diff} = \left| -V_{IN} + D_{n-1}\frac{V_{REF}}{2} + ... + \frac{V_{REF}}{2^i} \right|, 1 \le i \le n \tag{4.30}$$

where V_{IN} is the input voltage, V_{REF} is the DAC reference voltage and D_{n-1} is the binary digit (either '0' or '1'). Eq.(4.29) is rewritten as:

$$\begin{aligned} V_O &= A_V V_{I,diff} \exp\left(\frac{t_{d,DLC}}{\tau_{DLC}}\right) \\ \frac{(V_O = V_{DD})}{A_V V_{I,diff}} &= \exp\left(\frac{t_{d,DLC}}{\tau_{DLC}}\right) \end{aligned} \tag{4.31}$$

Assuming that $t_{d,DLC} = 1/2f_s = T_s$ (i.e., the DLC works continuously) the required $g_{m,DLC}$ is:

$$g_{m,DLC} = \frac{C_{DLC}}{t_{d,DLC}} \sum_{k=1}^{n} \ln\left(\frac{V_{DD}}{\left(-A_V V_{IN} + A_V\left(V_{REF}/2^k\right)\right)} \right) \tag{4.32}$$

The sigma (Σ) term is expanded as:

$$\sum_{k=1}^{n} \ln\left(\frac{V_{DD}}{\left(-A_V V_{IN} + A_V\left(V_{REF}/2^k\right)\right)} \right) = \sum_{k=1}^{n} \left(\ln(V_{DD}) + \ln(A_V V_{IN}) - \ln\left(\frac{A_V . V_{REF}}{2^k}\right) \right) \tag{4.33}$$

$$\sum_{k=1}^{n}\left(\ln(V_{DD})+\ln(A_V V_{IN})-\ln\left(\frac{A_V.V_{REF}}{2^k}\right)\right)= \\ n.\ln(V_{DD})+n.\ln(A_V V_{IN})-\sum_{k=1}^{n}\left(\ln(A_V.V_{REF})-\ln(2^k)\right) \quad (4.34)$$

$$= n.\left(\ln(V_{DD})+\ln(A_V V_{IN})\right)-n.\ln(A_V.V_{REF})+\ln(2).\sum_{k=1}^{n}(k) \quad (4.35)$$

$$\approx n.\ln(V_{DD})-n.\ln(A_V.V_{REF})+n\ln(2).\frac{n}{2} \quad (4.36)$$

Replacing the sigma (Σ) term of Eq. (4.36) into Eq. (4.32) is results in:

$$g_{m,DLC}=2nf_s.C_{DLC}\left[n\left(\ln\frac{V_{DD}}{A_V V_{REF}}\right)+\frac{n^2}{2}\ln 2\right] \quad (4.37)$$

The simplified total input-referred noise voltage of the DLC has a fundamental *kT/C* limitation given by [142]:

$$\overline{Vn^2}=4\gamma\frac{KT}{C_{DLC}} \quad (4.38)$$

where γ is a noise parameter (γ = 2/3 to 2). Equating this noise to the quantization noise, the minimum capacitive load of the DLC (under noise constraint) is:

$$C_{DLC}=48KT\gamma\frac{2^{2n}}{V_{FS}{}^2} \quad (4.39)$$

Substituting Eq.(4.39) in Eq.(4.37) the minimum value of $g_{m,DLC}$ and thus $I_d = g_{m,DLC}V_{eff}$ can be found [7]. The minimum DLC power consumption is:

[7] *For bipolar transistors* V_{eff} = kT/q (~26 mV), for MOS transistors in strong inversion $V_{eff} = (V_G - V_T)/2$, and for MOS transistor in sub-threshold $V_{eff} = mkT/q$ with *m* (sub-threshold slope factor) slightly larger than one. For typical transistors in 90–350 nm processes V_{eff} is 100–300 mV.

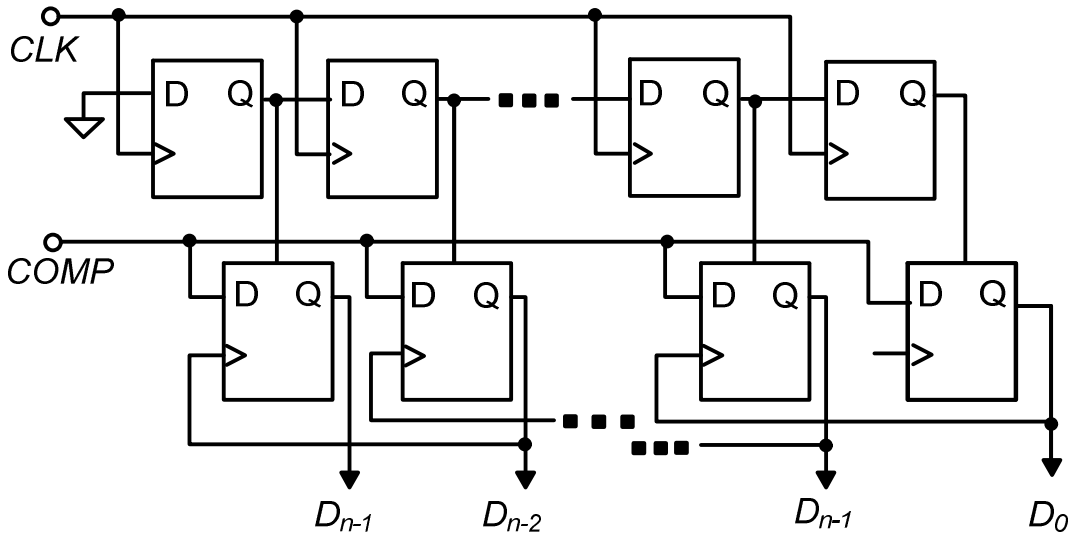

Figure 4.12: SAR digital logic.

$$P_{DLC,regeneration} = 96nf_s.KT\gamma\frac{2^{2n}}{V_{FS}{}^2}V_{DD}V_{eff}\left[\left(\ln\frac{V_{DD}}{A_V V_{REF}}\right)+\frac{n^2}{2}\ln 2\right] \tag{4.40}$$

In addition to the regeneration power during reset phase, the capacitive load at the comparator output is reset to the supply voltage V_{DD}. The power consumption during the reset phase is expressed as:

$$P_{DLC,reset} = n.f_S.C_U.V_{DD}^2 \tag{4.41}$$

The total amount of consumed power in operation phases is equal to:

$$P_{DLC} = P_{DLC,\,regeneration} + P_{DLC,\,reset} \tag{4.42}$$

4.2.4.1.2 SAR Register: Normally the register logic consists of 2*n* D-type flip-flops (DFFs) for *n*-bit resolution [143] (see Figure 4.12). It is assumed that each DFF can be modeled with a minimum of twelve transistors if they are replaced with NMOS pass transistors (almost six inverters) [144]. Typically an inverter consists of one minimum geometry nMOS and 3X wider pMOS transistors. $C_{\min}$ is defined as the input capacitance of a minimum-sized inverter; $C_{\min}$ = 1 fF and 5 fF for 90-nm and 180 nm technologies respectively. The entire register logic could be considered as a capacitive load ($C_{\text{logic}} = 24nC_{\min}$). To drive the capacitance within the sampling phase requires a

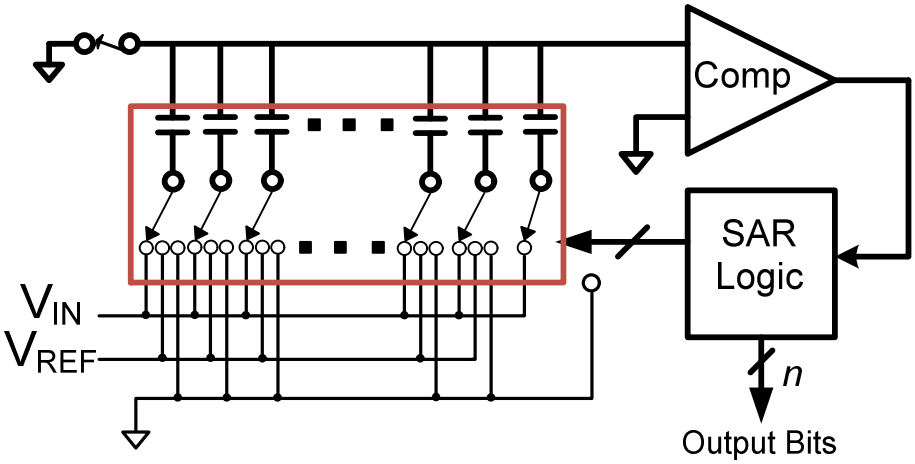

Figure 4.13: SAR ADC switching.

current of $I = (C_{\text{logic}} V_{DD}) / T_s$ which leads to the following minimum power consumption for the register logic:

$$P_{register} = 96\, n^2\, f_s C_{\min} V_{DD}^2 \tag{4.43}$$

4.2.4.1.3 Binary Weighted Charge-Scaling DAC: The DAC in a SAR ADC uses a binary-weighted capacitor array (with a unit capacitance C_u) to attenuate V_{REF} by charge scaling. In the conversion phase, the switches in the capacitor array are recursively set to '1' (close) or '0' (open) to adjust the binary-weighted values to the input sample value. In fact the DAC power consumption depends on the units contributing to the power consumption of SAR ADC including the loading value of the capacitive array, input signal swing and the employed switching approach. For an *n*-bit conventional SAR ADC, the average switching energy is [145]:

$$E_{avg,conv} = \zeta \sum_{i=1}^{n} 2^{n+1-2i} \left(2^i - 1\right) C_U V_{REF}^2 \tag{4.44}$$

where V_{REF} is the reference voltage, and ζ is a normalized switching scheme-dependent parameter. For conventional switching approach [145], ζ = 1 (for advanced approaches with bypassing methods ζ = 0.6-0.7). Hence, the power consumption due to the switching action is:

$$P_{Switch} = \frac{E_{avg,conv}}{T_S} = 2\zeta \sum_{i=1}^{n} 2^{n+1-2i} \left(2^i - 1\right) C_U V_{REF}^2 f_S \tag{4.44}$$

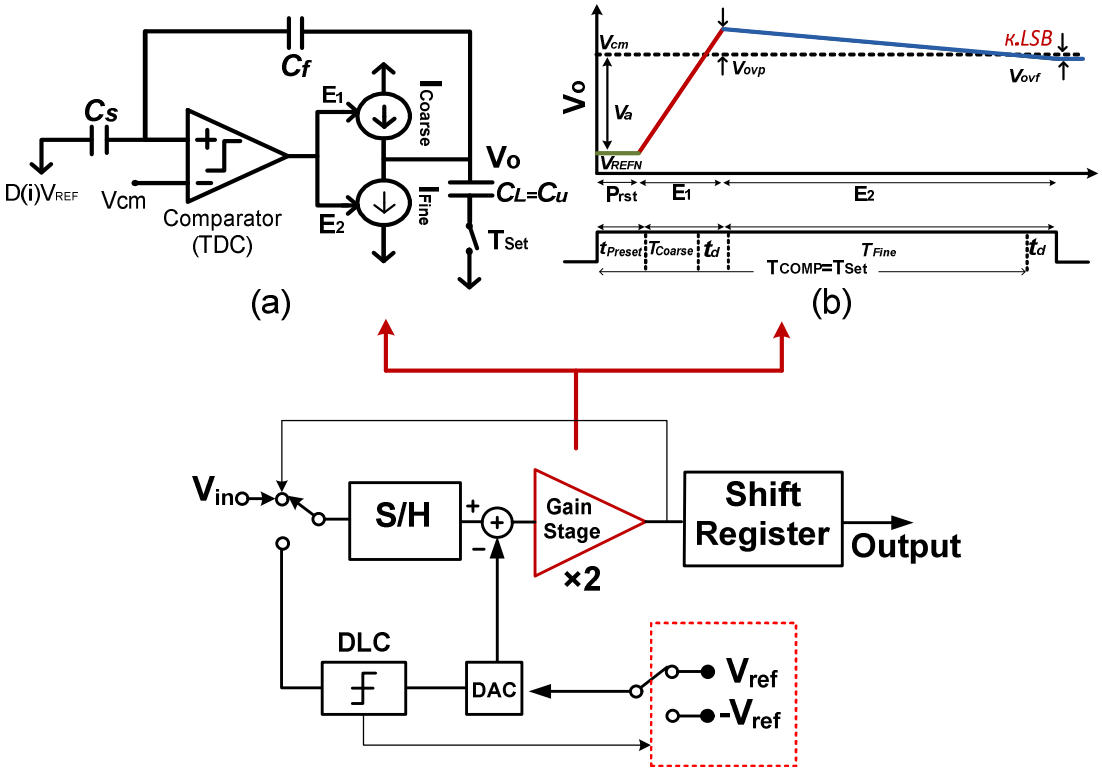

Figure 4.14: Block diagram of a cyclic ADC. (a) CBSC gain stage core and (b) timeline for output voltage (V_O).

4.2.4.2 CBSC Cyclic ADC:

Although the SAR ADC is generally used for converting bio-signals, there are drawbacks with this architecture including area occupation (which increases exponentially with resolution), the design of the capacitive array with matched elements and the necessary switching activity. An alternative ADC is the CBSC cyclic data converter which has not previously been considered for NFI design. The CBSC gain stage has been reported in [126-127] which replaces the op-amp with comparator and current sources, allowing more efficient use of power. As shown in Figure 4.14 a cyclic ADC uses a simple analog circuit that is repeatedly re-used to perform the conversion cyclically in time. This results in a very compact ADC but is obtained at the cost of a long conversion time. It requires n + 1 clock periods to provide an n-bit output. The overall power consumption ($P_{Cyclic,\,bound}$) of a CBSC cyclic ADC is estimated as follows:

$$P_{Cyclic,bound} \approx (n).\left(P_{TDC} + P_{Coarse} + P_{Fine}\right) \tag{4.46}$$

where P_{TDC}, P_{Coarse} and P_{Fine} are the power consumed by comparator, coarse current source and fine current source during their operation mode.

4.2.4.2.1 Current Sources

This section discusses the requirements for the current sources, necessary to achieve a certain accuracy in the gain stage during the charge transfer period. To meet accuracy, appropriate values of design parameters are analytically discussed and a procedure to select them is presented. Consider the positive half-circuit output voltage of a gain stage, as shown in Figure 4.14 (a), (b). The settling time required to complete the settling phase, $t_{Settle}=T_{Set}$, is [146]:

$$t_{Settle}=t_{prst}+T_{Coarse}+T_{Fine} \tag{4.47}$$

where t_{prst} is the preset time duration, T_{Coarse} is the coarse charge transfer phase (charging time) and T_{Fine} is the fine charge transfer phase (discharging time). The initial charging time, T_{Coarse}, is given by:

$$T_{Coarse}=\frac{V_a}{M_C}+t_{d1} \tag{4.48}$$

where M_C is the coarse phase ramp rate, t_{d1} is the delay time during the coarse phase and V_a is equal to the voltage difference between the common mode voltage, V_{cm}, and the negative preset level (V_{REFN}), ($V_a=V_{cm}-V_{REFN}$). During the fine transfer phase, the primary overshoot (V_{OVP}) produced at the end of the coarse phase, has to be compensated. The overshoot recovery time, T_{Fine}, is given by:

$$T_{Fine}=\frac{V_{OVP}}{M_f}+t_{d2} \tag{4.49}$$

where M_f is the fine phase ramp rate and t_{d2} is the delay time during the fine phase. V_{OVP} is given by.

$$V_{OVP}=\frac{I_{Coarse}.t_{d1}}{C_U}=M_C.t_{d1} \tag{4.50}$$

Here, I_{Coarse} is the coarse current and C_U is the total loading capacitance at the output of the gain stage. Eq.(4.49) can be rewritten as:

$$T_{Fine} = \frac{M_C}{M_f}.t_{d1} + t_{d2} \tag{4.51}$$

Substituting Eq.(4.48) and Eq.(4.51) into Eq.(4.47) gives the total charge transfer time in terms of the comparator delays, t_{d1} and t_{d2}, the coarse ramp rate M_C, and the fine phase ramp rate M_f, as:

$$t_{Settle} = \frac{V_a}{M_C} + t_{d1} + \frac{M_C}{M_f}.t_{d1} + t_{d2} \tag{4.52}$$

For simplicity, $t_{d1} = t_{d2} = t_d$ is assumed in the succeeding discussion. For a specific accuracy requirement the final overshoot is set by $\kappa.LSB$ where κ (e.g.1/8) is the accuracy factor which determines the allowable variation at the end of the charge transfer phase. The maximum allowed output voltage overshoot for the first stage is $V_{ovf} = \kappa.(V_{FS}.(2^{-n})) = \kappa.(LSB)$. This overshoot value determines the maximum final ramp rate for stage:

$$M_f = \frac{V_{OVf}}{t_d} = \frac{\kappa.LSB}{t_d} = \frac{\kappa.V_{FS}}{t_d.2^n} \tag{4.53}$$

By substituting Eq.(4.53) into Eq.(4.51), the required time to complete the fine charging transfer is given by:

$$T_{Fine} = \left(\frac{M_C}{\kappa.LSB}\right).\left(t_d^2 + t_d\right) \tag{4.54}$$

Assuming the sampling capacitor size is determined from section 4.2.3, the maximum fine phase current can be determined. In addition the total charge transfer time must be less than half of the sampling clock period ($T_s/2 = 1/2f_s$). It indicates that the I_{Coarse} and I_{Fine} are suitable to satisfy the charge transfer phase. Substituting Eq.(4.52) into the sampling clock constraint leads to:

$$\frac{1}{2f_S} = \frac{V_a}{M_C} + \frac{M_C}{M_f}.t_d + 2t_d \tag{4.55}$$

Rearranging Eq.(4.55) yields:

$$M_C^2 t_d - M_C.M_f.\left(\frac{1}{2f_S} - 2t_d\right) + M_f.V_a \tag{4.56}$$

Solving Eq.(4.56) for the coarse ramp rate M_C, results in:

$$M_C = \frac{\left(M_f . \left(\frac{1}{2 f_S} - 2t_d \right) + \sqrt{\left(M_f \left(\frac{1}{2 f_S} - 2t_d \right) \right)^2 - \left(4 t_d M_f . V_a \right)} \right)}{2 t_d} \tag{4.57}$$

The power consumption of coarse current source P_{Coarse} is given by:

$$P_{Coarse} = 2 \left(M_C . C_U \right) . V_{DD} . \left(T_{Coarse} + t_d \right) \tag{4.58}$$

where V_{DD} is the supply voltage. For the fine phase, the operation of an inverter during discharging can be considered. During this phase the overshoot voltage induced in the coarse phase is drawn from the capacitor network. In this situation the displacement current through the capacitor is caused by the pull-down current source and flows into ground. Thus the energy P_{Fine} is:

$$P_{Fine} = C_U \left(\left(M_C . t_d + Va \right)^2 - V_{CM}{}^2 \right) . \left(T_{Fine} + t_d \right) \tag{4.59}$$

4.2.4.2.2 Threshold Detection Comparator (TDC)

In this section power analysis for the threshold detection comparator (TDC) is examined. The comparator translates crossovers into a change in output logic value which controls the current sources. The TDC can be implemented by cascading several identical band-limited comparators [147] as shown in Figure 4.15. g_m is the transconductance and C is the capacitance of the output node at each stage, The delay between the comparator decision and the output logic unit for activation of the current sources is susceptible to process variations which can cause instability in the CBSC gain stage. In addition, the harmonic distortion at the output node of a CBSC gain stage is inversely proportional to the TDC delay time ($t_{d,\mathrm{TDC}}$). The effective open-loop gain A_{TDC} of a CBSC circuit [148] is inversely proportional to TDC delay ($t_{d,TDC}$):

$$A_{TDC} \approx \frac{1}{t_{d,TDC}} \tag{4.60}$$

An optimum design strategy is to minimize the TDC delay ($t_{d,TDC}$) by cascading the N identical

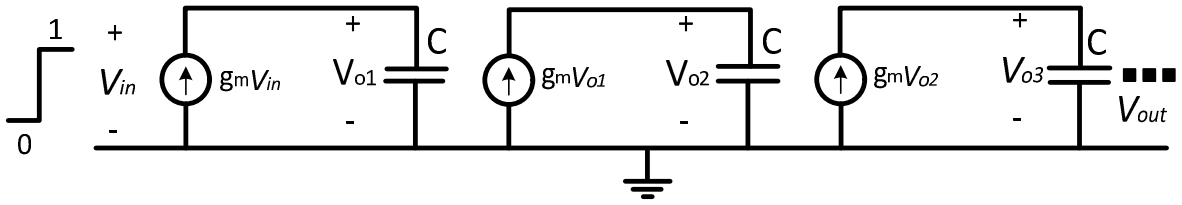

Figure 4.15: Illustration of a cascaded comparator. $A_{TDC}=V_{out}/V_{in}$.

amplifiers. The step response of a cascaded amplifier is:

$$V_{O1}=\frac{1}{C}\int_0^t g_m V_{in}\,dt=\frac{g_m}{C}V_{in}\,t \tag{4.61}$$

$$V_{O2}=\frac{1}{C}\int_0^t g_m V_{O1}dt=\frac{g_m}{C}\int_0^t \frac{g_m}{C}V_{in}dt=\frac{1}{2}\left(\frac{g_m}{C}\right)^2 V_{in}\,t^2 \tag{4.62}$$

$$V_{O3}=\frac{1}{C}\int_0^t g_m V_{O2}\,dt=\frac{g_m}{C}\int_0^t\left[\frac{1}{2}\left(\frac{g_m}{C}\right)^2 V_{in}\right]t^2 dt=\frac{1}{6}\left(\frac{g_m}{C}\right)^3 V_{in}\,t^3 \tag{4.63}$$

$$V_{O4}=\frac{1}{C}\int_0^t g_m V_{O3}dt=\frac{g_m}{C}\int_0^t\left[\frac{1}{6}\left(\frac{g_m}{C}\right)^3 V_{in}\right]t^3 dt=\frac{1}{24}\left(\frac{g_m}{C}\right)^3 V_{in}\,t^4 \tag{4.64}$$

Therefore, the output voltage of stage N is generalized as:

$$Vout=\left[\frac{g_m}{C}\right]^N\left[\frac{(t=t_{d,TDC})^N}{(N!)}\right]Vin \tag{4.65}$$

where for each comparator stage $\tau = C/g_m$ is the unity gain time constant and N is the number of stages. The bandwidth (g_m/C) is given by:

$$\left[\frac{g_m}{C}\right]=\sqrt[N]{(N!)\times\frac{V_{out}}{V_{in}}}\times\frac{1}{t_{d,TDC}} \tag{4.66}$$

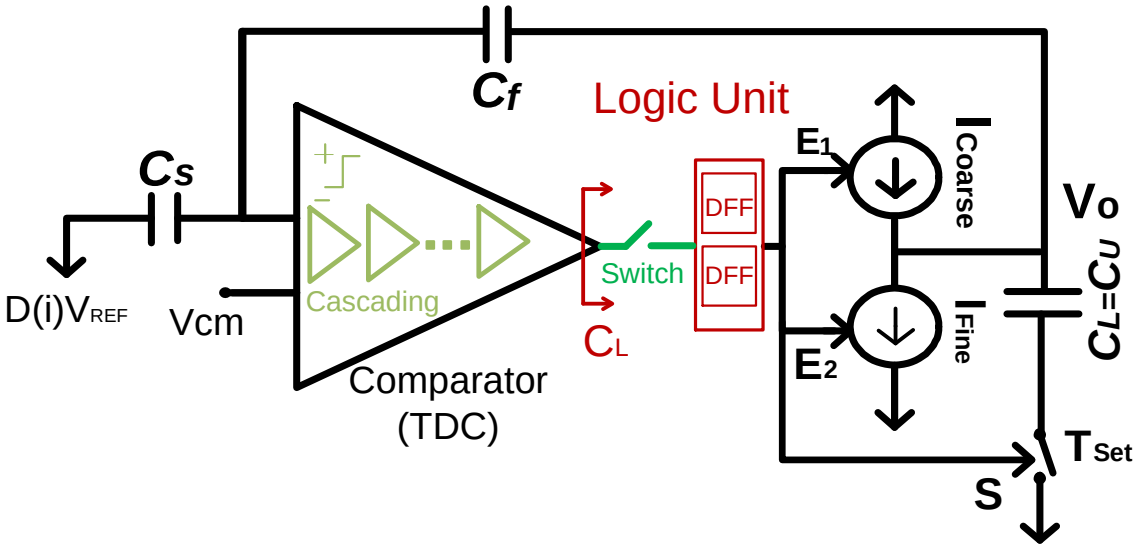

Figure 4.16: Driving logic by TDC (N=1).

where ($A_{TDC}=V_{out}/V_{in}$) is the total gain of the cascaded TDC. For a comparator composed of a cascade of gain stages, only the noise of the first stage is important. The input-referred noise of the subsequent stages will be suppressed by the gain of the first stage gain. The number of cascading stages is chosen as one (N=1) in this analysis, since there is no need for high speed conversion rates in biomedical applications (e.g. spike sorting requires 30Ks/s). For example in [149], a two stage cascading topology is considered to implement a highly accurate data converter for 26MS/s digitization rate. The use of a logic unit followed by the TDC in CBSC ensures that the settling issue is less significant compared with op-amp based architectures. The performance of the TDC is limited by its finite slew rate and its delay ($t_{d,TDC}$). The TDC bias current level must be chosen to provide a high slew rate. The required current to drive the logic unit input capacitance ($C=C_{Logic}$) to V_{DD} is ($I_{Comp}=C_{Logic}.V_{DD}/t_{d,\,TDC}$) where C_{Logic} is equal to the input capacitance of the logic unit which is composed of a switch and 2 DFFs (Figure 4.16). So the TDC must be capable of driving a capacitive load in the order of $48C_{\min}$. In addition, the $t_{d,\,TDC}$ is designed in the order of $1/20f_S$. The power consumption of TDC is calculated as $P_{TDC}=I_{TDC}.V_{DD}$.

$$P_{TDC}=\begin{matrix}\dfrac{C_{Logic}.A_{TDC}.V_{DD}^2}{t_{d,TDC}}.\left(T_{Comp}=T_{Set}\right)\\ \dfrac{C_{Logic}.A_{TDC}.V_{DD}^2}{t_{d,\,TDC}}.\left(T_{Coarse}+T_{Fine}+2t_d\right)\end{matrix} \tag{4.67}$$

where $T_{Comp} = T_{Set}$ is the time when comparator is active. Furthermore, the A_{TDC}-g_m (for N=1) can be investigated via E.q. (4.63).

4.3 MATLAB-FPGA Interfacing

Figure 4.17 shows the system used to accurately characterize the effect of NFI parameters on spike sorting. The FPGA board interfaces the developed NFI behavioral tool in MATLAB using a UART connection. The spike processor is programmed on an FPGA to cluster the input neural data. It reads the neural data from MATLAB via the interface, executes spike sorting, and finally the sorted data are sent back to MATLAB for cluster accuracy analysis. The MATLAB-FPGA Interfacing provides an assessment tool to avoid over-engineered NFIs by taking into account their impact on subsequent spike processing during the design process. The design specifications can then be adapted in order to maximize power efficiency and minimize hardware resource utilization.

4.3.1 Spike Sorting Processor

Instead of using Matlab the spike sorting processor based on the DD|$_2$-Extrema and online clustering method was implemented in an FPGA. This was to demonstrate the potential online hardware capability when applied in an ASIC. The Xilinx Artix-7 (XC7A200T) FPGA was programmed with Verilog using the Xilinx ISE 2014 tool. To satisfy power and area constraints a serial-parallel approach was adopted for the system architecture. Figure 4.18 shows the functional block diagram of the synchronous spike sorting processor. It implements spike detection, alignment to positive or negative peaks, mapping of aligned spikes to feature domains (feature extraction using DD|$_2$-Extrema [150]), and finally clustering by assigning feature vectors to their origins based on the $\ell 1$-norm metric. The detection block constantly receives the raw data for calculating the detection threshold. The captured spike is applied to the alignment block for

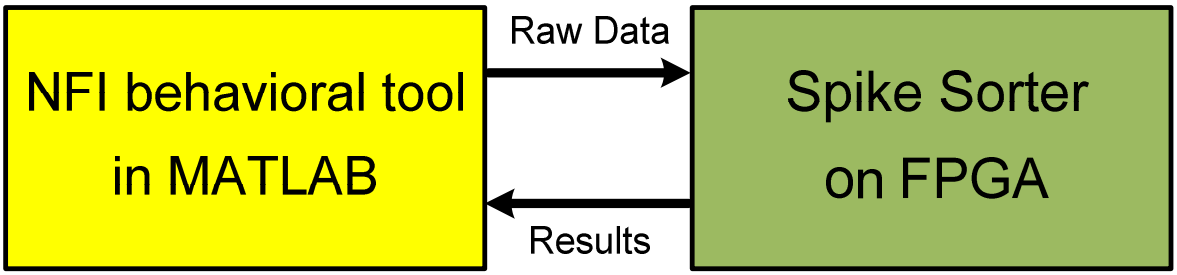

Figure 4.17: MATLAB-FPGA Interfacing used for NFI key parameters optimization.

calculating the alignment address based on a selected configuration such as maximum peak, mixed peak, maximum slope etc. The alignment address provides 45 samples for each aligned spike to the feature extraction block. *Aligned_valid* is used for buffering the *Aligned_spike*. The spike sorting processor performs clustering by reading the buffered feature vectors (*Feature_vec*).

Figure 4.19 shows the buffering structure of the feature vectors. It comprises a bank of register

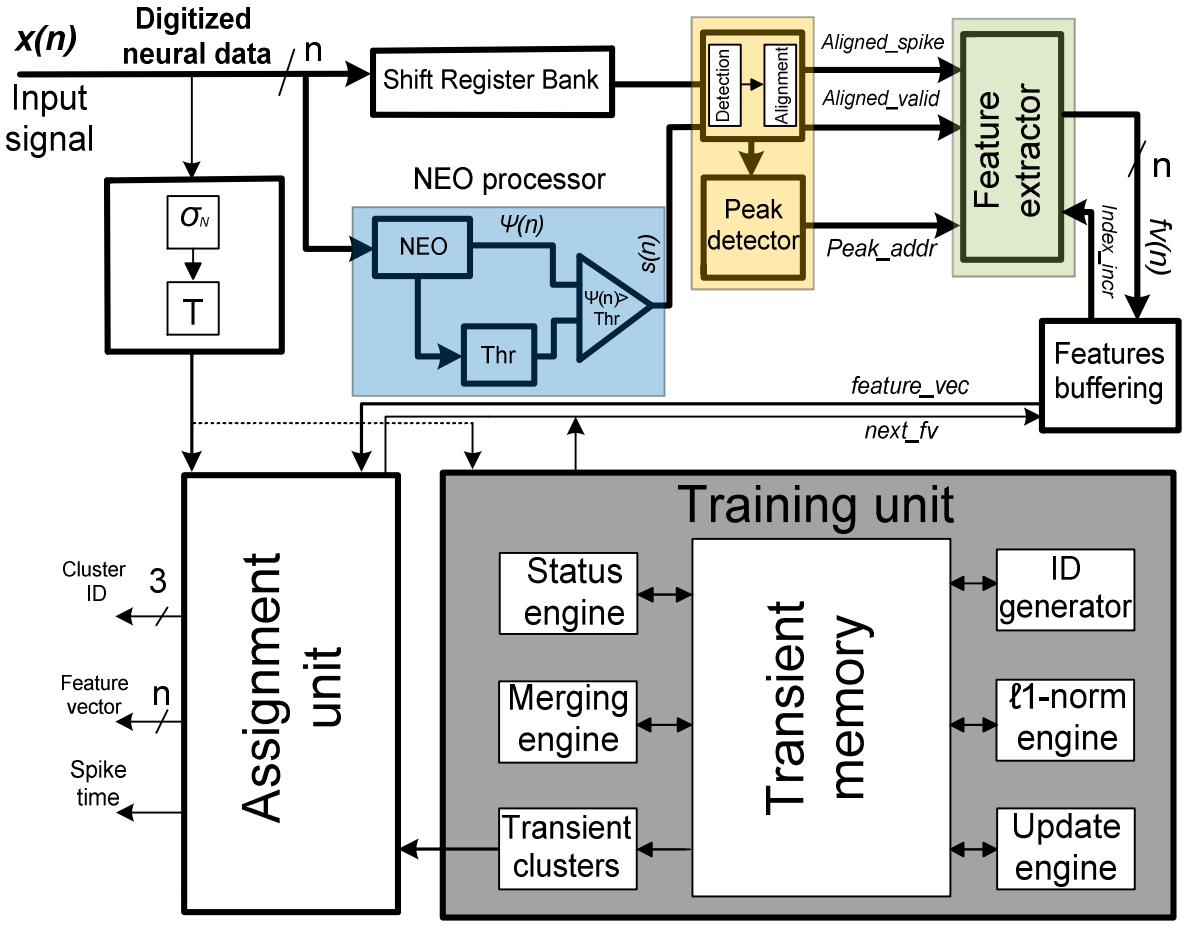

Figure 4.18: Block diagram of the spike sorting processor.

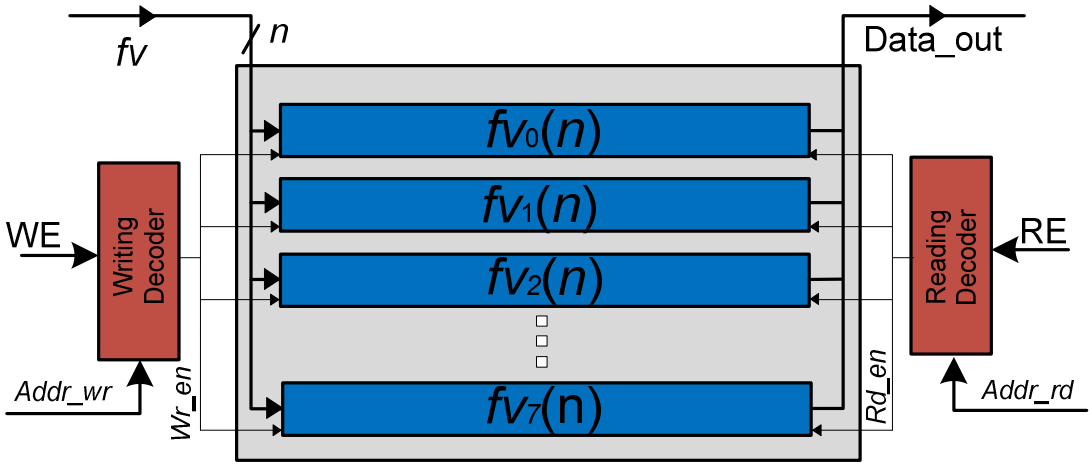

Figure 4.19: Block diagram of the features buffering memory.

memories. Each register is used to save a feature vector (*fv*). When a spike is saved in a register (e.g. *fv*0), the writing pointer is incremented by one to save the next *fv*. In the reading section, when the training or assignment for the current spike is performed, the next *fv* is called for processing. The decoder enables the address and the reading enable (*Rd_en*) signal for reading the *fv*. Incrementing the reading address (*Addr_rd*) at each clock cycle access to all the buffered samples is available for either the training or assignment phase.

The clustering step uses online sorting (O-Sort) [13] which is composed of training and assignment units to perform unsupervised, real time mapping of spikes to single neuron activity. The O-Sort steps are as follows:

1. Initialization – assign the first feature vector (*fv*) to a cluster.
2. The distance (*Dist*) between the next received spike *fv*(*n*) and other known cluster means [*c*(*n*)] is computed using the ℓ1-norm metric for the number of samples (N_S) representing *fv*(*n*) where

$$Dist = \sum_{i=1}^{r} \sum_{n=1}^{N_S} \left| fv_i(n) - c(n) \right| \tag{4.68}$$

   where *i* denotes the row number and *r* denotes the sample number of each feature vector.

3. If the smallest distance is less than the merging threshold T_M, the spike is assigned to the nearest cluster and that cluster's mean is computed as the weighted average of the input spike and the current cluster:

$$C_{update}(n) = \frac{W * c(n) + s(n)}{W + 1} \tag{4.69}$$

where W is the number of spikes representing a specific class and * denotes multiplication. Otherwise, a new cluster is generated.

4. Check the distances between each cluster and every other cluster. If any distance is below the sorting threshold T_S, merge those two clusters and re-compute its mean.
5. Once the cluster means are identified in the training phase, cluster assignment is initiated to simultaneously process and assign the incoming spikes to one of the recognized cluster means in the training phase according to the minimum distance (*Dist*).

Steps 2–4 are then repeated continuously. The design uses a threshold $T = 4(\sigma_N)$ for both T_M and T_S, where σ_N is the noise standard deviation of the data computed continuously with a long sliding window. The outputs of the clustering block, namely *Cluster ID*, *feature vector* (*fv*) and *Spike time*, are used for sorting performance calculation and feature space analysis.

The results of the spike sorting processor are based on 250 feature vectors composed of extrema of discrete derivatives. Compared to the only other spike sorting processor that provides real time, multi-channel clustering [76], the hardware model developed for $DD|_2$-Extrema has approximately 8X smaller memory size. The proposed spike sorting processor provides detection, alignment, feature extraction and clustering for real time processing.

4.4 Validation Tests and Results

4.4.1 Test Setup

As explained in previous section, the MATLAB-FPGA interface is established to accurately characterize the impact of front-end circuitry on back-end processing. The implemented NFI simulator in MATLAB sweeps the parameters and assesses the sorting accuracy based on the FPGA hardware implementation. The demand for sorting performance determines the minimum NFI specifications. Data is processed by the FPGA platform, and the results are written to the output buffers for accuracy analysis. For each spike, the spike sorter outputs the Cluster_ID, Assign_ID, Spike_time and feature vector (*fv*). Outputs are used for sorting accuracy analysis. After initializing the FPGA, the spike processor runs for a specific time (user programmable) to establish the cluster means in each channel. The entire dataset is then processed based on the tuned

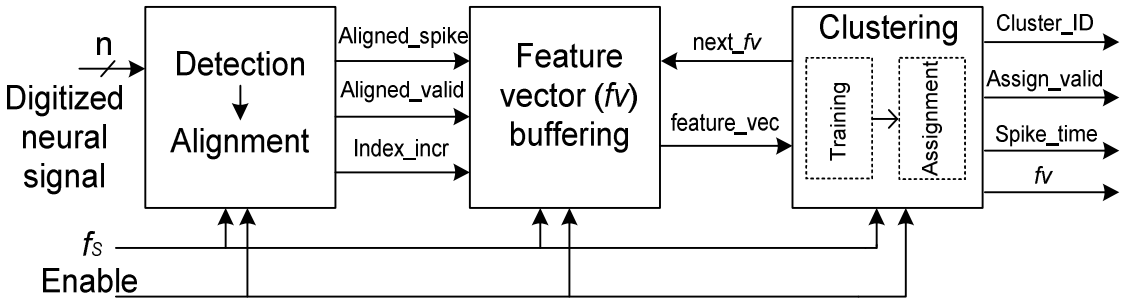

Figure 4.20: Block diagram of the processing elements on the FPGA.

parameters in detection and sorting units. The spike sorter FPGA implementation is depicted in Figure 4.20.

4.4.2 Design Optimization

In the following sections, key parameters of the NFI are optimized based on the spike sorting performance of the processor implementing the $DD|_2$-Extrema feature extraction method described in chapter 3. A second order bandpass Bessel filter was used in the NFI model to reject the undesired out-of-band frequency components, namely low-frequency local field potentials and high-frequency noise. The Matlab function *resample* was used to down sample the bandpass filtered data. The function *quant* was used to quantize the sampled data.

The low and high corner frequencies of the LNA were swept and the performance of the spike sorting processor was observed to define an optimized bandwidth for the LNA. In each iteration, the clustering accuracy (CA_{CC}) was calculated for the optimum values:

$$CA_{CC} = \frac{TPCC}{NTS} \times 100\ \% \tag{4.70}$$

where *TPCC* is the number of truly detected and correctly classified spikes and NTS is the number of truly detected spikes. *NTS* = DTS – (FPS + MS), where DTS is the number of detected spikes, FPS is the number of false alarm spikes due to noise or overlapping spikes, and MS is the number of missed spikes. The spike processor clustering test using template matching (TM) feature extraction is depicted in Figure 4.21.

The performance of the spike sorting processor was also evaluated when the sampling frequency and the resolution of the ADC were swept. The specifications of the key components of the NFI were optimally defined having both efficiency and efficacy in mind. The derived optimum values

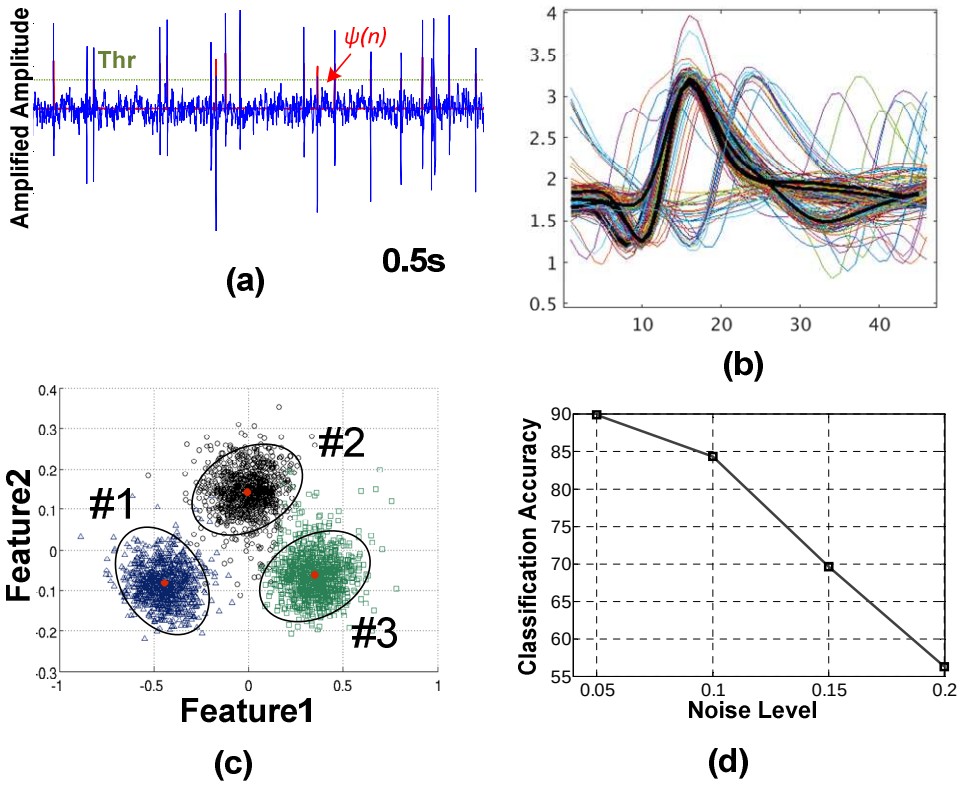

Figure 4.21: The spike sorting processor clustering test using template matching (TM) FPGA platform. The sample output from one of the channels is shown in this figure. (a) Segment of neural signal. (b) Distinguished cluster means from 250 spikes in training phase. The spike processor for TM is developed for this test. (c) A 2-D projection of the clustered spikes color-coded, each represents a cluster. (d) The relation of clustering accuracy to noise level.

from classifier robustness analysis were used to compare the NFI power limit of two types of data converter.

4.4.2.1 Defining Band-pass Response With Respect to Spike Sorting Performance

The obtained dataset introduced in chapter 2 (section 2.4.1) was already high-pass filtered. Low frequency noise was added to this data to examine the performance of the spike sorting processor given different low corner frequencies. Low frequency noise was extracted from a different dataset [151][8] and superimposed on the dataset under examination. Figure 4.22 shows the result of variations in high and low corner frequencies on CA_{CC}. It can be seen that for good performance,

[8]http://crcns.org/data-sets/hc/.exp

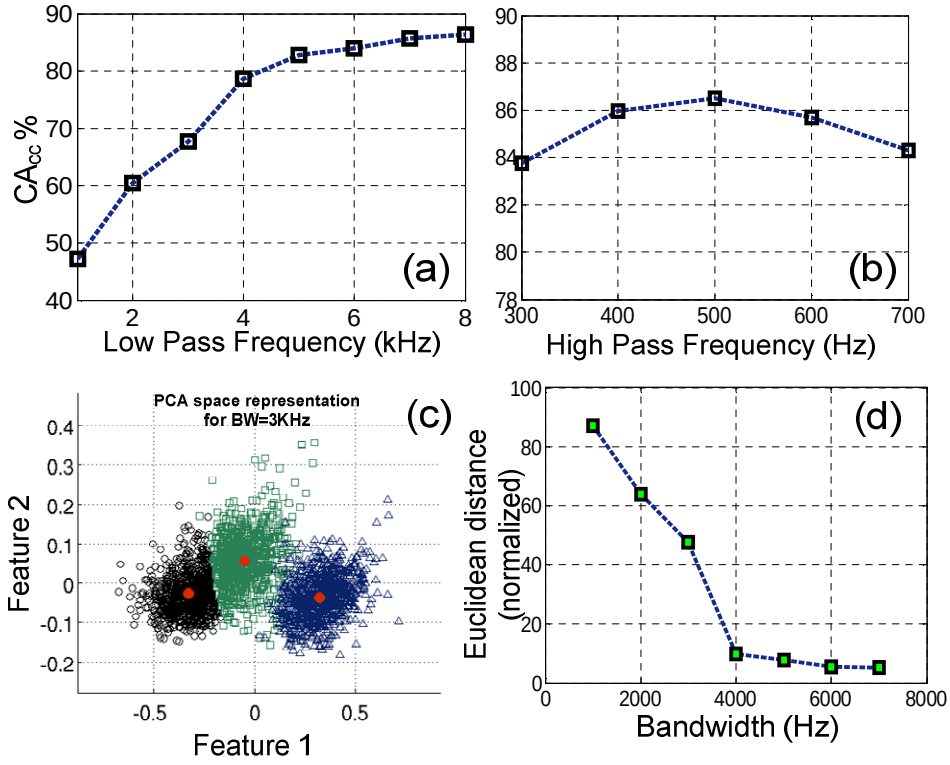

Figure 4.22: Effect of low pass (a) and high pass (b) filtering on spike sorting accuracy (mean over all spike datasets). The local field potential was extracted from *in-vivo* recording and superimposed on the dataset under examination. (c) PCA feature space for the feature vectors of three neurons. (d) Euclidean distance analysis of feature vectors.

the low corner frequency for the high pass should be set to approximately 500 Hz [Figure 4.22 (b)]. Similarly, the corner frequency of the low pass should set at approximately 7 kHz [Figure 4.22 (a)], below which the performance is degraded Principal component analysis (PCA) feature space can also be used to demonstrate the separation quality of the clusters. Increasing the high-pass cut-off frequency marginally increases the overall spike sorting accuracy. There is small degradation in the sorting performance when the LNA bandwidth ($BW_{LNA} = f_{LP} - f_{HP}$) is reduced to about 4 or 5 kHz. The sorting performance rapidly deteriorates for bandwidths of less than 3 kHz. Figure 4.22 (c) shows the PCA feature space when the bandwidth is set to 3 kHz. The Euclidean distance of the feature vectors for different sampling frequencies is shown in Figure 4.22 (d). A high Euclidian distance means higher error rate and lower accuracy. It shows that bandwidths of less than 4 kHz cause considerable deterioration in feature space (note that the spike energy concentration is in the frequency range of 6–7 kHz).

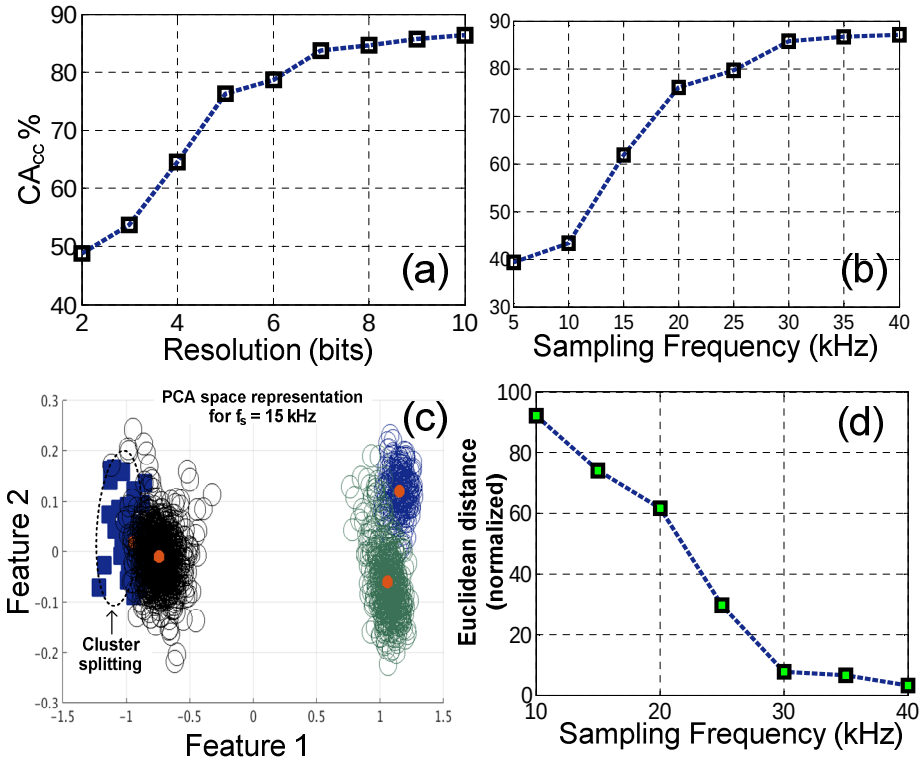

Figure 4.23: Effect of data conversion (a) resolution, and (b) sampling rate on spike sorting accuracy (mean over all datasets). (c) PCA features when the sampling rate is reduced to 15 kHz for three neurons. The effect of cluster splitting is highlighted in the figure. (d) Euclidean distance analysis of feature vectors.

4.4.2.2 Resolution (n) and Sampling Rate (f_s) Versus Spike Processing Performance

The performance of the spike sorting processor for different ADC resolutions was investigated. As shown in Figure 4.23 (a), when the ADC resolution increases the CA_{CC} of the spike sorting processor increases up to a limit beyond which there is little improvement. This limit depends on the degree of variations of the spikes where some of the variations are not represented in the digitized signal. The limit is approximately 7 bits in Figure 4.23 (a). Similarly the ADC sampling rate should be high enough to represent the shape of the spikes. This may be significantly higher than the Nyquist frequency because the variations in the shape of the spikes associated with different classes may be considerably subtler. In Figure 4.23 (b), a sampling frequency of about 30 kHz is sufficient to provide accurate spike sorting. The aim was to identify the lower limit of the ADC resolution and the sampling rate because higher levels correspond to higher power consumption.

The effect of reducing the sampling rate on the spike sorting process is also identified in PCA feature space. The sampling rate was decimated to 15 kHz using the *resample* Matlab function. The Euclidean distance analysis shown in Figure 4.23 (d) shows that the sorting performance degrades rapidly when the sampling rate decreases below 30 kHz. Reducing the sampling frequency causes creation of sub-clusters corresponding to one original cluster so cluster splitting results in large sorting performance degradation. The more the sampling frequency is reduced, the more spurious sub-clusters are generated. Figure 4.23 (c) shows the PCA feature space representation when the sampling frequency is reduced to 15 kHz. Additionally, it was observed that the PCA features space does not significantly change when the number of bits is reduced to as low as 4, which shows that the features are robust to the quantization bit depth.

4.4.3 NFI power bound

The overall minimum power consumption of a noise-limited NFI is:

$$P_{NFI} = ((\alpha + 1).\, P_{LNA}) + (P_{PGA}) + (P_{ADC}) \tag{4.71}$$

A survey of P_{NFI} for published neural recording systems versus bit resolution is shown in Figure 4.24. There is reasonable agreement between the calculated NFI power limit and published measured results. Notably, the BioBolt [152] has very similar power consumption to the proposed model. The reported specifications are 1.3 µW/ch and f_s = 31.25 kS/s which results in a figure-of-merit (P/f_s) of 42 pJ. The estimated NFI power limit using the CBSC cyclic ADC with the optimized design parameters (BW, f_s, n) is 8 pJ which is about 5 times smaller. In addition, for n = 7 the P/f_s ratio of the noise-limited NFI using the CBSC cyclic ADC is about (1.52X) (33%) times lower than the NFI using the SAR ADC. The difference between the NFIs can be investigated through the noise-power contribution of the data converter. It is assumed that the reported NFI parameters in [152] are designed based on power bound analysis, so they can be used for scrutinizing the theoretical power dissipations difference between the NFI using SAR and CBSC data converters for a given resolution. In [152], for digitization purposes, a traditional SAR ADC with sampling frequency tunability is implemented. It should be noted that, the analysis of noise-power in this chapter covers different SAR ADC implementation methodologies considering (NMF=α) and efficient power modelling of data converters. Based on [152], the reported LNA IRN for the bandwidth of (0KHz -10Hz) is 5.62µVrms. This results in an α range of (0.13-0.4)

which means (13-40%) power difference if the aim is obtain a fixed amount of LNA IRN. For the resolution (n<8), which covers the optimized NFI required resolution (n=7 or even n<7), the data converter behaves like a digital block. So the power ratio ($P_{bound\,,\,ratio}$) between the considered data converters can be written as:

$$P_{bound\,,\,ratio} = \frac{P_{SAR,bound}}{P_{Cyclic,bound}} \approx \frac{P_{DLC,regeneration} + P_{register}}{P_{Coarse} + P_{TDC}} \approx \lambda \tag{4.72}$$

where λ is a constant term which represents the power bound ratio between the converters and a range is considered for λ (3-6). This range which applies to lower-limit (advanced SAR ADC) and upper-limit (traditional SAR ADC) is extracted based on running different simulations in

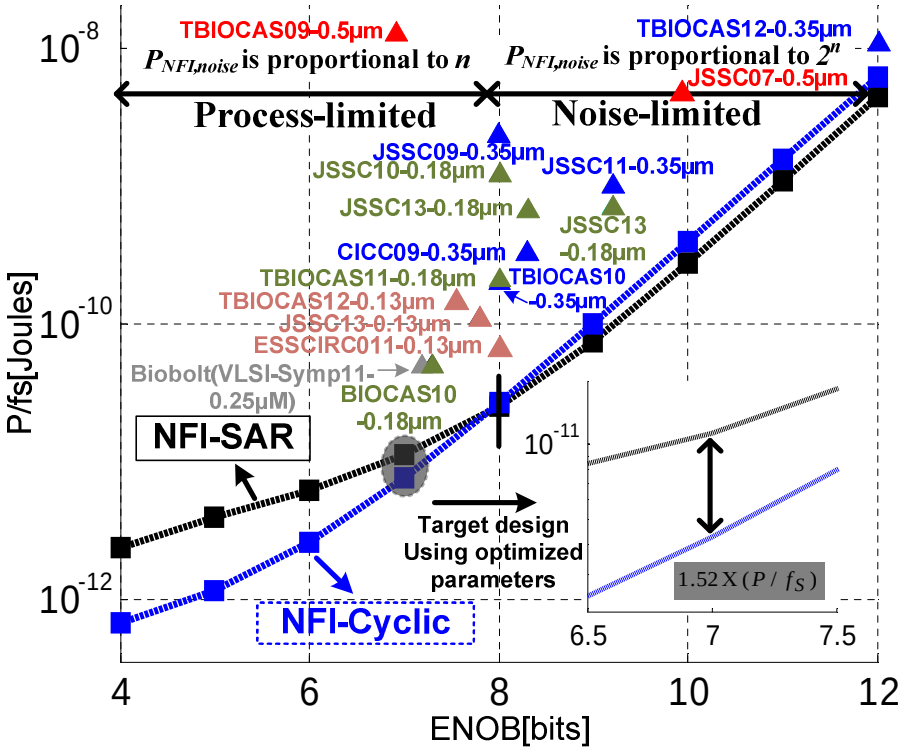

Figure 4.24: Estimated minimum power limit of NFIs based on SAR converter (dashed black line), CBSC cyclic ADC (dashed blue line) versus converter resolution. The following typical process parameters were used (180-nm CMOS): V_{FS} = 1 V, T = 300 K, V_{eff} = 180 mV and C_{min} = 5 fF. The NFI parameters were G_{LNA} =100 V/V, G_{PGA} = 10 V/V, NEF = 2 [7], $BW_{LNA} = f_{\text{LP}} - f_{\text{HP}}$ = 6.5 kHz and f_s = 30 kS/s. Data for published neural recording interfaces are also shown (▲) for comparison. α=0.5 is defined based on the reported $v_{ni,rms}$ =2.2 µV/Hz (0.5 Hz to 50 KHz in [131].

MATLAB. In the SAR ADC, for a resolution of $n \leq 8$, the ($P_{DLC,\,regeneration} + P_{register}$) dominates the total power consumption which is proportional to $Y.n^2C_{\min}$ where Y is a constant term which represents the other parameters contributing to the power bound of an SAR ADC. The overall minimum power limit of the CBSC cyclic ADC is proportional to $nC_{\min}$ (which shows the domination of the coarse current source). The CBSC operation is similar to a clocked-inverter[9] which charges/discharges a capacitor. This model is totally compatible with voltage scaling and technology scaling [153], [154] concepts. As reported in [152], the SAR ADC power consumption is 87.41nW which after dividing it by (λ =4), results in $P_{Cyclic,bound}$ =21.85nW.

The overall power ratio (*PR*) is:

$$PR\big|_{\alpha=0.13-0.4} = \frac{(\alpha+1).P_{LNA} + P_{PGA} + P_{SAR}}{P_{LNA} + P_{PGA} + \left(\frac{P_{SAR}}{\lambda}\right)} = (1.22-1.47)X \tag{4.73}$$

P_{PGA} is assumed to be same in both NFIs. According to the earlier discussion, there is good agreement between the observed power bound difference (1.52X) (33%) in Figure 4.24 and calculated theoretical minimum power range *PR* (1.22-1.47) X. In fact this type of digitization, radically improves noise–power product (NPP) which is demanded by recording and processing of 1K+ channels (see section 4.1). The NPP can be considered as a new figure of merit in design of biomedical NFIs.

4.4.4 Towards Parametric NFI Design

In previous sections, the noise-power effect of a data converter on NFI power bound has been studied. It also identified that using advanced techniques in the design of a data converter results in thermal noise minimization. This sub-section presents a brief explanation on the continuation of this research using the parametric NFI design. The block level system of the parametric NFI is shown in Figure 4.25. The digitized signal V_{tot} at the output of the ADC can be written as:

$$V_{tot} = V_{ndata} + V_{nnoise} + V_{LNA} + V_{PGA} + V_{ADC,thermal} + V_{ADC,qn} \tag{4.74}$$

where V_{ndata} is the neural input signal, V_{nnoise} is the pre-NFI system noise voltage (e.g., biological

[9] In ZCB gain stage, the operation is practically based on a clocked-inverter and it is estimated that in [153] the practical power conception is equal to theoretical power bound.

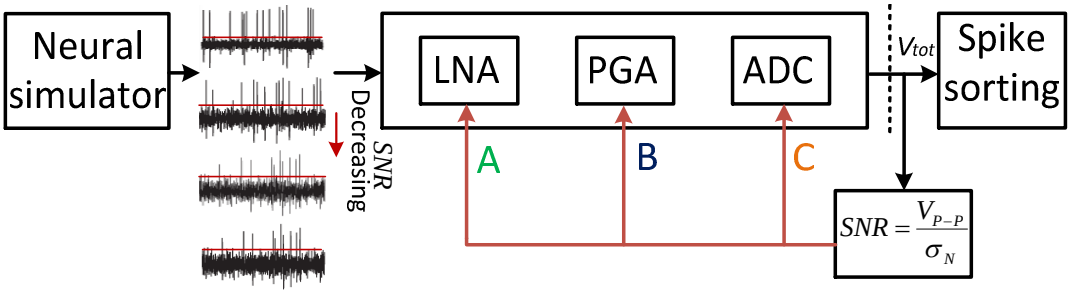

Figure 4.25: parametric NFI design. The system calibration is performed by recording channel *SNR*. A, B and C are the tuning parameters.

and biological noise), V_{LNA} is the LNA noise voltage, V_{PGA} is the PGA noise voltage, $V_{ADC,thermal}$ is the ADC thermal noise voltage and $V_{ADC,qn}$ is noise due to quantization. Removing the effect of PGA noise and thermal noise of the ADC (NMF/$\alpha \approx 0$) in Eq.(4.74):

$$V_{tot} = V_{ndata} + \overbrace{V_{nnoise} + V_{LNA}}^{\sigma_N} + \overbrace{V_{ADC,qn}}^{\sigma_Q} \tag{4.75}$$

The system noise (σ_N) and quantization noise (σ_Q) are assumed to be independent, making the total noise variance:

$$\sigma_{Ntotal}^2 = \sigma_N^2 + \sigma_Q^2 \tag{4.76}$$

In a parametric NFI design, the aim is to implement a calibration scheme via better estimation of the recording channel SNR ($SNR = \frac{V_{p-p}}{\sigma_N}$) as shown in Figure 4.25. Hence, it is desired to minimize the quantization noise to realize a single-parameter-dependent calibration method. The noise standard deviation (SD) of the recording channel (σ_N), is a reliable measure to perform the parametric tuning of NFI parameters as a result of reducing the effect of quantization noise (σ_Q). In most of the published systems, uniform sampling is used where the quantization steps are uniformly spaced and the bin size is ($V_{LSB} = \frac{V_{FS}}{2^n}$). In Figure 4.26(a), the distribution of quantization steps are uniform, so each quantization bin supplies the same amount of noise regardless of the

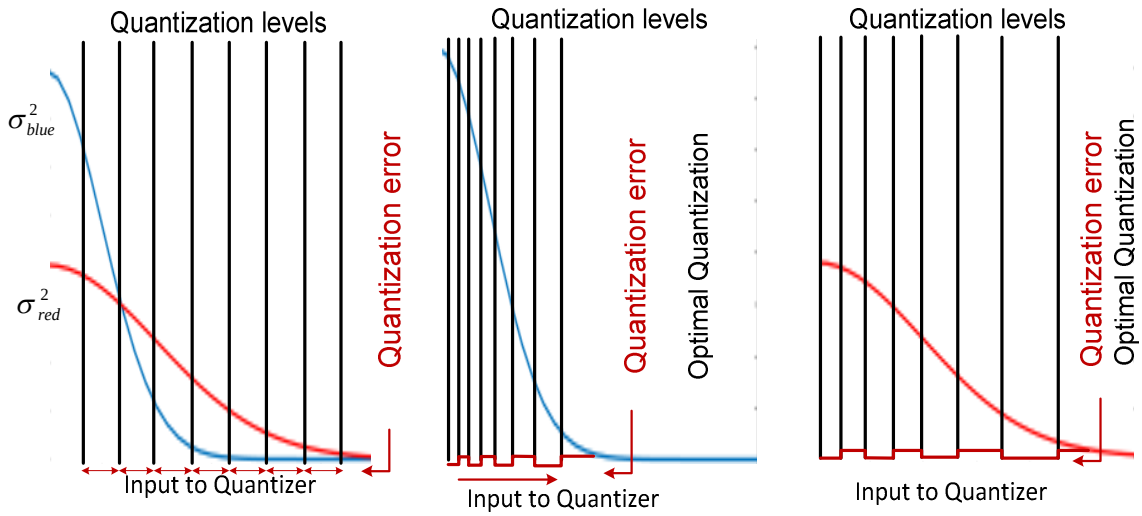

Figure 4.26: Examples of uniform and optimal quantization levels for narrow (σ^2_{blue}) and wide (σ^2_{red}) Gaussian pdfs. This curve reveals that the decision levels are densely located in high-probability region of the x-axis and coarsely in the low-probability region.

input probability density function (pdf) of neural data to the total quantization noise. On the other hand, the optimal quantizer [155] minimizes the quantization error (σ_Q) for each Gaussian pdf by more bins in higher probability regions and fewer bins in lower-probability regions Figure 4.26(b). It can be seen that the uniform quantizer reduces the quantization performance when there is a pdf/variance mismatch. Two important factors as a result of using the optimal quantization will be the future focus of this research:

1- Optimal quantization (*OQ*) and power-clustering performance: The application of quantization noise shaping provides the same quantization performance with lower resolution (*n*) by modifying the signal to quantization noise (SNQR). The effect of optimal quantization (*OQ*) on the performance of spike sorting has been observed. Evaluation of detection and sorting accuracy show that an optimal quantizer saves 2-3bits to resolve the signal for the same degree of sorting accuracy compared to uniform quantizer. *OQ* has the capability of noise shaping, and improves the signal to noise ratio which results in fewer resolution bits. Consequently it saves power and area in designing the NFI and spike processor.

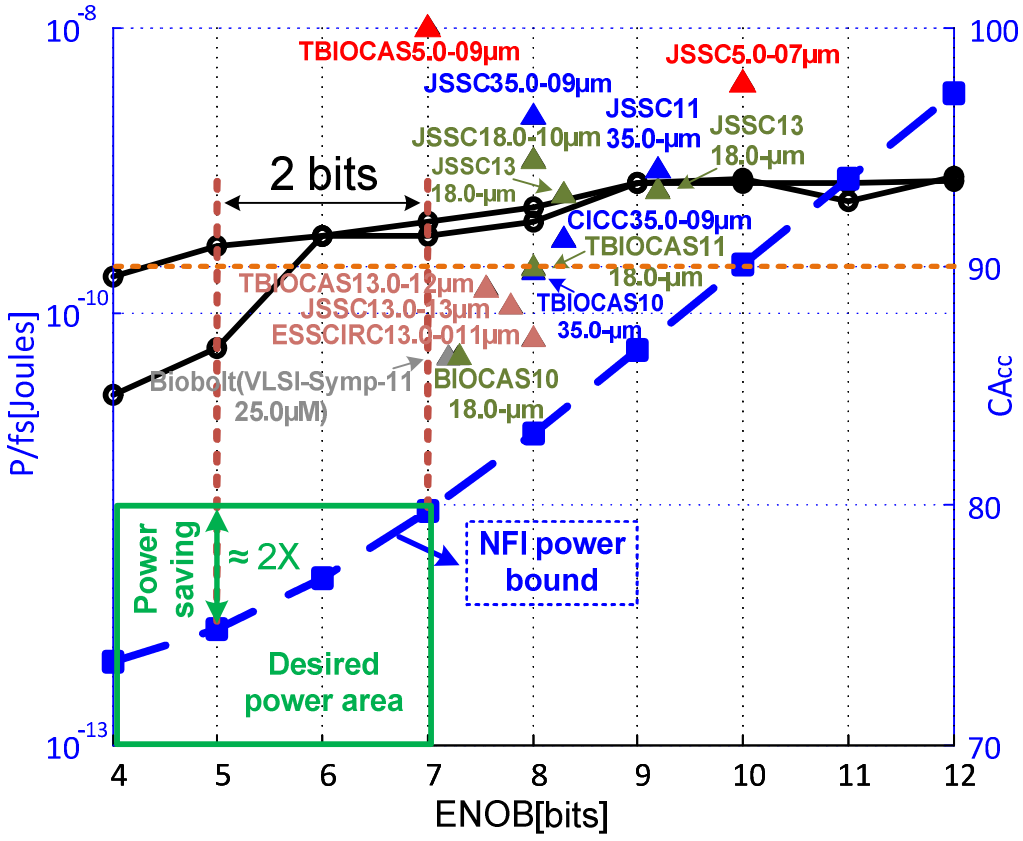

Figure 4.27: Power (blue)-CA_{CC} (black) analysis when *OQ* is used. Using *OQ* saves 2-3 bits in digitization. Desired power-area is illustrated with green square.

2- Optimal quantization (*OQ*) and NFI parametric design: As σ_Q^2 decreases, the total noise variance approaches σ_N^2, then the Eq.(4.76) is re-written as:

$$\sigma_{Ntotal}^2 \approx \sigma_N^2 \tag{4.77}$$

where, $\sigma_N^2 = \sigma_{nnoise}^2 + \sigma_{LNA}^2$. If σ_{nnoise}^2 is considered to be equal to σ_{LNA}^2, then:

$$\sigma_{Ntotal} \approx 1.4\sigma_{nnoise} \tag{4.78}$$

σ_{Ntotal} is a an accurate estimation for determining SNR in NFI parametric modelling. In the parametric model, there is potential to design an ultra-low power NFI (P_{NFI} < 500nW) compared to state-of-the-art [156], [157]. This would allow not only monitoring of >1K channels but also additional digital processing would be possible on the recorded data.

4.5 Conclusion

Future implantable devices which may have many hundreds of recording channels, involve increasingly complex processing, requiring power whose availability is very limited. This chapter has identified the minimum power limits that can be attained by considering fundamental concepts in the design of the NFI containing a low noise amplifier (LNA), a programmable gain amplifier (PGA) and an analog-to-digital converter (ADC) followed by a spike sorting processor. For the LNA the optimal low pass and high pass cutoff frequencies have been found to be about 500 Hz and 7 kHz, respectively. Examination of the lower power limit of NFIs has shown that the CBSC cyclic ADC is up to 30% more energy-efficient than the conventional SAR ADC. A power efficient spike sorting processor has been implemented on a FPGA capable of online and unsupervised clustering which has been used to identify the precise definition of NFI design parameters. The results have shown that 7-bit processing is sufficient for efficient spike sorting. The spike sorting process is robust to the ADC resolution but clustering accuracy is quite sensitive to the sampling rate. Analysis of the NFI to identify the optimized key parameters has suggested that the acquisition system has a minimum power limit set by spike sorting processing constraints rather than noise limitation. As discussed in section 4.2.2, there is significant potential for power saving by implementing an efficient management/calibration system in the using parametric NFI. The parametric power optimization methodology proposed provides a guide toward the design of future ultra-low power and extremely efficient acquisition methods.

CHAPTER 5

Biomedical Signal Processing Using Adaptive Techniques

5.1 Introduction

This chapter introduces a complementary processing framework to the non-adaptive classical synchronous processing systems (SPSs). This type of processor realizes an adaptive paradigm which in functionality or structure of the processor tailors itself via the characteristics of the input data and the defined processing frames. From the viewpoint of system level design, the proposed processing scheme combines three well-known processing approaches synchronous, asynchronous and self-tuning in the adaptive paradigm. As a result, unlike the SPS, which has been used for many years due to their simple realization, the adaptive processor's operation becomes signal dependent. The monolithic structure of SPSs does not allow the designers to implement a reconfigurable processing system which is robust to the input signal variations (e.g. signal noise standard deviation (σ_N)). At system level, any adaptive processor can be broken into two main parts: firstly the main processing path which utilizes the classical synchronous system (no ability to sense the signal characteristics) which operates with a master clock, and secondly the embedded processing frames (also referred to as modules or layers) which are defined based on the targeted application. Basically, the adaptive processing frames sense the input signal statistics and shape the processing path for maintaining optimal performance. Hence, the whole system is able to operate in two distinct processing modes: 1) synchronous processing and 2) adaptive processing mode. In summary, this chapter delivers a new class of a well-suited processing platform for biomedical applications, particularly for spike sorting. This chapter is organized as follows:

Section 5.2 focuses on developing embedding adaptive frames for spike sorting methodology developed in Chapter3 by characterizing the recorded neural data. Extraction of these frames are developed by the concept of re-definition of the spike processing through reverse-adjustment (RA). This section also compares the properties for this class of processing with the SPS. Section 5.3 provides detailed description of a complete design methodology for an implantable, unsupervised, on-chip and adaptive spike sorting processor. The circuit techniques for inserting the developed frames to the spike processor are described. In addition to the adaptive frames the choices on

architectures and circuit techniques regarding a multi-aspect optimization methodology (accuracy-power-area) are enumerated. The spike processor implementation details and measurement results are presented and compared with the state-of-the-art in Section 5.4. This section also has a brief description of the next generation of the proposed adaptive spike processors. Section 5.5 summarizes and concludes the chapter.

5.2 System-Level Illustration of Adaptive Spike Sorting

5.2.1 Development of Adaptive Frames for the Spike Sorting Application

Adaptive spike processing is introduced in this section as an advantageous alternative to the classical spike sorting systems. This section discusses the motivation behind the selection of embedding frames for spike sorting application to maintain the optimal clustering performance under any conditions. Hence the processing outcome is independent from the input signal. As explained in Chapter 3, the two key factors in spike sorting performance degradation are the noise of recorded data and the similarity index between the spike waveforms. The aim is to develop a spike processor in which the performance is adjusted to an optimal level (maintaining lowest clustering error) with the varying difficulty between the recorded spike waveforms in recording channel and different noise levels.

The concept of the frame extraction procedure is shown in Figure 5.1. Figure 5.1(a) is the block diagram of a traditional spike processor whose performance varies as a function of noise (*f*(noise)) and similarity of extracted spikes (*f*(similarity)). Figure 5.1(b) shows the spike sorting concept developed with added reverse-adjustment (RA) flow. In this class of spike processing, it is required that the resulting clustering performance (CA_{CC}) is independent ($\perp$) of noise and spike shape similarity between the detected spikes. As a result, the processing frames are developed based on the RA concept in spike sorting as shown in Figure 5.1(c). Adding the created frames (Frame1 and Frame2) to the traditional spike processor presents a fundamentally new approach for mapping the recorded spikes to the individual neurons. The newly created frames provide the functions of *adaptivity* or *reconfigurability* by in the recording channel distinguishing the noise property of neural data stream (Frame1) and the similarity between active neurons (Frame2) to maintain the

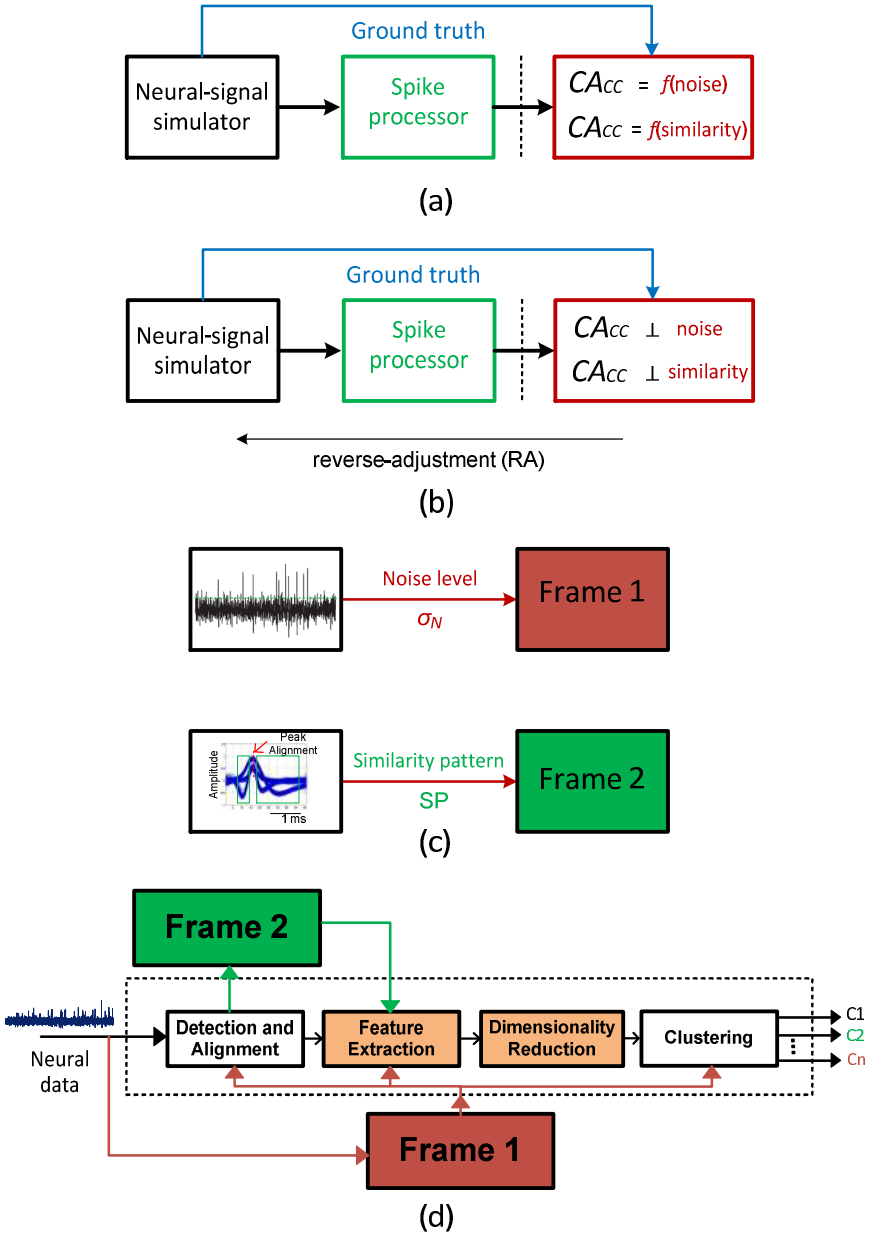

Figure 5.1: (a) Traditional spike sorting which in the performance is the function of noise (*f*(noise)) and similarity (*f*(similarity)). (b) Illustration of spike processor independent (⊥) of recorded data noise and similarity of spike waveforms. In this class of processing it is expected that the CAcc is matched with ground truth and at any condition set in neural signal simulator (c) Abstract view of mapping the proposed reverse-adjusted spike processor characteristics into the frames (Frame1 and Frame2) for implementing adaptive concept. Frame 1 increases the processor noise robustness while Frame 2 is adapts the similarity level between the extracted spike waveforms. (d) Transformation of developed spike processor in Chapter3 based on developed frames (Frame 1 and Frame 2). The created frames are embedded to the main processing line.

clustering performance at an optimal level. Figure 5.1(d) shows the two frames added to the spike sorting system previously described in Chapter 3. The adaptive processing provides an on-chip tuning mechanism for programming the key coefficients in the relevant building blocks. As noted earlier, this class of processing uses the design concepts from online, self-tuning, synchronous and asynchronous domains.

5.2.2 Transforming the Defined Adaptive Frames into Processing Units

As discussed in previous section it is desired to implement spike processor effectively independent of the main issues causing errors in the clustering performance namely noise variations in recording channel and the similarity measure between the existing spike waveforms (Figure 5.1(c)). The previously explained frames are translated into mathematic models in the appropriate circuit blocks. Frame 1 provides noise robustness to the processing chain by modelling the noise standard deviation of the recorded neural signal (σ_N) [10]. This value can be calculated by the median processing of the recorded neural data. Frame 2 provides similarity robustness by modelling with the localized difference extraction of the aligned spikes which is explained in 2.5.4. The similarity pattern extractor (SP) unit is initiated intermittently to update the similarity information of the existing spike waveforms in the recording channel. The (SP) is sent to the *FE* unit to synthesize the behavioural analysis of the extracted local differences (amount of dilation/contraction) for activation of the decomposition lines. Providing similarity robustness to the spike sorting chain.

5.2.3 Overall System Description

This section illustrates the system level implementation of an adaptive spike processor based on the concept detailed in previous section 5.2.1 and highlights the resulting properties Figure 5.2 shows the adaptive spike sorting diagram. The created adaptive frames are integrated into the sorting system. Frame 1 records the noise variance (σ_N) and defines the ($SThr=4*\sigma_N$) [89] which is distributed to the detection block, adaptive *FE* block and the sorting threshold look-up-table (*SThr* LUT) block. Frame 2 is developed for the SP unit in the adaptive *FE* block. SP is a generated similarity pattern generated between the aligned spikes. The addition of the two frames in the synchronous sorting system introduces an *adaptive architecture* to allow realization of a highly

[10] σ_N is calculated by median processor.

adaptive on-chip spike sorting with *on-chip parametric tunability*[11] which is can simply be added to the neural acquisition systems. In the spike-sorting process, the amplified, band-pass filtered and digitized neural data is fed to the spike processor. Normally the recorded signal contains multi-unit activity from all the neurons close to the implanted microelectrodes.

Each spike is extracted using a 2.5ms window and aligned to a common temporal reference. In the detection unit (which uses non-linear energy operator (NEO) [158]) a *conditional activation* (On-hold capability) method is utilized. The calculated threshold ($SThr=4*\sigma_N$) after median estimation is used to control the main detection block. With this method, the processing of worthless data is masked in the main detection block and the detection initiation is a function of input spike stream. Thus the power dissipation of NEO and the spike processor can be significantly reduced through asynchronous initiation of processing chain. So the processor operates at normal speed once the process starts and returns to standby mode when processor does not detect any spike activity. This capability helps to minimize power consumption without applying hand-shaking or power gating techniques.

In the adaptive FE block, extrema sampling of *adaptive discrete derivatives (ADDs)* is introduced to provide an efficient method not only in computational simplicity but also in accuracy to transform the recorded action potentials to a feature space that better separates spikes coming from

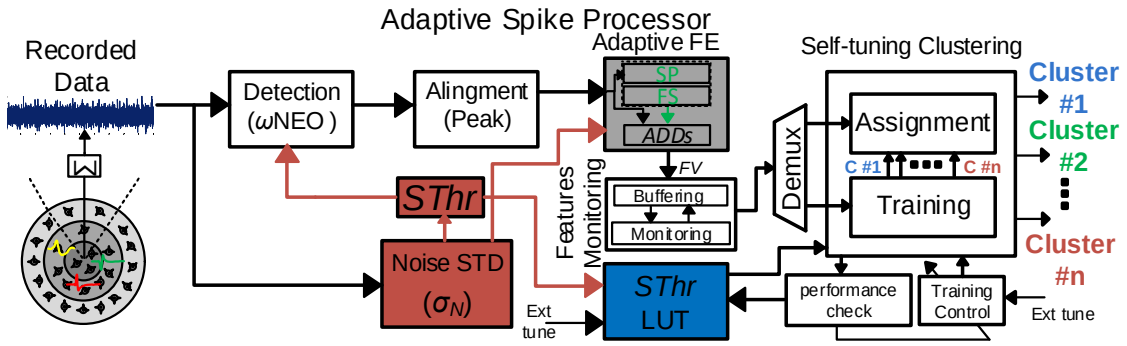

Figure 5.2: Adaptive spike sorting block diagram. The created frames in Figure 5.1(Frame 1 and Frame 2) are embedded in the spike processor as (Frame1= σ_N) and (Frame 2= SP). The introduced adaptive processor scheme obeys the built-in principals for on-chip tuning of sorting parameters. n=1,…,6 represents the number of existing clusters in the recorded data.

[11] This is also called processor with open input format. The processor is insensitive to the input signal variations. The spike processor is robust to the general spike sorting challenges [91].

different neurons. The selective spike decomposition is performed by generating a similarity pattern based on the aligned spike waveforms (SP). The (SP) is updated over time to monitor the similarity level between the extracted and peak-aligned spikes. The *FE* is tuned to the sub-bands (decomposition range) with the most informative samples. The maximum separation between the spikes is achieved by extrema sampling of selected sub-bands. The feature vectors (*FVs*) are sent to the features monitoring unit and subsequently to the clustering unit.

Finally, the *FVs* are sent to the clustering unit for training. The training engine is the most power consuming part of clustering due to its inherent complexity and the size of memory required for implementation. The training unit power dissipation is minimized using an *advanced power management methodology*: 1) using a unique register bank memory structure for built-in computation power suppression (self-power management), 2) choosing interleaving/logic-reusing topologies to obtain high level of integration in the training engine. It reduces the power-area simultaneously and 3) using feature extraction to reduce the feature space dimensionality. The feature space dimensionality (*K*) of *ADDs* is almost 8X lower than the state-of-the-art. The cluster means identified in this training phase are saved to the memory of the assignment unit. During the cluster-mapping phase, the input *FVs* are mapped based on their minimum distance to one of the identified cluster means saved during the training phase.

The output of the assignment block can be configured to the detected spikes, cluster means, or the spike IDs. As shown in Figure 5.2, two controlling units (performance check, training control and *SThr* LUT) are added to the clustering block to implement clustering performance monitoring in *validation phase* which is discussed in section 5.3.7. Having clustered the spikes, the performance check unit monitors and evaluates the clustered data based on the defined performance metrics. It decides whether the level if of the sorting threshold should be adjusted to an optimal level (T_{opt}) [12] and also triggers retraining to compute the cluster means if needed. To realize the validation phase, a look up table (LUT) based threshold adjustment scheme is designed within the online sorting (O-Sort) clustering method. It uses the noise analysis of recorded data and evaluation of the clustered spikes to obtain the *self-performance lock* capability recursively. In performance check monitoring if the conditions are met, the identified cluster means are not substituted with new patterns in the assignment unit unless the enable signal initiates the training phase due to either

[12] Deriving optimal sorting threshold is discussed in Chapter3.

clustering inaccuracy issues or variation in the action potentials. The sorting threshold (*SThr*) can alternatively be overridden by user-specified values.

This concept enables the implementation of fully online spike sorting accounting for the characteristics of the recorded signal and high clustering accuracy performance. The concept of adaptive processing is also extendable to all processor constituent building blocks, which would be embedded in the next generation of adaptive sorting DSP. In the next sections, design of all spike processor units and their operation are discussed in detail.

5.3 Adaptive Spike Sorting Processor for Accuracy Self-tuning and Inherent Power Suppression

The rest of this chapter provides the hardware implementation of an adaptive spike processor with the new processing framework and highlighted properties. The spike processor fabricated in a 180-nm CMOS process, it has a power consumption of 148 µW from 1.8 V supply voltage, and a 84.5% overall clustering accuracy (CA_{CC}). By providing 150/240X data reduction, a transmitter can also simply be integrated to the front-end and processing unit in implantation site.

5.3.1 Detection and Alignment

One unsupervised method to determine threshold value is the nonlinear energy operator (NEO) [158]. This class of detector calculates the energy variation of the raw signal to interpret the spike events in time. For the discrete signal, the NEO equation is defined as:

$$\psi(n) = x^2(n) - x(n+1).x(n-1) \tag{5.1}$$

where $x(n)$ is the input digitized signal and $\psi(n)$ is the NEO value at sampling time n. From the NEO equation, this operator highlights the large variations in power and frequency. The characteristic of spike activity is instantaneous amplitude-energy variation, and the NEO operator emphasises these localized variations. NEO significantly improves the signal to noise ratio (SNR) level in a noisy environment.

However, NEO is sensitive to the poor detection of spikes having low frequency components. To overcome this issue and increase robustness to the spike amplitude variations or to reduce NEO out of band noise sensitivity, a modified version of NEO is used by adding the constant value ω to

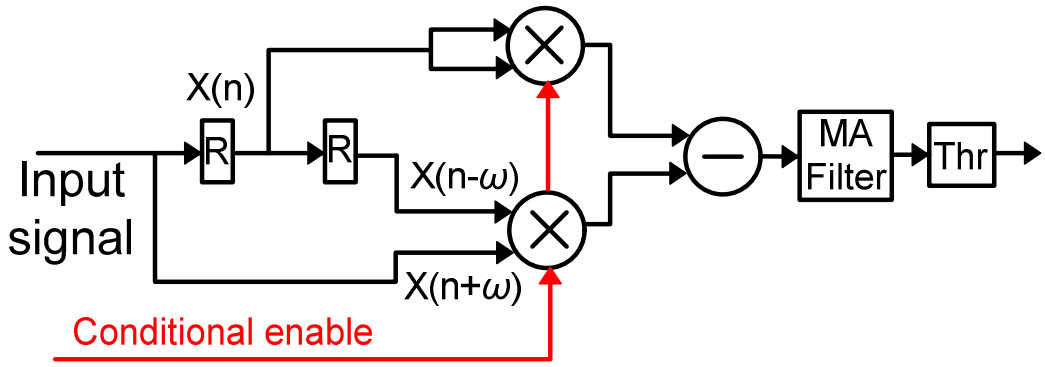

Figure 5.3: ωNEO conditional control. Conditional enable is initiated by (*SThr*).

Eq. (5.1) which becomes $\psi(n) = x^2(n) - x(n+\omega).x(n-\omega)$, where ω is experimentally found to be between 1 and 3.

In the spike sorting process, the noise variance calculation is needed to set the clustering threshold using absolute value method. To obtain high detection accuracy while keeping the power consumption low, a complementary approach is used in which the clustering threshold can be used as conditional activation function of the ωNEO. This method has two advantages: 1) conditional enabling is directly applied to the ωNEO which is composed of two multipliers. Thus when the input exceeds the clustering threshold (*SThr*), the true identity of the spikes is examined using ωNEO which provides a double check on accuracy. And 2) power saving due to dual-thresholding which results 30% power reduction based on Cadence synthesis simulation. The block diagram of the system is depicted in Figure 5.3.

The conventional method used for threshold calculation at the output of ωNEO is energy accumulation and dividing it to the window sample numbers. The power variations in different simulations show that the output of the ωNEO is sensitive to noise disturbances. The output of ωNEO can be influenced by noise variations. Normally the input signal to the ωNEO unit is composed of spike events which exhibit localized energy of a specific duration and other samples as a result of noise interference. The noise perturbations are projected in the calculated energy, and a simple filter can be added before the threshold calculation to reduce the effect of noise

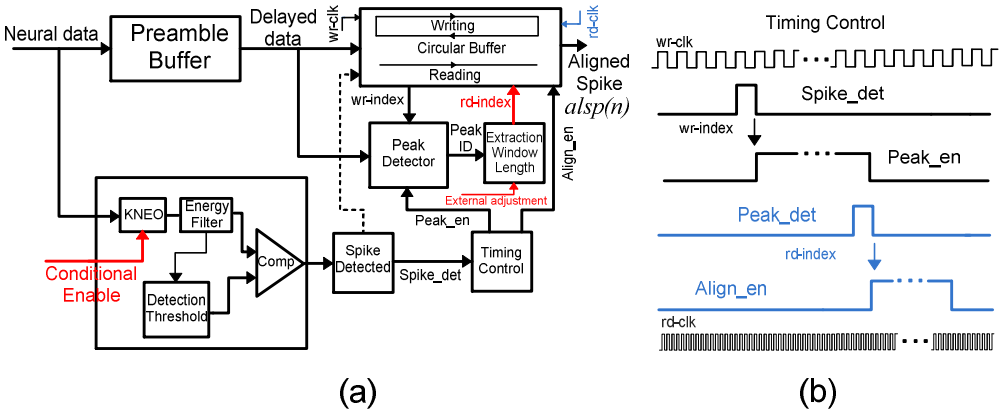

Figure 5.4: (a) Detection and alignment block diagram. (b) Operation timing diagram.

A simple moving-average filter can be applied to $\psi(n)$ to reduce the effect of projected noise. The filtered signal energy $\lambda(n)$ is then used for detection threshold calculation. The approach to set the detection threshold is a scaled version of the filtered energy that can be expressed as:

$$Thr = \alpha \frac{1}{N} \sum_{n=1}^{N_S} \lambda(n) \tag{5.2}$$

Where α is a constant, empirically chosen to be 8 in this implementation. In order to reduce the buffering of threshold calculation, the detection threshold is updated per window rather than per sample where N is the number of samples per window. The calculated threshold is used for the next segment of data; the accumulator will be reset and start again for the next data window.

Figure 5.4 shows the detection and alignment block diagram. The neural data to the detection block is fed to both a preamble buffer and ωNEO. The preamble buffer is a shift register composed of 24 delay cells. The purpose of using delay line is to synchronize the ωNEO output with the starting point of a spike. It buffers the samples before the spike exceeds the threshold level. The delayed data is constantly written to a circular buffer. When a spike is detected, the corresponding writing index (wr-idex) is sent to the peak detection block. With this method, the sample counting and the peak address are synchronized. The output of the peak detector (peak-ID) is used to define the extraction window length. The peak-ID serves as offset to correspond to the 15th sample of the 45 samples in the aligned window. Finally the rd-index which represents the first sample of

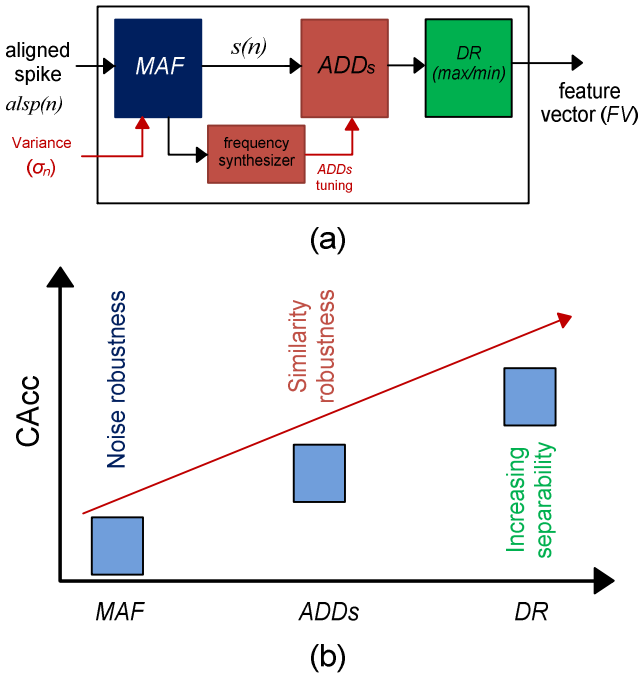

Figure 5.5: Adaptive Feature extraction unit. The *MAF* suppresses the effect of random and high frequency noise. The *ADDs* and frequency synthesizer blocks provide the adaptive decomposition. The *DR* block reduces dimensionality by retaining the most informative features of decomposed spike waveforms. Extrema (max/min) sampling is used in *DR* unit. It should be noted that other *DR* methods also can be employed for distinguishing the most appropriate features in the selected sub-bands (e.g. integral of a selected sub-band or spectral analysis of a decomposed signal).

window is set to the reading unit of circular buffer and aligned spike (*alsp(n)*) samples are transferred to the next processing unit which is feature extraction. The reading clock rate is 4x faster writing clock rate to ensure capturing spikes which are close in time.

5.3.2 Adaptive Feature Extraction

Feature extraction (*FE*) transforms the aligned spikes to a low-dimensional space which emphasizes the spike waveform differences. The block diagram of the modified and adaptive version of *FE* [150] is shown in Figure 5.5. The *FE* block consists of three main units namely a moving average filtering (*MAF*), adaptive discrete derivatives (*ADDs*) and dimensionality reduction (*DR*) unit. The *MAF* acts as a denoising filter to improve the *FE* robustness to random

noise (out-of-band noise) while retaining the crucial encoded information buried in spikes [13]. The output from Frame 1 (σ_N) is fed to MAF to increase the noise insensitivity and increase feature extraction separability by adjusting the length of *MAF*. The *MAF* is the most popular choice for hardware implantable devices due to its simplicity and efficiency. The *MAF* concept is based on averaging a specific number of samples of incoming aligned spike *alsp(n)* to produce the smoothed output signal *s(n)*:

$$s(n) = \frac{1}{M}\sum_{j=0}^{M-1} alsp\,(n-j) \tag{5.3}$$

where *M* is the filter length. The next block of the adaptive *FE*, are the adaptive discrete derivatives (*ADDs*) which calculate the slope at each sample point over a number of different time scales:

$$ADDs = amp[s(n) - s(n-\delta)] \tag{5.4}$$

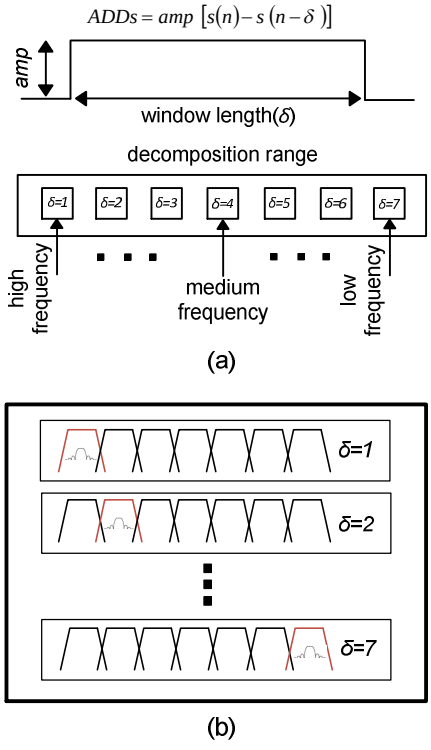

Figure 5.6: Demonstration of feature extraction processor employing spectral analysis a) parameterized *ADDs* (*amp*=1) and b) illustration of *ADDs* as an adaptive filtering.

[13]The MAF length is adjusted by the signal-to-noise (SNR).

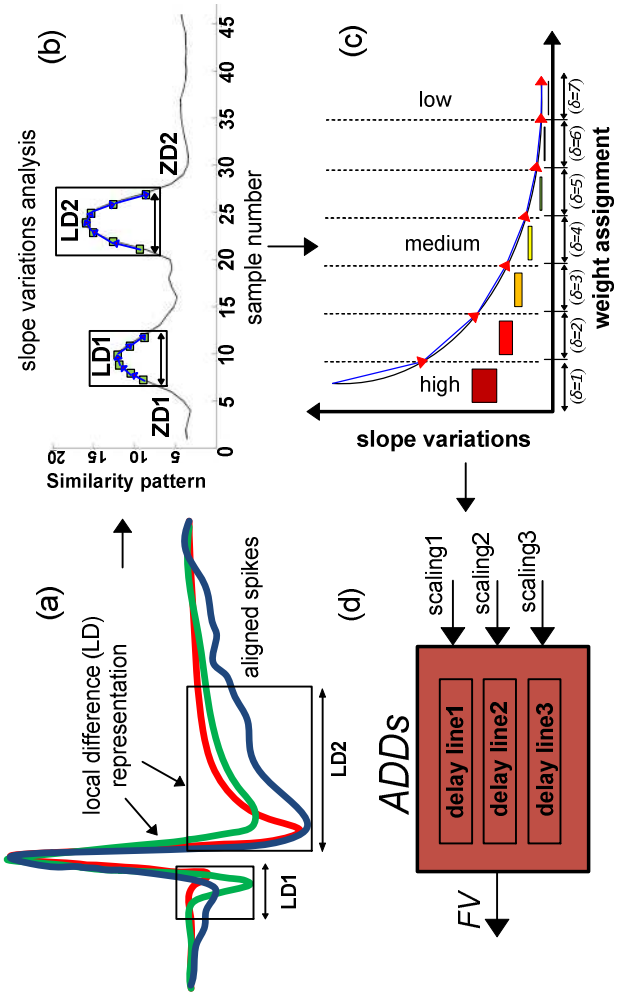

Figure 5.7: Frequency synthesizer and scaling factor tuning. (a) Peak aligned mean waveforms in C_Difficult1. Local differences are demonstrated with LD1 and LD2. (b) Generated sampling pattern from accumulated local differences. ZD1,2 identify the zero differences. (c) Weight assignment to the variations in the generated sampling pattern and extracting the scaling factors corresponding to the largest weights. The decomposition intensity is depicted by different colours from high (red) to low (black) (d) Tuning delay lines based on the selected scaling factors. *FV* is extracted based on the extrema of the delay lines (delay line1-3) outputs.

Where $(amp=1)$[14] is the amplitude of the decomposition window, s is the spike waveform, n is the sample point, δ is the scaling factor (time delay) and amp depicts the amplitude of *ADDs*. The

[14] *amp* can be used for weighted adaptive discrete derivatives (*WADDs*) decomposition implementation.

equation shows subtraction between the samples n and (n–δ). Multi-resolution decomposition of a spike can be obtained if the scaling factor (δ) is swept over a wide range as demonstrated in Figure 5.6. Here the adaptive discrete derivatives (*ADDs*) is proposed to retain features of focused decomposition from specific sub-bands for the succeeding clustering stage. In *ADDs*, the window length (δ) is not fixed and will be tuned based on the recorded spike waveforms over time. Adjusting the translation parameter (δ) is based on the frequency bands which correspond to the most useful information for clustering. To obtain high accurate selectivity in setting the translation parameter (δ), a sampling pattern generation method explained in [100][15] is used. Three useful factors can be observed from the generated sampling pattern based on the localized shape differences between the aligned spikes: 1) similarity (dissimilarity) measurement, 2) localized differences and 3) active frequency sub-bands in the generated sampling pattern.

The asynchronous sampling pattern generation filters out the non-useful frequency bands before applying the *FE*. Having generated the sampling pattern, slope variations analysis is accomplished to characterize and assign weight to the range of variations from high (δ==*1*) to low (δ==*7*). Once the weight allocation process is performed, three scaling factors with the highest weights are chosen for tuning the *ADDs*. The process is shown in Figure 5.7. The illustrated figure explains a simple approach of frequency synthesizer implementation for sub-band selection over a wide range

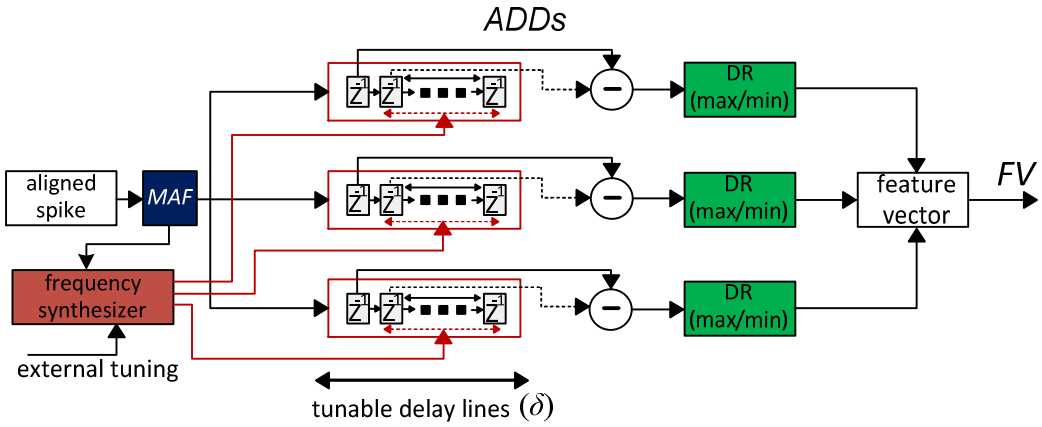

Figure 5.8: *ADDs* hardware implementation block diagram. Selecting the number of decomposition lines will be investigated further in future work. In this analysis three decomposition lines are used and six feature (K=6) are selected for clustering.

[15]This method is explained in Chapter2, section 2.3.5.

of frequencies which is used in the feature extraction block. The hardware implementation of *ADDs* is depicted in Figure 5.8. The main units are adjustable delay lines, subtractions and *DR* unit which performs extrema selection. The proposed *FE* is quite flexible in terms of frequency band selection and extraction of wide range of features. These processes result in robustness to spike similarity and noise level in the *FE* unit.

5.3.3 Feature Vector Monitoring

The captured features are fed into the next stage for buffering as illustrated in Figure 5.9. When the *FE*_done is set, the writing control block generates the address and enable signal for storing the incoming *FV* from *FE* unit (*FV*_in) to the memory (writing unit). A monitoring unit is deployed to constantly check the new incoming *FVs*. Based on the status flags (*write_status* and *read-status*) of the buffering unit the feature vector monitoring unit decides whether to pick new *FV* for training (assignment) or needs to scan the line for new *FV* (*FV*_in) in waiting mode. This approach allows the clustering unit to be put in stand-by mode during the *FV* intervals. When the *FV* is detected, the *read_flag* is set to one for activation of the reading control block. Once the reading of *FV* has been completed, the monitoring unit will be activated to repeat the process.

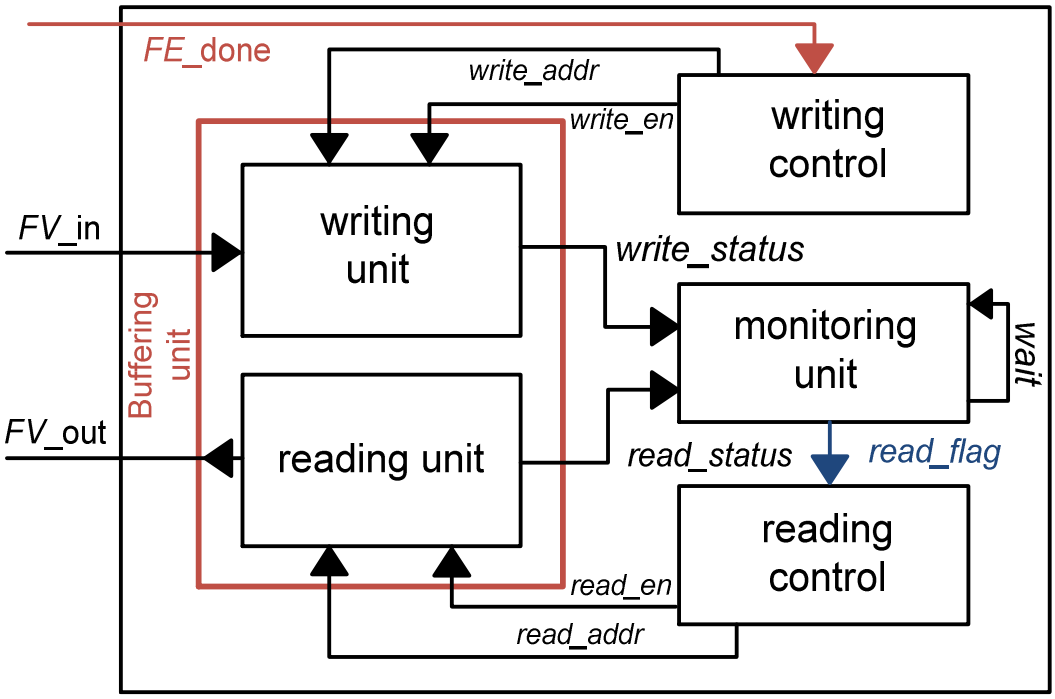

Figure 5.9: Feature vector monitoring unit.

5.3.4 Online Sorting (O-Sort)

Real-time identification of spike waveforms is required, to scrutinize the functionality of neural circuits. The algorithm in [13] is well suited for real-time neuron mapping. The method is called O-Sort which it provides the possibility of tracking neuron activity and closed-loop experiments. Although O-Sort is a fully online and unsupervised method, the average clustering accuracy is lower than the state-of-the-art clustering methods such as K-means [91]. A disadvantage of O-Sort is that it may split clusters into sub-clusters leading to a reduction in clustering performance. The created sub-clusters are not matched with any source and are considered as noise clusters. Therefore the aim is to modify the hardware implementation of O-Sort to achieve acceptable accuracy. Commonly known terms in O-Sort, namely, cluster splitting and artificial clustering, are used to implement fine tuning of the sorting threshold in the feedback loop. To keep the power consumption of O-Sort low, the core structure of the training unit, the transient memory, is designed in way that the status of memory locations are updated after each iteration in the training phase. Using this method, the unwanted memory locations are excluded from the next iteration phase. In addition to this unique and adaptive approach, other optimisation techniques to reduce power and chip area will also be examined. Another contributing factor in power and chip area reduction is the dimensionality of feature vectors (K=6) used for training which has almost 8X lower dimensionality compared with [76]. In summary, a modified version of O-Sort to reduce power consumption and chip area is presented. Figure 5.10 shows the flow chart of the O-Sort algorithm.

5.3.4.1 Unsupervised Clustering Using O-Sort

The training phase of the algorithm is initiated by assigning the first *FV* to its own cluster. The extracted features of the next spike are used to calculate the distance to the already created cluster. The distance between the *FV(n)* and created cluster *c(n)* is computed using the $\ell 1$-norm metric for the number of samples (N_S) representing the number of features in *FV(n)*:

$$C_{\min} = \sum_{n=1}^{N_S} \left| FV(n) - c(n) \right| \tag{5.5}$$

If the minimum distance at this stage (C_{min}) is smaller than the sorting threshold (*SThr*), the *FV* is assigned to existing cluster and the cluster mean is updated to be the weighted average of the first

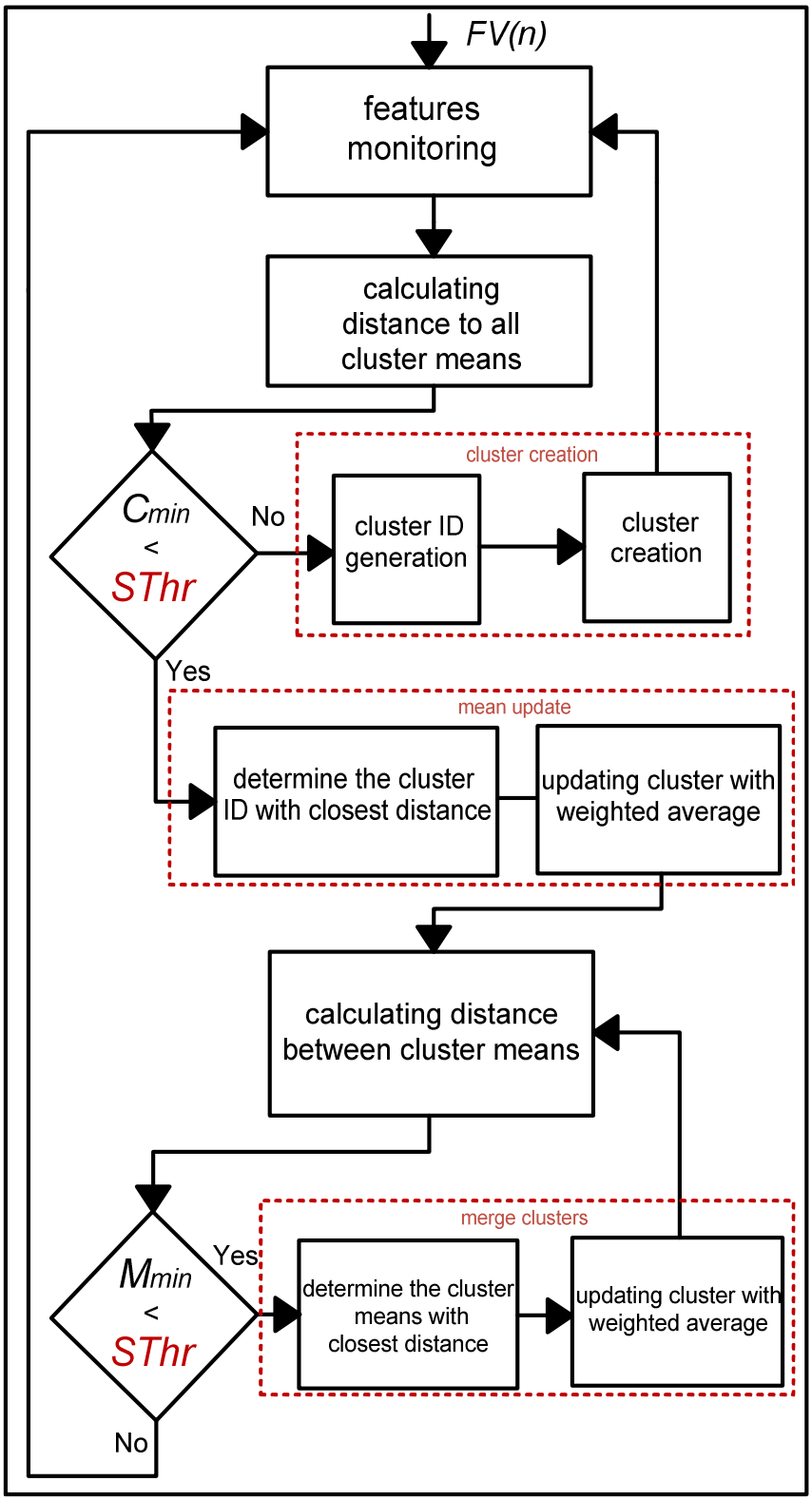

Figure 5.10: Flowchart for unsupervised clustering algorithm (O-Sort).

two spikes, otherwise a new cluster is automatically created and the *FV* is assigned to it. The *SThr* is defined as $SThr=4*\sigma_N$ where σ_N is the noise standard deviation of the data computed intermittently. The *SThr* is automatically and continuously calculated in median processor unit. This process is performed recursively for all the incoming *FVs* from feature monitoring unit. Every time a *FV* is assigned to an existing cluster, the cluster centroid is recomputed as the weighted average of the *FV* and the existing transient cluster is given by:

$$C_{Update}(n) = \frac{W \times c(n) + FV(n)}{W + 1} \tag{5.6}$$

where *W* is the number of *FVs* in a specific cluster. The updating scheme allows the tracking of changes in the incoming data and adaptively modify calibration of the system without user intervention. As interpreted from Eq.(5.6), during the training phase the *W* term becomes more significant compared to the second term. This means that after sufficient training time the contribution of the incoming *FV* is insignificant in the weighted average. This point is used in the hardware implementation to avoid over-processing for the clusters with (W>40). It is used as a condition to bypass the mean update and consequently merging check for the clusters with enough members.

During each iteration, after assignment and the mean update phase, a shift in the updated mean waveform over the feature space is expected which might cause overlap between the cluster means. After each mean update, the mutual distance between all possible pair of means is calculated. If the distance between any pair is less than (M_{min}< *SThr*), clusters are indistinguishable and are merged. The merging phase reassigns the weighted mean update to one cluster and eliminates the content of another cluster. The process of sorting is based on sequential comparisons between the mean waveforms thus after a specific training phase, the main cluster mean eventually starts converging.

O-Sort is simple in operation with good complexity-accuracy trade off and satisfies online sorting constraints which adaptively track the incoming spike data and autonomously determines the number of spike classes. This algorithm is adaptive and nonstationarity of data in time is applied to the cluster position and number of clusters.

5.3.4.2 Clustering Operation Methodology

A single-channel clustering timing diagram is shown in Figure 5.11. The clustering operation is divided into four sub-phases: hold time, training phase, validation phase and finally assignment phase. The clustering execution is starts with the hold time. In hold time which starts at (t_0) the median processor starts calculating the noise standard deviation (σ_N). As explained in the processor general structure, (σ_N) is used to calculate the threshold value for detection and sorting threshold look up table (LUT) units. The training begins at (t_1) when the threshold has been identified. The (*SThr*) is fed to the sorting LUT as the initial value for training phase. Over the training phase, the *FV* monitoring unit sends the *FVs* to the training unit to identify the number of active neurons in the recorded data. The validation phase (t_2) is initiated after training to evaluate the mapped data to the converged cluster means. In this phase the performance check unit is used to adaptively fine-tune the (*SThr*) for increasing the clustering performance which is explained later. Once the validation is performed (maximum of three iterations) the identified cluster means are transferred to the assignment unit. At (t_3) the recorded spikes are continuously mapped to their origins. The process time depends on the contributing factors from validation to assignment such as training and validation. The performance check unit (Figure 5.2) constantly evaluates the efficiency of the clustering to monitor either the error deviation in cluster means calculation or the changes of the action potentials over time. The retraining command is issued (t_4) when any of these issues occur. In addition to all adaptive settings, user-programmable tuning is also considered in case of worst-case scenarios.

This process is simply expandable for high channel count processing. One approach for multi-channel processing is to embed a processor in each recording channel. Thus, each processor

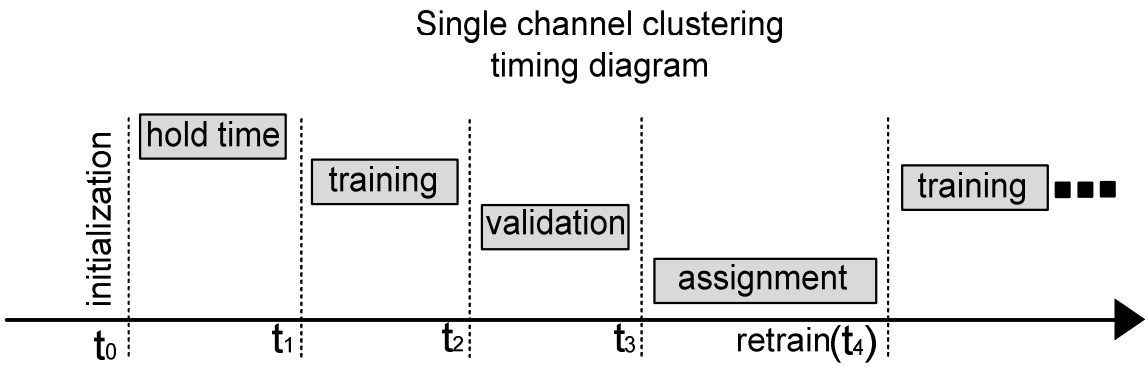

Figure 5.11: Multiphase clustering timing diagram.

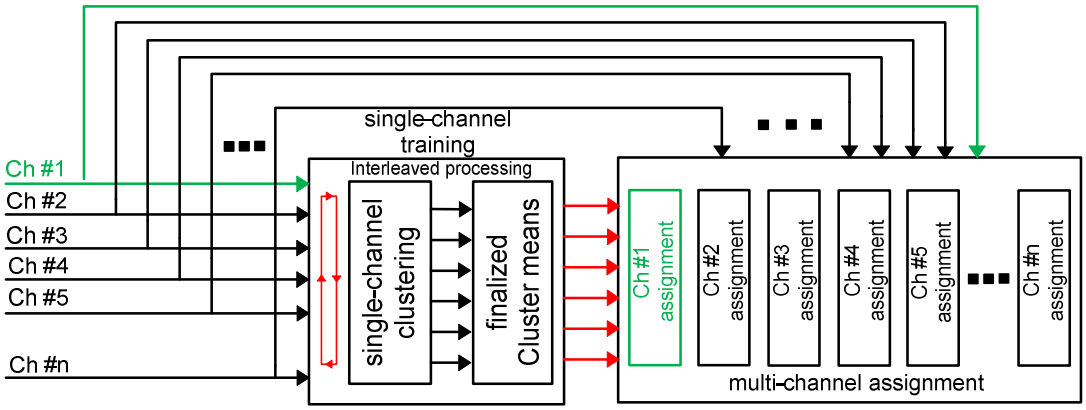

Figure 5.12: Example of multi-channel processing of recorded data.

periodically generates certain amount of information corresponding to the captured neural signal. This is an expensive method for processing since each channel requires a large amount of memory to cover all the transient clusters in training phase. The area and power consumption of the processor is dominated by the memories. To avoid these issues, the training unit in Figure 5.12 is shared between all the channels and the converged cluster means are buffered in the individual assignment unit for each recording line.

5.3.5 Training Unit Structure

The block diagram of the training memory structure and training unit main processing blocks (e.g Status engine) are shown in Figure 5.13. The transient memory core is implemented in a matrix format for simple highly flexible access to the memory locations. The structure is chosen based on factors such as adaptive power saving, conditional training and logic reusing which are explained in the next section. The training unit processing core includes implementations of the key blocks which are used in different phases of operations over the training period. Since O-Sort is a serial algorithm, each *FV* has to go through different paths for processing. In each path access to some units is necessary for performing different combinations of operations. For example for mean update operation, the sequences of states are ($\ell 1$-norm → mean update → merging check → merge update → waiting for next *FV*) which represents the worst-case processing chain. In the next section the operation of each block and its communication to the training memory is explained in detail.

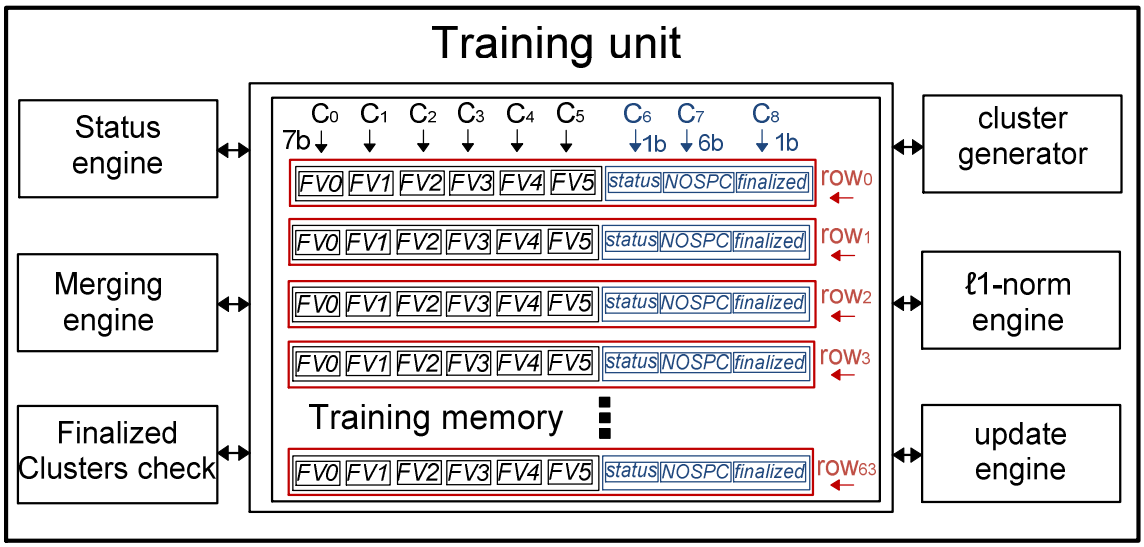

Figure 5.13: Training memory structure and main processing units. Each row of training memory consists of six locations for accommodating extracted feature vectors (FV_0-FV_5), 1bit status flag; 6bits represents number of spikes per cluster (NOSPC) and 1bit for finalized flag.

5.3.5.1 Training Memory Structure

The general structure of the transient memory is shown in Figure 5.13. There are important advantages in the proposed structure:

1- **Number of memory rows**: One of the most significant factors contributing to the power consumption is the number of memory locations for buffering the transient clusters in training phase. The depth of training (number of training locations) is designed to accommodate 64 transient clusters, so the chip power is dominated by the buffering of the cluster means. Using *ADDs* extracts six distinguishable features from the signal decomposition which results in substantial power and area reduction compared to spike template buffering.

2- **Adaptive power saving**: One of the embedded columns in training memory structure is status flag (see Figure 5.13). The status flag of zero (*Status*=0) shows that the memory row is empty and its contribution to the training process can be ignored. The flag is updated after cluster creation and merging phases to show that the buffered values in the memory locations are accessible for processing. It is updated recursively during training, and the memory locations with status of zero are excluded from the processing in $\ell 1$-norm and merging phases. The status flag is adaptively adjusted provides adaptive power saving over the training period. Another advantage of this flag

is in ID generation. Since flag shows whether the memory location is used or unused, the reuse of memory location to saves the number of memory locations required in the core of the training block.

3- **Conditional training**: In the mean update phase, the contribution of incoming *FV*s becomes less more insignificant after each iteration Eq.(5.6) The cluster mean value becomes less sensitive to noise variations and it is observed that if the cluster weight (*W*) is more than 40, the cluster centroids converge to their true values and the finalized flag (see Figure 5.13) is set to one. The finalized location will allows conditional mean updating and merging.

4-**Memory location reusing**: number of spikes per cluster (NOSPC) and status flag locations are used for memory location reuse in cluster creation, mean update and merging update phases.

These additions in the proposed training memory structure make it a very efficient choice for hardware implementation. The number of locations for buffering the features is defined via *FE* unit. The algorithm is completely adaptive in terms of power consumption and self-initializing. To satisfy the power requirement of implantable devices, register-bank memories are used to implement the proposed efficient memory core structure. The block diagram of the register bank memory is shown in Figure 5.14. The implemented design is composed of functional blocks such as writing decoder, reading decoder and the I/Os which are represented by *FV_in* and cluster

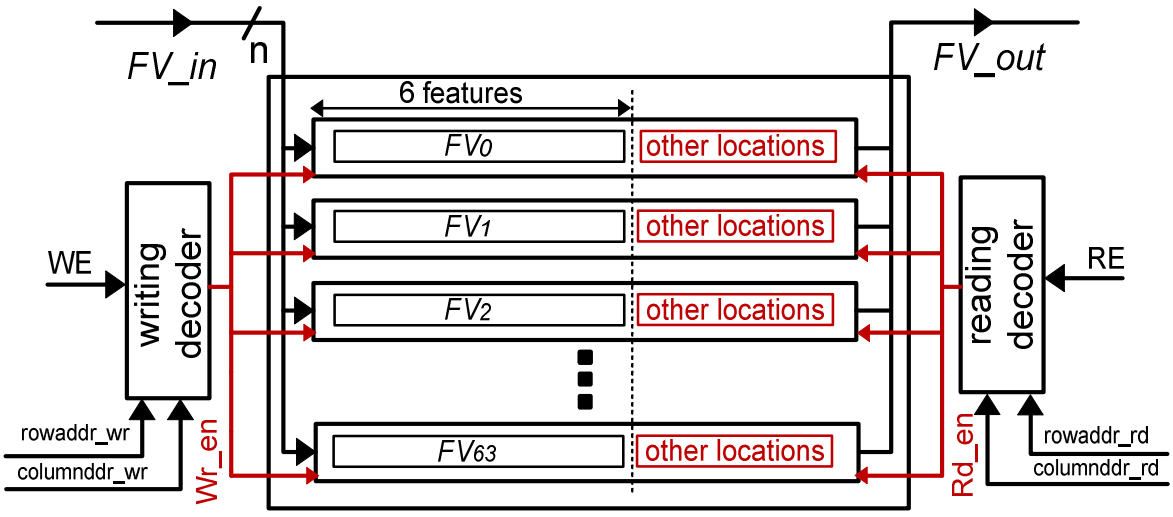

Figure 5.14: Block diagram of the register bank memory. Writing and reading of *FVs* are shown in this figure. Other memory locations include status, NOSPC and finalized flags.

FV_out. Based on the number of transient clusters in training, the number of memory locations are 64 in this design (FV_0-FV_{63}), which can be reduced to 45 using memory location reusing technique in the cluster generation phase. In the writing control block a 6 bits address (rowaddr_wr) decodes the row address of the memory array. columnddr_wr and Wr_en are also used to control the sequential writing process in different columns. Each row is used to buffer a *FV*. The writing process is performed in cluster creation, mean update and merging update phases. The same decoder and pointers are used for reading the process in transient memory as shown in Figure 5.14.

5.3.5.2 Status Engine

The status engine (Figure 5.15) screens the activity of training unit and monitors the duration of training using control flags which are explained in this section. The controlling I/Os are divided into three different categories: initiation of training, progress of the cluster convergence and flags regarding the termination of the recording channel evaluation. The training phase begins with training_ini. The retrain flag make the training engine flexible by providing training whenever is needed. Although the whole training phase is fixed for a specific number of spikes, training_coeff is used to manipulate the training time which can be overwritten manually or adaptively based on the sorting performance analysis as illustrated in Figure 5.15. The second bunch of terminals is related to the training progress. The considered flags, train_per1, train_per2 and train_per3 are the result of different units which are cluster creation, mean update and merging update phases. In the

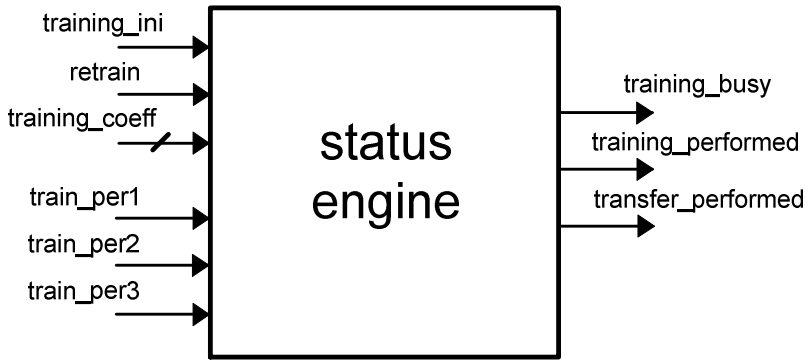

Figure 5.15: Status engine block diagram with inputs/outputs.

final category, training_busy defines the training and evaluation phases' progress. As long as this flag is set to zero, the training phase is on hold and when it is set to one the assignment phase is on hold. This flag is also used in the feature monitoring unit for switching the data transfer between assignment and training. The rest of the flags are relate to the termination of training and transferring of the converged cluster means to the assignment unit.

5.3.5.3 Cluster Generation (ID Generator)

Figure 5.16 shows cluster ID generation unit. The cluster creation unit performs ID generation for new transient clusters in training phase. New cluster generation occurs when the distance of the incoming *FV* is larger than the sorting threshold C_{min}> *SThr* (Figure 5.10). When this condition is satisfied, the ID generation unit starts looking for a free memory location to assign the *FV*. This process begins with sequential evaluation of memory locations step by step. In each step, based on the generated address in ID generation block (row-addr), the status flag and the NOSPC of each memory row (e.eg row_{20}) are called from transient memory for investigation. If the status flag of the memory location is zero, it means that the found location can be used for buffering the awaiting *FV*. Another condition is also considered to apply the logic reusing technique exploiting the status flag and NOSPC. In this condition if the status flag is high but the weight of the existing cluster

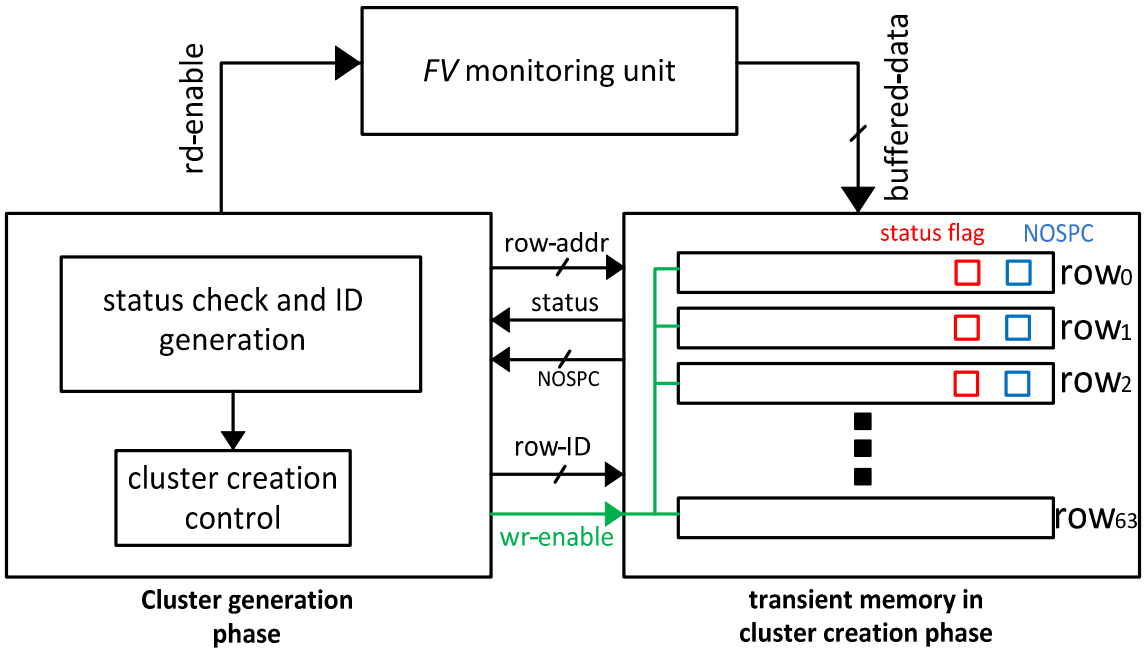

Figure 5.16: Cluster ID generation flow.

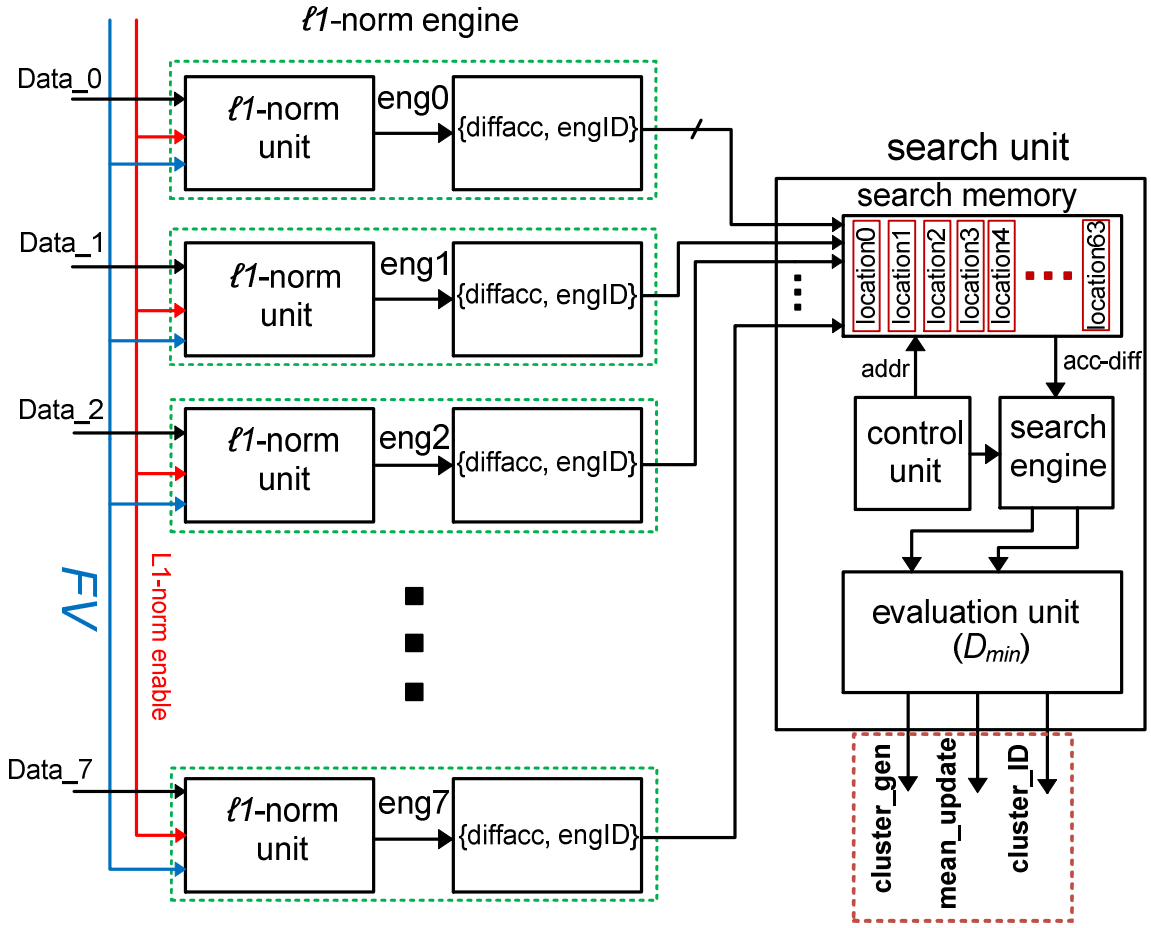

Figure 5.17: Block diagram of the $\ell 1$-norm engine. $\ell 1$-norm consists of eight parallel engines and it is reused eight times to calculate the $\ell 1$-norm difference accumulation for all the 64 rows (row_0-row_{63}) as shown in Figure 5.13.

(NOSPC) is less than five (NOSPC<5), it means that the created cluster is due to noise or overlapping spikes so the memory location is reused. This technique can reduce significantly the number of locations in the training memory. When the proper location is found, the proportional cluster ID is created (row-ID) for cluster generation.

5.3.5.4 $\ell 1$-norm Unit Structure

Figure 5.17 shows the block diagram of the $\ell 1$-norm engine. When a *FV* is sent to the training engine for processing, in a first step it is compared to the created transient cluster means in previous phases. Data_0 to Data_7 are the representative of transient clusters which are transferred to the $\ell 1$-norm unit for difference calculation and accumulation. The status flag in $\ell 1$-norm distance

metric is used as a condition for reading the memory rows which contain the transient clusters. The status flags of memory rows are updated during each training iteration. This merit allows the exclusion of non-important memory rows (status=0) from training and consequently results in inherent power management. The $\ell 1$-norm distance operator introduces two crucial advantages over Euclidean distance. Firstly $\ell 1$-norm simplifies the distance calculation operation in units where it is exploited compared to the squared distance and squared root calculations used in the Euclidean distance requiring more power consumption and extra wordlength for buffering the calculated values. Secondly, $\ell 1$-norm metric is more immune to noise which is discussed in [76]. In the $\ell 1$-norm engine the degree of interleaving is chosen to be 8-unit to reuse the logic blocks for multiple runs in order to reduce power and area simultaneously. Minimum distance

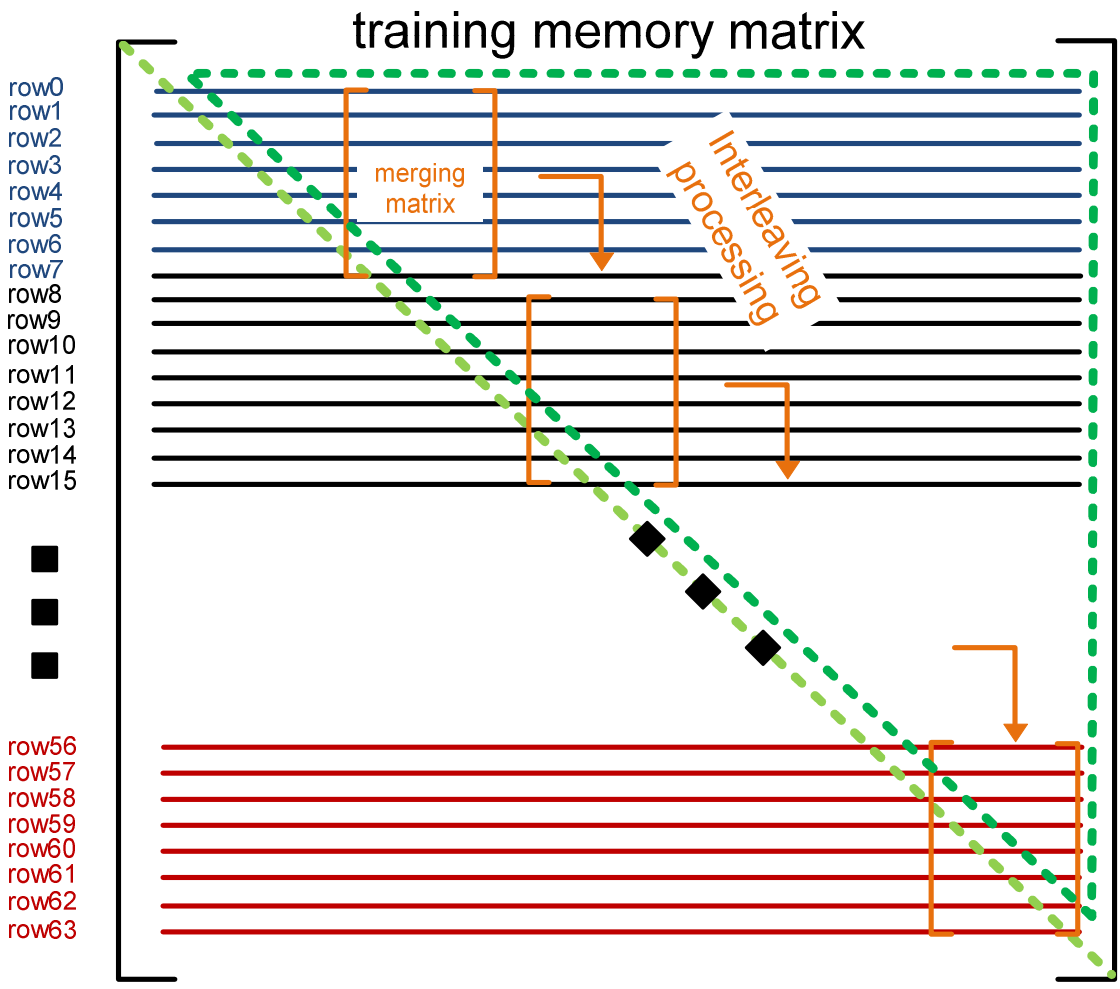

Figure 5.18: Demonstration of a moving merging matrix which is interleaved between all the transient memory rows (row_0-row_{63}).

determination (D_{min}) is performed sequentially on the buffered values in the search unit to either activate the cluster generation (cluster_gen) or to enable update the mean value (mean_update).

5.3.5.5 Merging Unit Structure

Each incoming *FV* is assigned to the closest cluster based on the $\ell 1$-norm between the *FV* and closest centroid in feature space and cluster mean is updated accordingly. The mean update phase might result in cluster shift in feature space. Due to the cluster shift, there might be overlap between the clusters. In this case, two clusters with distance of less than a threshold $M_{min} < SThr$ (Figure 5.10) in feature domain are indistinguishable and they are merged. To evaluate the merging possibility, the distance between all cluster means are calculated and the candidates are sent for merge update. The approach for finding the overlapping clusters is illustrated in Figure 5.18. An interleaving of eight parallel merging-check units is employed to move between the matrix memory rows. The merging engine is able to evaluate the overlapping between the clusters in eight rows simultaneously. One of the important points in merging design flow is that only the upper-half of the matrix memory is used for distance calculation and the lower-half of the matrix is masked in this analysis. The whole merging phase is performed recursively for eight consecutive runs to cover all the memory rows (see Figure 5.13). In mutual difference computation mode, all clusters are reconfigured and if the status flags of the memory rows are one, the mutual difference is taken and accumulated for further processing. A two-step sequential search method is developed to find minimum distance at the end of merging phase. At each run the minimum values are found in search_eng1-8 (see Figure 5.19), are stored in search memory as depicted in Figure 5.19. At the end of merging phase, it is decided whether to start merging update (merge_ini) or termination of training is issued for the current data (training_done). In case of merge update, the content of the transient memory rows based on the chosen IDs (first_ID and second_ID) are called for merge update phase. The centroid of the new merged cluster is then calculated as a weighted mean:

$$C_{MergeUpdate}(n) = \frac{W_1 \times c_1(n) + W_2 \times c_2(n)}{W_1 + W_2} \tag{5.7}$$

where c_1 and c_2 are the centroids, and W_1 and W_2 are the respective spike populations of each cluster. The updated centroid ($C_{MergeUpdate}$) is stored in one memory location and the content of other locations is erased to be reused for the cluster generation phase. In the first iterations of the

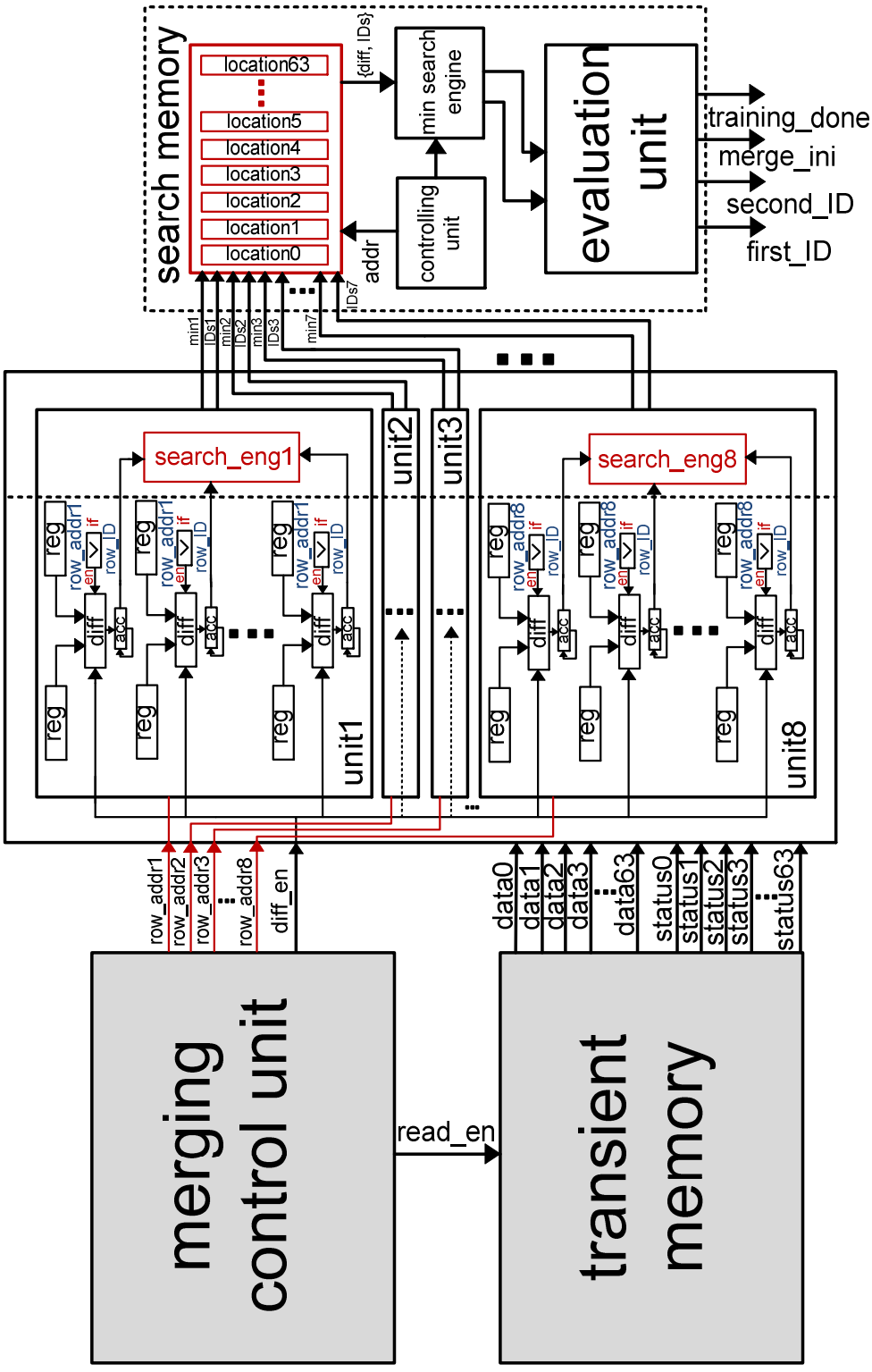

Figure 5.19: Hardware implementation of the merging unit. The first section depicts the

training phase merging occurs frequently due to the small clusters created in the cluster generation phase but after a specific time the main clusters start to converge to steady state values.

5.3.5.6 Finalized Cluster Means Transfer Unit

It should be mentioned that the amount of training time is different based on number of active neurons, spike shapes and their firing rates in the captured data. In the status engine, a specific number of spikes is considered for the duration of training (e.g.250 spikes). Over the training process, the IDs of the finalized clusters are saved in ID memory (Figure 5.20). It should be noted that the spike processor is capable of simultaneous assignment to six clusters. If within the specified processing time limit, the number of the finalized converged spikes reaches six, training is stopped and the identified cluster means are stored in the assignment unit. A second option is designed for sequential search to find the settled cluster means which meet the convergence criteria as discussed in section 5.3.4.1 (W>40). This phase is initiated when the number of the finalized cluster means are less than six. In this phase the conditions flags are evaluated for transferring the cluster means which are used in the cluster mapping phase according to the minimum distance of the incoming *FVs*.

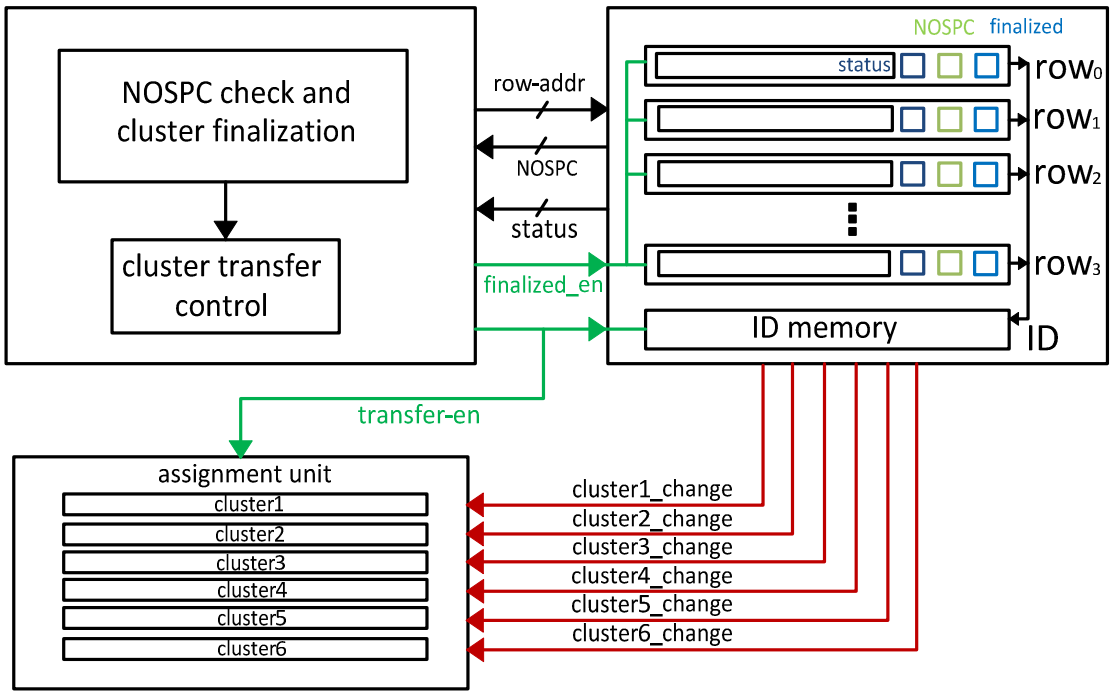

Figure 5.20: Converged cluster means check.

5.3.6 Assignment Unit

Figure 5.21 shows the assignment block diagram. Having identified the cluster means in the training phase, clusters 1-6 change are transferred to the assignment unit for updating the mean values. The assignment unit is capable of mapping up to six neurons in the stream of neural data. Having updated the cluster1-6 mean values at the end of the training phase, the spikes are constantly processed and assigned to one of the clusters according to the minimum $\ell 1$-norm distance measurement. The difference values (DIFF1-6) are computed simultaneously from the comparison of the incoming *FV* to each cluster mean (Cluster1-6) and the values are sent to the search engine (see Figure 5.21) to recognize the spike ID. The outputs of the assignment unit are clustered_*FV* and assign-notvalid flag which both are also used in the performance check block. The clustered_*FV* consists of six locations covering feature values and one location which

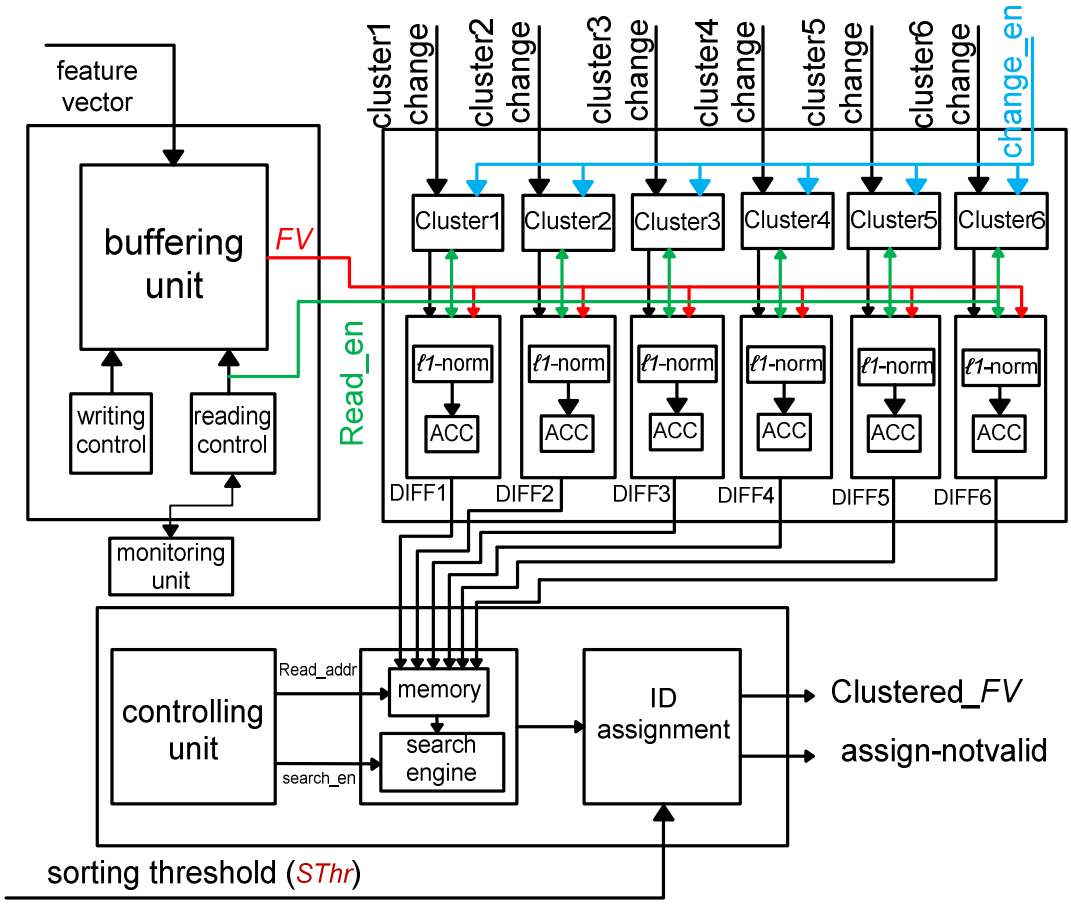

Figure 5.21: Assignment block diagram. Accumulator is depicted by ACC.

demonstrates the spike ID. The Assign-notvalid flag is set due to faulty noise *FV* assignment or it due to the unwanted issues in O-Sort operation such as artificial clustering.

5.3.7 Performace Check Unit

As explained in [13], there are two main approaches for calculating the sorting threshold, either threshold approximation which is computationally cheap or using statistical approach to find optimal threshold through the noise covariance matrix calculation. Running simulations for hardly distinguishable neurons or neural data with a high background noise level show that the clustering accuracy is severely affected when using the sorting threshold approximation approach. This normally causes two well-defined phenomena in clustering which are cluster splitting and artificial clustering. As shown in Figure 5.22, cluster splitting is the division of single cluster into multiple clusters while artificial clustering is merging of two single clusters (or more) into one cluster. With threshold values larger than (T_{max}) the probability of missing a cluster or artificial clustering is

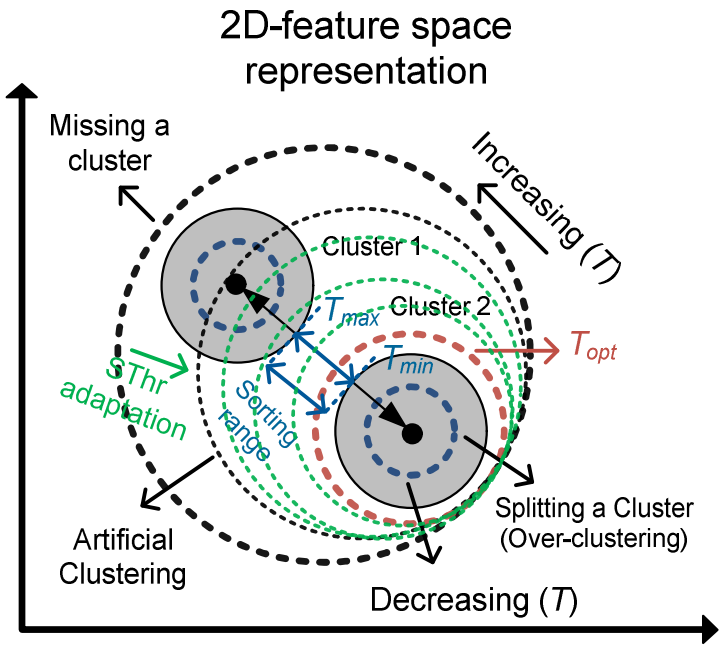

Figure 5.22: 2-D representation of feature space for two clusters. The effect of increasing (decreasing) the threshold is shown. Different threshold levels (T_{opt}, T_{Max} and T_{Min}) and sorting range are shown. With increasing the threshold ($T>T_{Max}$), the probability of missing a cluster and artificial clustering is high. In addition, with declining the threshold ($T<T_{Min}$), the main cluster would artificially be split into two or more sub-clusters.

high. There is also a lower limit (T_{min}) below which the over-clustering effect is expected. In this section, a novel and simple approach is presented for evaluation of the clusters status. Based on the clusters status analysis, an optimal threshold (T_{opt}) is adaptively set which enhances and locks the clustering unit on its mean average accuracy. The proposed method is called performance lock scheme. The hardware implementation of the proposed methods is shown in Figure 5.23. The *SThr* tuning chain is composed of clustering status analysis and *SThr* fine-tune blocks. The inputs to the status analysis block are the finalised cluster means (C_1-C_6), assign-notvalid and assigned *FV* (clustered-*FV*). In the *SThr* self-tuning design, assign-notvalid monitors the rate of assignment error and clustered-*FV* (six features representing the *FV*) and cluster ID from the mapping unit. Statistical operations are performed on the input values for fine-tuning the *SThr* such as calculating the correlation between the finalized cluster means and monitoring the assignment error rate (AER) over validation phase (see Figure 5.11). Indicators of the issues in clustering are interpreted from the statistically evaluated data as below:

1)- In case of artificial splitting of a single unit into multiple clusters, the correlation factor (*Cof*) between the identified cluster means is high (more than 0.9) and their corresponding firing rate is less than other active neurons. The *SThr* is increased (+) in this case to avoid cluster splitting.

2)-In case of artificial clustering, a spurious cluster is created and the rate of assign-notvalid flag is high. In addition to the AER, such cluster represents multi-unit activity and it has normal/high

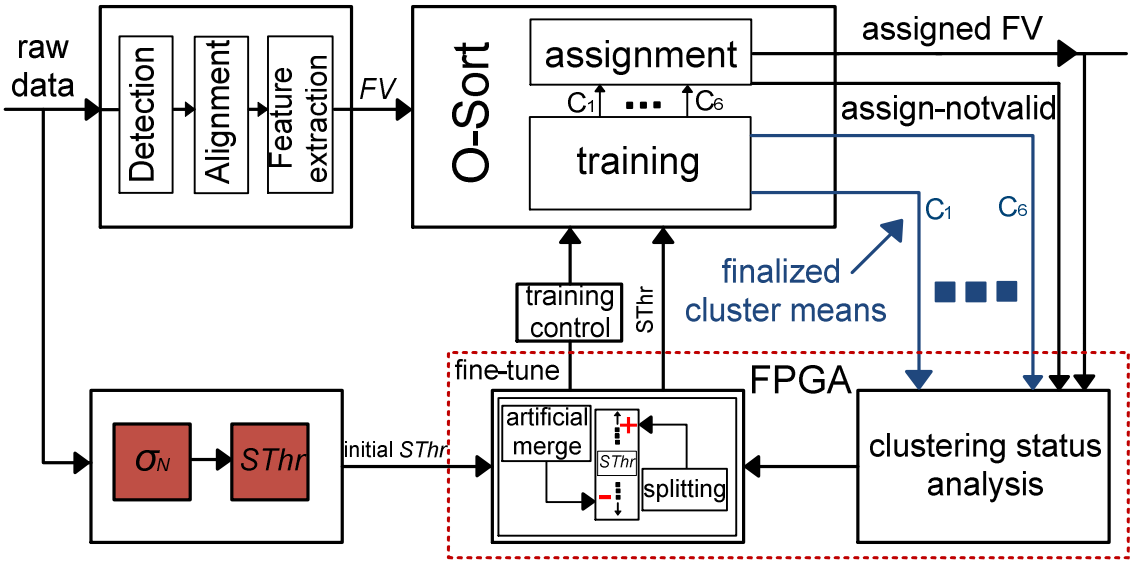

Figure 5.23: Sorting threshold (*SThr*) self-tuning methodology.

firing rate due to spike allocation from multiple neural activities. The *SThr* is adjusted to a lower level (-) for optimal clustering.

3- It also should be noted that the spike shapes may vary over time. The performance check unit also tracks this issue and reinitialises the training for detection of any changes in the recording data.

The adaptive threshold adjustment scheme is implemented on FPGA. The threshold fine-tuning steps are performed during the validation phase and the best threshold in terms of performance is chosen amongst three iteration phases. The progress of trimming the threshold is computed via the mentioned metrics over different runs. It should be noted that in each iteration, 10% variation is added/subtracted for *SThr* calibration. The threshold self-tuning method results in 5-8% median clustering performance improvement.

5.4 ASIC Implementation

5.4.1 FPGA-Based ASIC Verification

To evaluate the ASIC performance, a data streaming scheme using universal asynchronous receiver/transmitter (UART) was developed which continuously sends data to the FPGA and receives the processed data from FPGA (Figure 5.24). The control of data streaming is performed in MATLAB. Due to buffering limitations, the entire quantity of data cannot be transferred to the FPGA to driving the spike sorting DSP. The neural data is divided into different segments based on the input buffer of UART in MATLAB. The received data in UART is deserialized and based

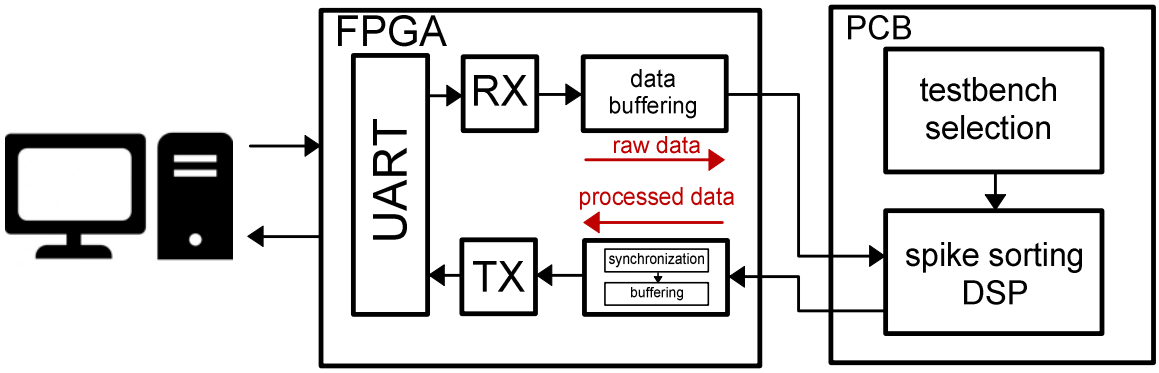

Figure 5.24: Data-streaming interface. RX is the symbol of receiver and TX is the symbol of transmitter.

on the controlling signals between UART/RX the incoming data is buffered in data buffering unit. The spike sorting DSP starts receiving the neural samples from the data buffering unit for processing. The test-bench is quite flexible and can be reconfigured to implement different test-benches. Based on the chosen test set up, processed data is sent back to the FPGA for synchronization. The output rate of the processed data in the defined test systems are different and they need to be synchronized for using in UART at a fixed transmit rate. The TX unit decides the timing intervals for transmitting data from FPGA to PC via the UART controlling signals. In the UART data is serialized and transferred to MATLAB. This process is repeated until all the data has been processed. The processed data from the spike sorter is evaluated in MATLAB to measure the detection and clustering accuracy. Figure 5.25 shows the developed processor and setup used for chip testing. The FPGA board interfaces to the ASIC board using universal asynchronous RX/TX system (UART). A MATLAB-based platform was used to drive the FPGA and read the corresponding outputs from the chip.

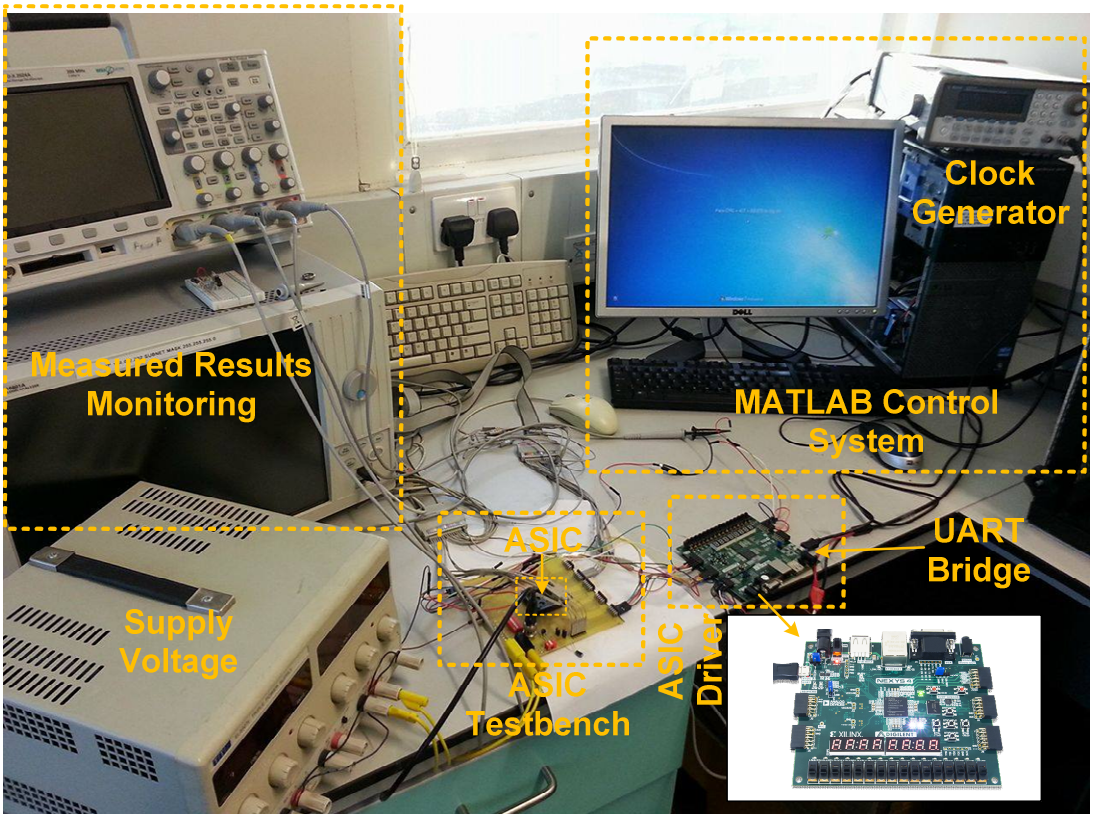

Fig. 5.25: Fabricated ASIC and MATLAB-based ASIC verification setup. FPGA is used to drive ASIC.

5.4.2 Chip Measurement Results

The adaptive spike sorting processor was fabricated in a 180-nm CMOS technology. The die is shown in Figure 5.26. It occupies an area of 10 mm^2. The processor uses four different clock rates (30kHz, 120kHz, 240kHz, 960kHz) to obtain the best processing efficiency which results in 148µW of power from a 1.8V supply voltage. The power density of the chip is 54.8 $\mu W/mm^2$ (the training area is excluded from the power density calculation) which is 14.6 times lower than the power density known to damage brain cells [159]. A standard dataset[16] with a known ground truth (spike times and cluster IDs) is used to validate the functionality of the fabricated processor. To evaluate the spike detection performance two well-known metrics are use: 1) the probability of detection, which is computed as the number of truly detected spikes (*TDS*) divided by the total

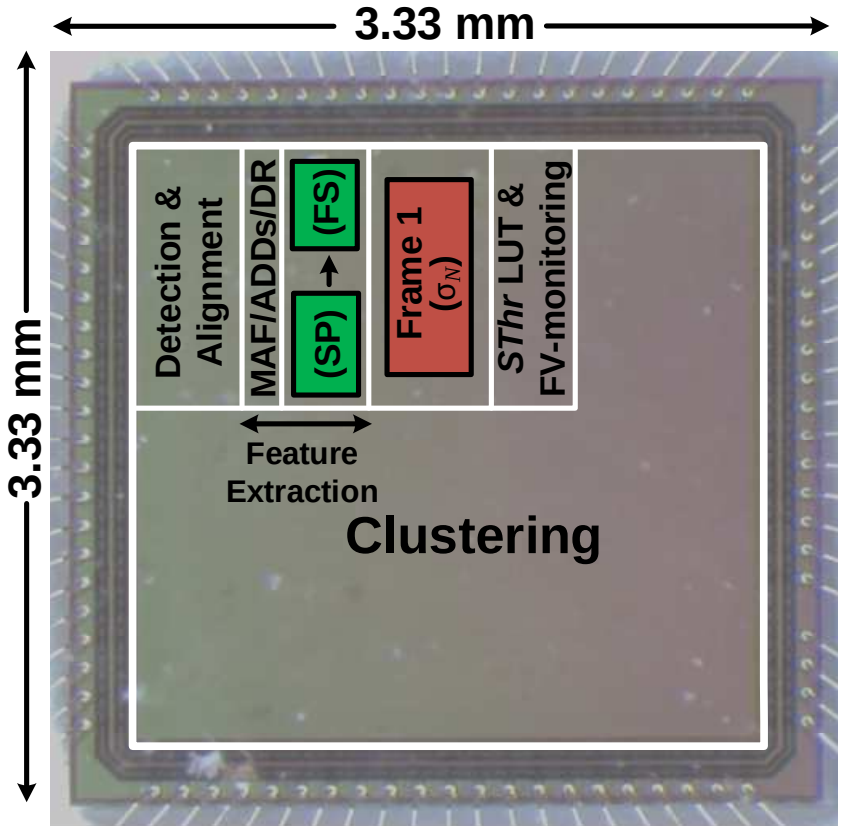

Figure 5.26: Illustration of ASIC die photo. The area distribution is illustrated based on the Cadence synthesize results.

[16] The datasets are explained in Chapter 2.

number of spikes (*TNS*): $P_D = TDS/TNS$. And 2) the probability of false alarm which is computed number of false detections (*FD*) divided by true positives (*TDS*): $P_{FA} = FD/TDS$.

To improve accuracy and power efficiency some modifications are made. The outputs of the detection unit was compared with the results obtained from Matlab simulations using the signals with known ground truths. The detection performance was characterized by the probability of detection of (92%) and the probability of false alarms of (1%). The ωNEO (see section 5.3.1) obtains the best detection accuracy when ω is chosen to be 2 or 3. This accuracy improvement is due to the modifications applied into the conventional NEO. In addition, the overall power consumption of ωNEO is reduced in simulations by 3X (30%) compared to standard NEO implementation. Figure 5.27 shows a sample recorded neural signal input on the logic analyzer and the corresponding collected aligned spikes in MATLAB.

The goal of *FE* is to choose a few distinguishable features for obtaining the best differentiation between the clusters. Normally features with multimodal distributions are chosen for efficient clustering using advance statistical methods such as asynchronous sampling [100] and Kolmogorov-Smirnov (KS) [160]. Having performed the *FE*, features with the largest deviation are sent for clustering. In *ADDs*, the hardware implementable version of KS is moved in front of the *FE* block for frequency band separability analysis in order to obtain high clustering accuracy. The proposed approach performs sub-band frequency selection and maximizes the separation probability between the spike features using proper scaling factor selection. Figure 5.28 shows the specific cases in which different scaling factors are used for decomposition. Using the identified scaling factors in *ADDs* introduces more discrimination for clustering. The extremas (max/min) of the decomposed spike waveforms can be used to represent the frequency band separability. The analysis of *ADDs* validates that the better performance is obtained through frequency sub-band selection approach. As depicted in Figure 5.28, the extracted max/min peak values in the identified frequency bands in two different datasets provide enough separability for clustering unit.

The identified scaling factors for each dataset can be compared with the manual decomposition approach (see Chapter 3, Table 3.1). For the Easy2 dataset, a comparison with Table 3.1 shows that the selected scaling factors are similar with the definition of combination 3 which achieves

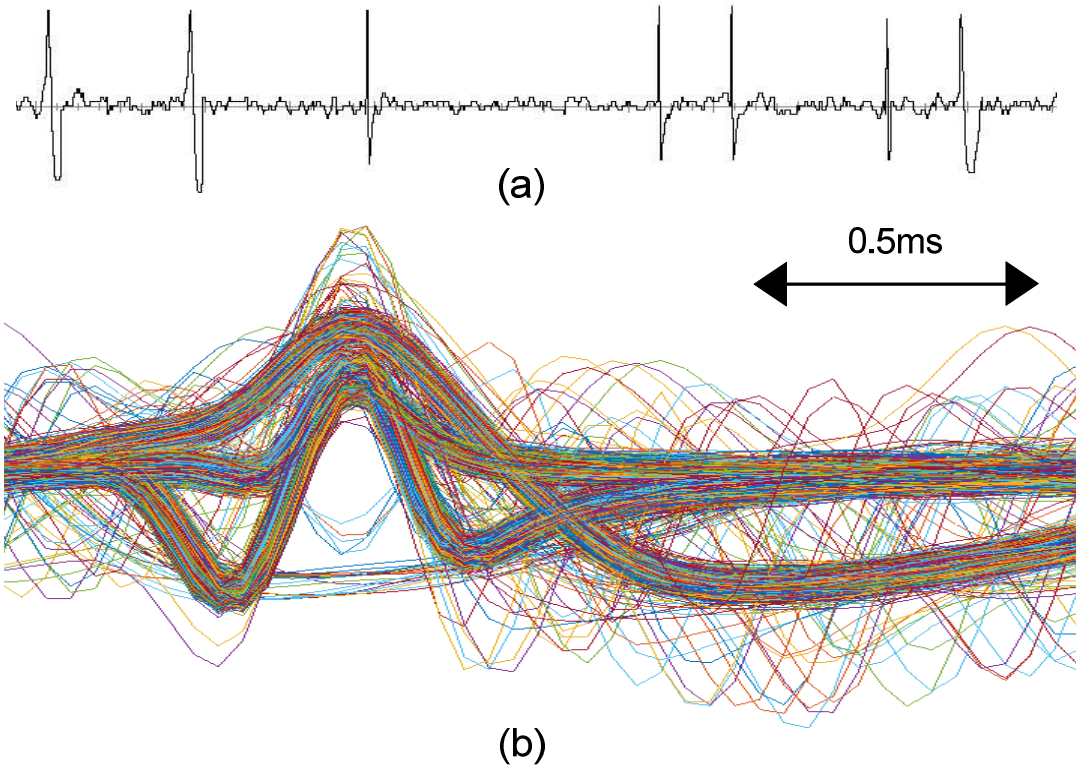

Figure 5.27: (a) A segment of neural signal depicted on logic analyzer and (b) the recorded spike waveforms in MATLAB using the interfacing system.

86% CAcc. For Difficult1, the chosen scaling factors are equal to the best combination (combination 3) in Table 3.1 which obtains 87.8% CAcc.

The clustering accuracy of the adaptive spike sorting chip was tested and evaluated across all datasets and noise levels. In the *FE* unit six features (K=6) are used for clustering where K shows the feature space dimensionality. As described previously, there is an initial phase for clustering self-tuning. The clusters are re-evaluated in the retraining phase if the implemented criteria are not satisfied. The evaluation of the spike sorting chip is performed over the steady-state phase, such that the final cluster means are converged. Figure 5.29 shows from (a) to (d) the 2-D projection of the clusters in different datasets form Easy1 to Difficult2. The boundaries of the clusters are marked with dotted lines. The overall clustering accuracy of 84.5% is obtained over all different

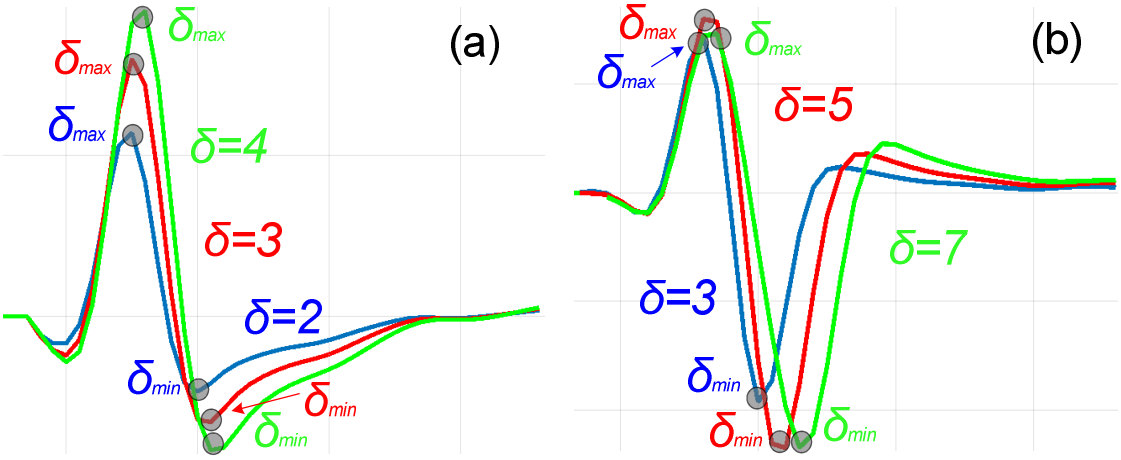

Figure 5.28: An illustration of extrema features using different sets of scaling factors. The scaling factors are selected based on ADDs methodology for (a) C_*Easy2_0.05* (b) *C_Difficult1_0.05*. The identified scaling factors for each dataset can be compared with the manual decomposition approach discussed in Chapter3.

datasets and noise levels. The adaptive processor accuracy compares well with the most powerful clustering methods such as *Waveclus* [89]. The average accuracy of the developed processor can be compared to median clustering accuracy of the state-of-the-arts classifiers such k-means or SPC. Figure 5.30 shows a plot of the estimated performance of the three clustering and classification methods.

The total system performance consists of contributions from the detection, *FE* and clustering units. As explained earlier, the architecture of training memory is designed for self-power management (status flag in Figure 5.13) purposes. The register bank memory implementation with the introduced architecture provides up to 2X power reduction compared to the memory implementation without considering the exploited status flags. This value is the average power reduction ratio, since the degree of power saved varies in different iterations. In addition to power reduction, the area is also reduced significantly due to the use of six features (K=6) for buffering the transient clusters.

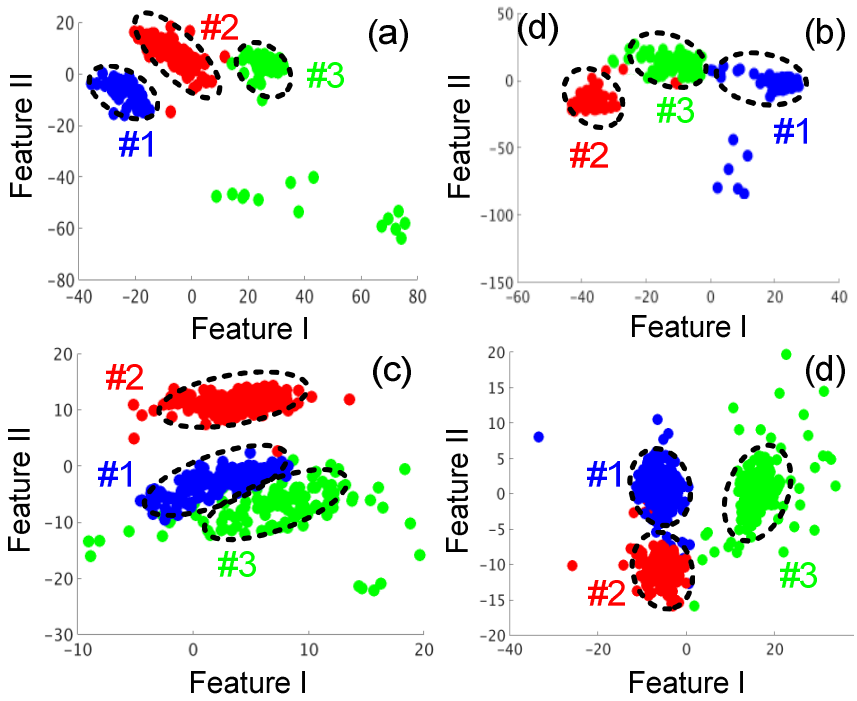

Figure 5.29: 2-D projection of clusters for (a) *C_Easy1_0.05*, (b) C*_Easy2_0.05*, (c) *C_Difficult1_0.05* and (d) *C_Difficult2_0.05*. (Spikes have been colored according to the ground truth).

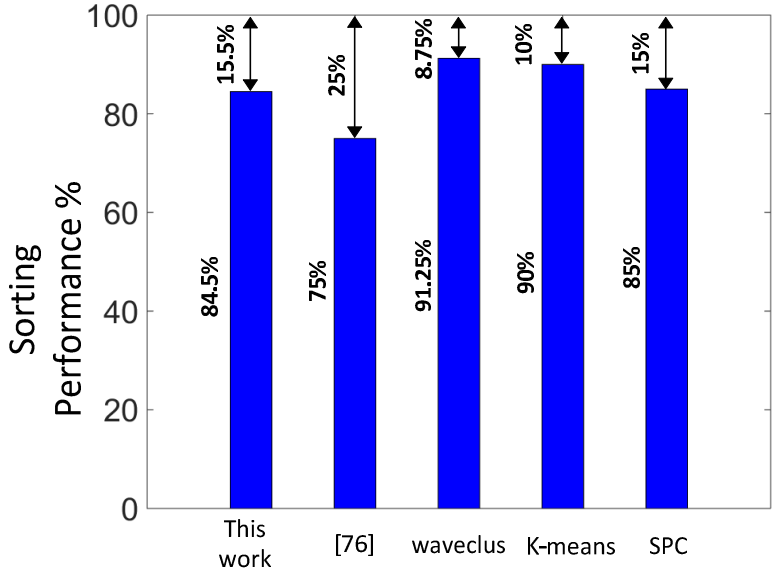

Figure 5.30: Performance comparison of implemented processor and the state-of-the-art spike processors. The mean accuracy of *K*-means and SPC is calculated based on the template matching *FE*.

Table 5.1:
Adaptive spike processor summary.

Technology	180-nm
Core V_{DD}	1.8V
Power	148µW
Operation frequency	30KHz
Processor type	Adaptive
Median P_D, F_A	92%, 1%
Median CA_{CC}	84.5%
Compression factor	150X, 240X *
Core size	10mm^2

* Compression factor in different modes.

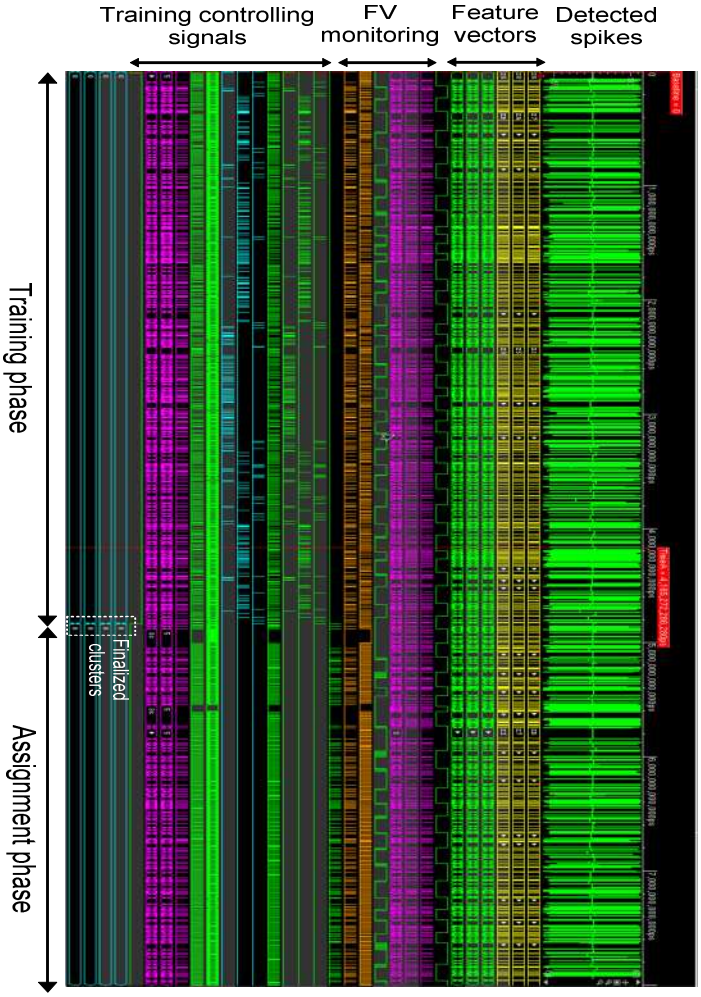

Figure 5.31: The operation phases of adaptive processor in Cadence simulator.

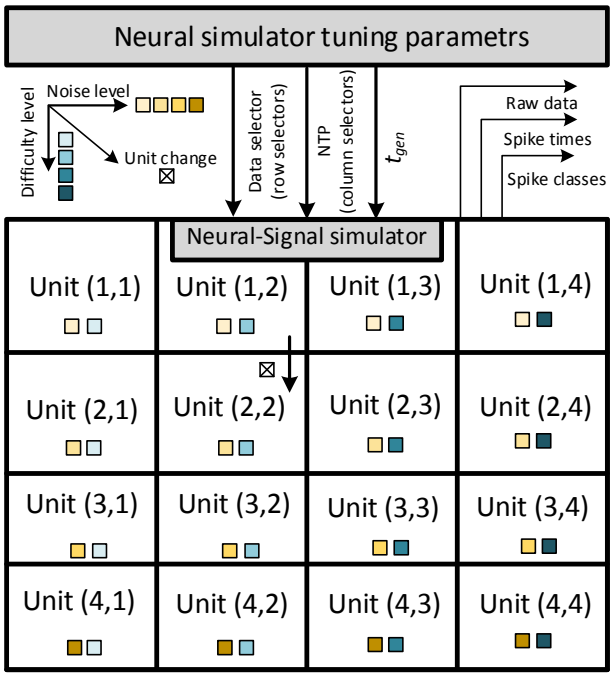

Figure 5.32: The data generation based on random data selection. Data is generated based on the tuning parameters (data selector, NTP and t_{gen}).

The main characteristics of the chip are summarized is in Table 5.1. Figure 5.31 shows the entire spike processing operation. The operations of different units are labelled in the figure. This unique balance between good accuracy and low complexity makes the adaptive spike processor a particularly good candidate for implantable devices.

5.4.3 Dynamic Test Methodology

In this section, a dynamic test protocol is adopted to fully evaluate the effectiveness of the new adaptive spike processor under variable input signal conditions. The input based on random data selection is proposed as an optimal approach for imitating the dynamic nature of neural data encountered in practice. For random data generation, all the datasets discussed in section 2.4.2 form a matrix of data as shown in Figure 5.32. In the neural simulator four tuning parameters are considered. They include a dataset selector (from Easy1, Easy2, Difficult1 and Difficult2), a noise tuning parameter (NTP) for adjusting the noise standard deviation (e.g. 0.05, 0.1, 0.015 and 0.2), a generation time (t_{gen}) and the number of active neurons selector (ActNeuSel). Based on tuning parameters, data is chosen randomly from each unit (e.g. unit (1,1) in Figure 5.32) and

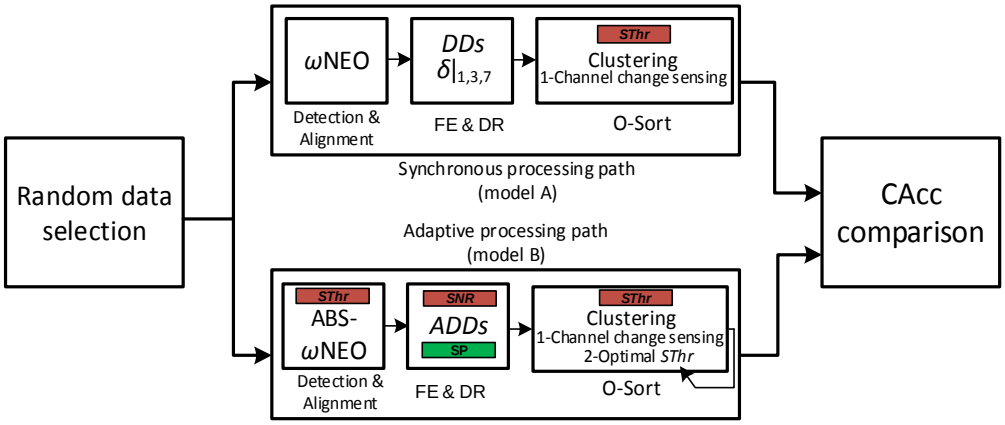

Figure 5.33: Dedicated set-up for comparing the performance between the non-adaptive and adaptive processors.

concatenated in time for a specific duration. The variable conditions in data represent variations over time such as non-stationarities in background noise, and appearance (disappearance) of active neurons in a recording channel. They are used for examine the robustness of the adaptive spike processor to input data variations.

The model used for data variation performance analysis is shown in Figure 5.33. In one path (model A), a synchronous processor without adaptive frames is used to test the classification accuracy performance. The constituent building blocks in a synchronous spike processor are NEO based detection, multi-resolution decomposition based on fixed scaling factors ($\delta|_{1,3,7}$) and O-Sort clustering. In a second path (model B), the spike processor with adaptive frames is used. The adaptive spike processor comprises conditional spike detection with dual thresholding (*SThr*-NEO), adaptive multi-resolution decomposition and O-Sort clustering with capability for setting the optimal threshold. In both processing paths, the O-Sort clustering unit is equipped with channel change sensing.

The clustering performance of both processors (adaptive and non-adaptive) is shown in Figure 5.34. The adaptive processor with the embedded adaptive frames (Frame 1 and Frame 2) and clustering unit with the capability of setting the optimal threshold outperforms the synchronous spike processor. The average classification accuracy of the adaptive processor is 10.9% higher compared to the non-adaptive model (84.5% versus 73.6%).

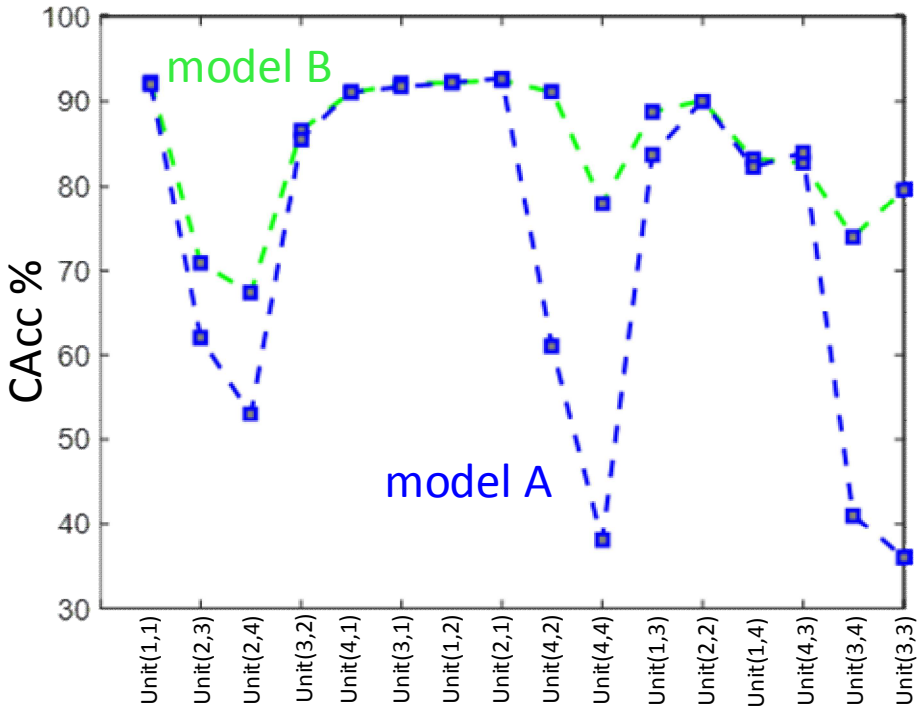

Figure 5.34: Comparison of clustering accuracy between the non-adaptive (model A) and adaptive (model B) processing chains. The data at input of the processor is generated based on the dynamic test methodology.

5.4.4 Comparison with Prior Work

Table 5.2 shows a performance summary of published previous spike sorting ASICs. This work introduces the first fully adaptive, online and unsupervised sorting processor with all the necessary building blocks to obtain the optimum performance with ultra-low power consumption. The fabricated processor is the first design with integration of synchronous/asynchronous/self-tuning capabilities. Although the system operation is synchronous, novel additions and circuit techniques are used to emulate the synchronous and asynchronous concepts in an adaptive form. Since the processors are implemented in different technologies, a power density (*Power/Area*) figure of merit (FOM) can be used for fair efficiency comparison. Power density can also be modified to express the processor efficiency to take account of the effective contributing factors in a processor

Table 5.2:
Comparison to the state-of-the–art processors.

Reference	[161]	[162]	[74]	[163]	[164]	[73](a)	[73] (b)	[76]	[165]	This Work
Detection	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Alignment	x	x	✓	x	x	✓	✓	✓	✓	✓
Feature Extraction	x	✓	✓	x	✓	✓	✓	x	x	✓
Clustering	x	x	x	x	x	x	x	✓	✓	✓
Compression factor	12.5X	80X	11X	15X	-*	-	-	240X	-	150X/240X
Power (µW/channel)	75	100	2.03	104	14.6	0.46	1.04	4.68	battery	148
Area (mm^2/channel)	0.11	1.58	0.06	-	0.01	0.03	0.02	0.07	-	2.7
Power density ($\mu W/mm^2$)	680	63.3	30	-	1460	15.33	52	66.8	-	54.8
Process (nm)	500	350	90	FPGA	90	65	65	65	DSP	180
Core Voltage (V)	3	3.3	0.55	-	1.08	0.25	0.25	0.27	-	1.8
CA_{CC}	x	x	x	x	x	x	x	75%	93%	84.5%
Self-tuning design	x	x	x	x	x	x	x	x	x	✓
EF	-	-	-	-	-	-	-	3.71	-	0.43

* The output of FE contains 53bits for transmission.

(a) Fabricated asynchronous processor in [73].

(b) Fabricated synchronous processor in [73].

design such as power, area, processor chain performance (*PCP*) and compression factor (*CF*). The processor efficiency factor *EF* is:

$$EF = \frac{Power}{PCP.CF.Area} \tag{5.10}$$

In addition to *EF*, the effect of technology scaling can be discussed for the digital power consumption. The power consumption downscaling factor (*DF*) is given by:

$$DF = \frac{\left(N_t.C_{avg}.V^2{}_{dd}.f_{clock}\right)_{H_{tech}}}{\left(N_t.C_{avg}.V^2{}_{dd}.f_{clock}\right)_{L_{tech}}} \quad (5.11)$$

where N_t is the number of transistors in design, C_{avg} is average capacitance load, V_{dd} is the power supply and f_{clock} is the operating clock frequency. H_{tech} and L_{tech} are the representatives of higher and lower technologies respectively. As an example, the *DF* comparison between the technology used in the fabricated processor is H_{tech} (180nm, 1.8V) and that in [76] L_{tech} (65nm, 0.27V), for the same f_{clock} and N_t is equal to $C_{avg}(H_{tech})/C_{avg}(L_{tech})$= 133. The N_t parameter is significantly affected by the feature space dimensionality (*K*) which results in number of logic cells for designing the clustering unit.

Compared to the state-of-the-art spike processor in [76], the adaptive design has 8.6X better *EF*. The estimated power consumption using *DF* is 4.2X lower than the reported power in [76]. The power density of the adaptive processor is 3.6X higher than in [73] due to the inclusion of clustering unit. Nevertheless, the clustered data provides a higher compression factor which is advantageous in terms of *EF* and transmission power. The processor in [76] has a clustering accuracy which is almost 10% lower than the overall performance of the adaptive spike processor in this chapter. Future implementation of the adaptive spike processor in a deep submicron technology would offer improved performance in terms of area and power consumption.

5.4.5 Interleaved Architecture

The optimal number of parallel processing engines in $\ell 1$-norm and merging units is obtained by interleaving processing methodology. Interleaving allows the reuse of a designed processing engine for optimizing the dynamic and leakage power which is due to the area of the silicon used in processor design. A detailed investigation of chip area and power consumption product reveals an optimal interleaving ratio (*R*). *R* is a trade-off between the leakage power, which is proportional to chip area, and dynamic power due to registers using in different operation phases. *R* is defined as total processing channels (e.g 64 in $\ell 1$-norm unit) divided by number of parallel processing engines (e.g. 4 or 8 parallel engines to perform $\ell 1$-norm phase). For *R*=1 the largest area and slowest operation frequency are obtained and the total power consumption is dominated by leakage

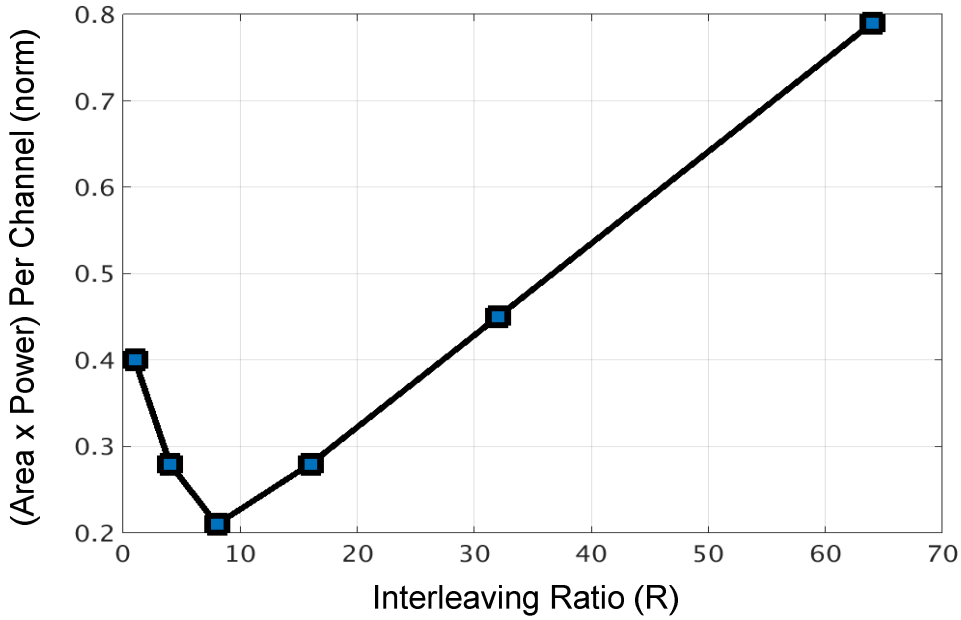

Figure 5.35: The normalized area and power trade-off versus interleaving ratio (R).

power. When *R* is increased, a faster operation clock is required which means more switching activity (dynamic power) but the leakage power is reduced. Therefore, at a specific *R*, the power-area product becomes minimum. It should be mentioned that the interleaving optimization methodology is strongly related to the fabrication process and circuit architectures. In this design, the interest is to optimize the area ad power simultaneously due to the hardware implantable circuit's constraints. Figure 5.35 shows the area-power product versus interleaving ratio (*R*). The *R* value is calculated by the extracted power and area information from the Cadence synthesize reports. It can be seen that the optimal ratio for power-area saving is 8 in $\ell 1$-norm and merging units. For more power saving in the training phase, *R* can be set at 4. The adaptive sorting processor is fabricated in 180-nm CMOS technology which exhibits less leakage current compared to more advanced technologies. It means that the dynamic power consumption plays the main role in defining the total processing power in the training phase because the higher frequency interleaving clock increases power consumption.

5.4.6 Power Management Techniques

This section elaborates on techniques which could further minimize power consumption of the spike sorting processor. Power gating [166] is a technique which may reduce the power dissipation

due to CMOS gate leakage in inactive processing units. For example, the power gating technique can be applied to the training unit in clustering once the active neurons in recording channel have been identified. Dynamic voltage scaling (DVS) [167] is another power efficient low-power design technique. In DVS, a higher supply voltage is considered when a processor is running at a high speed and a lower supply voltage is applied when it is not in its peak performance such as noncritical path. In the DVS technique, a dedicated power supply generator is essential which generates an adaptive supply voltage according to the operational frequency. For example, in the designed spike processor, a sequential DVS method may be used to activate the processing chain when a spike has been detected.

5.4.7 Towards the Next Generation of Adaptive Processing

5.4.7.1 Feature Extraction Development phases

The block diagram of the next *ADDs* model is depicted in Figure 5.36.

FV scaling: a crucial block in completing a highly reliable *FE* is the *FV* scaling unit. The main aim of this unit is to map the *FVs* from feature space to a distinguishable space. One possibility for implementation of *FV* scaling is using of the decomposition window amplitude (*amp*).

Dimensionality reduction methods: various number of geometric characteristics such as positive (or negative) signal energy, half-height position, right (or left) spike gradients, peak position, zero crossing points and positive(negative) lobes can be considered for alternative dimensionality reduction units. For example the conceptual illustration of power spectral density calculation for the decomposed ranges from high (δ=1) to low (δ=7) in the DR unit is shown in Figure 5.37.

5.4.7.2 Clustering Algorithm

The O-Sort clustering method can also be improved by applying the embedding frames in the clustering concept. The modification can be performed by employing other additional data clustering techniques, such as a tunable nearest neighbour setting, in the clustering unit to avoid

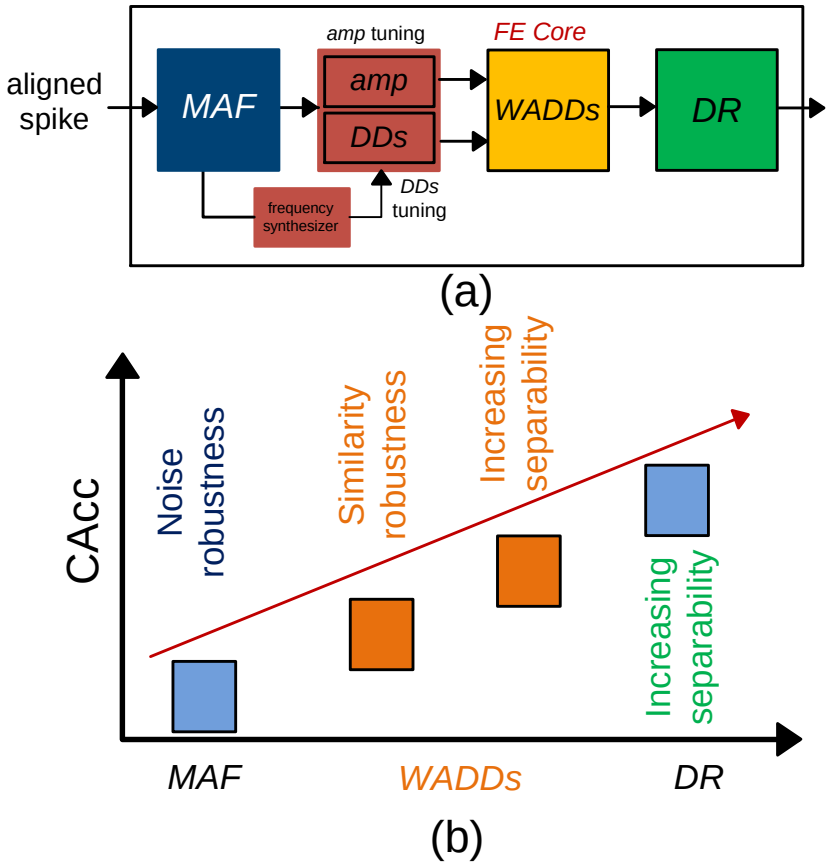

Figure 5.36: (a) Future implementation of *FE* (b) Separability level versus the utilized units. *WADDs* is the acronym of weighted adaptive discrete derivatives.

the cluster splitting and artificial clustering issues in O-Sort which may increase CAcc to over 85% without the use of *FE*.

5.5 Conclusion

In this chapter, an adaptive processing methodology is introduced to enhance the accuracy-power characteristics of SPSs by employing self-calibration of design features. As proof of the efficacy of the proposed processing framework, an adaptive spike processor is designed, fabricated in 180nm CMOS technology and evaluated using standard neural datasets. The adaptive adjustment strategy allows the alleviation of issues in spike sorting design such as low-power, low-area and high-accuracy by exploiting optimal processing techniques in each constituent block of the sorting chain. Conditional activation is considered for the detection unit to reduce the power consumption by 30%. Regarding detection performance, the probability of the presence of a spike is examined

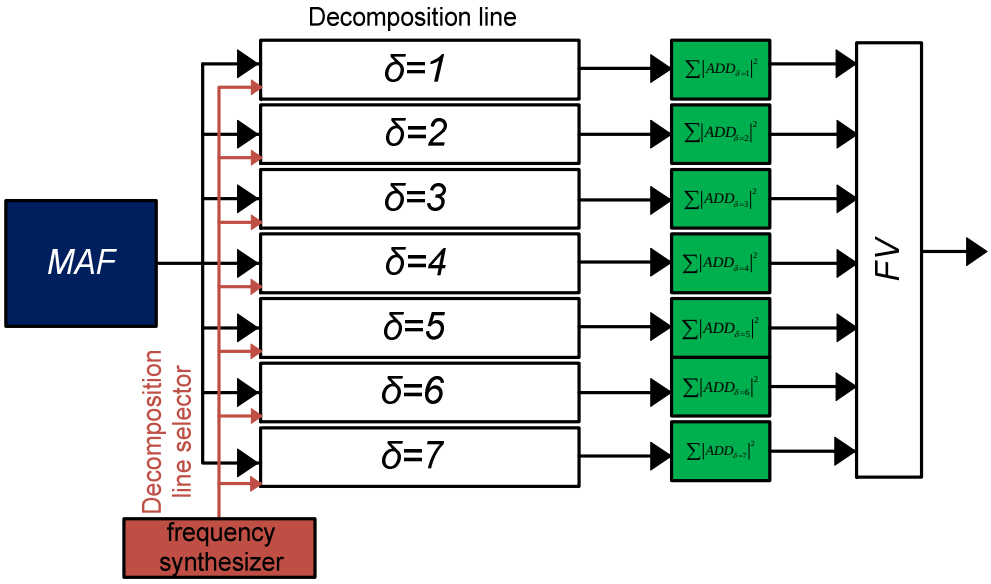

Figure 5.37: Demonstration of power spectral density calculation for the decomposed ranges from high (δ=1) to low (δ=7) in DR unit.

through two different metrics which are sorting threshold calculated based on noise standard deviation (σ_N), and modified power analysis (ωNEO) which results in significant detection improvement. This work is the first demonstration of a spike sorting processor with *FE*. In the *FE* block, a modified version of [148], *ADDs*, is used for selective spike waveforms decomposition. The *ADDs* is used to extract the features (positive and negative peaks) from the desired frequency bands. *ADD*s significantly improves the clustering accuracy by retaining the features with the highest level of non-Gaussianity while improving the in-band and out-band noise robustness using derivatives noise shaping property and moving average (*MA*) respectively. The number of locations for saving the spike features is reduced by almost 8X compared to processing the aligned spike waveforms.

In the clustering unit, a multi-aspect optimization methodology (power, area, accuracy) is proposed due to strict hardware implantable criteria. The register bank memory structure provides up to 2X dynamic power minimization in the training phase compared to the conventional implementation by using adaptive activation of the training memory locations. An interleaving technique is utilized to share computing resources in $\ell 1$-norm and merging units. Another technique employed is the reuse of logic in the cluster creation phase. This reduces the number of locations for buffering the transient cluster means. The clustering efficiency is improved using noise-tolerant $\ell 1$-norm

distance calculation and a modified version of O-Sort for sorting threshold calibration in the validation phase.

The prototype of the complete adaptive spike sorting processor is fabricated in 180nm CMOS technology with 84.5% overall accuracy and 148µW power consumption from 1.8V supply voltage. Better power performance characteristics are demonstrated compared to the state-of-the-art online and offline clustering methods. The focus of future work will be towards the development of more sophisticated *FE* methods, improving the clustering performance by development of highly accurate processing methods based on adaptive frameworks and realizing multi-channel processing.

CHAPTER 6

Conclusion

6.1 Original Contributions

This thesis has investigated a novel feature extraction method for hardware implementable purposes. Detailed in Chapter3 it uses discrete derivatives which is a simplified model of DWT as a reliable approach with low complexity for implanted hardware. The preservation of discrete derivative extremas successfully generates well-separated clusters. A standard dataset has been used to provide an efficient and unbiased comparison between the state-of-art spike sorting algorithms. This dataset contains spike waveforms similar to real recorded data with different levels of difficulty and noise. The results show that the feature extraction outperforms traditional and online feature extraction methods in terms of template similarity and noise immunity. To satisfy the main aim of increasing clustering performance while substantially reducing the complexity, the system is designed to be independent of multiplication from feature extraction to clustering ℓ_1-norm based O-Sort classifier. For a fair comparison against existing published systems, accuracy-complexity of all methods is discussed and compared using O-Sort.

To optimize the overall amount of power consumption in NFI, an optimization methodology has been introduced in Chapter 4 based on data conversion technique selection (e.g. CBSC) and NFI key parameters sensitivity examining on spike sorting accuracy. As shown in Figure 6.1, a hierarchical optimization scheme is considered to optimize the total power used in the neural acquisition path. In Chapter4 the thermal input referred noise of an ADC is considered as a contributing factor in LNA power consumption (NMF=α). Two types of data converters for biomedical applications have been used to evaluate the effect of noise-power of each data converter on NFI design (Layer 1). It is also explained that the behavior of the CBSC operation is similar to a clocked-inverter that charges/discharges the load capacitance and provides more power efficiency compared to the SAR ADC.

To estimate the theoretical NFI power bound, the effect of NFI key parameters (low-pass filtering, high-pass filtering, converter resolution and sampling frequency) have been examined on the spike sorting process (Layer 2). Defining the optimal NFI parameters were performed by utilizing a

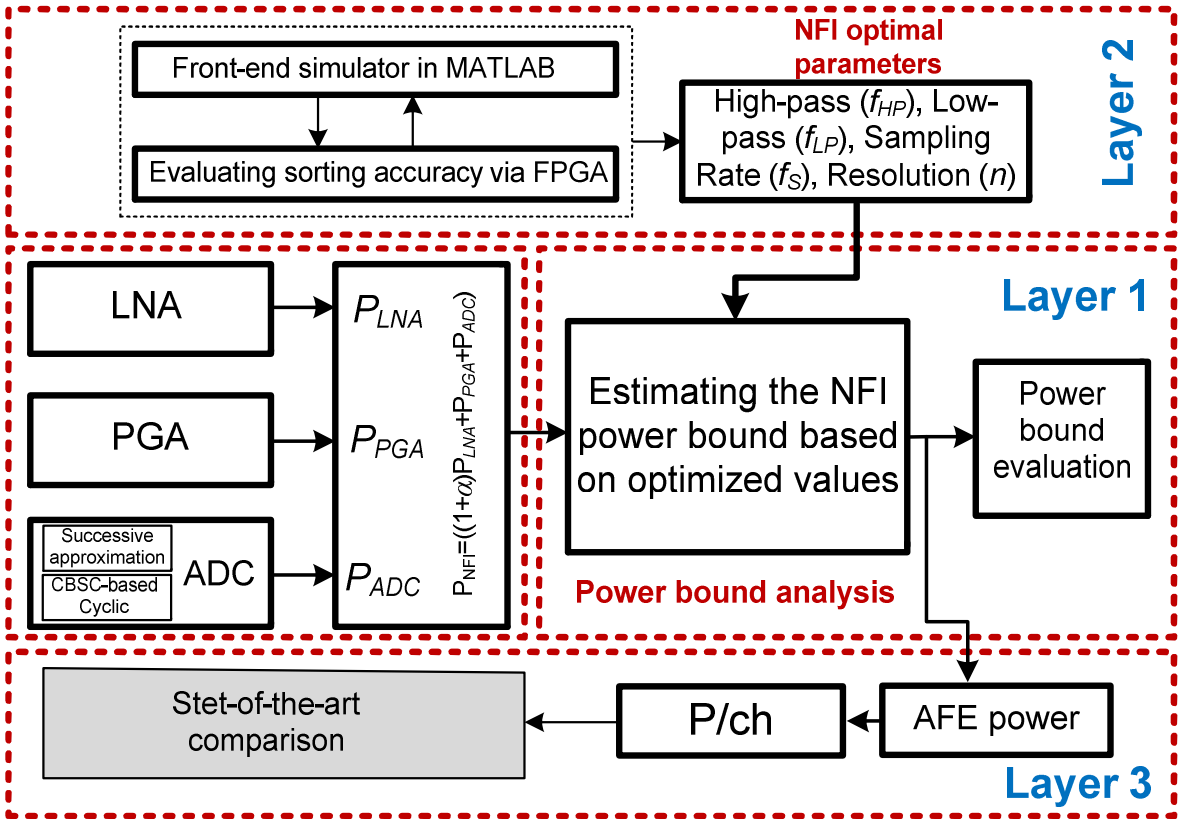

Figure 6.1: Proposed flow chart for NFI power optimization methodology.

variety of synthetic neural signals having different numbers of spike templates (e.g. different numbers of neurons), and different signal-to-noise ratios via an implemented spike processor on FPGA.

The NFI power bound (Layer3) has been estimated and compared with state-of-the-art. This chapter concluded that the NFI power consumption can be reduced by almost 30%.

The first generation of an adaptive spike sorting processor has been introduced and developed in Chapter 5 which significantly advances the state-of-the-art performance. The adaptive spike sorting enhances the accuracy-power characteristics by employing self-calibration of processing features by the concept of re-definition of the spike processing through reverse-adjustment (RA). The adaptive adjustment strategy improves performance in spike sorting such as low-power, low-area and high-accuracy by exploiting optimal processing techniques in each constituent block of the sorting chain. The processing method was designed especially for high density neural recording arrays (e.g., 512 recording channels) while satisfying the implantable devices restrictions. The

proposed method was implemented and verified in MATLAB/SIMULINK via complexity–performance metrics. Since the design of energy-efficient spike-sorting application specific integrated circuits (ASICs) is necessary to allow real-time multi-channel processing, the functionality verification of the proposed method was first examined on a programmable hardware development platform (i.e., FPGA).

The prototype of the complete adaptive spike sorting processor has been fabricated in 180 nm CMOS technology with 84.5% overall accuracy and 148µW power consumption from 1.8V supply voltage. The power-performance characteristics comparison to the state-of-the-art online and offline clustering methods has been demonstrated in Chapter 5.

6.2 Future Work

6.2.1 Multi-channel Processing

Future work will be on the development of multi-channel neural processing systems to satisfy strict limitations such as area and power consumption. The satisfactory performance of the complete processor will be followed by completing the implementation of an adaptive spike processor for multi-channel processing. The main aim is to design a scalable processor in order to accomplish processing of different number of channels (e.g 16, 32, 48, or 64 channels) for alternative BMI applications.

6.2.2 Brain Activity Analysis

The work presented in this thesis can be applied to the design of an implantable system as a powerful research tool for neuroscience applications. The aim is to investigate novel processing methods to increase the number of processing channels (e.g. >1K) for the same amount of power budget in the current state-of-the-art. This can be achieved by introducing a new processing paradigm for projection of recorded signals.

BIBLIOGRAPHY

[1] Action for Blind People. (2013, Sep.) Facts and figures about issues around sight loss. [Online]. Available: https://www.actionforblindpeople.org.uk/about-us/media-centre/facts-and-figures-about-issues-around-sight-loss/

[2] NA. Fineberg, PM. Haddad, L. Carpenter, B. Gannon, R. Sharpe, AH. Young, et al., “The size, burden and cost of disorders of the brain in the UK”. J Psychopharmacol 2013; 27: 761-770.

[3] L. Horsfall, I. Petersen, K. Walters, A. Schrag Time trends in incidence of Parkinson's disease diagnosis in UK primary care J Neurol, 260 (2013), pp. 1351–1357.

[4] Ponce F. A., Asaad W. F., Foote K. D., Anderson W. S., Rees Cosgrove G., Baltuch G. H., et al. . (2016). Bilateral deep brain stimulation of the fornix for Alzheimer's disease: surgical safety in the advance trial. J. Neurosurg. 125, 75–84. 10.3171/2015.6.JNS15716

[5] K. Ziegler-Graham, EJ. MacKenzie, PL. Ephraim, TG. Travison, R. Brookmeyer. , “Estimating the Prevalence of Limb Loss in the United States: 2005 to 2050,” Archives of Physical Medicine and Rehabilitation 2008; 89(3):422-9.

[6] M. Owings, LJ. Kozak, “National Center for Health S. Ambulatory and Inpatient Procedures in the United States,” 1996. Hyattsville, Md.: U.S. Dept. of Health and Human Services, Centers for Disease Control and Prevention, National Center for Health Statistics; 1998.

[7] HCUP Nationwide Inpatient Sample (NIS). Healthcare Cost and Utilization Project (HCUP). Rockville, MD: Agency for Healthcare Research and Quality; 2009.

[8] W. Wattanapanitch, M. Fee, and R. Sarpeshkar, “An energy-efficient micropower neural recording amplifier,” IEEE Trans. Biomed. Circuits Syst., vol. 1, no. 2, pp. 136–147, Jun. 2007.

[9] R. R. Harrison and C. Charles, “A low-power low-noise CMOS amplifier for neural recording applications,” IEEE J. Solid-State Circuits, vol. 38, no. 6, pp. 958–965, Jun. 2003.

[10] G. Charvet, et al., "A modular 256-channel Micro Electrode Array platform for in vitro and in vivo neural stimulation and recording: BioMEA" in EMBC, 2010 Annual International Conference of the IEEE, 2010, pp. 1804-1807.

[11] I. H. Stevenson and K. P. Kording, “How advances in neural recording affect data analysis," Nature neuroscience, vol. 14, no. 2, pp. 139-142, 2011.

[12] T. Sepke, J. K. Fiorenza, C. G. Sodini, P. Holloway, and H.-S. Lee, “Comparator-based switched-capacitor circuits for scaled CMOS technologies,” in ISSCC Digest of Technical Papers, 2006, pp. 220–221.

[13] 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. Neurosci. Methods,* vol. 154, no. 1-2, pp. 204–224, Jun. 2006.

[14] N. Kipnis, “Luigi Galvani and the Debate on Animal Electricity, 1791-1800,” *Ann. Sci.*, vol. 44, no. 2, pp. 107–142, 1987.

[15] J.C. Sanchez, J.C. Principe, T. Nishida, R. Bashirullah, J.G. Harris, and J.A.B. Fortes, “Technology and Signal Processing for Brain-Machine Interfaces,” *IEEE Signal Processing Magazine*, vol. 25, no. 1., Jan 2008, pp. 29-40.

[16] M. Ameer, M. Zamani, R. Bayford and A. Demosthenous, “Patient Specific Parkinson’s Disease Detection for Adaptive Deep Brain Stimulation,” Accepted in International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)**,** Milano, Italy, August 2015.

[17] M.D. Linderman, G. Santhanam, C. T. Kemere, V. Gilja, S. O’Driscoll, B.M. Yu, A. Afshar, S.I. Ryu, K.V. Shenoy, and T.H. Meng, “Signal Processing Challenges for Neural Prostheses,” *IEEE Signal Processing Magazine*, special issue on brain-computer interfaces, vol. 25, no. 1, Jan 2008, pp. 18-28.

[18] I. H. Stevenson and K. P. Kording, “How advances in neural recording affect data analysis," Nature neuroscience, vol. 14, no. 2, pp. 139-142, 2011.

[19] M. Capogrosso. *et al*. *Nature* 539, pp. 284–288, 2016.

[20] S. Morefiel, E. Keefer, K. Chapman, and G. Gross, “Drug evaluations using neuronal networks cultured on microelectrode arrays,” *Biosensors and Bioelectronics*, vol. 15, no. 7-8, Oct 2000, pp. 383-396

[21] “Medtronic deep brain stimulators," 2011. [Online]. Available: http://professional.medtronic.com/pt/neuro/dbs-md/prod/index.htm#.WFonM9KLTg8

[22] “Advanced Arm Dynamics," 2011. [Online]. Available: http://armdynamics.com/

[23] Y. Sankai, “Hal: Hybrid assistive limb based on cybernics," in Robotics Research. Springer, 2011, pp. 25-34.

[24] “Cochlear implants," 2011. [Online]. Available: http://www.nidcd.nih.gov/health/hearing/pages/coch.aspx

[25] L. Resnik, S. L. Klinger, and K. Etter, “The DEKA Arm: Its features, functionality, and evolution during the Veterans A airs Study to optimize the DEKA Arm," Prosthetics and orthotics international, vol. 38, no. 6, pp. 492-504, 2014.

[26] “Nautilus eeg cap,” http://www.gtec.at/Products/Hardware-and-Accessories/g. Nautilus-Specs-Features.

[27] J. L. Roland, C. D. Hacker, J. D. Breshears, C. M. Gaona, R. E. Hogan, H. Burton, M. Corbetta, and E. C. Leuthardt, “Brain mapping in a patient with congenital blindness- a case for multimodal approaches,” Frontiers in human neuroscience, vol. 7, 2013.

[28] E. Fernandez, B. Greger, PA. House *et al.* Acute human brain responses to intracortical microelectrode arrays: challenges and future prospects. *Front. Neuroeng.* 2014; 7: 24.

[29] L. A. Farwell and E. Donchin, “Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," Electroencephalography and clinical Neurophysiology, vol. 70, no. 6, pp. 510-523, 1988.

[30] 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, no. 51, pp. 17 849-17854, 2004.

[31] F. Galán, M. Nuttin, E. Lew, P.W. Ferrez, G. Vanacker, J. Philips and J.D.R. Millán, “A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots,” *Clin. Neurophysiol*2008, 119, 2159–2169.

[32] M. A. Lebedev and M. A. Nicolelis, “Brain-machine interfaces: past, present and future," Trends in Neurosciences, vol. 29, no. 9, pp. 536-546, 2006.

[33] J. J. Shih, D. J. Krusienski, and J. R. Wolpaw, “Brain-computer interfaces in medicine," in Mayo Clinic Proceedings, vol. 87, no. 3. Elsevier, 2012, pp. 268-279.

[34] L. F. Nicolas-Alonso and J. Gomez-Gil, “Brain computer interfaces, a review," Sensors, vol. 12, no. 2, pp. 1211-1279, 2012.

[35] B. Gosselin, “Recent advances in neural recording microsystems," Sensors, vol. 11, no. 5, pp. 4572-4597, 2011.

[36] E. C. Leuthardt, G. Schalk, J. R. Wolpaw, J. G. Ojemann, and D. W. Moran, “A brain-computer interface using electrocorticographic signals in humans," Journal of neural engineering, vol. 1, no. 2, p. 63, 2004.

[37] W. Wang, A. Degenhart, J. Collinger, R. Vinjamuri, G. Sudre, P. Adelson, D. Holder, E. Leuthardt, D. Moran, M. Boninger, A. Schwartz, D. Crammond, E. Tyler-Kabara, and D. Weber, “Human motor cortical activity recorded with micro-ecog electrodes, during individual finger movements," in Proc. IEEE International Conference of Engineering in Medicine and Biology Society, Sept 2009, pp. 586-589.

[38] R. Muller, H.-P. Le, W. Li, P. Ledochowitsch, S. Gambini, T. Bjorninen, A. Koralek, J. Carmena, M. Maharbiz, E. Alon, and J. Rabaey, “A minimally invasive 64-channel wireless μecog implant," IEEE Journal of Solid-State Circuits, vol. 50, no. 1, pp. 34-359, Jan 2015.

[39] J. Viventi, D.-H. Kim, L. Vigeland, E. S. Frechette, J. A. Blanco, Y.-S. Kim, A. E. Avrin, V. R. Tiruvadi, S.-W. Hwang, A. C. Vanleer et al., “Flexible, foldable, actively multiplexed, high-density electrode array for mapping brain activity in vivo," Nature neuroscience, vol. 14, no. 12, pp. 1599-1605, 2011.

[40] N. J. Hill, D. Gupta, P. Brunner, A. Gunduz, M. A. Adamo, A. Ritaccio, and G. Schalk, “Recording human electrocorticographic (ECoG) signals for neuroscientific research and real-time functional cortical mapping," Journal of visualized experiments: JoVE, no. 64, 2012.

[41] G. Buzsaki, C. A.Anastassiou, and C. Koch, “The origin of extracellular fields and currents–EEG, ECoG, LFP and spikes,” *Nat. Rev. Neurosci*. 13, 407–420, 2012.

[42] R. A. Normann, “Microfabricated electrode arrays for restoring lost sensory and motor functions,” in *Proc. Int. Conf. TRANSDUCERS*, Solid-State Sensors, Actuators and Microsystems, pp. 959–962, 2003.

[43] G. Mclaughlin and M. Imran, “Set it and forget it: Innovations in implantable medical technology,” IEEE SSC Magazine, vol. 4, no. 2, pp. 30-33, 2012.

[44] A. L. Hodgkin and A. F. Huxley, “Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo," *J. Physiol.*, vol. 116, no. 4, pp. 449-472, 1952.

[45] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500-544, 1952.

[46] J. W. Clarck, “The origin of biopotentials” in Medical Instrumentation, Application and Design, 4th ed., J.G. Webster, Ed. New Jersey: *John Wiley & Son.*, 2010, pp. 126-188.

[47] H. Lodish, A. Berk, S. L. Zipursky, P. Matsudaira, D. Baltimore, and J. Darnell, Eds,. Molecular Cell Biology. W. H. Freeman, 2000, vol. 2.

[48] J. J. Struijk, “Passive models of excitable cells,” in *Neuroprosthetics: Theory and Practice*, K. W. Horch and G. S. Dhillon, Ed. Singapoor: World Scientific Publishing Co. pte. Ltd., 2007, pp. 3-2"9.

[49] G. Buzsaki, “Large-scale recording of neuronal ensembles,” Nature neuroscience, vol. 7, no. 5, pp. 446-451, 2004.

[50] Richard O. Duda , Peter E. Hart , David G. Stork, Pattern Classification (2nd Edition), Wiley-Interscience, 2000

[51] G. Buzsáki, C.A. Anastassiou, and C. Koch, “The origin of extracellular fields and currents-EEG, ECoG, LFP and spikes,” *Nature Reviews Neuroscience*., vol. 13, no. 6, pp. 407-420, 2021.

[52] T. M. Sees et al., “Characterization of tissue morphology, angiogenesis, and temperature in adaptive response of muscle tissue to chronic heating,” Lab. Investigation, vol. 78, no. 12, 1998.

[53] S. Mandal and R. Sarpeshkar, “A bi-directional wireless link for neuroprosthesis that minimizes implanted power consumption,” in IEEE Conf. Biomed. Circuits Syst., Nov. 2007, pp. 45–48.

[54] J. L. Bohorquez, J. L. Dawson, and A. P. Chandrakasan, “A 350 μW CMOS MSK transmitter and 400 W OOK super-regenerative receiver for medical implant communications,” in 2008 Symp. VLSI Circuits Dig. Tech. Papers, 2008, pp. 32–33.

[55] Ma, Chao, Changhui Hu, Jiao Cheng, Lingli Xia, and Patrick Yin Chiang. "A Near-Threshold, 0.16 nJ/b OOK-Transmitter With 0.18 nJ/b Noise-Cancelling Super-Regenerative Receiver for the Medical Implant Communications Service." IEEE transactions on Biomedical Circuit and Systems, vol. 7, no. 6, pp. 841–850, 2013.

[56] K. Harris, D. Henze, J. Csicsvari, H. Hirase, and G. Buzsaki, “Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements," Journal of Neurophysiology, vol. 84, no. 1, pp. 401-414, 2000.

[57] Plexon, “Multichannel Acquisition Processor (MAP),” 2012. [Online]. Available: http://www.plexon.com/product/Multichannel Acquisition Processor MAP.

[58] [Online].Available: http://www.ced.co.uk/indexu.shtml.

[59] [Online].Available:http://www2.le.ac.uk/departments/engineering/research/bioengineering/neuroengineering-lab/spike-sorting.

[60] H. Abdi and L. Williams, “Principal component analysis," Wiley Interdisciplinary Reviews: Computational Statistics, vol. 2, no. 4, pp. 433-459, 2010.

[61] A. Hyvärinen , J. Karhunen, and E. Oja, Independent component analysis. Wiley interscience, 2001, vol. 26.

[62] Y. Ghanbari, L. Spence, and P. Papamichalis, “A graph-laplacian-based feature extraction algorithm for neural spike sorting," in Proc. IEEE International Conference of Engineering in Medicine and Biology Society, 2009, pp. 3142-3145.

[63] J. Sokolić, M. Zamani, A. Demosthenous, and M. Rodrigues. "A feature design framework for hardware efficient neural spike sorting." 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 1516-1519. IEEE, 2015.

[64] S. Shoham, M. Fellows, and R. Normann, “Robust, automatic spike sorting using mixtures of multivariate t-distributions,” Journal of neuroscience methods, vol. 127, no. 2, pp. 111-122, 2003.

[65] D. A. Reynolds. Gaussian mixture models. Pages 659-663, 2009.

[66] J. Holleman, A. Mishra, C. Diorio, and B. Otis, “A micro-power neural spike detector and feature extractor in. 13 m CMOS,” in Proc. IEEE Custom Integrated Circuits Conf. (CICC 2008), 2008, pp. 333–336.

[67] A. Zviagintsev, Y. Perelman, and R. Ginosar, “Low power architectures for spike sorting,” in Proc. 2nd Int. IEEE EMBS Conf. Neural Eng., Mar. 2005, pp. 162–165.

[68] A. Bonfanti, T. Borghi, R. Gusmeroli, G. Zambra, A. Oliyink, L. Fadiga, A. Spinelli, and G. Baranauskas, “A low-power integrated circuit for analog spike detection and sorting in neural prosthesis systems," in Proc. IEEE Biomedical Circuits and Systems Conference, 2008, pp. 257 -260.

[69] A. Mendez, A. Belghith, M. Sawan, “A DSP for sensing the bladder volume through afferent neural pathways,” IEEE Trans. Biomed. Eng., vol. 8, pp. 552–564, Jun. 2014.

[70] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, and C. Poon, " Real-time fpga-based multichannel spike sorting using hebbian eigen filters," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, no. 99, pp. 1-1, 2011.

[71] T.C.Chen, W.Liu, and L.G.Chenet al., "128-channel spike sorting processor with a parallel-folding structure in 90 nm process," in Proc. IEEE Int. Symp. Circuits and Systems, 2009, pp. 1253–1256.

[72] A. Bhaduri, Y. Enyi, and B. Arindam. "Pulse-based feature extraction for hardware-efficient neural recording systems." IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1842–1845, May 2016.

[73] T.-T. Liu and J. M. Rabaey, "A 0.25 V 460 nW asynchronous neural signal processor with inherent leakage suppression," IEEE J. SolidState Circuits, vol. 48, no. 4, pp. 897–906, Apr. 2013.

[74] V. Karkare, S. Gibson, and D. Marković, "A 130-uW, 64-Channel Neural Spike-Sorting DSP Chip," IEEE J. Solid-State Circuits, vol. 46, no. 5, pp. 1214-1222, May 2011.

[75] A. Kamboh, and A. Mason, "Computationally efficient neural feature extraction for spike sorting in implantable high-density recording systems," IEEE Trans. Neural Syst. Rehabil. Eng., vol.21, no. 1, pp. 1–9, Jan. 2013.

[76] V. Karkare, S. Gibson, and D. Marković, "A 75 μW, 16-channel Neural Spike Sorting Processor with Unsupervised Clustering," IEEE J. Solid-State Circuits, vol. 48, no. 9, pp. 2230-2238, Sep. 2013.

[77] M. Chae et al., "A 128-channel 6 mW wireless neural recording IC with on-the-fly spike sorting and UWB transmitter," in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers, 2008, pp. 146–147.

[78] Hyo-Gyuem Rhew, Jaehun Jeong, Jeffrey A. Fredenburg, Sunjay Dodani, Parag G. Patil, and Michael P. Flynn, "A Fully Self-Contained Logarithmic Closed-Loop Deep Brain Stimulation SoC With Wireless Telemetry and Wireless Power Management," IEEE Journal of Solid-State Circuits, Vol.49, Issue 10, pp. 1-15, August 28, 2014.

[79] K. G. Oweiss, D. J. Anderson, and M. M. Papaefthymiou, "Optimizing signal coding in neural interface system-on-a-chip modules," in Proc. 25th Annu. Conf. IEEE EMBS, Cancun, Mexico, Sep. 2003, pp. 3325–3328.

[80] A. M. Sodagar, K. D. Wise, and K. Najafi, “A fully integrated mixed-signal neural processor for implantable multichannel cortical recording,” IEEE Trans. Biomed. Eng., vol. 54, no. 6, pp. 1075–1088, Jun. 2007.

[81] F. Chen, A. P. Chandrakasan, and V. Stojanovic´, “Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors,”IEEE J. Solid-State Circuits, vol. 47, pp. 744–756, Mar. 2012.

[82] R. Agarwal, S. Sonkusale, Input-Feature Correlated Asynchronous Analog to Information Converter for ECG Monitoring, IEEE Transactions of Biomedical Circuits and Systems (TBCAS), Vol.5, Issue 5, pp.459-467, Oct 2011.

[83] J. Lee, H. G. Rhew, D. R. Kipke, and M. P. Flynn, “A 64-channel programmable closed-loop neurostimulator with 8-channel neural amplifier and logarithmic ADC,” IEEE J. Solid-State Circuits, vol. 45, no. 9, pp. 1935–1945, Sep. 2010.

[84] M. Shoaib, N. K. Jha, and N. Verma, “A compressed-domain processor for seizure detection to simultaneously reduce computation and communication energy,” in Custom Integrated Circuits Conference (CICC), 2012 IEEE. IEEE, 2012, pp. 1–4.

[85] M. Shoaib, N. K. Jha, and N. Verma, “Signal Processing with Direct Computations on Compressively-sensed Data,” IEEE Trans. VLSI Systems., vol.23, pp. 30–43, Jan. 2015.

[86] J. Lian, G. Garner, D. Muessig, and V. Lang, “A simple method to quantify the morphological similarity between signals,” J. Signal Process., vol. 90, no. 2, pp. 684–688, Feb 2010.

[87] I. Obeid and P. D. Wolf, “Evaluation of spike-detection algorithms for a brain-machine interface application,” IEEE Trans. Biomed. Eng., vol.51, no. 6, pp. 905–911, Jun. 2004.

[88] J. F. Kaiser, “On a simple algorithm to calculate the ‘energy’ of a signal,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP ’90), Albuquerque, NM, Apr. 1990, vol. 1, pp. 381–384.

[89] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, “Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering,” Neural Comp., vol. 16, no. 8, pp. 1661–1687, Aug. 2004.

[90] 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, no. 8, pp. 999–1011, Aug. 2003.

[91] M. S. Lewicki, “A review of methods for spike sorting: The detection and classification of neural action potentials," Network: Comput. Neural Syst., vol. 9, no. 4, pp. R53-R78, Nov. 1998.

[92] 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, no. 5, pp. 403-412, May 1997.

[93] 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, no. 4, pp. 738-746, 2006.

[94] M. Abeles and M. H. Goldstein, Jr., “Multispike train analysis,”Proc. IEEE, vol. 65, no. 5, pp. 762–773, May 1977.

[95] J. C. Letelier and P. P. Weber, “Spike sorting based on discrete wavelet transform coefficients,” J. Neurosci. Method.s, vol. 101, pp. 93–106, 2000.

[96] C. Valens, “A really friendly guide to wavelets.” http://www.robots.ox.ac.uk/ parg/mlrg/papers/arfgtw.pdf, 1999.

[97] R. Bestel, A. Daus, and C. Thielemann, “A novel automated spike sorting algorithm with adaptable feature extraction,” Journal of Neuroscience Methods, vol. 211, no. 1, pp. 168– 178, 2012.

[98] E. Hulata et al., “A method for spike sorting and detection based on wavelet packets and Shannonís mutual information,” J. Neurosci. Methods, vol. 117, no. 1, pp. 1–12, May.2002.

[99] H. W. Lilliefors, “On the Kolmogorov-Smirnov test for normality with mean and variance unknown,” J. Amer. Statist. Assoc., vol. 62, no. 318, pp. 399–402, June 1967.

[100] M. Zamani and A. Demosthenous, “Dimensionality Reduction Using Asynchronous Sampling of First Derivative Features for Real-Time and Computationally Efficient Neural Spike Sorting,” IEEE International Conference on Electronics, Circuits, and Systems (ICECS), Abu Dhabi, UAE, December 2013.

[101] 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. Rehabil. Eng., vol. 18, no. 5, pp. 469–78, Oct. 2010.

[102] F. Wood, M. J. Black, C. Vargas-Irwin, M. Fellows, and J. P. Donoghue, “On the variability of manual spike sorting,” IEEE Trans. Biomed. Eng., vol. 51, no. 6, pp. 912–918, 2004.

[103] J. Dai, X. Liu, Y. Yi, H. Zhang, J. Wang, S. Zhang, and X. Zheng, “Experimental study on neuronal spike sorting methods,” IEEE Future Generation Communication and Networking Conference, vol. 2, pp. 230-233, 2008.

[104] K. Harris, D. Henze, J. Csicsvari, H. Hirase, and G. Buzsáki, “Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements,” Journal of Neurophysiology, vol. 84, no. 1, pp. 401-414, 2000.

[105] K. Kim and S. Kim, “Neural spike sorting under nearly 0-dB signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier,” IEEE Transactions on Biomedical Engineering, vol. 47, no. 10, pp. 1406-1411, 2000.

[106] C. Zhang, X. Zhang, M. Q. Zhang, and Y. Li, “Neighbor number, valley seeking and clustering,” Pattern Recognit. Lett., vol. 28, no. 2, pp. 173–180, 2007.

[107] R. Xu and D. II Wunsch, “Clustering Algorithms in Biomedical Research: A Review,” IEEE Reviews in Biomedical Engineering., vol. 3, pp. 120–154, 2010.

[108] D. Novák, J. Wild, T. Sieger, and R. Jech, “Identifying number of neurons in extracellular recording,” in Proc. 4th Int. IEEE EMBS Conf. Neural Eng., Antalya, Turkey, 2009, pp. 742–745.

[109] Z. Nadasdy, R. Quian Quiroga, Y. Ben-Shaul, B. Pesaran, D. A. Wagenaar, and R. Andersen, “Comparison of unsupervised algorithms for on-line and off-line spike sorting,” presented at the *32nd Ann. Meeting of the Society for Neuroscience*, Orlando, FL, USA, 2002 [Online]. Available: http://www.vis.caltech.edu/~zoltan/

[110] M. A. L. Nicolelis, “Actions from thoughts,” Nature, vol. 409, pp. 403–407, 2001.

[111] S. E. Paraskevopoulou, D. Y. Barsakcioglu, M. R. Saberi MR, A. Eftekhar A, and T. G. Constandinou, “Feature extraction using first and second derivative extrema (FSDE) for real-time and hardware-efficient spike sorting,” *J. Neurosci. Methods*, vol. 215, no. 1-2, pp. 29–37, Jan. 2013.

[112] Y. Yuan, C. Yang, and J. Si, “The m-sorter: an automatic and robust spike detection and classification system,”*J. Neurosci. Methods*, vol. 210, no. 2, pp. 281–290, Sep. 2012.

[113] D. M. Schwarz, M. S. A. Zilany, M. Skevington, N. J. Huang, B. C. Flynn, and L. H. Carney, “Semi-supervised spike sorting using pattern matching and a scaled Mahalanobis distance metric,” *J. Neurosci. Methods*, vol. 206, no. 2, pp. 120–131, May 2012.

[114] Y. Ghanbari, P. E. Papamichalis, and L. Spence, “Graph-Laplacian features for neural waveform classification,” *IEEE Trans. Biomed. Eng.*, vol. 58, no. 5, pp. 1365–1372, Jun. 2011.

[115] C. Pouzat, O. Mazor, and G. Laurent, “Quality metrics to accompany spike sorting of extracellular signals,” *J. Neurosci. Methods*, vol. 31, no. 24, pp. 8699–8705, Jun. 2011.

[116] P. Dayan and L.F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, Cambridge MA, Inc., 2001.

[117] C. Pouzat, O. Mazor, and G. Laurent, “Using noise signature to optimize spike-sorting and to assess neuronal classification quality,” *J. Neurosci. Methods*,vol. 122, no. 1, pp. 43–57, Dec. 2002.

[118] A. Zviagintsev, Y. Perelman, and R. Ginosar, “Low-power architectures for spike sorting,” in *Proc. 2ndIEEE Int. Conf. EMBS on Neural Engineering.*, Arlington, VA, USA, 2005, pp. 162-165.

[119] R. R. Harrison, P. T. Watkins, R. J. Kier, R. J. Lovejoy, B. Greer, and F. Solzbacher, “A low-power integrated circuit for a wireless 100-electrode neural recording system,” *IEEE J. Solid-State Circuits.*, vol. 41, no.1, pp. 123–133, Jan. 2007.

[120] D. Cirmirakis, D. Jiang, A. Demosthenous, N. Donaldson, and T. Perkins,, “A fast passive phase shift keying modulator for inductively coupled implanted medical devices,” in *Proc. 38th Eur. Solid-State Circuits Conf.*, Bordeaux, France, 2012, pp. 301–304.

[121] I. H. Stevenson and K. P. Kording, “How advances in neural recording affect data analysis,” *Nat. Neurosci*., vol. 14, no. 2, pp. 139–142, Feb. 2011.

[122] A. Berényi, Z. Somogyvari, A. J. Nagy, L. Roux, J. D. Long, S. Fujisawa, E. Stark, A. Leonardo, T. D. Harris, and G. Buzsaki, “Large-scale, high-density (up to 512 channels) recording of local circuits in behaving animals,” *J. Neurophysiol.*, Dec. 2013 (Epub ahead of print).

[123] D. J. Chew, L. Zhu, E. Delivopoulos, I. R. Minev, K. M. Musick, C. A. Mosse, M. Craggs, N. Donaldson, S. P. Lacour, S. B. McMahon, and J. W. Fawcett, “A microchannel neuroprosthesis for bladder control after spinal cord injury in rat,” *Sci. Transl. Med.*, vol. 5, no. 210ra155, Nov. 2013.

[124] D. A. Borton, M. Yin, J. Aceros, and A. Nurmikko, “An implantable wireless neural interface for recording cortical circuit dynamics in moving primates,” *J. Neural Eng*., vol. 10, 026010 (16pp), Feb. 2013.

[125] F. Medeiro, B. Pérez-Verdú, A. Rodríguez-Vázquez, and J. Huertas, “A vertically-integrated tool for automated design of ΣΔ modulators,” IEEE J. Solid-State Circuits, vol. 30, pp. 762–772, July 1995.

[126] M. Zamani, S., J-Ashtiani, M. Dousti and M. N.-Moghadasi, “A10b, 20-MS/s, 2.6mW fully differential CBSC pipelined ADC in 0.18μm CMOS,” IEICE Electron. Express, vol. 7, no. 23, pp. 1694-1701, December 2010.

[127] M. Zamani, M. Taghizadeh, M. Naser-Moghadasi and B.S. Virdee, “A 5th-Order ΣΔ Modulator with Combination of Op-Amp and CBSC Circuit for ADSL Applications,” Analog Integr. Circuits Signal Process., vol. 71, no. 1, pp. 143–150, March. 2012.

[128] L. Brooks and H.-S. Lee, “A zero-crossing-based 8 b 200 MS/s pipelined ADC,” in Proc. IEEE Int. Solid-State Circuits Conf., Feb. 2007, pp. 460–461.

[129] N. Dolev, et al., “A 12-bit, 200-MS/s, 11.5-mW Pipeline ADC using a Pulsed Bucket Brigade Front-End,” Symp. VLSI Circuits, pp. C98–C99, June 2013.

[130] M. Steyaert, W. Sansen, and C. Zhongyuan, “A micropower low-noise monolithic instrumentation amplifier for medical purposes,” IEEE J. Solid-State Circuits, vol. 22, no. 6, pp. 1163–1168, 1987.

[131] R. R. Harrison and C. Charles, “A low-power low-noise CMOS amplifier for neural recording applications,” IEEE J. Solid-State Circuits, vol. 38, no. 6, pp. 958–965, Jun. 2003.

[132] R. J. Baker et al., CMOS Circuit Design, Layout and Simulation, Wiley Interscience, 1st Edition, 1998.

[133] T. Sepke, P. Holloway, C. G. Sodini, and H.S. Lee, “Noise Analysis for Comparator Based Circuits,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 56, no. 3, pp. 541–553, Mar. 2009.

[134] W. P. Zhang and X. Tong, “Noise Modeling and Analysis of SAR ADCs,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst. , vol. 23, no. 12, pp. 2922–2930, Dec. 2015.

[135] P. Nuzzo, F. De Bernardinis, P. Terreni, and G. Van der Plas, “Noise analysis of regenerative comparators for reconfigurable ADC architectures,” IEEE Trans. Circuits Syst. I, Reg. Papers , vol. 55, no. 6, pp. 1441–1454, Jul. 2008.

[136] J. Kim, B. S. Leibowitz, J. Ren, and C. J. Madden, “Simulation and analysis of random decision errors in clocked comparators,” IEEE Trans. Circuits Syst. I, Reg. Papers , vol. 56, no. 8, pp. 1844–1857, Aug. 2009.

[137] D. Schinkel, E. Mensink, E. Klumperink, E. van Tuijl, and B. Nauta, “A double-tail latch-type voltage sense amplifier with 18 ps setup + hold time,” in Proc. IEEE ISSCC , Feb. 2007, pp. 314–605.

[138] X. Zou, X. Xu, L. Yao, and Y. Lian, “A 1-V 450-nW fully integrated programmable biomedical sensor interface chip,” IEEE J. Solid-State Circuits , vol. 44, no. 4, pp. 1067–1077, Apr. 2009.

[139] T. Sundström, B. Murmann and C. Svensson, “Power dissipation bounds for high-speed Nyquist analog-to-digital converters,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 3, pp. 509–518, March. 2009.

[140] B. Murmann, “Limits on ADC power dissipation,” in Analog Circuit Design, M. Steyaert, A. H. M. Roermund, and J. H. van Huijsing, Eds. New York: Springer, 2006.

[141] D.A. Johns and K. Martin, “Analog Integrated Circuit Design” (John Wiley and Sons, Inc., 1997)

[142] I. E. Opris, “Noise estimation in strobed comparators,” Electron. Lett,. vol. 33, no. 15, pp. 1273–1274, Jul. 1997.

[143] T. O. Anderson, “Optimum control logic for successive approximation A-D converters,” Comput. Design, vol. 11, no. 7, pp. 81-86, July 1972.

[144] N. Weste and K. Eshraghian, Principles of CMOS VLSI Design. Addison-Wesley, 1993.

[145] C. C. Liu, S. J. Chang, G. Y. Huang, and Y. Z. Lin, “A 10-bit 50-MS/s SAR ADC with a monotonic capacitor switching procedure,” IEEE J. Solid-State Circuits, vol. 45, no. 4, pp. 731–740, Apr. 2010.

[146] M. Zamani, C. Eder, and A. Demosthenous, “Power Dissipation Limits of CBSC-Based Pipelined Analog-to-Digital Converters ,” in Proc. of IEEE Very Large Scale Integration (VLSI-SoC) , pp. 352-357, Oct 2013.

[147] J.-T. Wu and B. A. Wooley, “A 100 MHz pipelined CMOS comparator,” IEEE J. Solid-State Circuits, vol. 23, no. 6, pp. 1379–1385, Dec. 1988.

[148] M.-C. Huang and S. L. Liu, “A fully differential comparator-based switched-capacitor modulator,” IEEE Trans. Circuits Syst. II, Exp. Briefs , vol. 56, no. 5, pp. 369–373, May 2009.

[149] S.-K. Shin and Sepke et al., “A fully-differential zero-crossing-based 1.2 V 10 b 26 MS/s pipelined ADC in 65 nm CMOS,” in Symp. VLSI Circuits Dig. Tech. Papers , Jun. 2008, pp. 218–219

[150] 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. Rehabil., vol. 22, pp. 716–726, Jul. 2014.

[151] http://crcns.org/data-sets/hc/.exp

[152] S. I. Chang, K. AlAshmouny, M. McCormick, Y. C. Chen, and E. Yoon, “BioBolt: A Minimally-Invasive Neural Interface for Wireless Epidural Recording by Intra-Skin Communication,” in Proc. Int. Symp. VLSI Circuits, 2011, Kyoto, Japan, June 2011,pp. 146–147.

[153] Hae-Seung Lee. Limits of power consumption in analog circuits. In VLSI Circuits, 2007 IEEE Symposium on, pages 6-9, 2007.

[154] S. Lee, et al., “A 12b 5-to-50MS/s 0.5V-to-1V Voltage Scalable Zero-Crossing Based Pipelined ADC,” Proc. IEEE ESSCIRC , pp. 355-358, 2011.

[155] J. Max, “Quantizing for minimum distortion,” IEEE Trans. Inform. Theory, vol. 6, no. 1, pp. 7–12, Mar. 1960.

[156] A. Rodríguez-Pérez, J. Ruiz-Amaya, M. Delgado-Restituto and Á. Rodríguez-Vázquez, "A Low-Power Programmable Neural Spike Detection Channel with Embedded Calibration and Data Compression". IEEE Transactions on Biomedical Circuits and Systems, Vol. 6, pp. 87-100, April 2012.

[157] M. Delgado-Restituto, A. Rodríguez-Pérez, A. Darie, C. Soto-Sánchez, E. Fernández-Jover and A. Rodríguez-Vázquez, "System-Level Design of a 64-Channel Low Power Neural Spike Recording Sensor". IEEE Transactions on Biomedical Circuits and Systems ISSN 1932-4545; Digital Object Identifier 10.1109/TBCAS.2016.2618319.

[158] S. Mukhopadhyay and G. Ray, “A new interpretation of nonlinear energy operator and its efficacy in spike detection,”IEEE Trans. Biomed-ical Eng., vol. 45, no. 2, pp. 180–187, Feb. 1998.

[159] T. M. Seese, H. Harasaki, G. M. Saidel, and C. R. Davies, “Characterization of tissue morphology, angiogenesis, and temperature in the adaptive response of muscle tissue to chronic heating,” Lab. Invest.,vol. 78, no. 12, pp. 1553–1562, Dec. 1998.

[160] H. W. Lilliefors, “On the Kolmogorov-Smirnov test for normality with mean and variance unknown,” J. Amer. Statist. Assoc., vol. 62, no. 318, pp. 399–402, June 1967.

[161] R. Olsson, III and K. Wise, “A three dimensional neural recording microsystem with implantable data compression circuitry,” IEEE J. Solid-State Circuits, vol. 40, no. 12, pp. 2796–2804, Dec. 2005.

[162] M. Chaeet al., “A 128-channel 6 mW wireless neural recording IC with on-the-fly spike sorting and UWB tansmitter,” in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers, 2008, pp. 146–147.

[163] M. Rizket al., “A single-chip signal processing and telemetry engine for an implantable 96-channel neural data acquisition system,” IOP J. Neural Eng., pp. 309–321, Sep. 2007.

[164] T.C.Chen, W.Liu, and L.G.Chen et al.,“128-channel spike sorting processor with a parallel-folding structure in 90 nm process,” in Proc. IEEE Int. Symp. Circuits and Systems, 2009, pp. 1253–1256.

[165] D. Pani, G. Barabino, L. Citi, P. Meloni, , S. Raspopovich, S. Micera, and L. Raffo, “Real-time neural signals decoding onto off-the-shelf DSP processors for neuroprosthetic applications,” IEEE Transactions on Neural systems and rehabilitation engineering., vol. 24, no.9, pp. 993–1002, May. 2016.

[166] D. Markovic and R. W. Brodersen, DSP Architecture Design Essentials. Springer, 2012. 7.1.2

[167] T. Burd and R. Brodersen, “Design Issues for Dynamic Voltage Scaling,” *Proc. of Int. Symp. on Low Power Electronics and Design*, (*ISLPED*, Rapallo, Italy), pp. 9-14, 2000.