Imperial College London
Department of Electrical and Electronic Engineering

Real-time neural signal processing and low-power hardware co-design for wireless implantable brain machine interfaces

Zheng Zhang

Submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London
October 18, 2023

Declaration of Originality

I declared this thesis is my own work, except where stated by reference or in the text.

Zheng Zhang

Copyright Statement

Abstract

Intracortical Brain-Machine Interfaces (iBMIs) have advanced significantly over the past two decades, demonstrating their utility in various aspects, including neuroprosthetic control and communication. To increase the information transfer rate and improve the devices' robustness and longevity, iBMI technology aims to increase channel counts to access more neural data while reducing invasiveness through miniaturisation and avoiding percutaneous connectors (wired implants). However, as the number of channels increases, the raw data bandwidth required for wireless transmission also increases becoming prohibitive, requiring efficient on-implant processing to reduce the amount of data through data compression or feature extraction.

The fundamental aim of this research is to develop methods for high-performance neural spike processing co-designed within low-power hardware that is scaleable for real-time wireless BMI applications. The specific original contributions include the following:

Firstly, a new method has been developed for hardware-efficient spike detection, which achieves state-of-the-art spike detection performance and significantly reduces the hardware complexity. Secondly, a novel thresholding mechanism for spike detection has been introduced. By incorporating firing rate information as a key determinant in establishing the spike detection threshold, we have improved the adaptiveness of spike detection. This eventually allows the spike detection to overcome the signal degradation that arises due to scar tissue growth around the recording site, thereby ensuring enduringly stable spike detection results. The long-term decoding performance, as a consequence, has also been improved notably. Thirdly, the relationship between spike detection performance and neural decoding accuracy has been investigated to be nonlinear, offering new opportunities for further reducing transmission bandwidth by at least 30% with minor decoding performance degradation.

In summary, this thesis presents a journey toward designing ultra-hardware-efficient spike detection algorithms and applying them to reduce the data bandwidth and improve neural decoding performance. The software-hardware co-design approach is essential for the next generation of wireless brain-machine interfaces with increased channel counts and a highly constrained hardware budget.

Acknowledgements

First and foremost, I would like to express my profound gratitude to my supervisor, Prof. Timothy Constandinou, whose invaluable guidance and unwavering support have been pivotal to my progress. Your constant encouragement and patient mentorship have broadened my understanding and judgment in research. Thank you for your relentless belief in me and my abilities, for challenging me to reach higher, and for providing the tools and environment that have nurtured my professional and personal growth.

To my wonderful colleagues in NGNI lab, I extend my sincere thanks for your help and cooperation. I would like to extend particular gratitude to Dr. Oscar Savolainen and Dr. Peilong Feng, whose invaluable insights and steadfast support have profoundly consolidated my work and enriched my scholarly journey. I am truly blessed to have shared this journey working with all these talented individuals.

I also want to express my heartfelt gratitude to my cherished friends. You have stood by me in the best and the toughest of times, offering your unwavering friendship and support. I will forever treasure our incredible adventures in China, the UK, and throughout Europe. These journeys we have embarked upon together have created indelible memories that will always occupy a special corner of my heart.

Lastly, my deepest gratitude is reserved for my parents, my girlfriend, and my beloved cat. Your boundless love, sacrifices, and faith in me have empowered me to overcome the most daunting challenges during my PhD journey. This thesis has only been possible thanks to your enduring support, both financial and emotional. You have been the unwavering pillars of strength and encouragement, making this accomplishment not just mine, but ours.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

ADC Analog-to-Digital Converter

ADF Absolute Difference Filter

AI Artificial Intelligence

ASO Amplitude Slope Operator

CMOS Complementary Metal-Oxide-Semiconductor

CNN Convolution Neural Network

CPU Central Processing Unit

DBS Deep Brain Stimulation

DED Delta Event-Driven

EAP Extracellular Action Potential

EED Explicit Event-Driven

ESA Entire Spiking Activity

FDA the United States Food and Drug Administration

FDR False Discovery Rate

FDR False Discovery Rate

GED Group Event-Driven

GPU Graphic Processing Unit

iBMI Intracortical Brain-Machine Interface

IC Integrated Circuit

LFP Local Field Potential

LUT Look-Up Table

MCU Microcontroller

MUA Multi-Unit Activity

NHP Non-Human Primate

NN Feedforward Neural Network

NN Feedforward Neural Network

QRNN Quasi-Recurrent Neural Network

RNN Recurrent Neural Network

SNR Signal-to-Noise Ratio

SUA Single-Unit Activity

SVM Support Vector Machine

Chapter 1

Introduction

1.1 Motivation

iBMIs have made significant progress in recent years, demonstrating the ability to allow humans to interact with their environment directly. These interfaces have been used to restore lost abilities, enhance existing abilities, and even provide humans with new capabilities [1]. BMIs also offer a new tool for studying human neural systems and analysing neurological disorders [2, 3].

Wired BMIs, such as the Blackrock Neuroport system [4], are the most well-developed type of BMI. It was first developed in 2004 and is still in use for various challenging tasks, such as decoding human handwriting [5] or speech [6, 7]. However, their portability is limited by the need for wired connections and on-workstation processing. Recent developments in implantable BMIs, such as the one developed by Neuralink [8], have tried to improve portability by using a wearable central processing node connected to flexible probes. The Neuralink system still carries the risk of percutaneous connections leading to inflammation and brain damage. Current research in iBMI technology focuses on increasing channel counts to access more neural data, which could lead to a higher information transfer rate and better robustness/longevity through redundancy. Efforts are also underway to reduce the invasiveness of BMIs through miniaturisation and avoidance of percutaneous connectors, such as by using wireless communication and distributed electrodes [1, 9, 10].

Algorithms should be designed with hardware efficiency in mind, so called a software-hardware co-design. Both power and device size are hugely constrained on implants. Within the circuits, the area utilisation is determined by the size of the analogue front-end, Analog-to-Digital Converter (ADC), processing core, transmission modules and other peripherals, while the total power consumption consists of static power, processing power and transmission power. A multidisciplinary effort is needed to fulfil the constraint. Advanced Integrated Circuit (IC) design and Complementary Metal-Oxide-Semiconductor (CMOS) technology

play crucial roles in reducing the size and power. Furthermore, the hardware-efficient algorithms are also essential, particularly in mitigating the power limitations associated with transmission – the primary system bottleneck. This is one key motivation for this project.

Specifically, transmission power is directly proportional to the data bandwidth. Considering a single-channel raw neural signal sampled at 24 kHz with a 16-bit ADC, the bandwidth per channel amounts to 384 kbps; the wireless transmission power for each channel can reach tens of micro-Watts, which is impractical for high channel count brain implants. Some form of data reduction such as feature extraction and compression is necessary to reduce data bandwidth and eliminate the transmission power bottleneck. Any additional power required for on-implant processing must not exceed the power reduction resulting from transmission optimisation. Furthermore, the utilisation of resources, particularly memory, for implementing these algorithms must remain within reasonable limits to minimise area occupation.

Various methods have been developed to extract features from raw neural signals. However, there are limited options for on-implant use. Hardware-efficient feature extraction and compression can effectively reduce the transmission power, processing power, and area occupation while maintaining the brain activity decoding performance. Such a software-hardware co-design is crucial for designing next-generation BMIs. The above challenges and benefits motivate this project.

1.2 Research Objectives

Spike-based features are widely used in implantable BMIs. These systems face significant challenges in adapting to dynamic brain environments and maintaining long-term decoding performance. Implantable devices are amongst the most technically challenging electronic systems, subject to constraints such as limited energy capacity, reliability, and intrinsic safety requirements. The first step in a spike-based BMI system is spike detection, which extracts the most informative features, namely spikes, from the raw signals. An efficient and effective spike detection algorithm has the potential to significantly reduce data bandwidth (2-3 orders of magnitude) while maintaining system performance. This research aims to develop hardware-efficient spike detection suitable for real-time spike-based BMI with ultra-low power consumption and minimal resource usage. Building upon the spike detection algorithm as a foundation, this study explores opportunities to further reduce the wireless transmission bandwidth and power through compression techniques, ultimately making wireless BMI feasible. Novel machine learning models are expected to improve long-term decoding stability, ensuring robust and reliable performance over extended periods. By addressing these challenges, this research tries to advance spike-based BMI systems, enabling their seamless integration into implantable devices with improved adaptiveness, power efficiency, and long-term performance.

To achieve all these, we have defined the following research objectives:

- **Develop hardware-efficient real-time spike detection for safe and reliable implantable use.**
  Spike detection has the potential to significantly reduce data bandwidth, but current on-implant spike detection algorithms require excessive power and area exceeding the implant resource budget. This study proposes new hardware-efficient spike detection algorithms which to reduce complexity while maintaining performance compares to the state-of-the-art.

- **Enhance spike detection adaptiveness to varying brain environments.**
  Existing spike detection algorithms typically use fixed thresholds based on noise statistics. This fails to adaptive to varying noise levels in different brain environments and over time. The goal of this study is to overcome this limitation by comparing existing thresholding techniques and proposing new algorithms to enhance the adaptiveness of spike detection while maintaining low complexity.

- **Investigate the relationship between spike detection and neural decoding for improved system performance.**
  Spike detection plays a critical role in spike-based BMIs, as its output provides the input for neural decoding. However, the understanding of the relationship between spike detection and neural decoding remains unclear. This study aims to comprehensively investigate such a relationship, providing valuable insights for system design in spike-based BMIs. By investigating this relationship, new opportunities can potentially be identified to enhance the overall performance of spike-based BMIs.

1.3 Thesis Outline

This thesis describes the journey of developing hardware-efficient adaptive spike detection software-hardware co-design. It begins with simplifying the conventional path through multiplication-free fixed-point spike detection. Then, the adaptiveness of statistical-based spike detection algorithms is evaluated, leading to the development of an idealised model for this type of spike detection algorithm. This model reveals the limitations of noise-statistical-based thresholding and inspires the exploration of new thresholding mechanisms that can better adapt to changing noise levels. The firing-rate-based spike detection algorithm is proposed, which significantly outperforms statistical-based methods in terms of spike detection accuracy, adaptiveness, and hardware complexity. Finally, the relationship between spike detection and neural decoding is investigated, leading to the development of new algorithms

for firing rate compression and decoding that reduce data bandwidth and improve long-term decoding stability. The outline of this thesis is given as follows:

- **Chapter 2 Brain-Machine Interfaces: Background and state of the art**

  This chapter provides an overview of the background information on BMIs, including their development history, state-of-the-art, and future perspectives. It covers the various types of recordings and datasets used in BMI systems and the signal processing workflow, which includes techniques for enhancing the signal, extracting features, decoding brain activities, and their associated challenges. Additionally, the chapter explores the different hardware used in BMI systems, their advantages and disadvantages.

- **Chapter 3 Simplifying the conventional path: Multiplication-free fixed-point spike detection**

  This chapter presents a new approach for spike detection that simplifies the conventional path by using fixed-point representation and avoiding multiplications. The proposed algorithm demonstrates similar spike detection performance as more complex algorithms. Power consumption has also been measured on embedded targets, showing promising results for potential implementation on implants. The use of fixed-point representation and the elimination of multiplications in this spike detection algorithm can lead to more hardware-efficient and low-power BMI systems.

- **Chapter 4 The puzzle of adaptiveness: Statistical-based spike detection deadend**

  In this chapter, the limitations of conventional statistical-based spike detection algorithms in terms of adaptiveness are evaluated. An idealised spike detection model is built using a synthetic dataset, and the results suggest that the widely used statistical-based thresholding technique can suffer from parameter deviation at varying noise levels. The findings highlight the need for a new thresholding technique that can achieve fully automatic spike detection without requiring manual calibration.

- **Chapter 5 New approach: Firing-rate-based neural spike detection**

  This chapter presents a novel approach to spike detection using a firing-rate-based algorithm and its corresponding hardware implementation. Rather than setting thresholds based on noise statistics, the algorithm relies on the hypothesis that neurons in a given brain region fire at a normal rate. By leveraging neuroscientific findings to establish a reasonable detection rate, the threshold can be automatically updated to provide accurate detection. Results indicate that the proposed algorithm outperforms previous approaches, achieving improved detection performance and adaptiveness. Furthermore,

the FPGA and ASIC implementations demonstrate an order of magnitude reduction in power consumption and area occupation compared to existing state-of-the-art implementations.

- **Chapter 6 Spike detection and beyond: The relationship between spike detection and neural decoding performance**

  This chapter investigates the relationship between spike detection and neural decoding. The result suggests that spike detection is no longer simply a matter of setting a threshold to distinguish spikes from noise at an optimal level. Instead, it is a trade-off between data bandwidth and neural decoding performance. In light of these findings, new firing rate compression algorithms and decoding algorithms are proposed. These techniques reduce the data bandwidth, keep it within wireless transmission budgets, and improve long-term decoding stability.

- **Chapter 7 Conclusion and future directions**

  This chapter concludes this thesis, summarises the original contribution and points out the future potential building on this research.

Chapter 2

Brain-Machine Interfaces: Background and state of the art

This chapter provides an overview of the general background of iBMIs. It introduces the historical development of BMIs, research techniques for recording electrophysiology, neural signals, features, and datasets used in BMIs, as well as the long-term degradation of implant electrodes and signal characteristics. Finally, the commonly-used signal processing methods and decoding models for spike-based BMI systems are introduced, as well as how algorithms are co-designed with the hardware in these systems.

2.1 Brain Machine Interface: the past, now and future

BMIs, alternatively referred to as brain-computer interfaces or neural interfaces, are systems that bridge the nervous systems with external devices for neural function restoration, neural degeneration repair, communication recovery, or even human ability augmentation. The development of these interfaces has reached significant milestones over the past decades, from theoretical concepts and initial experiments to practical technology for clinical or customer use.

Early research in the late 20^{th} century studied the nature of neural signals and explores how to use them to interact with computers. In a pioneering study by Fetz et al. in 1969 demonstrated that the firing rate from the single or multiple neurons of Non-Human Primates (NHPs) could be modulated with visual stimulation and food reinforcement [12]. This study

A portion of the content in this chapter has been published in [11], with my full contribution to the software/hardware algorithm design and result analysis

was one of the earliest works proving the concept for BMI that neural activities could be controlled voluntarily by the BMI. Georgopoulos et al. in 1986 found that each neuron in the arm area of the primate motor cortex only controlled one single direction of movement [13]. This finding provided the foundation for movement decoding using motor cortex signals. One of the first real-time systems controlled by neural activities was developed by Nicolelis et al. in 1999, in which a robotic arm could be controlled by the neural activity of rats for pressing the lever to obtain water [14]. This study demonstrated the possibility of using BMI for movement restoration.

The BMI entered a rapid development stage in the early 21^{th} century as the Utah array and other microelectrode arrays became more refined for better signal quality, higher resolution, more recording channels, and better biocompatibility, longevity, and safety. In 2004, the Utah array was first implanted in humans with the launch of the BrainGate project [4]. The $4 \times 4\,\text{mm}$ array was placed in the arm area of the motor cortex of four paralysed patients. The experiment showed the ability to control a computer cursor [4] or robotic arm [15] based on spike signals extracted from the raw neural signal. Neuromodulation emerged as another significant application of BMI technology. The electrical signals were delivered into a specific area of the brain for patients with neurological disorders. It showed reduced symptoms such as tremors or rigidity, providing an alternative treatment for the patients when the normal medications are not effective [16]. Related studies showed new potential for BMI in clinical use for the treatment of neurological disorders such as Parkinson's disease or epilepsy.

In the current era, there has been a trend towards translating research outcomes into commercial and more practical applications. This transition has been facilitated by advances in neural recording technologies, the miniaturisation of hardware, and the integration of Artificial Intelligence (AI) with BMIs. Neuralink is a representative company designing high-bandwidth, implantable BMIs with the short-term goal of treating neurological disorders and ultimately enhancing human cognition [8]. With the development of BMI technologies, significant outcomes have been achieved in controlling paralysed [17] or prosthetic arms control [18] for paralysed patients; controlling computer cursor [19, 20] for brain typing; decoding handwriting [5] and speech intention [6, 7, 21] for communication recovery and enhancement; controlling a "third thumb" for human ability augmentation [22], and more. The current state-of-the-art BMIs for communication have hit 62 words per minute with an error rate of 23.8% in a large vocabulary set [7] and the most advanced prototype for robotic arm control has achieved ten degrees of freedom [23].

The development and commercial translation of current BMI systems faces several challenges. A key objective of the field is to develop recording systems with higher channel counts to capture an increased amount of neural information [1]. However, the telemetry power for transmitting the neural signal to the external processing node becomes an issue due to the hardware constraints [24]. The stability of the electrodes is another challenge. Foreign body

response in the brain, implantation trauma, and micromotion between the implants and brain tissue can gradually or significantly affect the recording quality, making it difficult to obtain high-quality long-term recordings, especially for wired implants [24, 25]. These challenges have motivated high-bandwidth BMIs with thousands of channels to be distributed and wireless, while on-implant pre-processing is required for bandwidth reduction [1].

The development of BMIs requires multidisciplinary expertise including neuroscience, computer science, microelectronics, materials, mechanical engineering, surgical and clinical medicine, and more. The goal of BMI systems can vary widely, ranging from supporting experimental neuroscience with a focus on signal integrity to conducting closed-loop experiments that require near-real-time processing. Other efforts concentrate on developing portable, power-constrained BMI devices with a strong emphasis on longevity and safety. Additionally, studies explore the use of different hardware to accelerate neural signal processing.

In this section, the different types of brain signals will be introduced, including the dataset used in this thesis. Considering the broad aspects involved in the area of BMI, we specifically focus on reviewing studies that concentrate on designing hardware-efficient neural signal processing platforms for high channel count implantable BMIs. This area of research is highly relevant to the objectives of this thesis and ensures alignment with the core topic.

2.2 Recording technology

The advances in different recording technologies have increased the number of recorded neurons and provided better signal quality and resolution. Microelectrode arrays (Utah array), high-density probes (Neuropixels Probes), microwire probes, and flexible probes are four types of widely used recording technologies.

2.2.1 Utah array

Utah arrays have long been pivotal in the neuronal recording. Developed in the late 20^{th} century by researchers at the University of Utah, Utah arrays comprise multiple microelectrodes arranged in a grid pattern, permitting simultaneous recording from a certain spot of neural tissue on the cortical surface. It features 100 electrodes arranged in a 10×10 array for 4×4 mm (96 channels with 4 references). Blackrock Neurotech, a spin-off company from the University of Utah, commercialises the Utah Array. Their Neuroport is currently the only device approved by the United States Food and Drug Administration (FDA) for human brain implanting before 2023, and most of the BMI studies on humans use this type of electrodes [5–7, 17, 21]. A standard experiment procedure is shown in Fig.2.1. This proce-

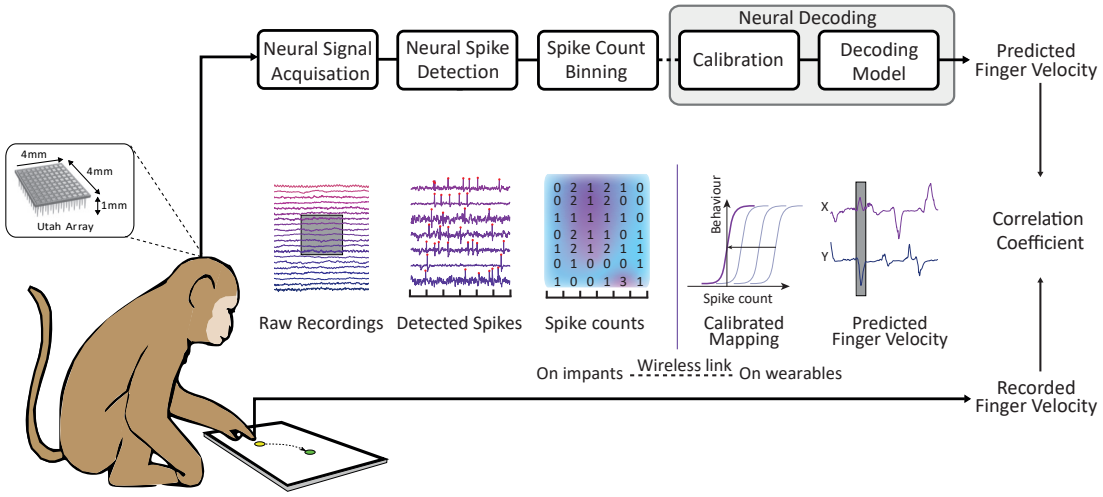

Figure 2.1: A standard experiment procedure using Utah array. The non-human primate is operating a ticker to control the cursor reaching target destination. The finger velocity is decoded from the neural signal after applying signal processing and machine learning models.

dure is also followed by this thesis while datasets are from public available resources as in Section.2.3.3 instead of being collected on our own.

2.2.2 Neuropixels probe

Neuropixels probes represent a significant leap in invasive recording technology. They are high-density, silicon-based probes with up to 384 recording sites, enabling recordings with high spatial resolution. The Neuropixels 1.0 comprises a single 10 mm long shank of 70 $\times$ 20 µm cross-section, hosting 960 low-impedance, CMOS compatible, Titanium Nitride sites. The integration of voltage signal filtering, amplification, multiplexing, and digitisation on the 6 $\times$ 9 mm base of the probe eliminates noise and allows for the direct transmission of digital data [26]. The miniaturised Neuropixels 2.0 is reported in [27] with 1/3 the size of the original probe and the same capability. This design has resulted in an unprecedented capacity to isolate and record spiking activity from hundreds of neurons per probe when implanted in rats and mice. Neuropixels probes have been used in various neuroscience research areas in studying the relationship between neural activities and spontaneous behaviours [28], decision-making processes [29], and more [30–32].

2.2.3 Microwire electrodes

Microwire electrodes represent an alternative approach to invasive recording. Microwire electrodes are thinner and more flexible than microelectrodes. The Paradormics company introduces a novel system combining platinum-iridium microwire electrode arrays with a

CMOS voltage amplifier array. This system is capable of simultaneous recording from 65,536 channels, sampled at a rate of 32 kHz with 12-bit resolution [33]. The system's performance has been demonstrated through bench testing and *in-vivo* experiments, achieving microwire-based recordings with the highest channel count in sheep and rate [34], making it one of the most powerful tools for cortical recording.

2.2.4 Flexible probe

The recent emergence of flexible probes, such as Neuralink's technology, offers a promising alternative to traditional recording technologies. Founded in 2016, Neuralink is at the forefront of developing flexible, thin-film electrodes that can be robotically inserted into the brain, causing minimal tissue damage [8]. The Neuralink device features over 1000 channels for simultaneous recording and stimulation, providing a recording capacity that rivals lead of the high-density Neuropixels probe. It also holds an edge due to the potentially reduced tissue damage courtesy of its flexible design. In real-time NHP experiments, subjects were able to control cursor movement through the Neuralink devices, revealing the potential this technology holds for future neuroscience research and neuroprosthetics. Neuralink was recently approved by FDA for the first-in-human trial in May 2023.

2.3 Neural recordings and features

This section reviews different types of widely used recordings and neural signal features.

2.3.1 Intracellular recordings

Intracellular recordings are the signals recorded inside a neuron by inserting the measurement electrode, typically a glass micropipette electrode, into the cell. Intracellular recordings provide high-resolution signals of individual neurons, which are considered to be a valuable tool in neuroscience research, and the recordings enable studies to learn single neuron activities and their response to external stimulation [35–37] .

2.3.2 Extracellular recordings

Extracellular recordings are measured outside the neurons. This type of recording records the activities of multiple neurons, making the study of local neuronal networks viable. Extracellular recordings are most preferred in BMI applications because they offer a less invasive approach and are more feasible for in-vivo studies involving free-moving animals or humans. However, the amplitude of the neuronal activities can be attenuated from 1,000 to 2,000

times less [38], making it require additional processing to extract useful features. These features for extracellular recordings include Local Field Potentials (LFPs) Extracellular Action Potentials (EAPs), Single-Unit Activities (SUAs), Multi-Unit Activities (MUAs) and Entire Spiking Activities (ESAs).

LFP reflects the joint activities of neuronal assemblies with a frequency typically below 300 Hz. Its oscillatory activities, phase relationships, and spectral power are linked to different states of consciousness, cognitive tasks, or neurological disorders [39]. LFP is particularly preferred in Deep Brain Stimulation (DBS) to the clinical treatment like in Parkinson's disease, guiding the stimulation to be delivered in a closed-loop manner [40].

EAP is one of the most important features in neural recordings. It is typical in 500 Hz - 3 kHz frequency range, containing the electrical activities (spikes) fired by neurons. MUA is obtained by thresholding the AP to detect these spikes, while SUA is obtained by clustering the detected spikes and assigning the spikes to different neurons according to the waveform, known as spike sorting. Both SUA and MUA are widely used in BMI systems for behaviour or intention decoding [5–7, 17, 21]. ESA is a newly used feature in BMI applications. It is obtained by rectifying and lowpass the EAP signal at around 50Hz. It represents the envelope of the spike activities, which has the potential to provide high decoding performance with low data bandwidth [41, 42].

2.3.3 Publicly available datasets

There are a large number of datasets available in the community. The datasets used in this thesis, including two real recordings obtained with Utah array [43] and Neuropixels probes [44] and a synthetic dataset [45] synthesised with Utah array signal templates.

The Utah array recordings are publicly available in [43]. This dataset was collected on the Motor Cortex of a non-human primate when the subject was operating a ticker to control the ticker operating a cursor reaching task on a computer as shown in Fig.2.1. The finger movement displacement and velocity were also recorded. It contains 5 hours of recordings over 200 days, from June 27, 2016, to January 13, 2017. The recording is collected at 24,414 Hz. Along with this dataset, threshold crossings are provided, which are obtained using 3-5 times standard deviation values.

The Neuropixels dataset is publicly available at [44]. This dataset was obtained from rodent visual cortex and collected by the Cortex Lab at University College London. Each shank of the probes contained 384 channels, with each channel sampled at 30 kHz.

While in vivo recordings are valuable in providing a realistic representation of neural activity, their usefulness is somewhat limited by the lack of ground truth. This makes the quantitative evaluation and benchmarking of algorithms challenging. For this reason, synthetic datasets have emerged as essential tools in these processes.

Quiroga et al. presented an influential synthetic dataset [45]. It is widely used as a benchmark for assessing spike detection and spike sorting algorithms [46–48]. This dataset simulates the activity of three neurons using a Poisson process, with an average firing rate of 20 Hz per neuron. Notably, both the noise and spike templates in this dataset originate from Utah array recordings. This feature enhances the verisimilitude of the synthetic data, aligning it more closely with real neural recordings. It has 16 signals divided into four groups: Easy1, Easy2, Difficult1, and Difficult2. Within each group, the noise standard deviation to spike peak ratio is 0.05 to 0.2 with a step of 0.05. The sampling frequency of the original recordings is 24 kHz.

Sample recordings with their amplitude and gradient distribution (after absolution) is given in Fig.2.2. Though distributions are different across recordings, all distribution contains a noise cluster near the origin and several spike clusters. Based on this observation, various spike algorithms [42,49,50] have been design to estimate different statistics for setting a threshold discriminating spikes from the noise.

2.4 Long-term electrode and signal degradation

Long-term operation of the BMI systems is challenging as the signal can degrade over time. We attempt to improve long-term performance algorithmically. An overview of how the electrode and signal degrade in the long term is provided in this section. The findings and observations in this section are mainly from Barrese et al. in [51].

By examining the electrode conditions and recording quality from 27 NHPs, they evaluated the, causes of electrode failures, recording qualities, and electrode impedance overtime. The results suggests that Utah array is feasible to record spiking signals from Utah for 5.5 years, but the recording quality, both in terms of the number of channels and signal amplitude, consistently decreases over time. Most of failures (56%), happened within the first year after implantation. Acute mechanical issues were the predominant class of failures, making up 48% of the total, and within this class, connector problems were the main cause, contributing to 83% of these failures. As for observable biological failures, which constituted 24% of the total failures, the most common issue was a progressive meningeal reaction that caused the array to separate from the parenchyma, accounting for 14.5% of the total failures. Among the total of 78 implants, 33% provided chronic recordings for over a year. Of the 26 arrays that managed to record for more than a year, a 61% continued to record for a second full year. Additionally, 27% sustained recordings for a third year, 15% were functional for four years, and 8% stayed functional for more than five years.

The short and long time scale analysis of the pattern from the electrical measurements indicates changing patterns. Such patterns provide clues for the causes of failures and approaches that mitigate them. Impedance rises during the first two weeks after implantation

and stays high for about 100 days.Then, they exhibit a continuous and steady impedance decline with time. Initially high for the first few months, the number of channels with signals (around 60) and signal amplitude also gradually decreased over the years. Similarly, noise on the electrodes showed a decrease over time, leading to a flat Signal-to-Noise Ratio (SNR) when all data is averaged. Meningeal encapsulation and the failure of insulating materials are the primary contributors to the chronic decrease in impedance and deterioration of signal quality.

2.5 Advances in neural signal processing

Fig.2.3 shows a standard procedures of BMI systems and summarises methods involved in each step.

Various pre-processing and feature extraction algorithms have been proposed. In the frequency domain, pre-processing involves frequency-band filtering to select different EEG, ECoG and LFP bands of interest. Different frequency bands have been observed to have high correlations with attention, sleep and neurological disorders [57]. Short-time Fourier transform and Wavelet transforms [45, 58, 59] have both been used to analyse tempo-frequency features of the signals and capture local patterns of neural spikes. Phase-lock value has been found to be an efficient indicator for abnormal brain status detection [60]. Principal component analysis (PCA) [61] is sometimes used to reduce the dimensionality of the extracted tempo-frequency features.

Spike-based signals (MUA, SUA), which indicate local neuronal activities [62], can be pre-processed in the time domain. Conventional spike pre-processing methods use the absolute value, derivatives, power or standard deviation [45]. Some operators (e.g. Nonlinear Energy Operator [53]) have been proposed to emphasise spikes and suppress noise. On top of the pre-processed signal, many different spike detection [42, 49, 50, 63] and sorting [64–67] algorithms have been explored to resolve the spikes from the noise. There is also a low-power design to extract the ESA signal demonstrating the power-saving opportunity from a reduced sampling rate using ESA [42]. Pre-processing, feature extraction and dimensionality reduction distil the valuable information (e.g. informative frequency bands or spikes) from the noisy signal and reduce the bandwidth for further processing or transmission. Some studies [68] also utilise lossless compression techniques like Huffman encoding to reduce the data rate further to accommodate the low processing power and transmission power requirements.

Pre-processing signals are beneficial to downstream neural decoders, however a robust and high-performing decoding algorithm is also key to the success of the whole system. Conventional decoding approaches are adaptive filters like the Wiener filter [4], Wiener cascade filter [69], Kalman filter [70] and Unscented Kalman filter [71]. Recently, with improvements in artificial intelligence, machine learning and deep learning have been introduced

for neural decoding. Various architectures have been explored and have achieved promising levels of decoding performance. These architectures include Support Vector Machine (SVM) [72], Neural Network (NN) [73], Convolution Neural Network (CNN) [74], Recurrent Neural Network (RNN) [5, 75], LSTM [76], Quasi-Recurrent Neural Network (QRNN) [41] and attention-based models [77]. In terms of application to neurological disorders, there are also various closed-loop control algorithms [56] that have been proposed for adaptive neuromodulation to treat diseases such as epilepsy, Parkinson's and essential tremor [78–80].

2.6 Software-hardware co-design considerations for BMI systems

The concept of software-hardware co-design is important in facilitating the seamless integration of algorithms and hardware to leverage the inherent strengths of each component. The algorithms implemented on the hardware must not only fulfill their intended functions but also be tailored to harness the unique characteristics and capabilities of the hardware platform, ultimately ensuring the algorithm to be hardware efficient.

The most powerful BMI systems take advantage of the high computing power provided by high-performance Central Processing Units (CPUs) and Graphic Processing Units (GPUs) in workstation computers. Some applications involve offline spike sorting acceleration [81] or computationally intensive online decoding, such as in handwriting or speech decoding [5, 7]. However, the high power consumption and lack of portability of these approaches reduce their potential for commercial use. They also require specialised training to operate and calibrate. As such, these technologies are mostly only used in laboratory settings.

Various real-time applications have also appeared. Some applications stream the recorded brain data to wearable Microcontrollers (MCUs) for real-time signal processing, neural signal decoding and closed-loop control. The Neuralink company has successfully decoded a pig's gait pattern and enabled a monkey to play games with such implantable devices [8]. There are also various healthcare applications for closed-loop neuromodulation. Based on the Medtronic Activa PC neuromodulator, the Activa PC+S was designed in 2012 [82]. Such designs have been widely used for closed-loop DBS treating different kinds of neurological disorders [83–85]. In 2018, an upgraded version, Summit RC+S [86], was introduced with an enhanced data rate and real-time streaming capability. Neuropace Responsive Neural System is another closed-loop neuromodulation system, which was approved by the FDA for neurological therapy in Nov. 2013 [87].

There is a trend in the design of the next generation of implantable neural interfaces towards increased channel count, reduced invasiveness, and increased longevity and scalability [24]. Increased channel count increases the data bandwidth and makes it impossible to

stream the raw data wirelessly to the MCUs. On-implant signal processing becomes essential to reduce the bandwidth to make it possible for wireless transmission. To minimise the invasiveness and to increase the longevity of the implants, one should optimise both the size and power consumption. The electronic systems must also be programmable to guarantee their scalability. There is added complexity given that performance should not be compromised while fulfilling all these requirements. Neuroscientists and engineers have made joint efforts to overcome these difficulties. To create programmable, high-performance, low-cost, tiny size and low-power systems, researchers use MCUs, FPGAs and ASICs to fully utilise their features in different functional units within neural interface systems. Schaffer et al. has used FPGAs for accelerating the real-time spike sorting [88, 89]. In 2018, a standalone online spike sorting system was designed combining FPGA and ASIC circuits [67]. A fully implantable neurostimulator was proposed in [90, 91], which used an FPGA for front-end on-implant neural signal pre-processing and an MCU for the event classification and stimulation policy control.

Power consumption is clearly a significant concern in designing next generation neural interfaces, and the distribution of power consumption in a neural interface system can vary. The static power consumption is from the hardware itself. It includes, for example, the recording electrodes, amplifiers, ADCs and FPGA/MCU/computer static running power. The data transmission power is a significant concern, especially when wireless transmission is used, as wireless link often is to reduce infection risks from percutaneous connections. Processing power is another primary source of power consumption that has been investigated. A diagram of three types of BMI systems and their pros and cons is given in Fig. 2.4 along with typical power consumption distributions.

ASIC designs are the optimal solution for minimising power consumption and chip area. However, due to the ease of the design process compared to ASICs, researchers tend to use FPGAs to validate the ASICs' performance before developing ASICs. FPGAs also have the advantage of flexibility which suits programmable applications well. MCUs have significantly higher power consumption. However they are easier to integrate with front-end analogue and digital components, are more accessible for development, and are low cost. The selection of platform architecture and implementation methodology, and associated trade-offs is critical for implantable and/or on-node processing.

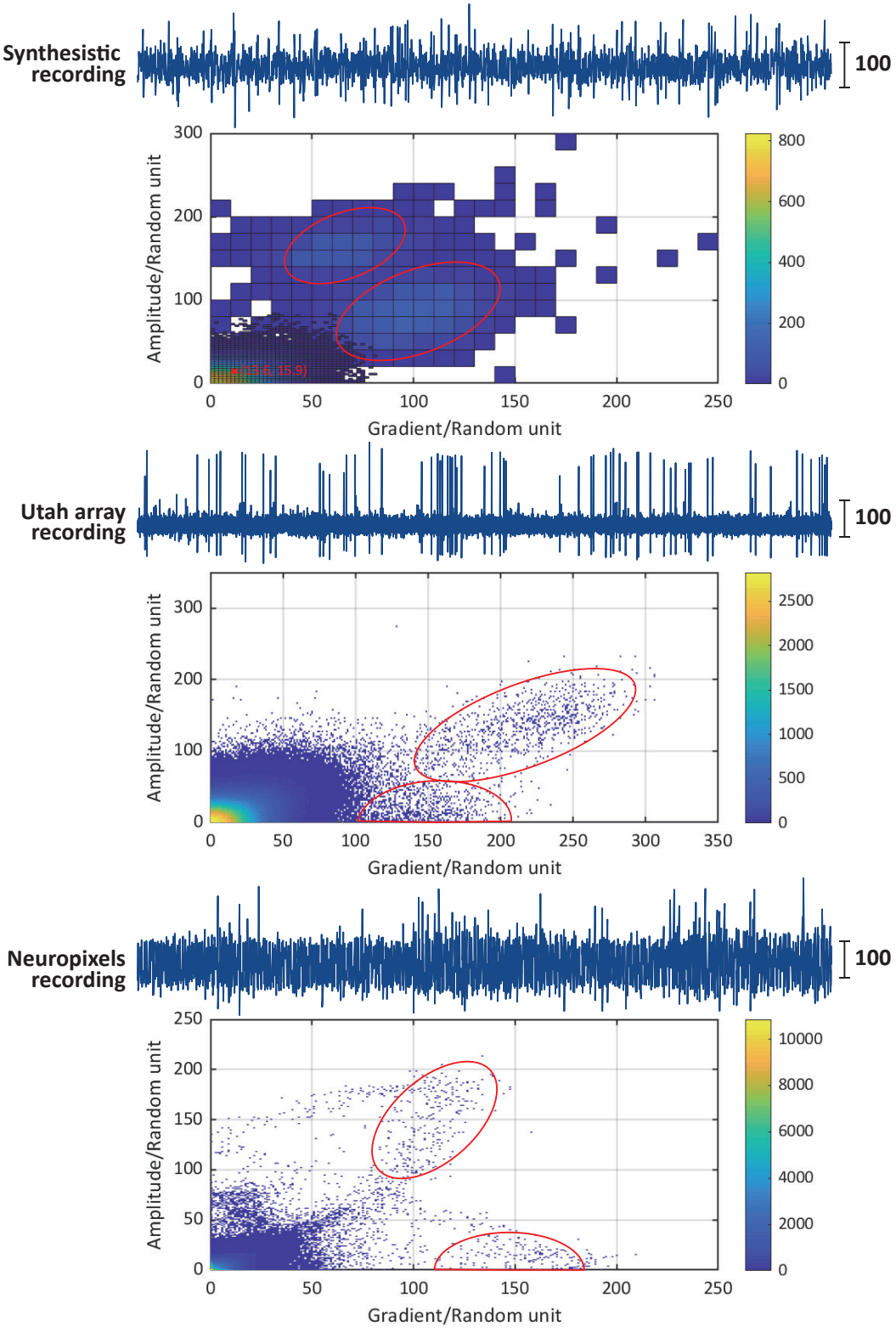

Figure 2.2: The recording snapshot and signal amplitude-gradient distribution of the synthetic recordings, Utah array recordings and Neuropixels recordings. The red star labels the mean of the noise distribution of the synthetic dataset. Red circles highlights the potential spike distributions. The recordings are scaled to have the same average peak spike amplitude, so they are all in random units.

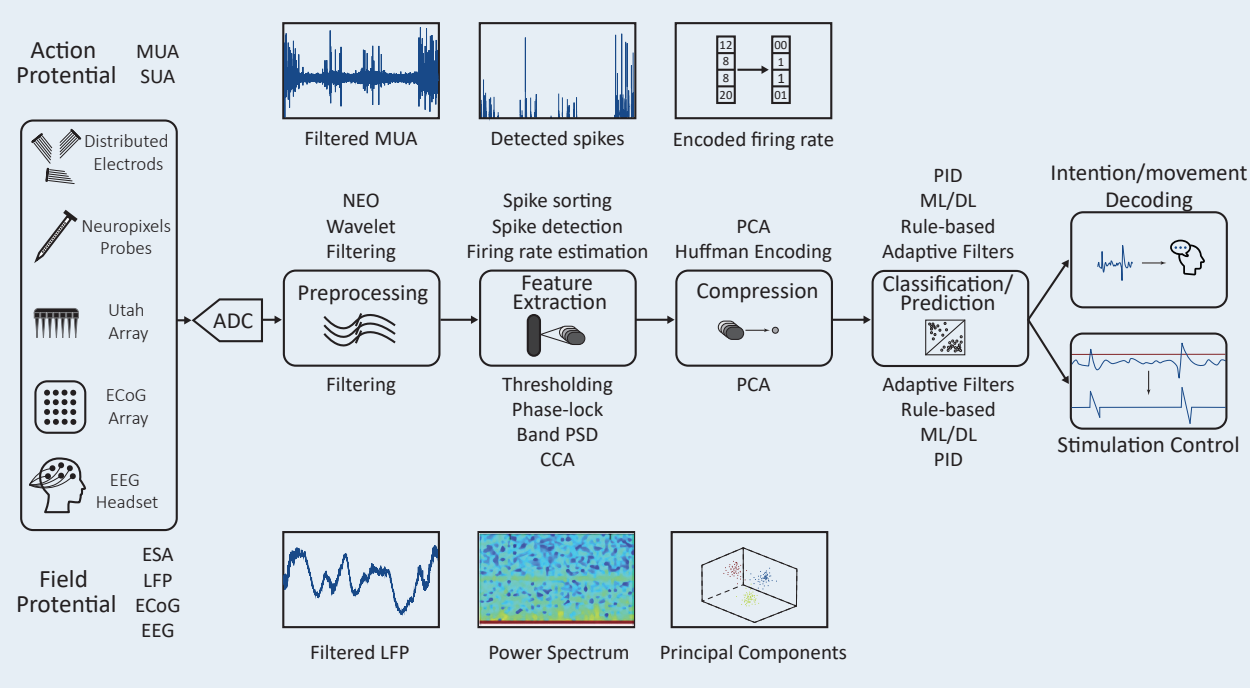

Figure 2.3: A standard workflow for BMI systems. Modules shown: different recording microelectronics, pre-processing, feature extraction, dimensional reduction/compression and classification/prediction techniques, and two use cases for intention/movement decoding and neuromodulation control. Plots at the top and bottom illustrate examples for each step. Terms: MUA: Multi-Unit Activity. SUA: Single-Unit Activity. ECoG: Electrocorticography. EEG: Electroencephalography. ESA: Entire-Spike Activity; this is extracted from action potentials but is processed with field potential techniques [52]. NEO: Nonlinear Energy Operator [53]. PSD: Power Spectrum Density. CCA: Canonical Correlations Analysis, an algorithm commonly used for analysing EEG Steady-state visual evoked potentials [54]. PCA: Principal Component Analysis, a technique to reduce the data dimensionality and is widely used in spike sorting [55]. PID: Proportional–Integral–Derivative control, a useful approach to closed-loop neuromodulation control [56]. ML/DL: Machine Learning/Deep Learning.

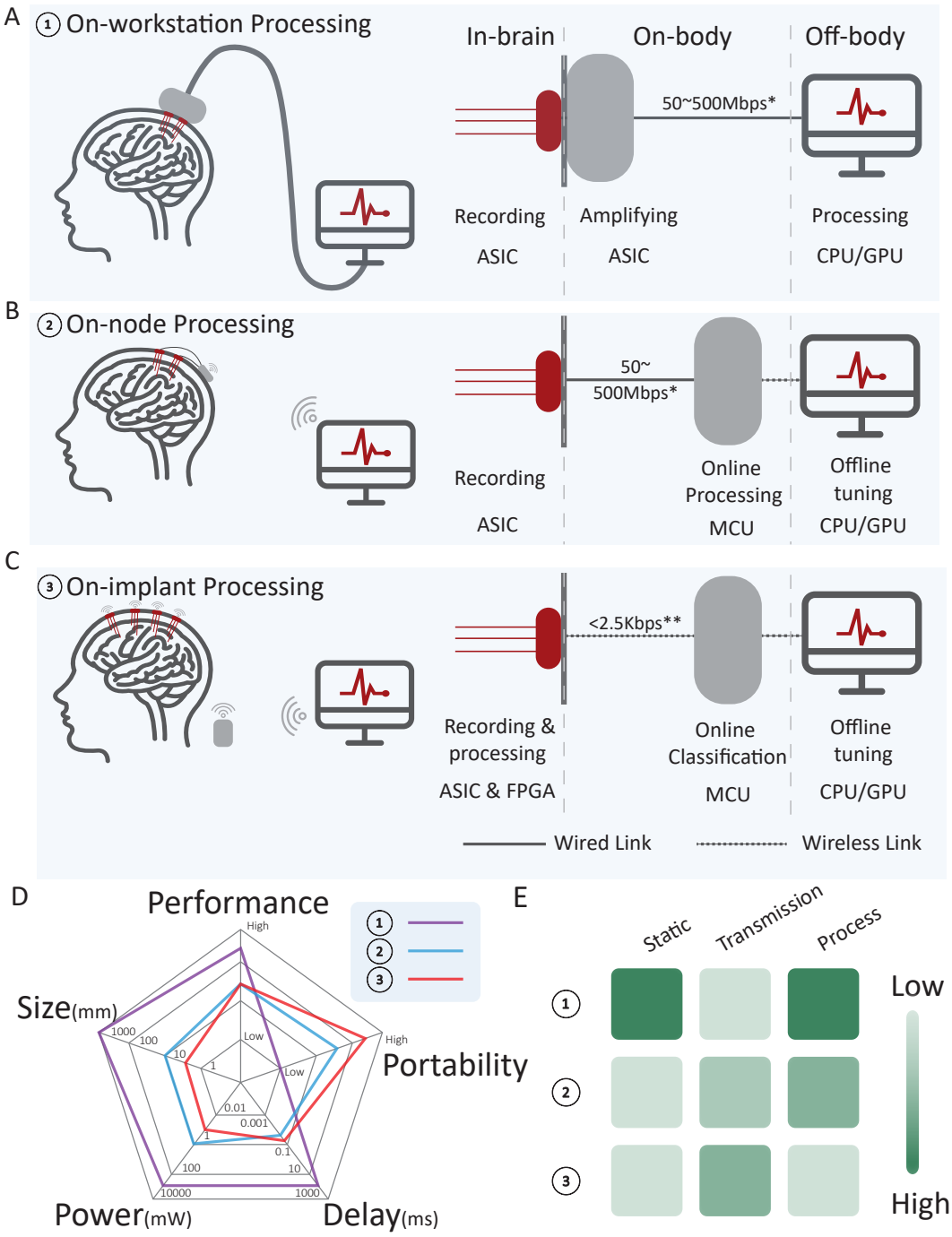

Figure 2.4: A: Wired BMI systems stream and process the amplified neural activities on the workstation. B: Wired online signal processing systems stream and process the neural data on MCUs. The model is tuned on a workstation. C: Distributed wireless online systems pre-process the recording on-implant for bandwidth reduction, reducing the bandwidth of the signal wirelessly sent to the local classifier and response controller (e.g. wearable devices). D: A radar diagram for the three systems showing performance, size, power, delay and portability. E: A comparison of different types of power consumption for the three BMI systems. * Assuming the sample frequency is 24,414 Hz with 128 to 1024 channels with 16-bit data stream. ** Assuming the firing rate is 20 Hz with binary data stream, with a total of 128 channels. This data rate can be even reduced with binning and/or compression.

Chapter 3

Simplifying the conventional feature extraction: Multiplication-free fixed-point spike detection

In this chapter, we delve into the early stages of designing a hardware-efficient and adaptive spike detection algorithm. Our approach simplifies the conventional statistical-based spike detection algorithm by leveraging a fully fixed-point representation and eliminating the need for multiplications. The design philosophy presented in this chapter has been inherited in our later designs, showcasing its potential impact on future implantable BMI systems.

3.1 Introduction

There are several challenges to detecting the spike. The Local Field Potentials (LFP) can leak to the Multi-unit Activities (MUA) band even though a filter is implemented at the analogue front end. After digitisation, the first step to spike detection is to remove the LFPs. The LFP-removed signal can be noisy. Pre-processing for removing the noise and emphasising the spikes can be useful for enhancing the signal quality and improving the detection performance. A threshold is then applied to the pre-processed signals, and the intervals exceeding the threshold are detected as spikes. A robust threshold that can adapt to different noise levels is essential for the success of the spike detection algorithm. Besides

A portion of the content in this chapter has been published in [92] with my full contribution to the software/hardware algorithm design and result analysis.

detection performance, the algorithm complexity is another concern. Algorithms for implantable neural interfaces have a strict area and power budget constraints. Therefore, spike detection should be designed with the minimum number of operations and the least amount of memory.

This chapter presents a highly hardware-efficient (low complexity, low computation) adaptive spike detection algorithm for implantable BMI applications. This includes a mean subtraction filter to first eliminate any lower frequency components (e.g. LFP) from the signal without introducing additional phase distortion. We also propose a novel operator, Amplitude Slope Operator (ASO) as a hardware-efficient alternative to NEO for enhancing the SNR of the EAP signals. The adaptive threshold is calculated periodically by taking a running mean that excludes any detected spikes that have been detected but also runs a concurrent subthreshold detection to exclude a portion of the undetected spikes within the background activity. The algorithm was originally developed in MATLAB using floating-point arithmetic and has been ported to a C implementation using fixed-point arithmetic. This has been implemented on two embedded targets (ARM Cortex M0+ and ARM Cortex M4 microcontrollers) to demonstrate the real-time capability, spike detection performance, and low power/low complexity implementation.

3.2 Proposed spike detection algorithm

The algorithm for adaptive spike detection has been developed in three phases: (1) initial methods conceived, tested, and optimised using MATLAB with floating point arithmetic; (2) then translated to a fixed point representation; and (3) migrated to embedded targets using the C programming language. Quiroga's synthetic dataset [45] as introduced in Section.2.3 has been used for evaluating the proposed algorithm.

The workflow of the proposed spike detection system is illustrated in Fig. 3.1.

3.2.1 Mean subtraction filter

The aim of the mean subtraction is to remove the LFP – the low-frequency 'background noise' in signals observed due to the aggregated network activity across the tissue. The LFP mainly lies at frequencies below 300 Hz. To remove this, we propose using a 16-point moving average filter to find the local mean and subtract this mean from the current value to align the signal. The buffer size 16 is selected by trading-off between the spike detection performance and hardware cost, where increasing the buffer size further only brings less than 3% accuracy improvement with significantly increasing memory usage.

Instead of calculating the mean by summing a series of successive samples, we achieve this by incrementally updating a weighted average as described in Eq. 3.1. This approach

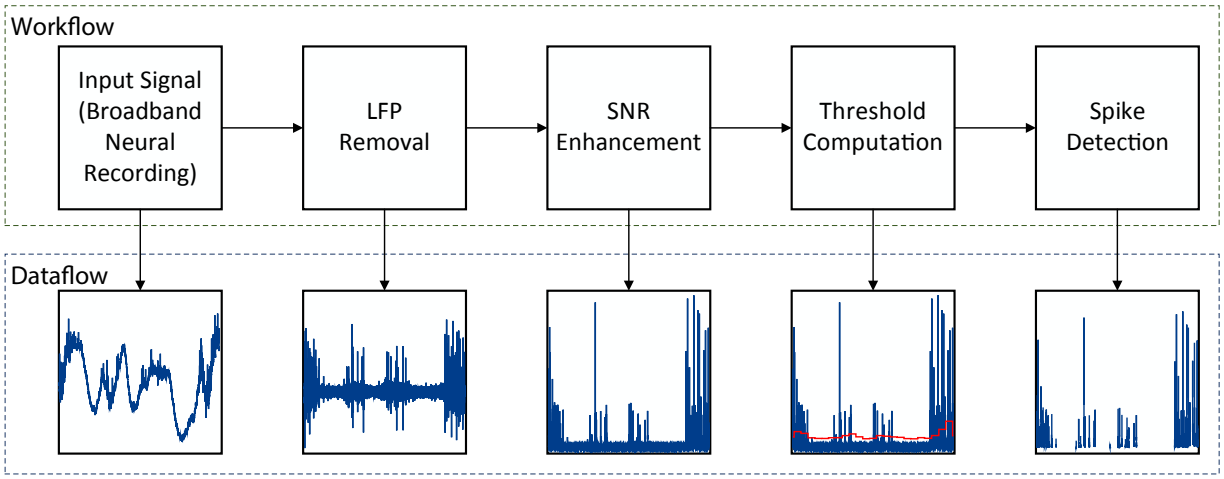

Figure 3.1: Overview of the proposed adaptive spike detection algorithm.

both simplifies the required computation and reduces memory requirements. The result of the LFP removed signal is obtained by Eq. 3.2.

$$\mu_n = \mu_{n-1} - \frac{x_{n-16} - x_n}{16} \tag{3.1}$$

$$y_n = x_n \ - \ \mu_{n-1} \tag{3.2}$$

where μ_n is the n^{th} mean of the signal, x_n is the n^{th} read value and y_n is the mean-removed x_n. With this design, the summing operation in a filtering operation can then be replaced with 2 additions (or subtractions). The number of required operations is therefore reduced by a factor of 8. The complexity is reduced from $O(N)$ to $O(1)$ and the computation is distributed across one duty cycle.

3.2.2 Amplitude slope operator

Inspired by the Multi-resolution Nonlinear Energy Operator [93], described by Eq. 3.3, we propose a new operator, the ASO as shown below. By assuming the signals to be slow varying, we are able to approximate y_{n+k} as y_n obtaining Eq. 3.4. After extracting the y_n from both terms, ASO in Eq. 3.5 is obtained. It uses one less multiplication and k fewer sample buffers, which makes it suitable for real-time applications.

$$z_n = {y_n}^2 - y_{n+k}y_{n-k} \tag{3.3}$$

$$\approx {y_n}^2 - y_n y_{n-k} \tag{3.4}$$

$$= y_n \left(y_n - y_{n-k}\right) \tag{3.5}$$

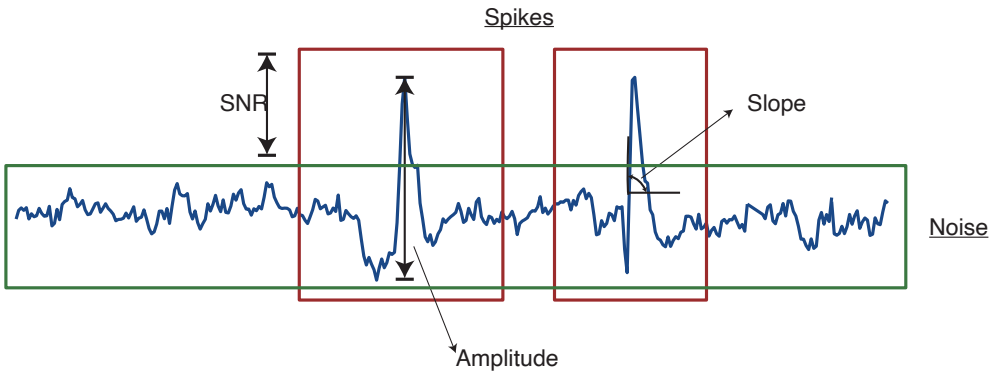

Figure 3.2: The differences between spikes and noise, where the spikes are likely to have more distinguishable amplitudes and slopes.

where z_n is the emphasised data and y_n is the LFP-removed input data. In this equation, y_n stands for the amplitude and $y_n - y_{n-k}$ stands for the slope at this point.

The NEO is proportional to the signal energy and frequency square according to [53], so is able to emphasis the spikes. Referring to Fig. 3.2, a spike is different from noise because it has a higher amplitude and a more significant slope or gradient, i.e. abrupt changes. Therefore, intuitively the ASO amplifies the signal intervals that satisfy both these conditions whilst suppressing signal intervals where only one or neither of these conditions are satisfied. Such a design is also supported by the signal amplitude-gradient distribution in Fig.2.2, where the spike is statistically discernible from the noise in both amplitude and gradient.

3.2.3 Adaptive thresholding

The spike detection step itself is performed by comparing the emphasised signal to a threshold value. This threshold should be determined by the local statistics, providing a measure of the local noise level. A spike is detected when the input signal crosses this threshold value.

There are several challenges in computing the threshold value, particularly in real-time hardware applications. Firstly, the inherent need for statistics in computing the threshold demands a lot of memory in a low-power/low-complexity setting. Secondly, the fact that each recording electrode observes its own unique SNR means each channel needs to be individually trained or calibrated. Thirdly, the signal observed at any given electrode itself changes over time requiring repeated re-calibration. These challenges motivate the development of an adaptive threshold method that uses an algorithmic approach to improve adaptiveness.

Developing an iterative function that estimates the noise level without requiring prior history poses its own challenges. Samples where contain spikes for example can contribute to erroneously raising the threshold value. It is thus essential to provide blanking to capture a robust noise level. The overall threshold comparison process is described in Fig. 3.3. The block labelled *'Spike Detected'* reports the detected spike locations, and details of the block

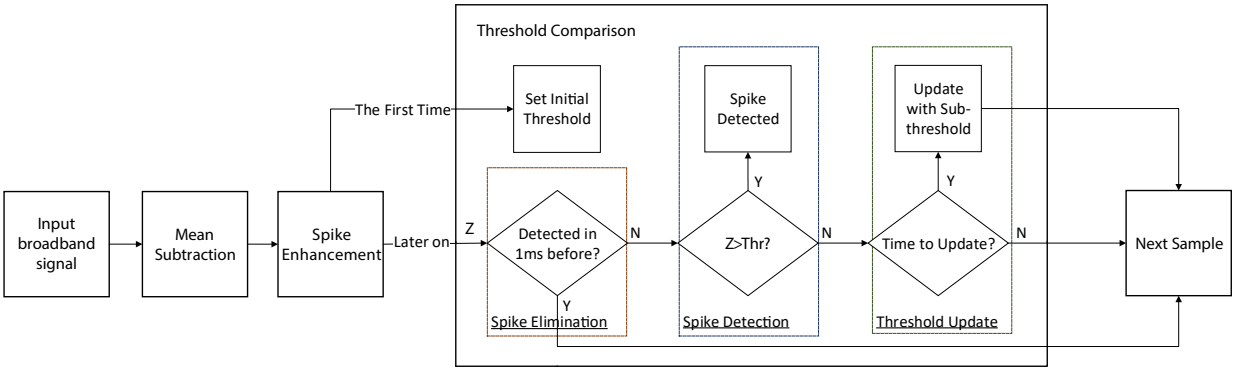

Figure 3.3: Block diagram of the proposed algorithm.

labelled *'Update with Sub-threshold'* are shown in Fig. 3.4.

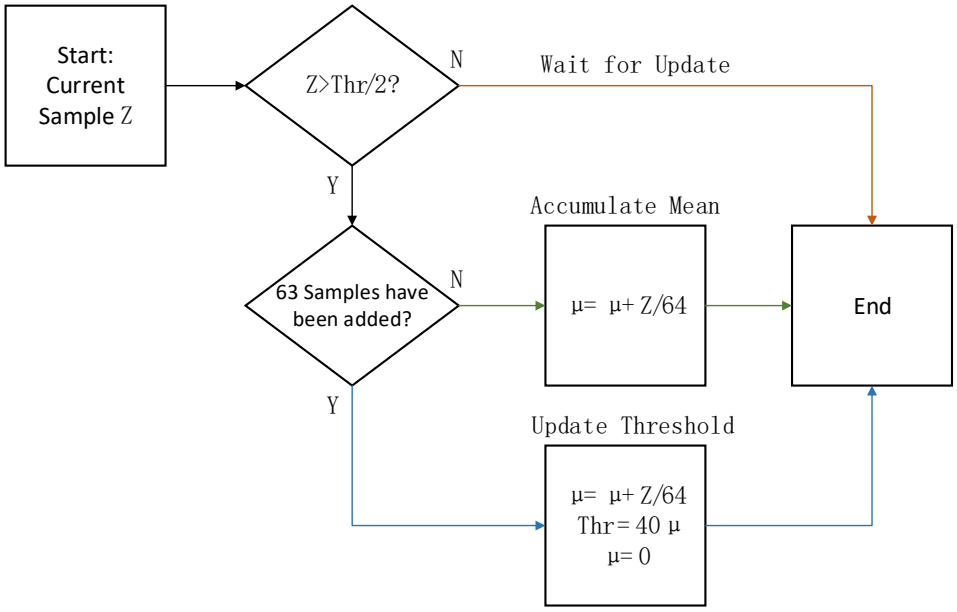

Figure 3.4: Block diagram of the update algorithm

The algorithm operates as follows: an initial threshold value is set by calculating the median value across the first 64 samples. Using the median is essential here because the buffered values are likely to contain spikes, and this provides some robustness to outliers. We compute the initial threshold by adapting the method reported in [45], with the threshold described by Eq. 3.6. This has been modified (multiplier increased to $\times 22$) to operate on the enhanced/emphasized signal (after ASO pre-processor) instead of raw data.

$$Thr = 22\sigma, \quad \sigma = \text{MEDIAN}[\frac{y\,(n)}{0.6745}] \tag{3.6}$$

Although the median is essential for initially setting the threshold, the trade-off between

accuracy and complexity needs to be considered for the operation thereafter (i.e. regular threshold update). The median, mean and standard deviation measures are widely used in set a spike detection threshold. The mean is selected for the regular threshold update after trading-off between the spike detection performance and computational costs. This is performed by taking an average across a 64-sample window, as described in Eqs. 3.7 and 3.8.

$$\mu_{thr} = \frac{1}{64} \sum_{i=n-64}^{n-1} z_i \tag{3.7}$$

$$Thr = 40 * \mu_{thr} \tag{3.8}$$

The 40× multiplier has been determined through empirical tests (i.e. an exhaustive search) also considering hardware implementability. Interestingly, the factor $40/64 = 0.625$ is a close approximation of $e^{-1/2}$, which related to the probability of the Gaussian distribution at one standard deviation point. More explanation is given in Appendix.D The threshold update is duty cycled to occur every 0.6 s, striking a trade-off between the adaptiveness and average power requirements. This duty cycle frequency can potentially be tuned to be significantly lower (as signals do not generally drift over seconds, but more on the scale of hours or days).

Since using the signal mean to determine the threshold value is sensitive to outliers, e.g. the presence of spikes would add error to the noise estimation, we implement two techniques to eliminate the effect of spikes on the threshold computation:

1. *Spike Exclusion (SE)*: This effectively blanks the samples associated with any detected spikes when calculating the local mean. More specifically, this is implemented by invoking a 1 ms 'refractory period' whenever a spike is detected (it is observed that the vast majority of spikes have a positive phase of approximately 0.7 ms). This disables any updates to the threshold computation during this period and as such excludes the spike samples from the mean computation. Implementing this feature additionally avoids multiple detections of a single spike.

2. *Sub-threshold Exclusion (STE)*: For spikes that are not detected but clearly above the 'noise' level, the previous *spike exclusion* step does not exclude this from the mean computation. Therefore, a separate spike detection is performed using half the threshold value to further exclude any distinct spike signals. This has been adopted in preference to reducing the spike detection threshold level to maintain a good balance between overall sensitivity and false detection rate. It worth note that these small spikes are actually informative for neural decoding. The benefits and drawbacks of detecting or excluding these spikes will be shown in Section.6.2.

3.2.4 Fixed-point migration

To improve the efficiency of hardware implementation (i.e. reducing complexity and power consumption), a fixed-point representation is highly desirable.

To convert the previously described algorithm that is originally developed in MATLAB, all 64-bit floating point variables are replaced with 16-bit short variables. This furthermore provides the opportunity to replace multiplication (and division) operations with logical bit shifts wherever scaling ratios are appropriate. The reduced precision in the arithmetic however poses some issues that need to be addressed as below:

Mean subtraction Filter: The fixed point conversion first impacts the mean subtraction operation (to filter out LFP signal), where the reduction in precision leads to the mean value fading to a relatively small value that can lead to a significant error. This can be resolved by regularly recalculating the mean such that its precision is restored before it is allowed to 'fade'. Given the 8-bit precision, the recalculation interval is found to be 40 ms (determined empirically).

However, as the moving average filter is based on a 16-sample sliding window, its implementation can be simplified by replacing the division operation with a 4-bit logical right shift operation.

Amplitude Slope Operator: The ASO function can also be simplified in a similar manner using a logical shift. The ASO operator is thus modified to:

$$z_n = (y_n \; - \; y_{n-1}) << \varphi(y_n) \tag{3.9}$$

where φ refers to the operator that finds the last power of 2 and $<< x$ stands for x-bit left-shifting. Therefore, the need for multiplication can be altogether eliminated.

Adaptive threshold: The computation of the initial threshold is modified to account for the fixed point representation, described in Eq. 3.10.

$$thr_{init} = \text{MEDIAN}[y(n)] << 5 \tag{3.10}$$

The threshold value here is rounded down to 32 times of the local median value ($22/0.6745 \approx 32$, according to Eq.3.6), implemented using a 5-bit left-shift of the median.

The updated threshold value (during the detection process) is then calculated according to Eq. 3.11.

$$thr = \mu_{thr} << 5 + \mu_{thr} << 3 \tag{3.11}$$

For the mean computation required for the *sub-threshold elimination*, we reverse the order of operation (i.e. first right shift by 6 bits and then accumulate) to avoid the possibility of an overflow given the 16-bit width.

Table 3.1: A summary of the embedded target

	CPU	RAM/KB	Flash/MB	CLK/MHz	ADC/bits	DAC/bits	Current µA/MHz
K64F	ARM Cortex M4	256	4	128	16	12	250
K05Z	ARM Cortex M0+	32	1	48	12	12	45

3.2.5 Embedded implementation

To assess the hardware efficiency of the proposed algorithm we implement the algorithm on a commercially-available embedded platform. Here, the power consumption can provide a relative measure of computational complexity, whereas the memory requirements can provide a measure of hardware complexity (i.e. silicon area).

We have implemented the proposed algorithm on two different microcontroller families (ARM Cortex M0+ and ARM Cortex M4), using appropriate development platforms (Freescale FRDM - KL05Z [94] and FRDM - K64F [95] respectively), for the power consumption, memory requirement, and run times measurement of the proposed algorithm. The FRDM - KL05Z platform is designed with efficiency in mind, which is ideal as an ultra-low power demonstrator, while The FRDM - K64F board is a powerful MCU with low power consumption. The board details are given in Table.3.1. The firmware has been programmed in C using the MBED OS online compiler development environment. All variables are defined using either unsigned short or short data types (16-bit integers). Since the typical frequency range of observed spikes (measured extracellular action potentials) is 300 Hz to 3 kHz, the sampling frequency is set to be 7 kHz to avoid spike aliasing and keep the power consumption as low as possible [96]. For spike sorting applications however a higher sampling frequency would be desirable.

The algorithm itself has been developed with computational complexity and hardware efficiency in mind. The previous section focused on translating this to fixed point arithmetic to further reduce complexity. One additional consideration however is to achieve an approximately constant computational load (operations per sample) to ensure efficient hardware utilisation (e.g. clock optimisation).

Although the threshold update itself happens once each 0.6 s, it is essential to spread the processing operations required throughout this 0.6 s period. The threshold update process achieves this by interleaving operations during each sample period. This is implemented by executing one of 3 different branches (shown previously in Fig. 3.4): 'Wait for update', 'Accumulate mean' and 'Update threshold'. This approach both alleviates the need for loops and additionally reduces memory requirements.

The subroutine for detection (mean subtraction, spike enhancement and threshold comparison) is invoked at the sampling rate (i.e. 7 kHz), controlled by a timer (low power timer

in K64F). The MCU is set to a low-power sleep mode between samples and woken up on each new sample. The timing for the threshold comparison duty cycle and spike elimination is achieved by polling. This is to avoid using additional timers. The threshold is then updated every 4,200 samples, and 5 samples will be skipped after a spike is detected for applying spike elimination.

The ASO implementation is different in each of the MCUs. This is because the K64F has a dedicated assembly instruction (Count Leading Zeros, CLZ) that can be leveraged to determine the last power of 2. This reduces the multiplication and division operations to bit shifting. The KL05Z however, does not support CLZ so needs to be implemented in software, therefore requiring even more instructions than one single multiplication. The multiplication therefore remains in the KL05Z implementation with ASO calculated according to Eq. 4.2.

The threshold comparison is implemented identically to the MATLAB implementation.

3.3 Results

This section presents results demonstrating the operation of the algorithm, spike detection performance (e.g. sensitivity, accuracy, false detection rate), and hardware efficiency. The power consumption, run time, and memory requirement for the hardware is then measured to demonstrate the suitability for implantable BMI applications.

3.3.1 Evaluation metrics

Accuracy (Acc), Sensitivity (Sens) and False Detection Rate (FDR) are three metrics commonly used for spike detection performance evaluation, which are formulated as:

$$Sens = \frac{TP}{TP + FN} \tag{3.12}$$

$$FDR = \frac{FP}{TP + FP} \tag{3.13}$$

$$Acc = \frac{TP}{TP + FP + FN} \tag{3.14}$$

where TP, true positive, is the number of spikes that have been correctly detected; FP, false positive, is the number of moments that have been incorrectly detected as spikes; FN, false negative, is the number of true spikes that the algorithm has not detected.

Acc is an overall metric that balances Sens and FDR. However, in spike detection, we generally prefer high-sensitivity algorithms because a higher sensitivity means less information loss. The wrongly detected spikes can be filtered out after detection, but missed spikes cannot be restored.

Table 3.2: Performance of spike detection on synthetic dataset

	Dataset	Easy1				Difficult1				Average
	Noise Level	0.05	0.1	0.15	0.2	0.05	0.1	0.15	0.2	-
Fixed-point Implementation	Acc	0.87	0.86	0.82	0.87	0.81	0.86	0.92	0.92	0.87
	Sens	0.93	0.94	0.89	0.91	0.93	0.92	0.93	0.95	0.92
	FDR	0.07	0.09	0.09	0.05	0.14	0.08	0.02	0.03	0.06
Float-point Implementation	Acc	0.90	0.93	0.89	0.87	0.92	0.91	0.93	0.94	0.92
	Sens	0.93	0.93	0.89	0.86	0.93	0.93	0.93	0.95	0.93
	FDR	0.03	0.00	0.00	0.01	0.01	0.03	0.00	0.01	0.01
Offline spike detection (reported in [45] for comparison)	Acc	0.94	0.95	0.95	0.90	0.82	0.92	0.91	0.77	0.91
	Sens	0.95	0.96	0.95	0.90	0.95	0.93	0.92	0.77	0.93
	FDR	0.02	0.00	0.00	0.00	0.15	0.01	0.00	0.00	0.01

3.3.2 Detection performance on synthetic dataset

The results of the floating and fixed-point implementation are given in Table. 3.2 (Only Easy1 and Difficult1 are included for conciseness). We have also compared our results with [45,97–99]. Spike detection performances are similar (Acc ranging from 90% to 99%). We provide results for one of these datasets in Table. 3.2 (other datasets excluded for conciseness). We however calculate the average scores for each of the different SNR levels across all four dataset, with results shown in Fig.3.5. This clearly shows how the proposed algorithm is robust to different noise levels. In [45], the scores could overfit to some noise levels and behave worse in other cases (especially in 0.2 noise level). The performance variance is much smaller in our implementation. From the table, one can notice that the average performance is similar between the float-point implementation and the detection method in [45]. Migrating to the fixed-point implementation, the sensitivity only degrades for 1% and the FDR is increased for 5%. Such scarification gives us a power reduction of 2/3 as will be demonstrated in the later section.

3.3.3 Mean subtraction v.s. High-Pass IIR Filter

It has been shown that in [96,100,101], causal IIR filters will cause phase distortion, i.e., the phase response of the IIR filter is nonlinear in frequency, which can change the shape of the spikes and reduce the detection accuracy.

However, with the mean subtraction, the phase response of the corresponding filter is approximately linear. A comparison between the phase response of the proposed mean subtraction filter (MS) and a 2-pole causal Butterworth filter (IIR) used in [96] is shown in

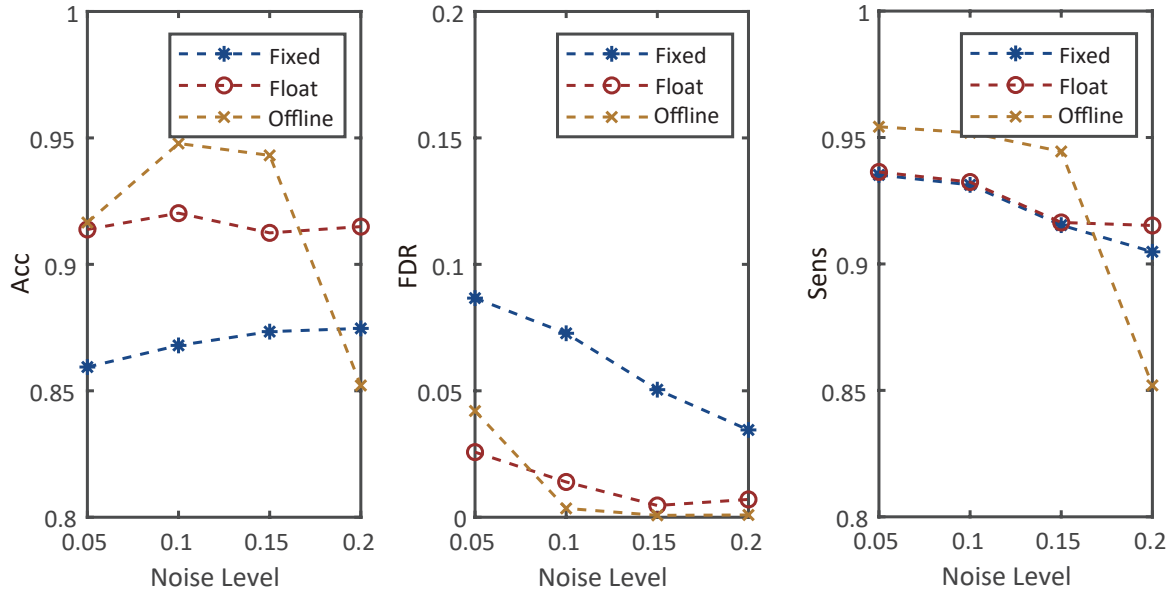

Figure 3.5: A comparison of different algorithms on four noise levels

Table 3.3: Performance of spike detection on different settings. MS - Mean Subtraction, IIR - second order Butterworth filter, ASO - Amplitude Slope Filter, NEO - Nonlinear Energy operator, SE - Spike Elimination, STE - Subthreshold Exclusion.

	Acc	Sens	FDR
MS - ASO - SE&STE	0.92	0.93	0.01
IIR - ASO - SE&STE	0.86	0.92	0.07
MS - NEO - SE&STE	0.90	0.90	<0.01
MS - ASO - SE	0.88	0.89	0.03

Fig. 3.6a. From the filtered results in Fig. 3.6b, it can be observed that by using the proposed filter, the spike shape is preserved, which is beneficial to spike detection. A comparison of the detection performance is shown in Table 3.3.

Here it can be observed that the proposed LFP removal technique can outperform an IIR filter implementation, in particular the FDR, which is improved by some 6 %.

3.3.4 ASO v.s. NEO

The proposed SNR enhancement function (ASO) is compared to a commonly used energy operator (NEO) to assess its suitability as a pre-processing step for spike detection. A sample spike recording is shown in Fig. 3.7 including both the ASO and NEO processed signals. It can be observed that the ASO function produces higher peak values in comparison to the

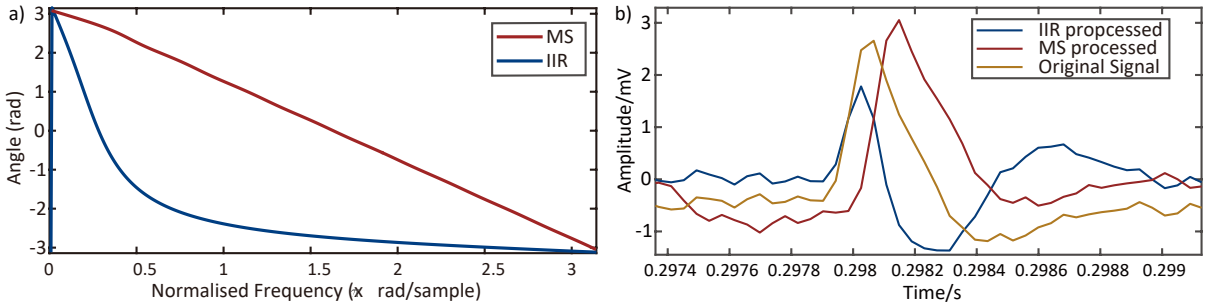

Figure 3.6: a) Phase response of the proposed mean subtraction filter (MS) and a second order Butterworth filter (IIR), where the phase response of the proposed filter has significantly better linearity than that of the Butterworth filter. b) Effect of two different filter types on a spike waveform. This illustrates that the proposed filter does not attenuate the spike peak or distort the spike shape

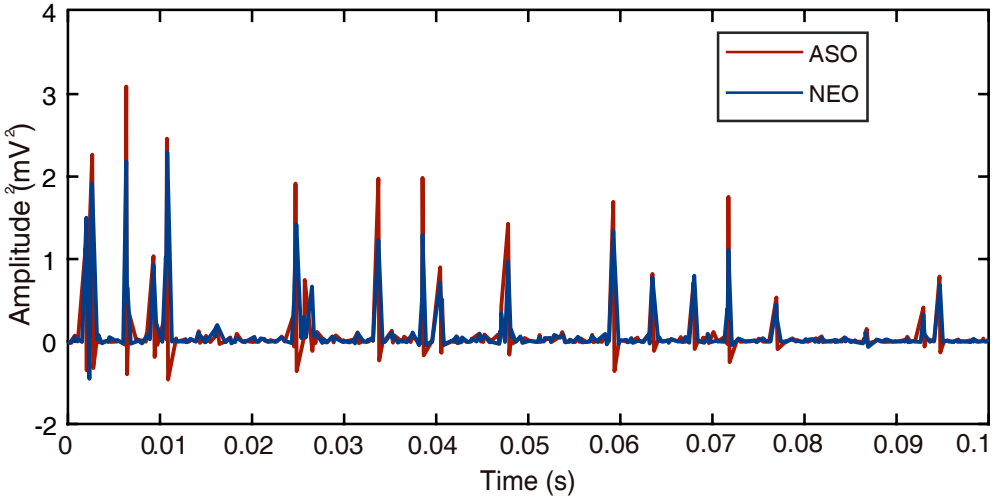

Figure 3.7: Comparison between the ASO and NEO pre-processor functions for SNR enhancement.

NEO.

According to Table. 3.3, a higher sensitivity can be achieved using ASO, however this is at the cost of also a higher FDR. Overall, the detection accuracy is improved for 3% by using ASO.

3.3.5 Spike elimination and subthresholding exclusion

As described previously the noise estimation is used to define the adaptive threshold level. This is based on a running average, excluding any samples referring to detecting spikes and background activity. This is achieved by applying Spike Elimination (SE) and Subthresholding Exclusion (STE).

This process is illustrated in Fig. 3.8 demonstrating the impact of applying these estimation enhancement techniques. The upper plot here shows the SNR-enhanced signal (after

ASO pre-processing) and samples that are used for the noise estimation after SE and STE. It can be observed that by using SE and STE, most of the spikes are removed from the samples used for calculating the threshold. The lower plot shows the threshold based alone on ASO pre-processed signal, and with SE and both SE/STE techniques. The detected spike trains are shown below the plot alongside the ground truth data. This qualitatively demonstrates the effectiveness of the noise estimation algorithm as implemented. It is clearly observed that with the use of the exclusion techniques (SE/STE), the threshold is much more stable and spike detection performance is improved.

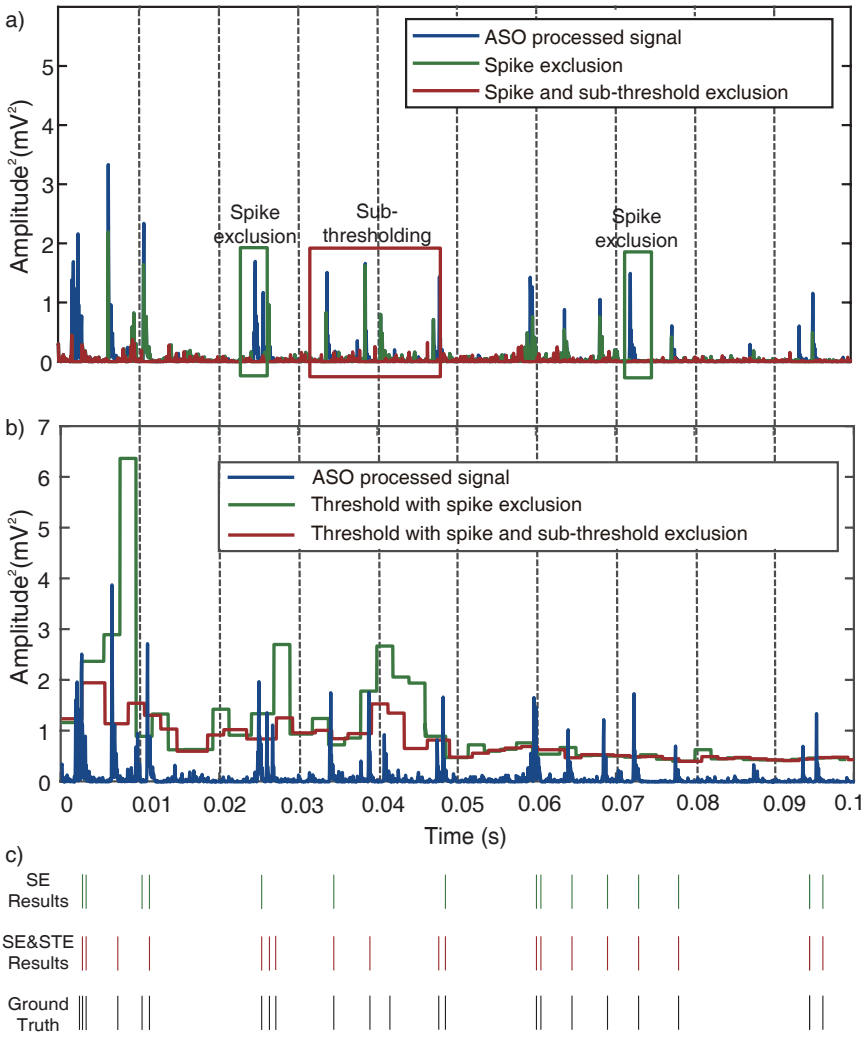

Figure 3.8: Noise estimation process using the SNR enhanced signal. Shown are: (a) SNR enhanced signal (using ASO function) with excluded samples annotated (i.e. for noise estimation); (b) noise level estimation based on averaging samples corresponding to noise (using different techniques); (c) spike train of detected spikes (using different techniques) including ground truth.

The overall performance evaluation is provided in Table 3.3. It can be observed that all the performance metrics are improved by applying both SE and STE techniques: 4 % for Acc, 2 % for FDR, and 4 % for Sens on average.

3.3.6 Hardware evaluation

To demonstrate the suitability of the proposed algorithm for hardware implementation, we have used embedded targets to measure the resource utilisation (memory), power consumption and run time. The test data has been transferred from MATLAB to an arbitrary waveform generator, with output observed using an oscilloscope. A snapshot of the oscilloscope is given in Fig. 3.9. The system here is tested with a broadband input signal (including both LFP and extracellular action potential recording) with an increasing noise level (added Gaussian noise with SNR decreasing from 5 dB to -5 dB), then with a step change (to no added noise). This demonstrates the effectiveness and ability of the algorithm to adapt to signals with changing SNR.

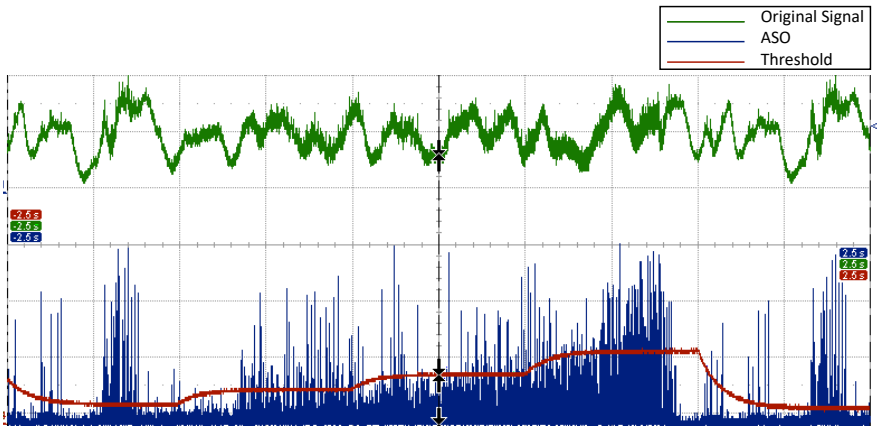

Figure 3.9: Real-time implementation of the adaptive spike detection algorithm on the Freescale FRDM-KL05Z microcontroller platform. Shown are: the original input signal (green trace), SNR enhanced signal (i.e. pre-processed using ASO function) (blue trace) and adaptive threshold (red trace).

Memory utilisation: In total there are 27 variables declared that include the 16-sample buffer (for mean subtraction). The zero initialised data is therefore 27×2 bytes $= 54$ bytes for the fixed-point implementation and 27×4 bytes $= 108$ bytes for the floating-point implementation for both KL05Z and K64F target, with the program memory requiring approximately 3 kb flash. It is worth to be mentioned that in the implementation the threshold calculation does not require a buffer, saving the need for a further 63 variables.

Power consumption and runtime: To improve precision in both energy and timing measurements (taken using an oscilloscope), we repeat each function 500 times, such that an average value for a single operation can be determined. All tests are initially done on the K64F platform (ARM Cortex M4 microcontroller) with the 'final' optimised algorithm and then compared to an equivalent implementation on the KL05Z platform (ARM Cortex M0+).

- NEO vs. ASO: This is first tested for a floating point implementation - the average power consumption for these functions are 0.87 mW and 0.14 mW respectively. The

Table 3.4: Average power, energy per operation and run-time of each function for floating and fixed point implementations

Operation		Floating point, K64F			Fixed point, K64F			Fixed point, KL05Z		
		Power (mW)	Energy (pJ)	Time (ns)	Power (mW)	Energy (pJ)	Time (ns)	Power (mW)	Energy (pJ)	Time (ns)
Subtract mean		3.20	509	160	1.17	187	160	0.15	87	578
ASO		0.14	8	60	0.14	14	100	0.15	9	60
Threshold	Wait for update	0	0	54	0	0	43	0	0	124
	Accumulate mean	0.94	134	143	0.14	19	133	0.04	12	343
	Update	1.62	281	174	0.94	147	157	0.21	113	546
Recalculate mean		-	-	-	1.20	759	612	0.08	85	1000
Average/total amount per sample		1.90	519	275	0.75	204	306	0.13	97	769

run-time for each operation is 60 ns. This corresponds to an energy per SNR enhancement operation of 52.2 pJ and 8.4 pJ. As expected the ASO implementation consumes significantly less power.

- Floating point vs. fixed point: The algorithm is then tested using both a floating and fixed-point implementation. The average power, energy per operation and run-times for each implementation are given in Table 3.4. One key observation is that the power requirements for ASO operation does not reduce for the fixed point implementation, in fact the run-time increases. This is possibly due to the availability of hardware multiplication in the ARM Cortex-M4 or the pipeline hazard of multiple MAC institutions. The average power consumption of the floating-point implementation is 1.9 mW (519pJ per sample), compared to the fixed-point implementation consuming only 0.75 mW (204pJ per sample). The run-time is slightly increased in going from floating point to fixed point implementation from 275 ns to 304 ns (per sample). This would allow for over 450 channels to be implementable using a single microcontroller with 7 kHz sampling rate.

- K64F (ARM Cortex M4) vs. KL05Z (ARM Cortex M0+): The same algorithm is then implemented in KL05Z. The average power, energy per operation and run-times for each operation are also given in Table 3.4. The power is highly reduced with KL05Z since the clock frequency is decreased from 120 MHz to 48 MHz. The average power is measured to be 0.13 mW, which is 17% that of the K64F fixed point implementation and 7% of the floating point implementation on K64F. The run-time however increases to 768 ns. This however still can support nearly 200 channels sampled at 7 kHz.

Table 3.5: Comparison with computational methods for spike detection

	Algorithm	Sens	Low complexity
Our work	ASO	92%	✓
[108]	NEO	>90%	✓
[114]	SNEO	>90%	✓
[113]	SNEO	90%	✓
[115]	Sigma&Delta	99%	✓
[105]	PCA	99%	✗

3.3.7 Comparison to state-of-the-art

Although there exists a significant amount of previous work on spike detection [102–104], the vast majority focuses on computational methods for offline analysis. Although there are several examples in the literature of hardware implementations [49, 105–111], it is not possible to assemble a fair and comprehensive comparison. This is in part due to the diverse hardware methods available (e.g. ASIC implementation, embedded processor, reconfigurable logic, computational emulation). In this section, we therefore select a narrow representative sample to include some qualitative comparison.

Computational methods: Five specific algorithms from the literature have been selected for comparison with the presented algorithm, provided in Table 3.5. These methods achieve a slightly higher sensitivity to the method described herein (compared to fixed point implementation) but with significantly higher computational complexity. For example, not to mention the complexity of PCA decomposition in [105], which required offline training and real-time projection, the complexity of template matching [112] is at least O(MN), where M stands for the template length, and N is the signal length. For NEO [108] and SNEO [113, 114], this is reported to have a complexity of O(N), which is comparable to the proposed method, however, the LFP removal are not considered.

Hardware methods: The hardware implementation is compared with four different works that target embedded processors [109, 116] and an integrated circuit implementation [106, 117], shown in Table 3.6. Orders of magnitude of power are reduced compared to the studies using MSP430 but this is also due to the different CPU architecture. Compared to the work in [116] which uses KL25Z MCU featuring the same ARM Cortex M0+ core, the proposed implementation has achieved even lower power consumption given that that work does not support pre-processing and employed static threshold.

Table 3.6: Comparison with hardware methods for spike detection

	Power /mW	Hardware	Pre-peocessing	Adaptive Thr
Our work	0.12	KL05Z	✓	✓
[116]	0.255	KL25Z	✗	✗
[109]	>41.9	MSP430	✓	✗
[118]	16.5	MSP430	✓	✓
[106]	0.11	0.6 µm CMOS	✓	✓
[117]	$5e^{-5}$	0.13 µmCMOS	✓	✓

3.4 Discussion

The results above demonstrate the good performance of the proposed spike detection algorithm and the hardware cost reduced from hardware-efficient design. However, there is still improvement potential.

3.4.1 Multiplication-free fixed-point spike detection

By using a fully fixed-point representation and replacing the multiplication with a logical shift, a substantial reduction of over 50% in power and memory usage has been achieved in on-implant spike detection. This represents a big step forward for on-implant spike detection. However, reduced precision is detrimental to noise estimation and bit-shift-estimated multiplication further reduces the spike detection performance. Though the sensitivity is less affected, 5% more false detection reduces the fixed-point algorithm detection accuracy to below 90%. While the hardware cost is significantly reduced, there is still a need to enhance spike detection performance to ensure the extraction of high-quality features from the raw neural signal.

3.4.2 Statistical-based thresholding

In this study, a two-step approach is employed for setting the threshold in spike detection. Initially, the median threshold is utilised to establish the initial threshold, eliminating the impact of the spike outliers. Subsequently, the mean threshold is employed to reduce power and resource usage. Additionally, Spike Elimination (SE) and Subthreshold Exclusion (STE) are incorporated with the mean threshold to mitigate interference from spikes. However, the practical implementation on FPGA or ASIC raises concerns. There are some concerns in practice when translating the implementation into FPGA or ASIC. The need for a dedicated

circuit solely for median calculation results in wasteful resource utilization since it is used only once. Furthermore, during parameter selection, it is observed that the optimal parameter setting for spike detection varies with different noise levels. The multiplier to the mean value tends to be smaller in lower SNR scenarios. The adaptiveness is limited and calibration is still required. These challenges motivate the subsequent chapter to investigate the optimal setting of spike detection parameters, aiming to go beyond trial and error methods and improve adaptiveness.

3.5 Conclusion

This chapter has presented a efficient spike detection algorithm and hardware realisation intended for autonomous, calibration-free high channel count systems. Key features of the algorithm include:

- It includes a mean subtraction filter that can minimise the phase distortion whilst removing the LFP.
- A novel pre-processor to enhance SNR with a lower computational complexity compared to the gold standard NEO operator.
- A novel and robust thresholding schema which can reduce the effect of the spikes and multiunit activities on the stability of the threshold.

The hardware implementation of the algorithm achieved the following:

- Power consumption is amongst the lowest reported (130 μW average) for a microcontroller implementation including pre-filtering, SNR enhancement, adaptive threshold, and spike detection.
- From a hardware portability view, to prolong the battery life, the power consumption is expected to be scaled down by 1-3 orders of magnitude (depending on technology) if translating to FPGA or ASIC implementation.
- Can be implemented using only fixed point arithmetic with no need for multiplications.
- Requires a program memory of 3 kB and under 0.1 kB RAM.
- Low computational complexity allows over 100 channels to be implemented on a single ARM Cortex-M0+ device.

While significant achievements have been made, it is important to acknowledge that the multiplication-free fixed-point implementation exhibits performance degradation in spike detection compared to other state-of-the-art algorithms. Furthermore, the challenge of adaptiveness remains unresolved, as calibration is still necessary when the brain environment undergoes substantial changes. However, the multiplication-free fixed-point implementation has already demonstrated remarkable reductions in both power consumption and area utilisation. This design philosophy will be carried forward and utilised in subsequent designs.

Chapter 4

The puzzle of adaptiveness: Statistical-based thresholding deadend

In the journey to develop hardware-efficient adaptive spike detection algorithms, this chapter represents a pivotal turning point. By evaluating conventional statistical-based spike detection algorithms and constructing an idealised model using synthetic data, it becomes evident that noise-statistical-based thresholding is unable to adapt to varying levels of noise within a reasonable power/area budget. This observation serves as a critical inspiration for the subsequent work in this thesis aimed at discovering new low-complexity thresholding mechanisms that can better adapt to changing noise levels.

4.1 Introduction

In the previous chapter, we introduced a statistical-based spike detection algorithm which achieves significantly reduced hardware costs. This algorithm follows the conventional approach of setting the threshold as $T = \alpha N$, where N represents the noise statistics and α is a user-defined parameter. This approach is favoured in on-implant implementations due to its simplicity [96]. However, the adaptiveness of this approach raises concerns. Some studies have attempted to find the optimal threshold for spike detection. In [122], it was found that the noise mean value is the best threshold driving factor among other noise statistics. However, defining a single optimal multiplier suitable for varying SNR levels is challenging.

A portion of the content in this chapter has been published in [119–121], with my contribution to the software/hardware algorithm design and result analysis. Section 4.2 was accomplished in collaboration with Mr Alexandrue Oprea, who conducted the algorithm accuracy comparison under my supervision.

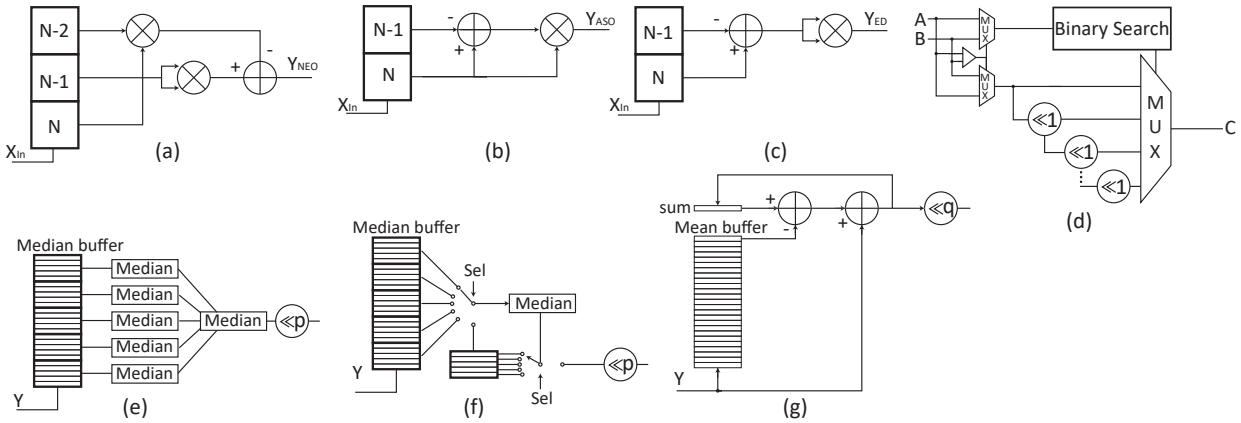

Figure 4.1: The signal flow diagram of baseline emphasisers and thresholding modules. (a) NEO (b) ASO (c) ED (d) Shift-based multiplier (e) Unrolled median recursion (f) Rolled median recursion (g) Accumulated mean

In [123], it is assumed that the neural recording distribution consists of an exponent component (noise) and a power component (spikes). The threshold is set at the intersection point of two estimated distributions. However, distribution estimation can be resource-intensive and power-consuming in hardware implementations.

Numerous spike detection algorithms are reported in the literature, including various spike emphasis techniques and threshold mechanisms. However, not all of these algorithms are suitable for on-implant use due to their computational complexity and hardware constraints, such as silicon area and power limitations. Before translating the algorithms into hardware instances for the design of wireless implantable BMIs, a comprehensive evaluation of the trade-offs between hardware and algorithmic considerations is necessary. In this chapter, we begin by investigating the performance of the thresholding method using different statistics. Through practical simulations, we demonstrate their adaptiveness and compare their hardware costs to evaluate their suitability for on-implant use. Subsequently, we construct idealised spike detection models to assess the theoretical performance of employing various statistics. Our objective is to identify an optimal threshold setting capable of adapting to different noise levels without the need for parameter training or calibration. By conducting these investigations, we aim to advance our understanding of spike detection algorithms and their potential for achieving robust performance in diverse neural recording environments.

4.2 Statistical-based spike detection performance practical simulation

We have selected some of the most widely used methods that can be implemented in fixed-point representation, with minimal multiplication requirements. Therefore, they could potentially be implemented on implants. This section will introduce the selected methods with the gate-level design diagram shown in Fig.4.1. Their spike detection performance and adaptiveness with different noise levels will be evaluated and the FPGA implementation resource occupation and power consumption will be reported. A sample rate of 7 kHz per channel with a 10-bit resolution in fixed-point representation is used following [96].

4.2.1 LFP filters

This chapter does not focus primarily on the LFP filter, as there are existing studies [46, 96] that extensively investigate the most efficient filter topologies for rejecting LFPs. A linear phase filter is ideal, as this does not introduce any phase distortion and therefore the spike shape is preserved. Phase distortion is especially critical for applications requiring single unit activity, as spike sorting relies on differences in spike shape dynamics. However, as neural decoding for iBMIs can be achieved effectively using only threshold crossings (i.e. MUA data), a linear phase response is not critical, whereas filter complexity is important. Therefore, a simple two-pole Butterworth filter is used in this chapter as [96] recommends.

4.2.2 Emphasisers

Using an emphasiser to enhance the SNR of the LFP-removed signal is critical for improving detection performance, in particular for low SNR recordings.

The implementation of the selected emphasisers are shown in Fig. 4.1(a) - (c). Nonelinear Energy Operator (NEO) in Eq.4.1 and its variations are the most widely used emphasisers. Amplitude Slope Operator (ASO) proposed in the previous chapter and Energy of Derivative (ED) [124] both used one less multiplication to emphasise the spikes and were reported to outperform NEO, so they were chosen for this work given in Eq.4.2 and Eq.4.3.

$$Y_n^{NEO} = X_n^2 - X_{n-1}X_{n+1} \tag{4.1}$$

$$Y_n^{ASO} = X_n(X_n - X_{n-1}) \tag{4.2}$$

$$Y_n^{ED} = (X_n - X_{n-1})^2 \tag{4.3}$$

Absolute values are taken after the operation allowing the algorithm to detect double sides peaks with a single threshold.

Other variations may require smoothing filters, buffering more samples, or switching between different emphasisers based on noise levels. The detection performance can be improved, but the additional hardware cost is expensive for implant use and thus not chosen.

In order to design a multiplication-free system, the multiplication results are estimated using bit shift according to Eq.4.4 the same as the previous chapter as shown in Fig.4.1.(d).

$$\begin{aligned} C &= B * A \\ &= \Sigma^{i}\{B << [(i-1) * b_i(A)]\} \\ &\approx B << (\mathrm{MSB}(A) - 1) \end{aligned} \tag{4.4}$$

where A>B, b_i stands for the binary value of number at i-th digit and MSB finds the index of the most significant bit.

Instead of multiplying two values, the larger value is left-shifted by N bits, where N is the index of the most significant bit of the smaller value. A binary search is adopted to find the most significant bit of one number. The circuit is shown in Fig.4.1.(d)

In order to test whether such estimation is more efficient than using LUT to implement a multiplier, we have implemented three different multipliers: 1) Digital Signal Processor (DSP) Multiplier 2) Look-Up-Table (LUT) Multiplier 3) Bit shift multiplier. These different implementations are compared from algorithmic and hardware perspectives.

4.2.3 Thresholding

Thresholding is the most important step in spike detection. A threshold is typically set by levelling up the signal statistics. These statistics include the mean, RMS/STD, and median values.

The mean value is preferred due to its simplicity. Instead of sum all values in the threshold buffer, the mean is calculated in an accumulated manner as shown in Fig.4.1.(g). The sum of the buffer is stored, the oldest value is subtracted from the mean, and the new value is added to the result. Such an operation can be down in one clock cycle with only two additions which reduces the hardware cost to a minimum. The threshold is then calculated through bit shift. Unlike storing the mean as done in the previous chapter, storing the sum requires a larger buffer. However, by storing the sum, the mean value does not fade and there is no need to recalculate it.

The median is used because it is less affected by outliers (spikes), but it is inefficient on hardware due to its $O(n^2)$ computational complexity. To address this shortcoming, A median recursion is proposed to estimate global median values from subset medians, reducing the complexity to $O(nlog(n))$. In more detail, as shown in Fig.4.1.(e)/(f), to calculate the median of 25 numbers, we divide all the numbers into five subsets and take the median value of each subset. Finally, we use the median of these medians as the output. This method

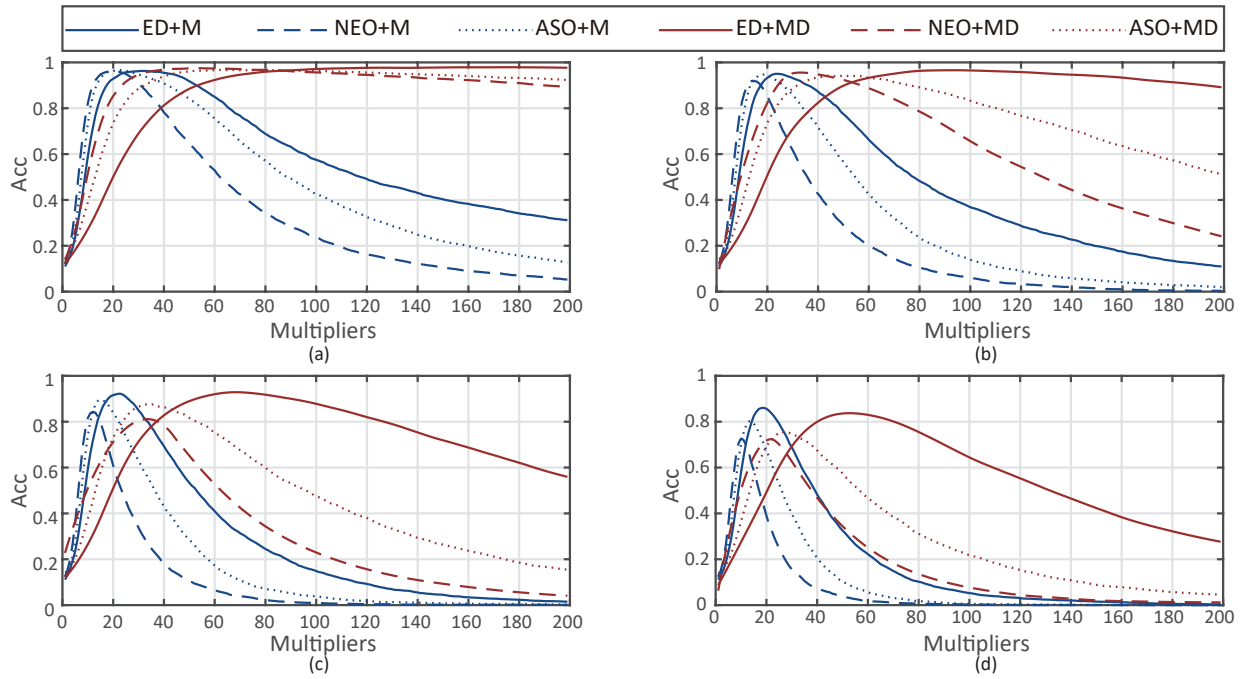

Figure 4.2: The performance of different combinations of the baseline methods ED - Energy of Derivative, NEO - Nonlinear Energy Operator, ASO - Amplitude Slope Operator, M - Mean, MD - Median, Acc - Accuracy. MD typically archives higher accuracy and flatter curves, especially when combined with ED. Four subfigures are the accuracy at different noise levels. (a) 0.05 (b) 0.1 (c) 0.15 (d) 0.2

does not guarantee finding the real median value, but it is close enough to provide a reliable threshold.

Three different median implementations are adopted. 1) Normal median 2) Rolled median recursion: The subset median search logic is shared across the subset. 3) Unrolled median recursion: Each subset has its own median search logic.

It requires either massive computation to obtain RMS values or multiple cycles for RMS estimation to saturate [47], neither of which is preferable for on-implant use and, therefore not considered here.

4.2.4 Spike detection performance

Fig.4.2 is the average detection accuracy of different combinations of ED, NEO or ASO, and 16-point mean or 25-point median at different levels, obtained from tests on the synthetic data [45] the same as previous chapter. The 16/25 buffer thresholds buffer size is considered to use acceptable resources for on-implant use. A detailed analysis of the emphasisers and thresholding mechanisms is given below.

Emphasisers: In Fig.4.2, ED (solid lines) is the most accurate emphasisers among the three baseline methods at all noise levels for both thresholding mechanisms. ED is also the most adaptive emphasiser in terms of multiplier choice robustness (flatter curves).

ASO (dotted lines) has comparable peak Acc, but its performance decreases faster with increasing noise levels, while the performance of NEO (Dash lines) degrades the most rapidly. The observation above implies that the gradient is more discriminative than amplitude in spike emphasising, particularly as noise levels rise and ED becomes more effective than NEO or ASO.

Thresholding: The performance of the estimated median (MD, Red lines) and the Mean (M, Blue lines) is comparable in terms of the highest detection Acc they can achieve. However, a typical shortcoming of statistically-based thresholding is that multiplier selection might impact detection performance, requiring manual threshold calibration in practice. As seen in Fig.4.2, the median is more robust to the choice of multiplier since it produces a flatter curve nearer to its peak. This characteristic is particularly apparent around 0.05 noise level when the method can nearly always achieve high Acc. In contrast, when utilising the mean, the Acc degrades rapidly as the multiplier moves away from the maximum values. The median outperforms the mean according to the above results because of its robustness to the outliers. Spikes can affect local statistics estimation, especially when the buffer size is limited. Though several samples are skipped after one detection to reduce such an impact, some residual energy of spikes and undetected spikes can still be contaminated, thus disturbing the threshold calculation. The outlier robustness of the median then plays a critical role in relaxing the impact of the spikes.

4.2.5 Spike detection adaptiveness

To further investigate the adaptiveness of the mean and median thresholding in practice, we simulated the scenario that recordings for parameter selection are statistically different from the recordings tested onward. The recordings of noise levels 0.05 - 0.15 are used to find the best parameter settings, and the recordings of noise level 0.2 are then tested on these settings. These test results for different buffer size is then plotted as the solid lines in Fig.4.3.(a). The dashed lines are the best detection accuracy algorithms can achieve if the settings are appropriate.

For median-based spike detection, the detection accuracy can be improved effectively with the increased buffer size, while it becomes less effective for the mean for a buffer size larger than 30. This is also because of the outlier robustness of the median operation. A large mean buffer can help estimate more accurate mean values but also increases the chance of including undetected small spikes.

If the parameters are appropriately set, median-based spike detection can achieve the highest spike detection accuracy when the buffer size is large enough, as the red dashed line shows. However, as the solid red line shows, when the noise level is different between calibration and experimental recordings, median-based threshold detection performance can

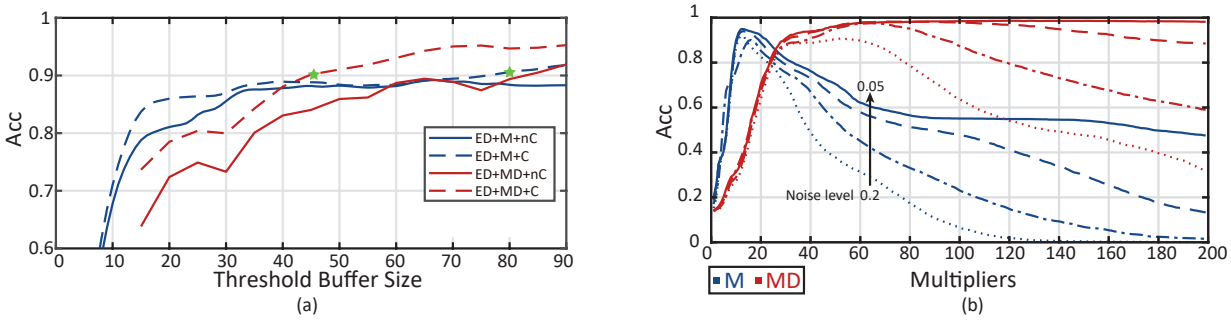

Figure 4.3: (a) The spike detection accuracy is evaluated with Calibration (C) and without calibration (nC). The performance of the ED+MD algorithm deteriorates significantly without calibration. (b) The spike detection accuracy is analyzed at the green stars in (a) where it achieves above 90% detection accuracy. Results indicate that the Mean thresholding displays better adaptiveness to varying noise levels, whereas the Median is the weaker.

be degraded for at least 5% according to our simulation. Mean-based thresholding is less affected from the training-testing deviation, but the overall performance is lower compared to the median-based one, especially when the buffer size is limited.

Fig.4.3.(b) shows how their detection accuracy varies with the multipliers or the target detection rate at different noise levels. It explains why mean-based thresholding can achieve better adaptiveness. The response of the accuracy to the multiplier is similar across different noise levels at the peak region. However, the median varies significantly. Therefore, even though the median can achieve higher detection performance when threshold buffers are enough, it is not suitable in practice due to low adaptiveness.

In the meantime, from Fig.4.3.(b), we can observe one drawback of the mean thresholding that it is sensitive to the inaccurate threshold settings (similar to what is observed in the thresholding subsection in Section. 4.2.4). As the responses of the mean are sharper around the peak, if the multiplier is inaccurately set, the impact on spike detection accuracy is more significant than that of the median.

4.2.6 Estimation error

In order to reduce hardware complexity, we used recursive median and shift-based multipliers as approximations for the actual median values and multiplications. Noting that the spike detection accuracy reported in the sections above is tested without using these estimations, we will now quantify the impact on spike detection performance.

The results, shown in Fig.4.4.(a), indicate that the approximation of the median introduces about 5% degradation, and the trends are similar across all three emphasisers. When using estimated multiplication, as the Fig.4.4.(b) shows, the degradation is more severe for the NEO, as two successive multiplications amplify the estimation error. The ED and ASO

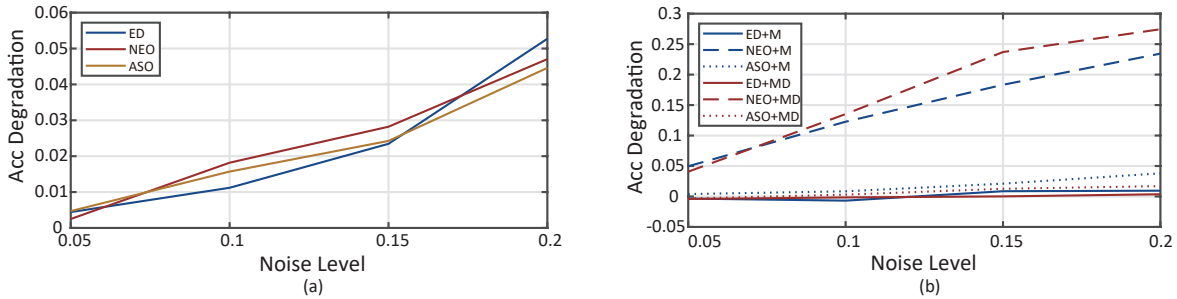

Figure 4.4: (a) Performance degradation when using median recursion to estimate median values. (b) Performance degradation when using shift-based multipliers to approximate the product.

methods only involve a single multiplication and show only minor degradation.

4.2.7 Hardware cost comparison

The FPGA implementation provides us with a rapid workflow to compare hardware complexity, identify bottlenecks, and optimise hardware implementation. The algorithms are implemented on a Digilent Artix-7 FPGA development board featuring the XC7A35TICSG324-1L FPGA core built on 28 nm technology using Vivado 2022.2. Resource utilisation provides a relative measure of the area and power consumption of the ultimate ASIC design. We synthesised the algorithms into different modules. The resource utilisation is shown in Table. 4.2. Detailed analysis is given below:

Emphasisers: Out of the three emphasisers, NEO requires almost twice the number of LUTs and RAM compared to the other two, due to an additional multiplication and sample buffer requirement. On the other hand, ED occupies slightly fewer LUTs compared to ASO.

We also investigated how three different implementations of the multipliers can affect the resource utilisation. The LUT-based multiplier provides a fair reference for the corresponding ASIC design.Compared to the LUT-based multipliers, half of the LUT can be saved. When DSPs are used, most of the LUT resources can be saved, but NEO requires two DSPs.

The dynamic power breakdown is shown in Fig. 4.5 according to the power estimator with the implementation simulated at 100 kHz. Similar to the resource occupation, it provides a similar insight. ASO and ED can save nearly half of the power, but interestingly, ASO consumes slightly less power than ED, even though ED requires fewer LUTs. Such difference is mainly from that ASO consumes less signalling power than that of ED. It is potentially due to their placing and routing design.

Another notable observation is that using shift in ASO and ED does not significantly reduce the total dynamic power consumption. The logic power is the main bottleneck in the LUT implementation, and a simpler processing logic can have the greatest impact on power reduction. When shifts are used to replace multiplications, though it uses less logic,

Table 4.1: FPGA implementation resource utilisation

		LUT	Registers	RAM Width (Bits)	DSPs
LFP filter	Butterworth filter	196	0	20	0
Emphasisers	NEO LUT Mul	272	0	20	0
	NEO shift	147	0	20	0
	NEO DSP	50	0	20	2
	ASO LUT Mul	140	0	10	0
	ASO shift	75	0	10	0
	ASO DSP	29	0	10	1
	ED LUT Mul	121	0	10	0
	ED shift	48	0	10	0
	ED DSP	10	0	10	1
Thresholding	Mean (16/64)	95	0	380/1300	0
	Median unroll (25/50)	1495/2579	0	500/1000	0
	median roll (25/50)	468	184	500/1000	0
	median real 25	>7500	-	-	-
Full system	ED+Mean	140	24	390	0

the signalling becomes more complex. As a result, dynamic power is reduced by only 20%. In such cases, optimal placing and routing become crucial factors to consider.

Overall, ASO and ED can have similar hardware costs, while the gold standard NEO operator can occupy and consume twice as many resources and dynamic power. By using shift to replace multiplication, the area occupation can be significantly reduced with only about a 1% performance degradation

Thresholding: The 16-sample mean and 25-sample median are two compact algorithm settings, while the 64-sample mean and 50-sample median are two settings that can achieve around 90% detection accuracy at the low noise level as shown in Fig. 4.3.(a).

The RAM bandwidth always increases linearly with the buffer size increase. The median consumes much more LUTs compared to others. However, while the conventional median operation for 25 numbers can consume over 7500 LUTs, more than 5 times reduction can be made through the median recursion. Such recursion implementation of median estimation makes the median operation achievable in hardware, especially in the rolled version, where it can be implemented using less than 1000 logic cells, but it still requires significantly more LUTs compared to using mean.

The power of the thresholding algorithm is highly data-dependent and therefore is not

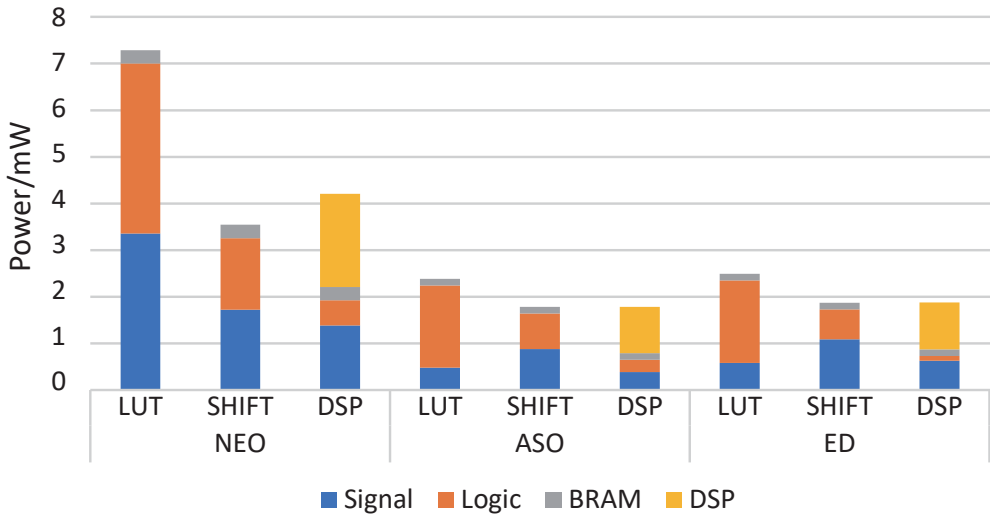

Figure 4.5: Power breakdown for different implementations of NEO, ASO, ED and ADF emphasisers simulating at 100 kHz.

estimated.

Overall, ED is the best emphasiser as it achieves the highest decoding accuracy and adaptiveness while requiring the least amount of resources. However, a trade-off must be made when choosing between mean and median. Mean achieves better adaptiveness with significantly fewer resources but only marginally acceptable spike detection accuracy. In contrast, median has higher spike detection accuracy, but it comes with significantly increased hardware complexity and reduced adaptiveness in different noise levels. Considering the highly constrained resource and power budget, mean should be preferred for on-implant use. In the meantime, replacing multiplication with shift should be adopted, as it can nearly half the resource occupation and reduce the power consumption by about 20% while only degrading the accuracy for 1% at maximum.

4.3 Spike detection idealised modeling

In the previous section, mean thresholding was chosen as the optimal trade-off among spike detection performance, adaptiveness, and hardware cost based on recordings at four different noise levels. However, the issue of relatively lower spike detection accuracy still remains, and the optimal selection of the multipliers is unknown. It is also unclear whether using standard deviation can improve spike detection performance and adaptiveness. To address these questions, we generated our own synthetic dataset with more noise levels and employed four different thresholding models based on noise mean, standard deviation, and peak mean values, assuming perfect knowledge of the signal. We then swept the threshold multipliers to identify the theoretically optimal settings for achieving the best spike detection accuracy

and adaptiveness. This approach allows us to provide guidance for threshold setting with less reliance on trial and error.

4.3.1 Synthetic dataset

In order to have more comprehensive control of the signal characteristics, we have generated a synthetic dataset in different noise levels for assessing the detection performance. The synthesising procedure is based on [45]. We use real recordings [125] from the motor cortex of NHPs sampled at 24,414 Hz to generate a synthetic dataset with ground truth. Synthetic recordings are formed by adding the noise templates with spikes templates. The noise templates are truncated from the LFP-removed real neural recordings in which periods spikes do not appear. The noise standard deviation (STD) is normalised to 1 and modified according to the desired SNR. The spike templates are extracted from the real recordings using WaveClus [55] and there are 1,000 different templates with varying amplitudes for selection. The arrival of spikes can be simulated as a Poisson distribution with λ equal to the firing rate. By simulating multiple Poisson distributions of spike arrivals, multi-unit activities can be generated. One spike is randomly selected from the template spikes and chained with former spikes at desired arrival time. The gaps is filled with zeros. After chaining all spikes from different cells, the spike amplitude will be normalised according to the spike peak mean values resulting in the unit average peak amplitude. Adding the spikes with the noise according to the desired SNR and firing rate forms one synthetic recording.

One synthetic recording is defined with three parameters: firing rate, number of cells and SNR. The firing rate is the single-cell spike rate which determines the λ of each Poisson distribution, and the cell number determines the number of Poisson distributions to be simulated. SNR is defined as the ratio of the mean value of spike peak amplitude and the STD of the noise.

4.3.2 Spike detection models

In order to find the optimal threshold derived from the signal statistics, we have assumed to have the perfect estimation of the noise and spike statistics. Three indicators: μ_{noise}, σ_{noise}, μ_{peak}, which are (absolute) noise mean, noise STD and spike peak mean, are used to set the threshold:

$$T = \alpha\mu_{noise} \tag{4.5}$$

$$T = \alpha\mu_{noise} + \beta\sigma_{noise} \tag{4.6}$$

$$T = \alpha\mu_{peak} \tag{4.7}$$

$$T = \alpha\mu_{peak} + \beta\mu_{noise} \tag{4.8}$$

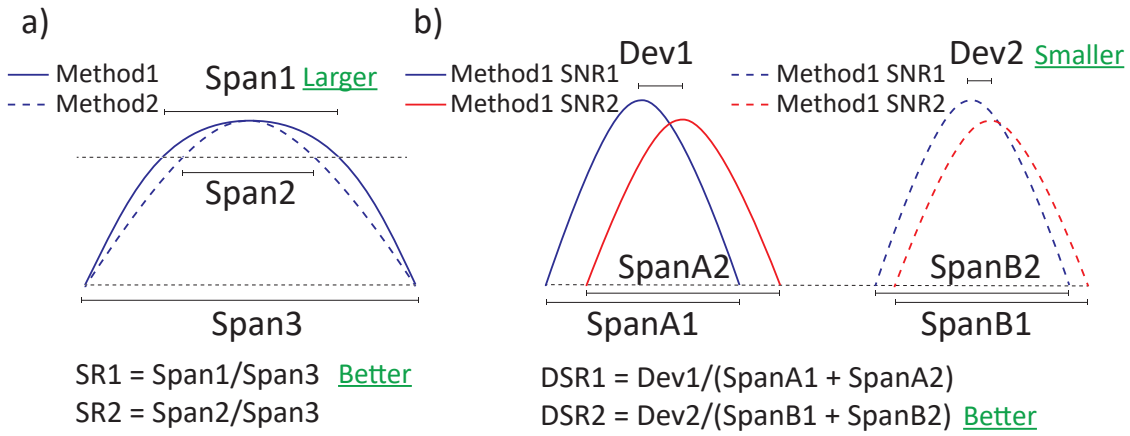

Figure 4.6: **A)** Span ratio of two different methods at the same SNR levels. In the case of multiple SNR levels, the sum of the spans at different noise level is taken. Larger span always relates to flatter curve and consequently better robustness. **B)** Deviation to span ratio of two different methods at two different SNR levels. At the same span, smaller deviation always relates to closer peak performance points and consequently better adaptiveness.

where α and β are user-defined parameters.

The detection occurs when the signal amplitude exceeds the threshold. As the spikes typically last for 24 timestamps (1 ms), detection will be inactivated for 15 timestamps to avoid re-detection. Detections falling within 10 samples around the ground truth are True Positives (TPs), others will are Flse Positives (FPs), and undetected spikes are False Negatives (FNs).

4.3.3 Evaluation metrics

To evaluate the performance of different threshold settings, we assess the detection accuracy (Acc), Span Ratio (SR) and Deviation-to-Span Ratio (DSR). Acc is formulated as:

$$\text{Acc} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{4.9}$$

which describes the detection performance jointly considering sensitive and false detection rates.

Based on the observation in Section.4.2, we found the algorithm adaptiveness can be evaluated in two aspects. One is how the algorithm performance is affected when the threshold parameter is inaccurately set. It will be referred to as "robustness". The other one is how the algorithm performance is affected when the parameter is set in one noise level and applied to the other levels. This will be referred to as "adaptiveness". Two metrics, SR and DSR, are established to quantitatively evaluate these two aspects and they have been shown visually in Fig.4.6.

SR describes the robustness as in Eq.4.10. The *Span* is defined as the average multiplier difference in different noise levels that achieves the accepted accuracy. We prefer settings leading to large *Spans*, which means the multipliers values vs. Acc curve is flat, and it is less sensitive to the inaccurate setting of the threshold multipliers. However, the range of the multipliers in different settings can be different. To ensure a fair comparison among different settings, we take the ratio of *Span* at 0.8 Acc to the *Span* at 0.7 Acc as the metric *SR* to evaluate the robustness of non-optimal setting of the threshold within the same SNR level. More significant *SR* means better robustness. It can also be regarded as a measurement of the detection performance when the threshold is sub-optimal.

$$\begin{aligned} \mathrm{Span}_A &= \frac{\sum_{\mathrm{SNR}} \left(\max(\mathrm{Mul}_{\mathrm{Acc>A}}) - \min(\mathrm{Mul}_{\mathrm{Acc>A}})\right)}{\sum_{\mathrm{SNR}} \left(\max(\mathrm{Acc}) \ > \mathrm{A}\right)} \\ \mathrm{SR} &= \frac{\mathrm{Span}_{0.8}}{\mathrm{Span}_{0.7}} \end{aligned} \tag{4.10}$$

where *Mul* is multipliers, $max(\cdot)$ and $min(\cdot)$ are operators for max and min values, and A is the accepted accuracy level.

The *DSR* describes the deviation of the parameters across different SNRs for obtaining the best detection accuracy, reflecting the adaptiveness. *Dev* is the maximum difference for the multipliers that achieve the best detection accuracy higher than the accepted accuracy in different SNR levels. A small *Dev* means the best multiplier is less deviating from one SNR level to another. The adaptiveness of such a setting is therefore better at different SNR levels. Taking the ratio between the *Dev* and *Span* makes it possible to compare the settings with different parameter spaces. The *DSR* is defined as in Eq. 4.11 and we here assess the deviation at the accepted accuracy level of 0.8, which gets involved in enough noise levels and fits the intuition of the minimum acceptable detection accuracy.

$$\begin{aligned} \mathrm{Dev}_A &= \mathrm{Mul}[\mathrm{maxL}(\mathrm{Acc}_A)] - \mathrm{Mul}[\mathrm{minL}(\mathrm{Acc}_A)] \\ \mathrm{DSR} &= \frac{\mathrm{Dev}_{0.8}}{\mathrm{Span}_{0.8}} \end{aligned} \tag{4.11}$$

where $maxL(\cdot)$ and $minL(\cdot)$ are operations to find the lag of max and min values, and Acc_A is the max detection accuracy in different SNR levels that are larger than A.

4.3.4 Results

We have simulated 18 sets of recordings in which 2 cells each fires at 20 Hz. Their SNRs vary from 5 to 40 with a step of 2. Each set contains ten 4 s recordings. The threshold in each run is set with varying α and β. Detection accuracy is averaged among 10 runs.

$T = \alpha\mu_{noise}$

We have swept α from 1 to 50, detection accuracy for different α and SNR is given in

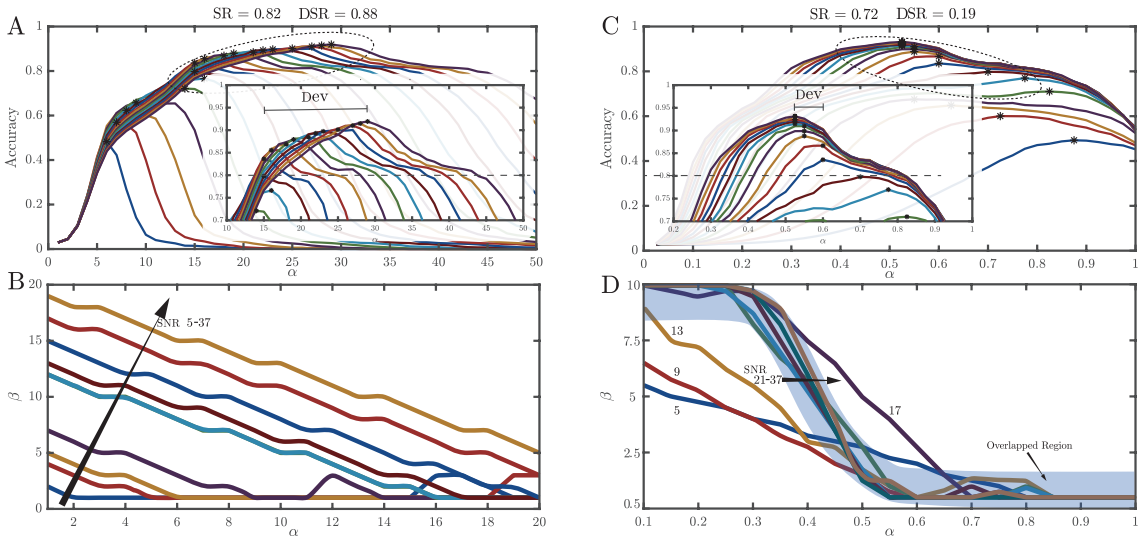

Figure 4.7: **A)** The detection accuracy using $T = \alpha\mu_{noise}$ for threshold with changing α and SNR. The curves from the bottom blue to the top purple are the SNR from 5 to 40 with a step of 2, and the stars indicate the top settings that achieve the highest accuracy in different SNR. (the same for C). **B)** The combination of α and β that achieves the best detection accuracy using $T = \alpha\mu_{noise} + \beta\sigma_{noise}$ for threshold in different SNR. The curves from bottom blue to top yellow are the SNR from 5 to 37 with a step of 4 (the same for D). **C)** The detection accuracy using $T = \alpha\mu_{peak}$ for threshold with changing α and SNR. **D)** The combination of α and β that achieves the best detection accuracy using $T = \alpha\mu_{peak} + \beta\mu_{noise}$ for threshold in different SNR. The overlapped region denotes the region of SNR levels and $\alpha - \beta$ combinations that suffers less from the parameter deviation for setting the optimal threshold.

Fig. 4.7.A. Curves from top to bottom are cases SNR from high to low. The stars indicate the top settings that achieve the highest accuracy in different SNR. One can notice that there is a significant deviation for the optimal α from 15 to around 30 as the SNR increases. This indicates that the optimal threshold for spike detection does not increase linearly with the noise levels, i.e. we cannot use a constant multiplier for different noise levels expecting it always achieve the best detection accuracy.

$T = \alpha\mu_{noise} + \beta\sigma_{noise}$

As the optimal threshold increases non-linearly with the noise mean, we tried to introduce the STD for thresholding. Both α and β are swept from 1 to 20. Results are shown in Fig. 4.7(B). The curves shown are the best combinations of the settings that achieve the highest accuracy in different SNR levels. The parameter deviation still exists in both the mean and STD sides as there is no overlapping among the settings in different SNRs. We therefore question if the noise statistic is a good or only indicator for setting the threshold.

$T = \alpha\mu_{spike}$

Intuitively, one can set a suitable threshold in the med-way of spike peaks and noise ground, which means the spike peak level can be an indicator for setting the threshold. The α is set to vary from 0.1 to 2 with a step of 0.05. Results are shown in Fig. 4.7.C. DSR,

which describes the parameter deviation, is reduced dramatically compared to Fig. 4.7.A, indicating that the spike-based threshold suffers much less than the noise-based threshold in different SNR levels, which has better adaptiveness. However, the SR is reduced, which means the threshold is less robust to the inaccurate set of the threshold. One reason is that the spike peak mean is high, and minor changes in the multiplier can significantly change the threshold level; another reason is that this threshold is noise-invariant. When the noise is high, some large noises can exceed the threshold to increase the false detection.

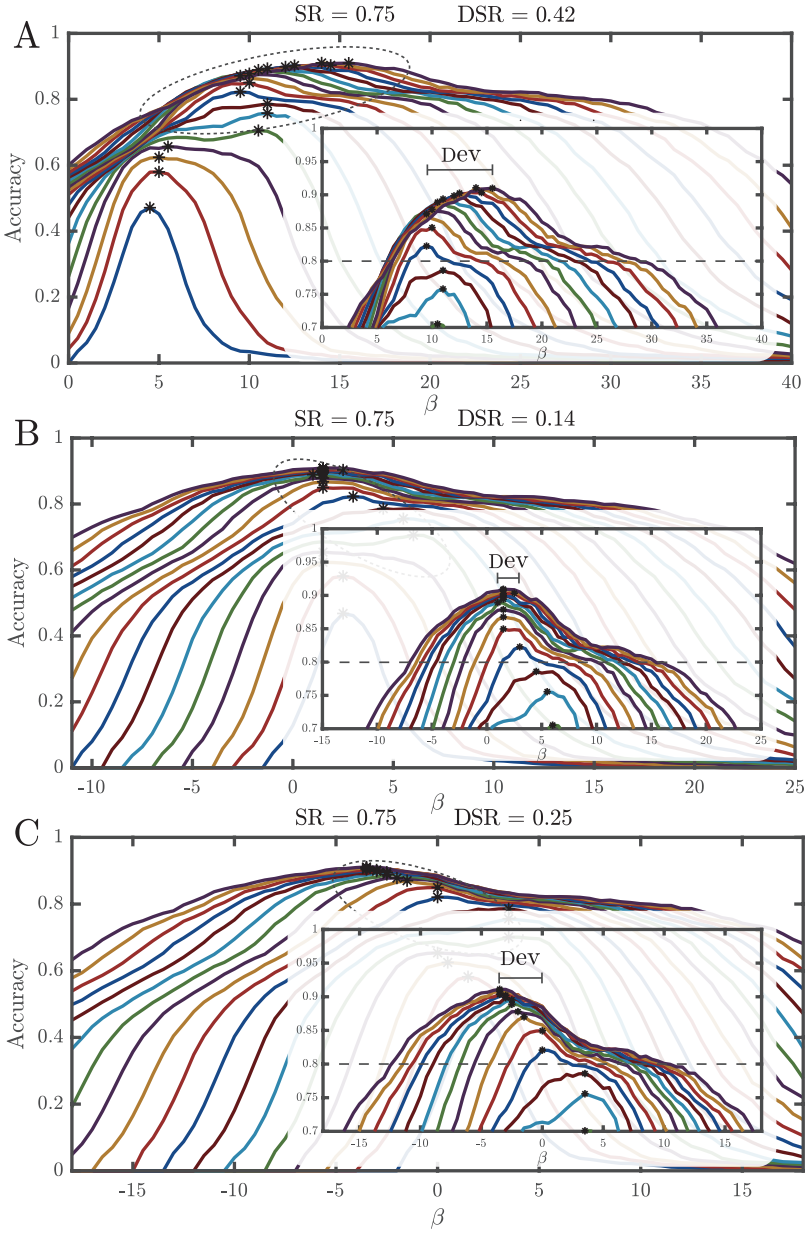

Figure 4.8: The detection accuracy using $T = \alpha\mu_{peak} + \beta\mu_{noise}$ for threshold with changing fix α, and varying β and SNR. The curves from the bottom blue to the top purple are different trials when SNR is increased from 5 to 40 with a step of 2. **A)** $\alpha = 0.25$. **B)** $\alpha = 0.5$,**C)** $\alpha = 0.6$.

$T = \alpha\mu_{spike} + \beta\mu_{noise}$

We can introduce noise awareness to the threshold by combining spike and noise statistics. With α varying between 0.1 to 2 and β varying between 0.5 to 10, the top combination achieving the highest accuracy is shown in Fig.4.7.D. The deviation is much less than using Eq.4.6 compared to Fig. 4.7.B. The accepted parameters are overlapped at the shaded blue region, which means the SNR in this region shares close optimal threshold settings. We selected three α values 0.25, 0.5, 0.6 and swept the β. The results are shown in Fig. 4.8. It can be observed that when α is 0.5, the DSR is minimal, and SR is also increased compared to it in Fig. 4.7.C. The parameter deviation can be increased when α deviates to 0.5 as DSR increases. However, the SR will still be maintained, which means the performance of different settings is consistent when the threshold is sub-optimal.

Such a joint spike-noise-based threshold utilises the peak values to set the coarse-grained baseline level of the threshold and uses the noise value to fine-tune the threshold. The coarse-grained baseline reduces the parameter deviation, and the fine-tuning provides noise awareness and increases the low SNR performance. Such a finding reveals the essence of optimal thresholding and tells us we have forgotten an important factor - the spike peak in the past. Moreover, this also fits our intuition as the threshold should be SNR-driven.

4.3.5 Application of the spike-noise based threshold to an independent dataset

According to Fig. 4.8.B, the suitable settings for α and β can be 0.5 and 1.5, respectively. In order to evaluate the generalisation of such a finding, we have applied these settings to a different dataset generated in [45], which is the dataset we used previously. The results are shown in Table 4.2. Compared to nearby settings, the maximum threshold still peaks at the selected settings, which means there is no deviation between this dataset and the dataset we generated. Compared to other works, such a method achieves the highest detection accuracy and better robustness to different noise levels, referring to the mean and STD values of the Acc in different noise levels.

Table 4.2: A comparison of different spike detection methods in varying SNR

Noise level	0.05	0.1	0.15	0.2	Mean	STD
$\beta = 1$	0.98	0.99	0.96	0.88	0.953	0.05
$\beta = 1.5$	0.99	0.99	0.97	0.92	0.968	0.031
$\beta = 2$	0.99	0.99	0.96	0.92	0.967	0.031
Previous work	0.89	0.97	0.92	0.89	0.918	0.037
[45]	0.92	0.95	0.95	0.85	0.918	0.047

4.4 Discussion

4.4.1 Noise-driven threshold

The optimal threshold set only with noise statistics suffers severely from the changing noise levels. The parameter deviation lowers the model's adaptiveness. The reason is that noise statistics is not the only determining factor on the optimal threshold. The best setting for one SNR can overfit such noise level and no longer works with the noise level changes. No matter whether the higher order statistics are used or not.

4.4.2 Spike-driven threshold

Using spike peak in guidance of setting the threshold can overcome parameter deviation and improve the adaptiveness. A suitable parameter that generalises well in different noise levels can be found. However, as the spike values are more significant than the noise, the threshold becomes more sensitive to the change of multiplier values and less robust.

4.4.3 Spike-noise-driven threshold

Jointly using the spike and noise statistics can trade-off between parameter deviation (adaptiveness) and sub-optimal threshold degradation (robustness). Using spike values can effectively reduce parameter deviation, and using noise values reduces the effect of sub-optimal thresholds on degrading detection performance.

4.4.4 The concern on spike peak estimation

Such an approach requires robust estimation of both noise and spikes. There are plenty of researches focusing on noise estimation but few studies are working on the spike peak estimation. Estimating the spike peak can be challenging as we have no prior knowledge of the peak amplitudes before detection. The threshold will be updated according to the local environment in real-time adaptive spike detection. Without a robust estimation of the spike peak amplitude, the false detection of the spikes could lower the estimated spikes peak amplitude and even lower the threshold leading to more false detections. Such positive feedback could eventually crush the spike detection algorithm. In addition, extra algorithm complexity will be added for spike estimation.

4.5 Conclusion

This chapter provides a comparison of a number of empirical spike detection emphasisers and statistical thresholding techniques in terms of algorithm performance and hardware cost. The findings in this chapter include:

- Noise-mean-based thresholding has the lowest hardware complexity, but lacks adaptiveness to different noise levels, is sensitive to inaccurate threshold settings, and achieves relatively lower spike detection accuracy in practice.
- Noise-median thresholding can achieve good spike detection accuracy in practice, but it is sensitive to parameter setting and requires too many resources for an implanted application.
- A spike-noise-driven threshold can theoretically achieve good spike detection accuracy and high adaptiveness. Accurately estimating the spike peak within an on-implant budget could be challenging.

Our results suggest that none of the tested spike detection algorithms can achieve good spike detection performance, noise adaptiveness, robustness and lower hardware complexity simultaneously. These concerns highlight the need for a new spike detection thresholding mechanism other than using statistics.

Chapter 5

New approach: Firing-rate-based spike detection

As the journey towards hardware-efficient adaptive spike detection algorithms continues, this chapter represents a significant milestone. The introduction of the firing-rate-based spike detection algorithm marks a departure from conventional statistical-based approaches and leads to significant improvements in terms of spike detection accuracy, adaptiveness, and the reduction of hardware complexity. The proposed algorithm is thoroughly evaluated and compared through MATLAB simulations, FPGA and ASIC designs, demonstrating its advantages for on-implant use.

5.1 Introduction

Although the statistical-based algorithms previously described in Section.4.2 are widely used, there do exist several drawbacks. Firstly, there is no heuristic way to find optimised parameters to local statistics such that to set the reliable threshold. The standard approach is searching through synthetic data by trial and error . It is however not guaranteed that chosen parameters will be applicable to real recordings, in particular with different noise levels. As shown in Section.4.3, it is impossible to establish a reliable threshold that can adapt to different noise levels using statistics-based thresholding methods. Another problem is that calculating local statistics can require significant memory to maintain sufficient history –

A portion of the content in this chapter has been published in [120,121,126] with my contribution to the software/hardware algorithm design and result analysis. Dr Peilong Feng conducted the hardware migration from the FPGA implementation to the 180 µm and 65 nm ASIC design.

memory requirements increase linearly with the number of channels and sampling frequency. Finally, [42] suggests that the neural decoding performance is quite robust to the spike detection algorithm missing small spikes (i.e. there is only a small reduction in accuracy). Fewer spikes detected mean less data to be communicated. Statistics-based spike detection is also sensitive to the scaling factor that is used to map local statistics to the threshold level – it is not clear how to balance the scaling factor to deliberately detect fewer spikes and trade-off accuracy with data bandwidth.

To address these challenges in conventional methods, we propose a firing-rate-based spike detection algorithm. Instead of needing an arbitrary scaling factor to define the threshold from local statistics, the proposed algorithm requires a detection rate interval. We hypothesise that the threshold is reliable if it detects spikes at a rate close to the actual local firing rate. The target detection rate can therefore be set according to neuroscience observations heuristically. For example, it has been reported in [127] that the firing rate of human motor neurons varies from a nominal rate of about 10 Hz when the subject is performing minimum effort tasks, to over 20 Hz for maximum effort tasks. Additionally, it has been reported that on average 3 to 4 neurons can be observed per Utah array electrode in motor cortex [43] (i.e. through spike sorting efforts). Thus, the average firing rate of the ticker reaching task (low effort) is expected to be around 40 Hz. A target detection rate interval can then be set and the threshold is dynamically updated to maintain the detection rate within this interval. In the meantime, the firing-rate-based algorithm can deliberately neglect spikes (i.e. set a slightly lower detection rate) for better power efficiency. The firing rate in formation is used in [112]. This algorithm uses the minimum and maximum expected firing rate to set a threshold boundary and search the best separation of the spikes and noise according to the peak distribution. This algorithm is only applicable offline as the threshold is setup from the distribution of massive recording samples.

Two versions of the proposed online firing-rate-based spike detection algorithm will be introduced in this chapter. The simple version is more hardware-efficient while the complex version reduces the impact of the inaccurate initial settings of the target detection rate but requires more computation.

5.2 Firing-rate based spike detection (Basic version)

This algorithm is designed to have extremely low computation and resource requirements, and we aim to use only fixed-point operations, no multiplications/divisions, and minimal memory. Such requirements prevent us from using complex mathematical models and statistics to derive the optimal threshold but to see the essence of spike detection with the most simple operations. The proposed algorithm includes an Absolute Difference Filter (ADF), removing the LFP while enhancing the spikes and a firing-rate-based thresholding mechanism

updating the threshold automatically.

5.2.1 Absolute difference filter

The broadband neural recordings consist of LFPs, EAPs, and noise. Different broadband components can be separated using highpass or bandpass filters. However, it is challenging to design a reliable fixed-point filter with limited orders.

Filtering the LFP, enhancing spikes and suppressing the noise are especially critical for statistical-based thresholding as the threshold is typically derived from the noise statistics. The mean subtraction filter, Butterworth filters and different operators including NEO, ASO and ED has been used previously. However, the pre-processing becomes less critical when the firing rate information is introduced as the threshold is independent of the noise. That potentially allows us to use an even simpler operator than those introduced in the previous chapters.

According to the result in Section.4.2.4, where ED outperforms other operators revealing the gradient to be a more discriminated feature than the amplitude. We decided to only use the double-side gradient (without squaring), which is described in Eq. 5.1. It is possible to remove the LFP and enhance the spikes simultaneously, providing good-enough signal SNR for firing-rate-based spike detection.

$$Y_n = abs\{X_n - X_{n-k}\} \tag{5.1}$$

where X_n stands for n^{th} input to the pre-processing and Y_n stands for n^{th} output, and k is the interleaving of two samples. This value is sample-frequency dependent and $k = 2$ was found empirically to be the best choice at 7 kHz.

Such an operation has been explored by many studies, but can fail when the gradient of the LFP overwhelms that of the spikes. However, in practice, the front-end recording system will highpass filter the signal with analogue filters. Only limited LFP power can leak to the passband, which prevents the given filter from failing.

Using the absolute difference filter, the signal is LFP-removed and SNR-enhanced in one single step with only one subtraction and one absolute operation (reverse-and-plus-one for negative values). Compared to the operators such as NEO, ASO and ED, no multiplication is required. It is expected to achieve significantly reduced resource usage.

5.2.2 Firing-rate-based thresholding

The proposed adaptive threshold is set based on the fact that the firing rate of a certain brain region is on average stable as mentioned in the introduction of this chapter.

Though the firing rate can fluctuate during the active or silent period, it is relatively stable in the long term. By setting a reasonable target detection rate and threshold update strategy, we can control the threshold to saturate at the level, automatically detecting spikes at the desired rate.

The threshold update strategy is described in Eq.5.2.

$$Thr_{i+1} = \begin{cases} Thr_i & R_2 < DR < R_1 \\ Thr_i * (1+p) & DR > R_1 \\ Thr_i * (1-p) & DR < R_2 \end{cases} \tag{5.2}$$

Where Thr is the threshold, $[R_2, R_1]$ is the target detection rate interval and DR is the current detection rate.

In more detail, the target detection rate $[R_2, R_1]$, initial threshold T_0, and updating period P are set at the start. In one updating period, if $Y_n > T_n$, a spike is detected, and 5 samples (in the 7kHz sample frequency case) are skipped to avoid multiple detections of one spike. The detection counts DR are increased by one. If $DR > R_1$, the threshold will increase by p, and a new updating period is started. At the end of one update period, if $DR < R_2$, the threshold will decrease by p. Otherwise, the threshold will be unchanged. In short, we set an acceptable detection number range in one update period, and the threshold will be decreased or increased if the detection rate is too low or high. Even though the initial threshold can be non-ideal, the threshold will eventually saturate at a level that fulfils the target detection rate.

p determines the saturated speed and threshold resolution. A larger p means faster saturation but a less accurate threshold. Adaptively changing p can be implemented to mitigate such a trade-off but not used considering the computation overhead. In both datasets, p is set to 6.25%, i.e. 1/16, which can be easily calculated with 4-bit right shifting as all numbers are in fixed-point representation.

P determines the adaptiveness of the system to the varying signal condition. The threshold can be less sensitive to signal varying if P is large, while it can be disturbed by the burst firing if P are too short. The frequent threshold updating also leads to more computation and higher power consumption. P is empirically set to 1 s.

The target detection rate interval $[R_2, R_1]$ is the most important parameter determining the acceptable detection rate range and eventually determines the saturated threshold level. There are several benefits to setting the target detection rate rather than conventionally deriving from multiple times to noise mean/median/root-mean-square/standard deviation values.

1) Compared to finding a multiplier, selecting one detection count interval can be more heuristic, building on previous neuroscience observations like [127] other than trial and error (exhaustive search).

2) It guarantees the detection of useful information while the detection result of the statistical approach is not predictable. It could happen using multiple times of signal statistics that the threshold can grow too high or low in some channels. That can lead to continuous detection or zero detection resulting in system failure.

3) It improves the detection in long-term implantation. The scar tissue around the probe can push neurons further away and weaken neural activity. The conventional approaches based on statistics can miss these spikes while the proposed algorithm can lower the threshold adaptively to detect these spikes and fulfil the target detection rate.

4) The target detection rate range can potentially be used to improve the decoding performance. In more detail, if the resultant detection rate of a certain period approaches or even exceeds R_1, such period is more likely to be active, while if the resultant detection rate of another period is under R_2, such period is more likely to be silent (the subject is inactive). The traditional method cannot provide such extra information (at least not in real-time), and we have proposed a Rate-Modulated LSTM based on such information, introduced in the next chapter. Notice that the target detection rate mentioned later will be the interval upper limit R_1 unless specified and $R_2 = R_1/2$.

In most experiments below unless specified, R_1 is set to 60, assuming to detect 3 neurons' activities firing at 20 Hz each and the effect of varying R_1 is also evaluated.

5.3 Firing-rate-based spike detection with target detection rate auto updating (Complex version)

The previous algorithm requires a manually selected target detection rate. In this section, we introduce a mechanism that can automatically update the target detection count when the initial setting is inaccurate. Inspired by the noise-spike-based threshold in the last chapter, we tried to track the spike peaks. The estimated peak level allows the algorithm to perceive whether the threshold level and target detection rate are appropriate. For example, if the threshold is updated to approach or even exceed the peak level to fulfil the target detection rate, the target detection rate must be too small compared to the real spike firing rate.

5.3.1 New firing-rate-based thresholding

The threshold needs time to saturate at an appropriate level. The previous algorithm set a small p for a finer threshold level which requires several seconds for the threshold to be saturated. However, a slow saturation speed can be problematic here. As the spike peak mean is tracked, false detection before saturation can lead to an inaccurate estimation of the spike peak mean. Therefore, a new thresholding strategy is designed to allow faster saturation speed by using a larger p. However, the spike detection outcomes from the

saturated threshold level then become not as accurate as using small p. This firing-rate-based thresholding begins with setting the initial threshold (Thr_0) and target detection rate interval $[R_2, R_1]$. These parameters' initial values can be set automatically based on several seconds of spike detection using traditional methods. They can also be set manually or even randomly. It was tested that the algorithm is largely insensitive to the initial settings, which is detailed later.

The new threshold updating strategy is more complex than the previous design as (5.3).

$$Thr_{i+1} = \begin{cases} Thr_i & A1 : R_2 < DR < R_1 \\ Thr_i * 1.5 & A2 : DR > R_1 \\ Thr_i * 0.5 & A3 : DR < R_2 \ \& \ C2 \\ Thr_i * 0.75 & A4 : DR < R_2 \ \& \ C1 \end{cases} \tag{5.3}$$

where C1 means the threshold was increased in the last update, C2 stands for the complementary set of C1. We also add an A4 brunch to help the threshold to saturate as shown in Fig. 5.1.

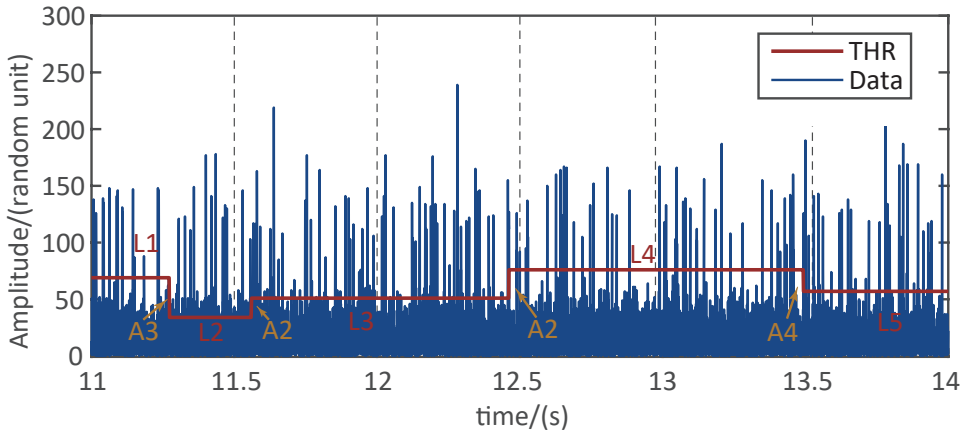

Figure 5.1: A demo of the threshold update process. This is a segment of one recording between 11 s and 14 s, and was chosen as it serves well to demonstrate the updating procedure. At L1, the current detection rate is below the desired range. As such, the threshold should follow the A3 branch (where it is reduced by half) to L2. However, the detection rate is larger than the max spike rate, and the threshold is then increased by $\times 1.5$ according to A2 to L3 (note that the A2 increment is only 1.5 times and not 2 times. This prevents the threshold from just oscillating between L1 and L2). The detection rate is still too high at L3, and so the threshold is increased again to L4, but then the detection rate is too low. The threshold should then be decreased. However, if it follows A3 (a half), the new threshold will be lower than L3. As such the reasonable level should be between L3 and L4, and so we designed the A4 path to let the threshold level off at L5. During these updates, we have seen two saturation trends: L1-L2-L3 and L3-L4-L5. With this branching system, the threshold can level off at a suitable place that is within the acceptable range of detection rates.

Spike peak mean tracking and target spike count auto-updating are used to address the following three issues: (1) Because the threshold update gradient is quite large, under

certain conditions the threshold may move down to a level close to or below the noise floor and this results in a high false detection rate; (2) if the target detection rate is too high, the threshold could move down to a level around the noise floor, therefore resulting in a high false detection rate – similarly to (1); and (3) if the target detection is set too low, the threshold could move upwards to a level near the actual spike peak and therefore result in low sensitivity. These three conditions are illustrated in Fig. 5.2. The black dashed lines show the thresholds derived without resolving these three issues.

5.3.2 Spike peak mean tracking

In the last chapter, we investigated the optimal threshold level where it should be around $0.5 \times$ Spike Peak Mean + $0.25 \times$ Noise Mean. Once we know the spike peak mean, a reasonable threshold level is known. To ensure that the algorithm can detect whether the threshold/target spike count is appropriately set, we track the spike peak mean. The spike peak estimation is based on a weighted average filter, where 75% is weighted by the mean amplitude of past spikes and 25% by that of the current spike.

The spike peak mean value is updated whenever the threshold is updated. Once the spike peak level is known, a minimum accepted peak amplitude can be set (in this case $0.5 \times$ Spike Peak Mean). This turns out to be very useful: if the peak of a detected spike is significantly lower than the current estimated spike peak level, the detected spike is most likely to be noise and is therefore ignored. As a result, the false detection rate can be reduced significantly. It is clearly observable that there are large amounts of falsely detected spikes with the old setting in Fig. 5.2 (A & B) Con. 1. However, as the spike peak mean value is tracked, we can ignore spikes with amplitudes below half the spike peak mean value. As such, a spike needs to exceed both the set threshold and half the amplitude of the spike peak mean to be considered a spike, reducing the false detection rate as shown in Fig. 5.2 (A & B) Con. 1.

5.3.3 Target spike count auto-updating

Conditions (2) and (3) result from an incorrectly set initial target detection rate. To avoid the algorithm being negatively impacted by these conditions, the threshold is not updated when it tends to exceed the spike peak or drop under half spike peak (refer to conditions (2) & (3) in Algorithm 1). Instead, the target detection rate is decreased/increased to compensate for the incorrectly set initial R_1. The resulting red thresholds (new THR) in Fig. 5.2 (B) and (C) during Con. 2 and Con. 3 are better than the black thresholds (old THR), as the spike peaks and noise floor are more accurately separated. Pseudocode of the new threshold updating mechanism is provided in Algorithm 1, illustrating how the threshold and target detection rate are updated. With this spike peak tracking and target spike count auto-updating technique, we have integrated the target detection rate and threshold update into

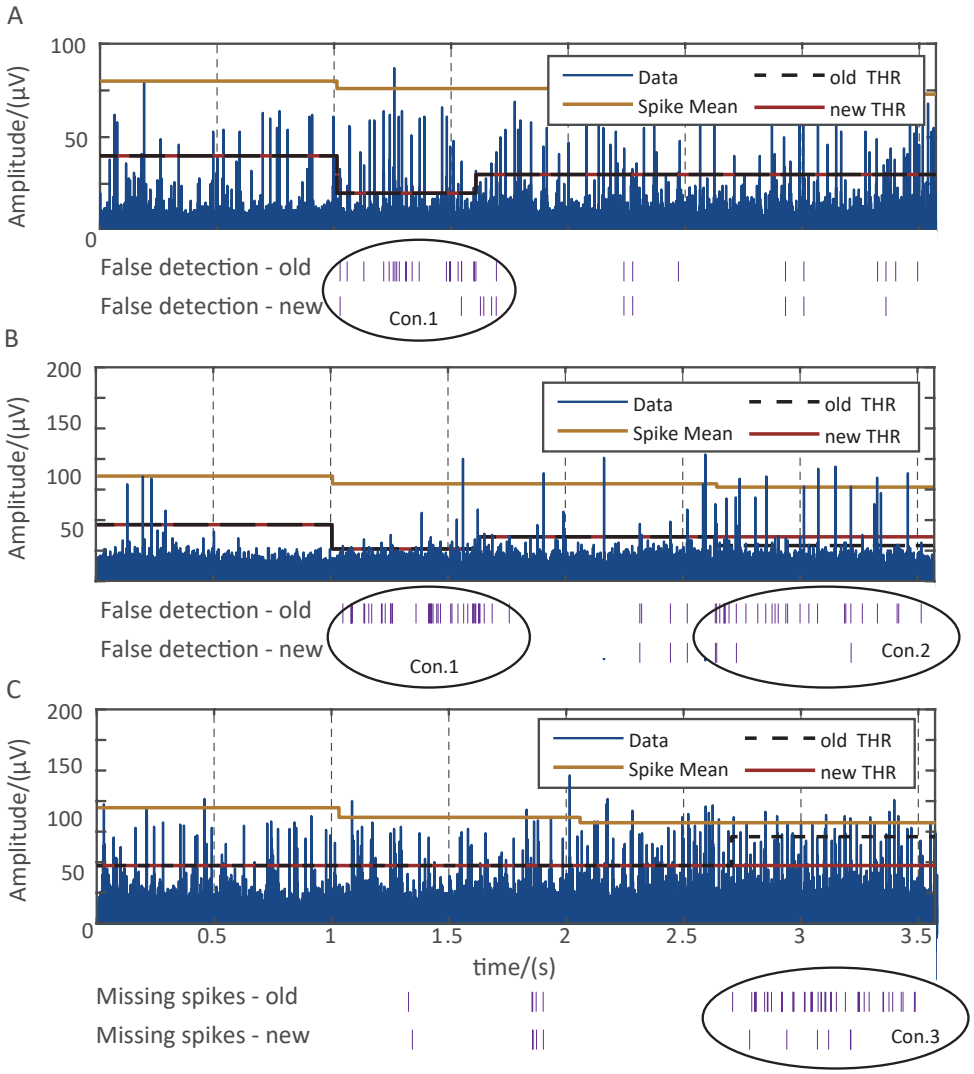

Figure 5.2: The three conditions under which the performance of the algorithm can degrade, showing the detection results with and without the addition of rules that mitigate the degradation. A) Before the threshold settles to a desirable level, the threshold can go too low and detect many false spikes. Solution: with spike mean tracking, we ignore spikes that are smaller than half of the measured spike mean, reducing the false detection rate. B) If the MSR is set too high, the threshold can go too low, thus incorrectly classifying noise as spikes so as to satisfy the max spike rate. Solution: the threshold is kept as it is even though the detection rate is below the expected range, while the MSR is decreased. C) If the MSR is too high, the threshold can go too high, and so some spikes can be missed. Solution: the threshold is left unchanged even though the detection rate is higher than the MSR, and the MSR is increased instead.

a closed loop. This eventually converges to a reasonable rate and threshold regardless of the initial setting.

Algorithm 1 Automatic Threshold Updating

Input: *Spike Count*, *Max Spike Count* and *Spike Mean*

Output: *Thr*

```
1: if Spike Count > Max Spike Count then
2:     if Thr + Thr/2 < Spike Mean then
3:         Max Spike Count + +                          ▷ (3)
4:     else
5:         Thr ← Thr + Thr/2                            ▷ A2
6:     end if
7: else if Spike Count < Max Spike Count/2 then
8:     if Thr < Spike Mean/2 then
9:         Max Spike Count - -                          ▷ (2)
10:    else
11:        if Thr is just increased then                ▷ C1
12:            Thr ← Thr - Thr/4                        ▷ A4
13:        else                                         ▷ C2
14:            Thr ← Thr/2                              ▷ A3
15:        end if
16:    end if
17: else
18:    Thr ← Thr                                        ▷ A1
19: end if
```

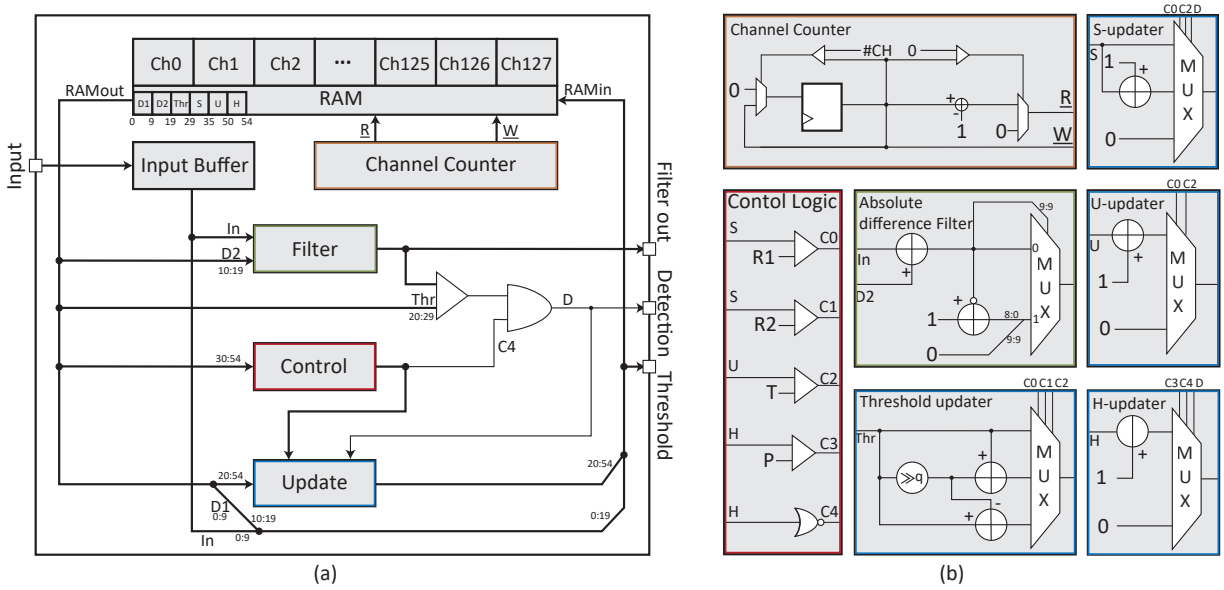

Figure 5.3: A diagram for the proposed spike detector. (a) The system overview. The system comprises an input buffer that temporarily stores the input samples, a RAM that maintains the status of various channels, a channel counter that rotates among channels, a filter that improves the signal-to-noise ratio (SNR), a control unit that updates the RAM status, and a control unit that generates the control signals for threshold updating. The filter output, threshold, and detection binary stream are the final outputs from the system (b) Detailed digital logic for different modules.

5.4 Hardware implementation

Both firing-rate-based spike detection algorithm has been implemented on FPGA. However, only the simple firing-rate-based spike detection algorithm is given below in the main content because it is preferred for on-implant use (simpler and effective), while the complex version of firing-rate-based spike detection hardware implementation is given in the Appendix.B for consciousness. The diagram of the implementation is given in Fig.5.3. It consists of a RAM, storing the temporal values of all channels; a channel counter, scheduling the read and write of the RAM for different channels to share; filters, the combinational circuit for absolute difference filter; control logic, the combinational circuit issuing the control signal; and update logic, the combinational circuit for duty cycling, updating the threshold and counting the number of detections.

5.4.1 RAM

The system status random access memory stores the current status of each channel, and the channel counter schedules channels sharing the same combinational logic. The proposed algorithm only needs one clock cycle to process each sample, so there is no need for any register except the memory output buffer.

Table 5.1: Bit width and descriptions for the values stored in RAM

Register Name	Width	Description
D1	10	ADF buffer for the first previous sample
D2	10	ADF buffer for the second previous sample
Thr	10	Current threshold
S	7	Number of detections within the current duty cycle
U	13	Count for the update duty cycle
H	3	Count to hold after detection avoiding redetection

The RAM is a single-clock, dual-port block RAM. On each rising clock edge, the memory stores the updated status of the current channel read from the combinational logic and outputs the current status of the next channel into the output buffer. The channel counter updates the current and next channel addresses for the memory cyclically. Table. 5.1 lists the various statuses stored in memory. This requires a total of 53 bits per channel.

5.4.2 Absolute difference filter

Both the LFP signal removal and spike signal emphasizing are achieved using the ADF as Eq.5.1 and no additional digital filters are used to remove the LFP.

The ADF operates by reading the new input sample from a rolling buffer together with previous samples from the RAM. The filter output is obtained by subtracting a previous sample (two samples back) from the new sample. If the difference is negative, the absolute value is calculated according to a two's complement representation.

The filter output is compared with the threshold value stored in RAM. When the filter output exceeds the threshold and no spike is detected in five samples before, a valid spike detection signal is generated.

5.4.3 Control logic

The control unit reads three values (S, U, and H) and generates five control signals. C0 - C4, according to the logic circuits plotted in the control logic block. C0 and C1 control two conditions in Eq.5.2 for the threshold to update. C0 is set when the current number of detections exceeds R1, and C1 is set when the current number of detections is below R2. C2 is set at the end of one update duty cycle. C3 and C4 control the signal entering and exiting the period when a spike is detected. C3 is set when it reaches the preset length of a spike after a spike is detected, and C4 is set when H is zero, i.e. not in a period of the presence of the spike.

5.4.4 Update logic ('updaters')

Four updaters update the values of Thr, S, U and H. When the number of detections S within the current duty cycle exceeds the detection rate upper limit R1 (C0 is high), the threshold is increased by $2^{-q} * \text{Thr}$ (q is four). When S is below the lower limit of R2 (C1 is high) at the end of each duty cycle (C1 is high), the threshold is reduced by $2^{-q} * \text{Thr}$. Otherwise, the threshold stays unchanged for the next duty cycle.

When a validation spike is detected (D is high), the S is increased by one as long as it does not reach R1 (C0 is high), while it does reach the R1 or one duty cycle is over (C2 is high), S is reset. There is a special case when a valid spike is detected at the end of one duty cycle. In this case, S is set to 1. In all other cases, S stays unchanged.

U is increased by one every clock cycle and reset when the threshold needs to be updated, i.e. C0 is high, or C2 is high. H is increased by one when a D is high or when C3 is low, and C4 is high. Otherwise, it stays at zero.

The new values are concatenated as follows: [Input, D1, new Thr, new S, new U, new H] replacing the old values [D1, D2, S, U, H] of the current channel stored in RAM. The values for R1 and R2 have been set to 30/60 according to the expected firing rate of the recordings as stated in the introduction in the chapter. As a spike typically lasts for 7 samples (1 ms), skipping 5 samples after detection can effectively avoid re-detection while maintaining sensitivity. P is thus set to 5. The threshold updating duty cycle is empirically set to 1 s. T is thus 7000.

5.4.5 Timing

The hardware implementation is designed such that a single spike detection system can serve multiple channels through multiplexing input and maintaining channel-specific memory. Therefore, a 128-channel spike detection system operating at a 7 kHz sampling rate would require a system clock speed of 896 kHz (i.e. 1 clock cycle is needed for each independent sample). As an example, Fig.5.4 shows a timing diagram illustrating how the threshold value in RAM can be updated for channels 0 and 1. D1, D2, S, U, and H are updated following a similar procedure.

At the CLK posedge, the threshold value 50 is read from RAM location $\underline{\text{R}}$ (CH0) as RAMout.Thr. Here, we suppose the threshold is to be decreased by 1/16. The threshold updater updates RAMout.Thr as 47 into RAMin.Thr. At the next CLK negedge, the new R/W channel addresses will be updated. At the coming CLK posedge, the RAMin.Thr is written into the $\underline{\text{W}}$ (CH0) position of RAM as RAM.CH0.Thr. In the meantime, the threshold value of CH1 is read into the RAM output buffer and the threshold value will be updated accordingly.

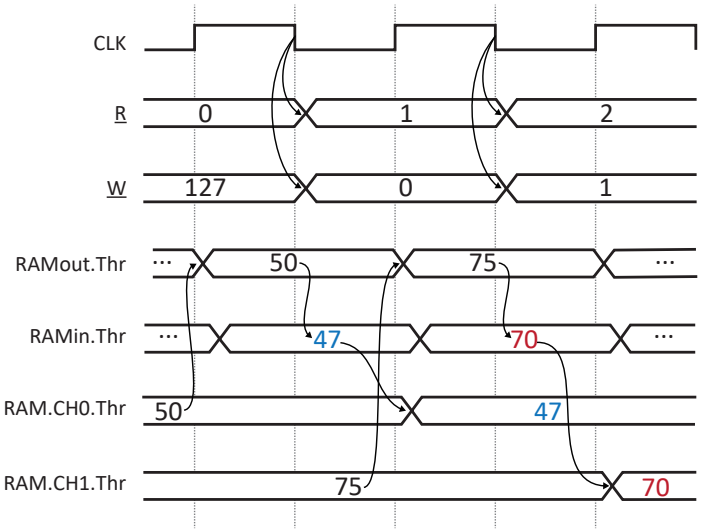

Figure 5.4: A timing diagram is presented to depict the procedure of threshold updating. The memory read/write addresses are updated at the negative edge of the clock. At the positive edge of the clock, the previous threshold value (50) is retrieved from the RAM based on the read address. The new threshold (47) is calculated through a combination logic. At the next clock positive edge, the updated threshold is saved back to the RAM based on the write address, the same as the previous read address. As a result, the previous threshold value is overwritten by the new value.

5.4.6 System scalability

As the proposed spike detection system is efficient both in terms of hardware complexity (number of logic gates, memory) and computation (only 1 clock cycle is needed per input sample), this implementation is easily scalable to higher channel counts. The number of channels supported can be increased by linearly increasing the RAM and clock frequency. The Look-Up Table (LUT) usage is less affected because all channels share the same filter, control logic, and updaters in a time-sharing manner. Without considering the on-implant area and power constraints, the system can be scaled up to N channel as long as (5.4) is satisfied.

$$T_{delay}(RAMout, RAMin) + T_{RAM} < \frac{1}{7000 * N} \tag{5.4}$$

where $T_{delay}(RAMout, RAMin)$ is the propagation delay from RAMout to RAMin and T_{RAM} is the delay of RAM read and write. The system can be easily scaled up to more than 1000 channels (7MHz), without violating timing constraints.

5.5 Results

The results suggest that the firing-rate-based spike detection with spike peak mean tracking and target detection rate auto-updating (Noted as FR-C) can provide good robustness to

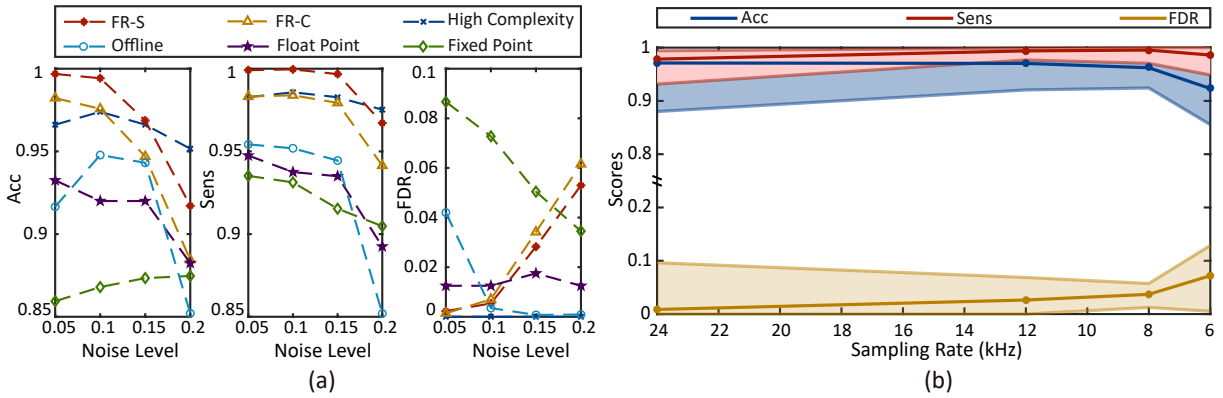

Figure 5.5: a) Average Spike detection accuracy, sensitivity, and false detection rate across synthetic datasets of the proposed algorithm compared to our previous work [11], and exemplar methods including a high complexity algorithm [128], an offline algorithm [45], a low complexity float-point algorithm [48] and the low complexity fixed-point algorithm in 3 at different noise levels. b) The spike detection performance at the different sampling rates. There is nearly no degradation when the sampling frequency is above 8kHz and only minor degradation at 6kHz.

inaccurately defined initial values. However, the simpler version (Noted as FR-B) achieves 1% higher spike detection accuracy when the target detection rate is correctly set, significantly improves long-term detection stability, and reduced hardware resources by 67%, which is preferred for on-implant use. Therefore, this section only includes a comprehensive evaluation of the FR-B, while the most of evaluation of the FR-C is included in Appendix.B.

We compare the FR-C with the baseline statistical-based spike detection algorithms and other state-of-the-art algorithms w.r.t spike detection accuracy and adaptiveness on the synthetic dataset. We also compare the neural decoding performance, and long-term detection stability on the Utah array recording. In addition, the algorithm is also validated on Neuropixels recordings.

All algorithms are validated using MATLAB R2020a (v9.8) for detection performance evaluation. To ensure the MATLAB algorithm is equivalent to a hardware implementation (referring to both FPGA and ASIC), we defined all parameters to use 10-bit integers and ensured all arithmetic functions use fixed point operations leading to identical performance across different platforms.

5.5.1 Spike detection accuracy on synthetic dataset

We first tested the FR-B on the synthetic dataset to quantitatively evaluate the algorithm compared to other state-of-the-art spike detection algorithms.

The performance of FR-C and FR-B at different noise levels are plotted in Fig.5.5a along with other algorithms. The FR-B show higher detection accuracy compared to FR-B because the smaller p value allows the threshold to adapt into a finer level detecting spike

more accurately. From that figure, we can also observe that it achieves better detection accuracy compared to the offline spike detection [45], real-time float point algorithm [48] and fixed point algorithm in Chapter.3.

The FR-B performs better than the high complexity algorithm [128] in 0.05 to 0.15 noise levels. It is only less notable at 0.2 noise level compared to the high complexity algorithm. The performance is mainly degraded by the increased false detection rate. FDR can be improved by introducing more complexity spike enhancing operations, but it is shown in [129] and will be demonstrated in the next chapter that sometimes a higher FDR setting can be beneficial for neural decoding.

FR-B is operated at a relatively low sampling frequency. Low sampling frequency can significantly reduce the power of the hardware implementation while compromising detection performance. To evaluate how the spike detection performance can be degraded with the reduced sampling frequency, we have downsampled the raw signal at 24kHz to 2-4 times. The results are shown in Fig.5.5b). We can observe that there is nearly no degradation when the sampling frequency is above 8 kHz and start to degrade at 6 kHz (but still above 0.9 detection accuracy). As the energy of the spikes is mostly concentrated below 3kHz, 7kHz is the lowest sampling frequency for accurate spike detection according to the Nyquist theorem with 1kHz oversampling. Similar findings are also drawn from [47] and [96].

We also compare the FR-B with the shift-based ED emphasiser combined with mean or median with different threshold buffer sizes, which are the 'baseline algorithms' introduced in Section.4.2. The results are shown in Table.5.2

Compared to the 16-sample Mean (M16) or 25-sample Median (MD25). The FR-B achieves much higher detection accuracy across four noise levels. The detection performance of FR-B is minorly degraded (1.3%) if the noise level increases from 0.05 to 0.15. It is 5.3% for both M16 and MD25. At 0.2 noise level, the proposed algorithm can still maintain over 90% detection accuracy while it becomes 85% and even lower for the other two. It only becomes comparable to FR-B when using 50-point median to set the threshold, and using more buffers for mean-thresholding cannot improve its accuracy even further.

5.5.2 Spike detection adaptiveness

An interesting observation in Fig.5.5 is that the FR-B performs even better than the FR-C. The reason is that its updating scale is smaller leading to a more accurate threshold. However, with the spike peak mean tracking and max spike count auto-updating, it gains the advantage of better adaptiveness. Fig.5.6 shows the detection accuracy of two algorithms with different initial R_1 and the mean-based threshold with different multipliers. Compared to the mean-based thresholding the simple version of the firing-rate-based spike detection algorithm already has improved adaptiveness as the peak accuracy is aligned at around 56Hz.

Table 5.2: The spike detection accuracy of the proposed and baseline algorithms at different noise levels

Noise levels	M16	M90	MD25	MD50	ADF+FR
0.05	0.966	0.95	0.983	0.987	0.98
0.1	0.957	0.94	0.968	0.983	0.974
0.15	0.913	0.90	0.93	0.976	0.967
0.2	0.851	0.90	0.835	0.907	0.919
Avg	0.922	0.923	0.929	0.963	0.96

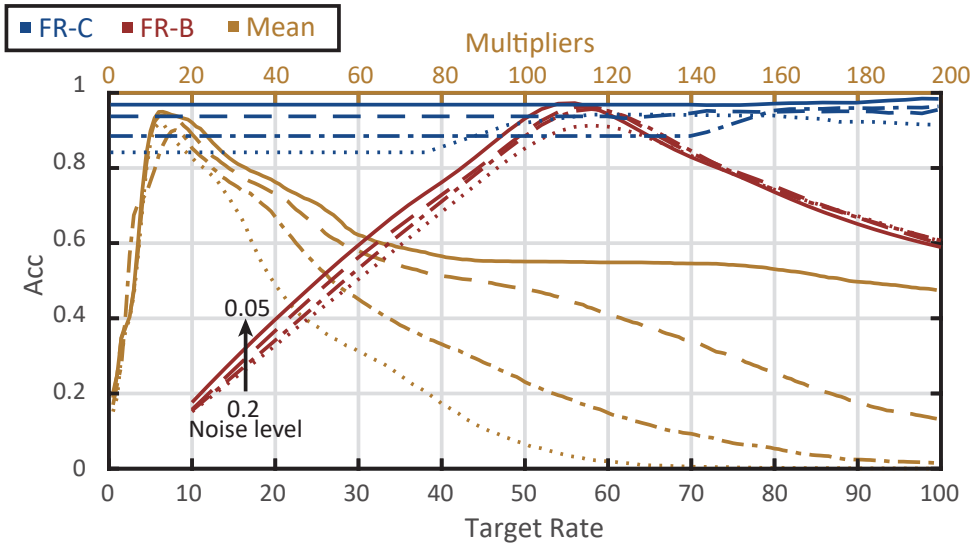

Figure 5.6: The spike detection accuracy varying with target detection rate or multiplier values for firing-rate-based algorithm basic version (FR-B), firing-rate-based spike detection with spike peak mean tracking and max spike rate auto-updating (FR-C) and mean-based spike detection (Mean) at different noise level on the synthetic dataset.

By using spike peak mean tracking and max spike rate auto-updating, the FR-C becomes robust to the initial setting. Because the spike peak levels are known, we can guarantee that the threshold is updated within a reasonable range.

Taken together, the FR-C can be regarded as having two modes. Mode I: when the MSR is inappropriate and the threshold tends to be unreliable, the threshold can still stay within the reliable region, and the MSR will update until it is also appropriate. Mode II: when the MSR is appropriate, the threshold will converge to a level so as to detect the spikes at the desired rate. These two modes operate in tandem and guarantee high performance.

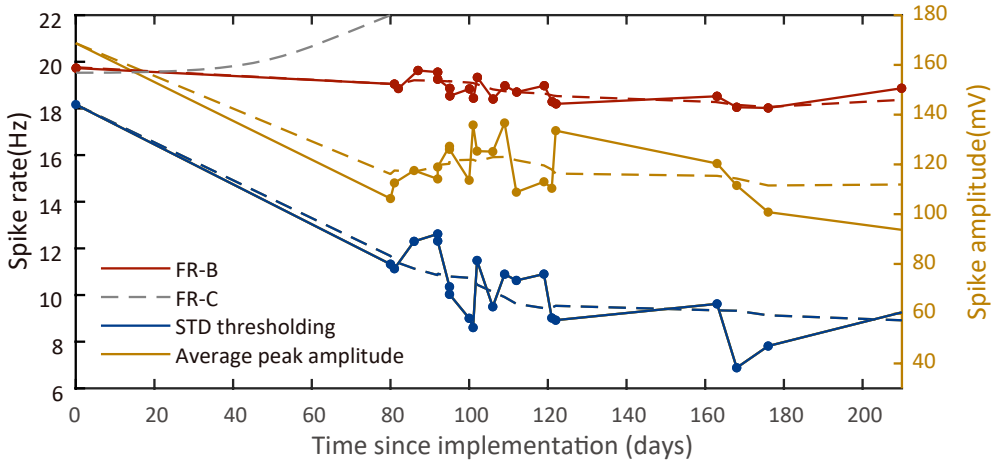

Figure 5.7: Spike rates of the Utah array recording detected from this work detected at 20 Hz (Red), previous (Grey), STD thresholding (Blue), and the average spike peak amplitude (Yellow) over 200 days of recording started from 27 July 2016. The grey curve quickly disappears, indicating that the FR-C method is not long-term stable.

5.5.3 Long-term detection stability on Utah array recordings

Long-term stability is another important aspect of implantable BMI applications. Fig.5.7 shows the real detection rate of three different spike detection outcomes on the Utah array recordings [43] using FR-B, FR-C and adaptive STD threshold (Provided with this dataset).

It can be clearly observed from Fig.5.7 that the detection rate of the STD threshold (Blue) is reduced from 18 kHz to 10 kHz, However, the detection rate of FR-B (Red, the target detection rate is set at 20 Hz) is relatively stable over 200 days of recordings.

The growth of scar tissue can push the neurons further away from the implants. The neural activities are thus weakened over time, leading to less significant spikes [51]. Statistical-based spike detection can therefore detect fewer spikes over time.

We observed inadequate performance in long-term detection stability of FR-C (Gery) because the threshold of FR-C can reduce to an unreasonable low level detecting too many spikes if not reset frequently. To be robust to the inaccurate initial settings, FR-C sets the threshold according to both the spike peak mean values and target detection rate. However, due to the sub-optimal spike peak mean value estimation method, false detection of noise peaks can reduce the estimated peak mean value below the real spike peak level. After a long time of detection, the spike peak means can become too low. This even lowers the threshold, creating more false detections. Such positive feedback can eventually cause the algorithm to fail unless the system is reset. This impact is especially prominent when the signal-to-noise ratio is low.

The FR-B however eliminates the dependence on spikes. Although the target detection rate should be properly set to achieve optimal spike detection performance, the algorithm

Table 5.3: Decoding performance when using the result from STD-based spike detection and the proposed spike detection algorithm

	STD w calibration	FR w/o calibration	FR w/ calibration
WCF	0.636	0.663	0.668
LSTM	0.738	0.753	0.759

can now provide long-term stable detection outcomes as shown in Fig.6.4. The threshold can be automatically adjusted to a lower level to detect the activities from neurons that have been pushed further away by the scar tissue maintaining the preset target detection rate. It is reasonable to assume that such stable detection results are expected to provide better long-term decoding performance compared to conventional statistical-based spike detection.

5.5.4 Decoding performance on Utah array recordings

The vast majority of spike detection algorithms are validated using synthetic data - as the ground truth is known *a priori*. Rarely are they validated using real recordings, as manual labelling can be time-consuming. Some studies use the recording of paired electrodes to obtain the neural firing ground truth of one signal neuron. However, this only allows us to evaluate the algorithms' sensitivity, not the false detection rate or accuracy. Other studies have attempted to use new metrics to evaluate spike detection performance in practice [130] or design algorithms to automatically label the spikes [131], but these methods have not been widely accepted.

The Utah array dataset mentioned in Section.2.3.3 recorded from the motor cortex of an non-human primate when controlling a ticker to operate cursor reaching tasks. Two standard neural decoders: Long-Short-Time Memory (LSTM) and Wiener Cascade Filter (WCF) are trained following [41] to decode the subject hand velocity from the spike detection outcome of the neural signals. We used Cross-Correlation (CC) between the actual and predicted hand velocity to evaluate the corresponding neural decoding performance and implicitly evaluate the spike detection performance, formulated as Eq.5.5.

$$\mathrm{CC} = \frac{\sum_{t=1}^{N}(Y_t - \bar{Y})(\hat{Y}_t - \bar{\hat{Y}}_t)}{\sqrt{\sum_{t=1}^{N}(Y_t - \bar{Y})^2}\sqrt{\sum_{i=1}^{N}(\hat{Y}_t - \bar{\hat{Y}}_t)^2}} \tag{5.5}$$

where Y_t and $\bar{Y}$ are the true target velocities at timesteps t and the average, $\hat{Y}_t$ and $\bar{\hat{Y}}$ are the predicted velocities and the average. N is the total number of samples in this trial. CC is measured on both the x and y-axis and averaged. The models are evaluated using 10-fold

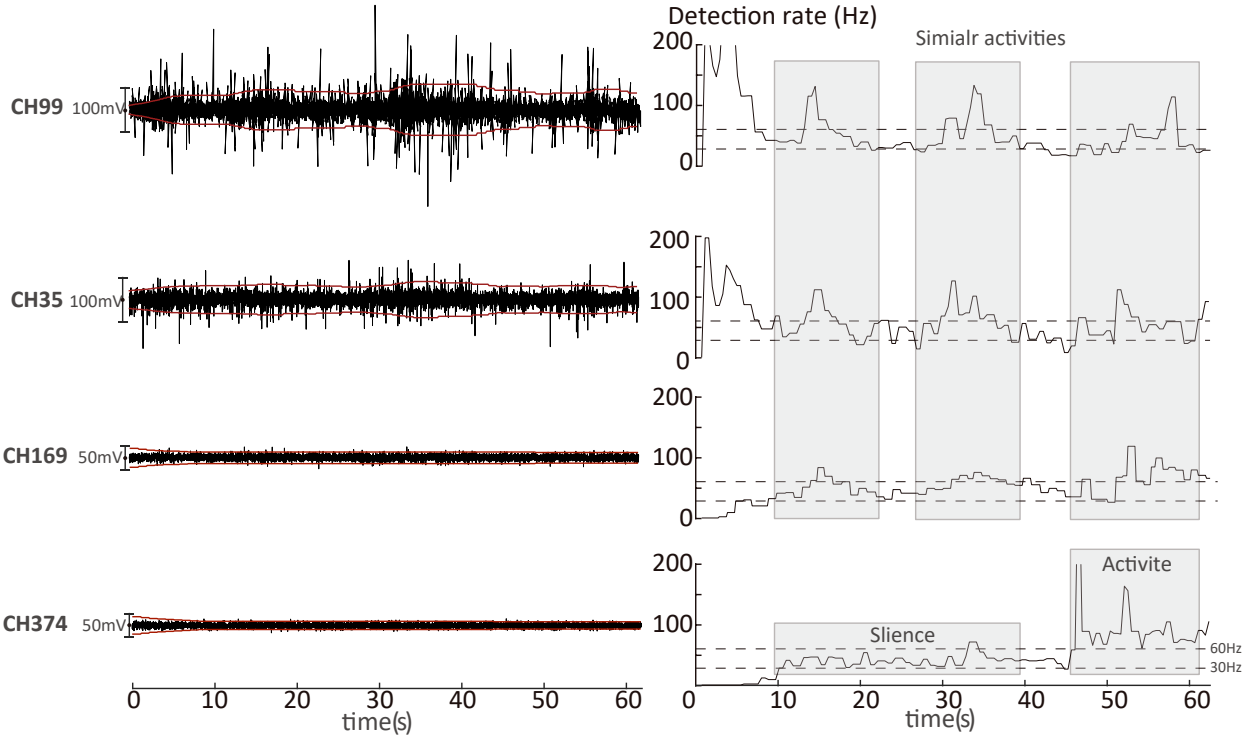

Figure 5.8: The Neuropixels recordings and adaptive threshold of four different channels with the corresponding instant detection rate. For better visualization, the recording is plotted at a lower sampling rate of 800 samples/s without taking absolute values, but the algorithm still operates at 7 kHz with absolute values being taken. Some spike peaks may appear missing in the plot due to the lower sampling rate, but they are accurately detected by the algorithm.

validation, and the final CC is obtained by taking the average of the CCs from all ten folds. The results are given in Table.5.3.

We can observe that for both conventional filter-based decoding and machine-learning-based decoding, with a fixed parameter setting across all channels (w/o calibration), the proposed spike detection algorithm achieves 1% to 3% higher decoding accuracy compared to the decoding results from calibrated STD-based spike detection. The corresponding mean or median-based result is expected to be even lower as their detection performance is supposed to be worse than the STD-based one. We also notice that channel-wised and day-to-day calibration only increased decoding accuracy by less than 1%, which means that after we find suitable parameters at the start, there is no need for further parameter calibration for the proposed algorithm in practice.

5.5.5 Validation on Neuropixel recordings

We have shown the effectiveness of the firing-rate-based spike detection algorithm on Utah array recordings. The synthetic dataset is also created from Utah array templates. To investigate the adaptability of our algorithm to other types of recordings, we also tested it on

Neuropixels recordings, which use a high-density electrode array, and were provided in [44]. Moreover, this high-density recording can provide a diverse collection of the different signal SNRs, allowing us to evaluate the algorithm's robustness in varying brain environments. However, since there is no ground truth available to evaluate the spike detection performance quantitatively, we visually inspected the spike detection results.

The detection outcomes are shown in Fig.5.8, where four different channels with varying SNRs are plotted at the same scale. The original recording is down-sampled to 800 Hz before plotting for better visualization, but the detection algorithm still operates at 7 kHz with the same target detection rate interval of [30 Hz,60 Hz].

Even though the neural activities are significantly attenuated across channel 99, channel 35, and channel 169, similar neural activities are detected from these channels according to the detection rate plot on the right. It is especially notable for CH169 that the spikes are nearly invisible from the noise. It has been shown in [129, 132] that detecting noise-level spikes is beneficial for neural decoding. Such an observation provides strong support for the robustness of the proposed algorithm to be used without calibration.

We can also observe that in CH374, the detection rate is relatively stable at the start. We assume the reason is that the neural activity could be attenuated to be invisible from the noise making this channel to be 'silence' for a while. Once the activities happen in a closer region, this channel is activated.

Another observation is that even though a target detection interval is set, the instant detection rate can exceed or fall below this range, rather than being confined within it. This is because the threshold is updated only gradually after the target interval is not met. Burst activities or inactive states can still be detected, even if the actual firing rate is outside that interval.

Although we cannot provide a quantitative analysis of the spike detection accuracy in Neuropixels recording, the visualisation and qualitative analysis above can demonstrate the potential of the proposed algorithm to be applied to new recordings at varying noise levels without calibration.

5.5.6 FPGA resource utilisation and power consumption

The downsampled input signal is created on MATLAB and generated from Tektronix AFG3102 Arbitrary Function Generator. A Digilent PMOD DA2 dual channel 12-bit DAC and Digilent PMOD AD1 dual channel 12-bit ADC are used for I/O conversion. The Tektronix DPO4034 Oscilloscope is used to monitoring the processed signal, threshold and the detection outcomes. The experiment setup is shown in Fig. 5.9.

The resource utilisation of different components of two firing-rate-based algorithms is given in Table.5.4. The utilisation of the ED+Mean spike detection is also reported, which is

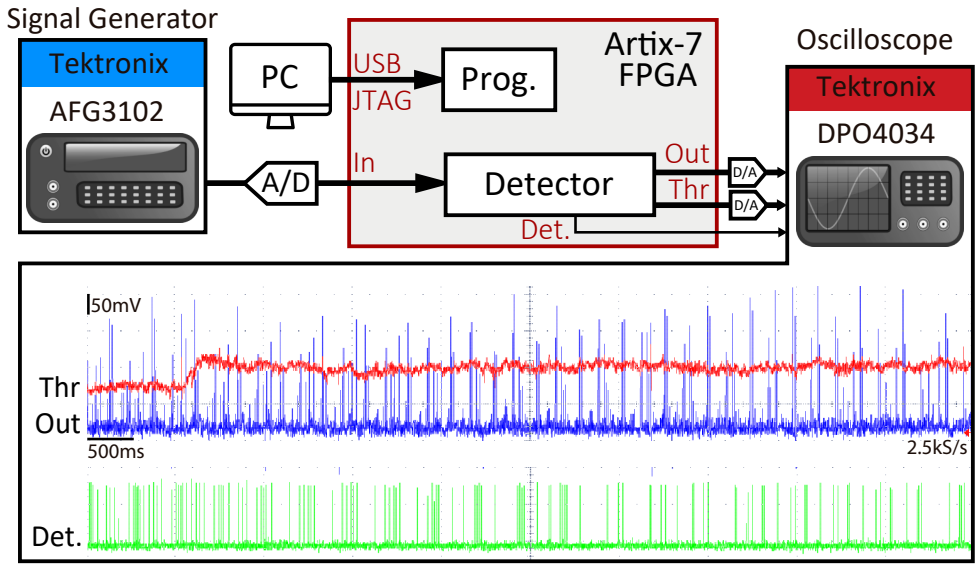

Figure 5.9: The experimental setup used to validate the spike detector FPGA implementation, showing the respective instruments that were used.

the most hardware-efficient spike detection implementation using statistical-based approach as it is shown in Section.4.2. ADF only occupies one-forth of the LUTs compared to the ED because no multiplication is used in ADF. Compared to the shift-based ED, ADF can still reduce LUT usage by 2/5. However, the RAM width is doubled because ADF uses two samples ahead. The mean-based thresholding requires nearly 100 LUT, while the FR-B thresholding only needs half and FR requires a similar amount of LUTs as the mean-based thresholding. Besides the LUT usage, the FR thresholding has a significant advantage in reducing RAM usage. RAM bandwidth can be reduced by an order of magnitude compared to the 16-point mean. This difference can be even larger when more samples are buffered for calculating the mean values. Given that the ED-Mean-based spike detection performs no better than the firing-rate-based spike detection, the resource usage report can illustrate the effectiveness of using the firing rate to set the threshold in reducing the hardware cost while maintaining and even improving the spike detection performance.

Finally, the FR-B consumes 0.21μW power per channel in Lattice ice40lp1k platform, which features a low-power FPGA with 40 nm technology. In comparison, the power consumption was 0.28μW per channel for FR-B. We should note that the Digilent Artix-7 FPGA platform can consume a considerable static power, which can lead to inaccurate power measurements. For that reason, we report the power consumption of a low-end FPGA platform to provide more representative values for power.

Table 5.4: FPGA implementation resource utilisation

		LUT	Registers	RAM Width (Bits)	DSPs
emphasizers	ED LUT Mul	121	0	10	0
	ED shift	48	0	10	0
	ADF	30	0	20	0
Thresholding	Mean (16/64)	95	0	380/1300	0
	FR-B	53	0	33	0
	FR-C	115	0	70	0
Full system	ADF+FR-B	78	16	53	0
	ADF+FR-C	193	106	90	0
	ED+Mean16	140	24	390	0

5.5.7 ASIC area occupation and power consumption

The register transfer level (RTL) implementation of the FR-B spike detection is mapped to two ASIC designs using TSMC 0.18 µm BCD Gen II and 65 nm LP technology. The synthesis and place & route were performed by Cadence Genus and Innovus by using standard digital cells. The implementation includes sequential and combinational logic cells. Based on the synthesized reports, sequential logic instances (mainly the memory) occupy four times the silicon area than combinational logic.

At 0.18 µm technology, the 128-channel FR-B spike detection occupies $0.93\times0.93\,mm^2$ and consumes 35.7 µW at 1.8 V supply voltage. When it is upgraded to 65 nm technology, it only occupies $0.31\times0.31\,mm^2$ and consumes 4.86 µW at 1.2 V supply voltage, resulting in a 7-fold reduction in power consumption and a 9-fold reduction in area. These two designs provide a good reference to study how the technology node impacts the power and area. and allow us to compare our work with state-of-the-art implementations using different technology.

As shown in Table. 5.5, our design has achieved the lowest power consumption and area occupation while still maintaining spike detection accuracy similar to the highest-performing design that has been included in the table. Compared the 65 nm implementation with the work in [47], the power consumption is nearly halved and the area occupied is reduced by about 1/4.

The power of our 0.18 µm design only consumes 1/20 of the power reported in [46] and occupies 2/3 less area. Although analogue designs are expected to have better hardware efficiency than digital designs, our digital design reduced the area occupation by nearly two orders of magnitude compared to the analogue design in [133], and more than one order of

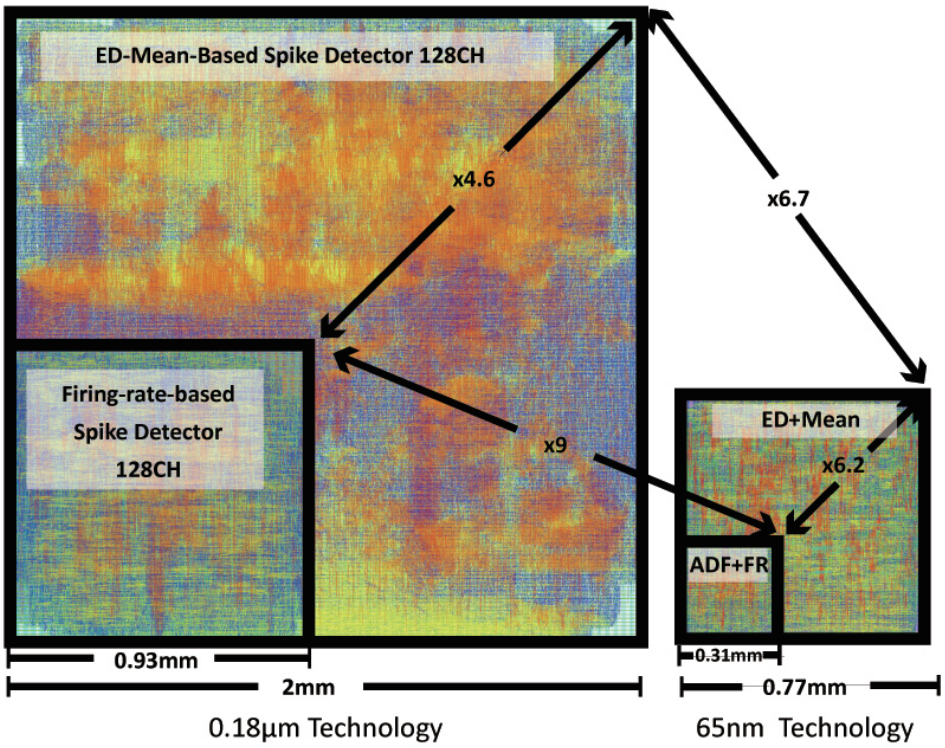

Figure 5.10: The layouts of 128 channel FR-B spike detector and ED-Mean-based spike detector in 0.18 µm and 65 nm technology. The ratio of the area occupation is annotated.

magnitude of the power compared to [111]. Even the 0.13 µm design in [117] requires twice as much power and area as our 0.18 µm design.

Another interesting observation in Table.B.2 can be found by comparing the work in [133] and [111]. They both use 0.18 µm analogue design, while the former one uses mean thresholding and the latter one uses RMS thresholding. The mean thresholding leads to significantly reduced power consumption but requires more area, as sample buffers are needed to calculate the mean values, increasing the area requirement. However, the latter one estimate RMS values without using data buffers, but it typically requires multiple clock cycles (e.g. 200 [47]) and complex logic, leading to less area occupation but significantly increased power consumption. However, firing-rate-based spike detection requires no buffer and can be calculated with only one clock cycle, making it the best choice for spike detection when jointly considering the power, area, and accuracy.

While we implemented our design in two different technology sizes, it is important to note that the hardware specifications, experimental setups, and power estimation strategies can vary, making it challenging to compare designs directly. Comparing our design to the ED+Mean16 algorithm allows us to control for these variables and compare the hardware efficiency difference resulting solely from the algorithm. We can use them as two generic models for comparing statistical and firing rate approaches for thresholding. Furthermore,

Table 5.5: Comparison to state-of-the-art ASIC adaptive spike detectors

	This work (0.18 µm)	This work (65 nm)	BioCAS2022 [47]	JNE2022 [46]	TBCAS2020 [133]	AEU2018 [111]	TBCAS2017 [117]
Technology (nm)	180	65	65	180	180	180	130
Implementation	Digital	Digital	Digital	Digital	Analog	Analog	Digital
Supply voltage	1.8	1.2	1.1	1.8	0.5	0.8	1.2
Clock frequency (MHz)	0.896	0.896	4	0.8	0.54	-	0.16
Preprocessing	ADF	ADF	Dual NEO	NEO	MAE	ED	NEO
Thresholding	FR	FR	STD	RMS	Mean	RMS	RMS
Channel	128	128	256	128	-	-	32
Resolution (Bits)	10	10	7	-	-	-	10
Sample frequency (kHz)	7	7	16	24	30	16	20
Power per channel (μW/Ch)	0.28	0.038[b]	0.07	4.9	0.116	5.1	0.52[d]
Area per channel (mm^2/Ch)	6.76×10^{-3}	7.51×10^{-4}	9.69×10^{-4} [c]	0.02	0.27	0.018	3.82×10^{-3} [c]
Accuracy	0.96[a]	0.96[a]	0.97	0.92	<0.97[e]	0.95	<0.95[e]

[a] The accuracy is 0.97 @ 12 kHz sampling rate.

[b] At best FoM (32 channels), this design only takes 0.01 μW/ch and 1.01 $\times10^{-3}$mm^2/ch.

[c] The area is obtained from multiplexing eight 32-channel modules.

[d] Only NEO processor and threshold estimator modules in the spike sorting design are included.

[e] Maximum accuracy on a different dataset.

we can investigate how these two approaches scale with increased channel counts. As a result, we implemented both designs in 0.18 µm technology with channel counts ranging from 8 to 512 channels. The layouts of 128 channel designs of both algorithms in two technology sizes are given in Fig.5.10 with the relative area ratios among them.

The 128-channel ED+Mean16 requires 2×2 mm^2 and consume 35.7 µW, which is 0.03 mm^2 and 1.77 µW per channel. It already achieves a similar area occupation and much lower power consumption compared to the work in [46] with the same spike detection accuracy (0.92). Compared to the firing-rate-based spike detection in this work, it requires 4.6 times more area and 6.3 times more power while the detection accuracy becomes 4% lower.

Fig.5.11 shows the scalability of these two implementations with an increased channel count. The area occupation per channel decreased slowly for both spike detectors because combinational circuits are shared across channels. The power consumption of ED+Mean16 scales much faster than ADF+FR, ED+Mean consumes 0.64 µW more power than ADF+FR per channel when the channel number is 8 and it becomes 6.7 µW per channel when the channel count increased to 512. The reason is that ED+Mean16 requires much more memory than ADF+FR and the memory is supposed to be the most power-consuming module [47]. The observation above demonstrates that the firing-rate-based thresholding has better scalability with increased channel count compared to the mean-based thresholding. As the RMS-based spike detection can consume even more power compared to the mean-based one, our design potentially also outperforms the RMS-based spike detection as well.

In this section, we have demonstrated improved hardware efficiency over other works.

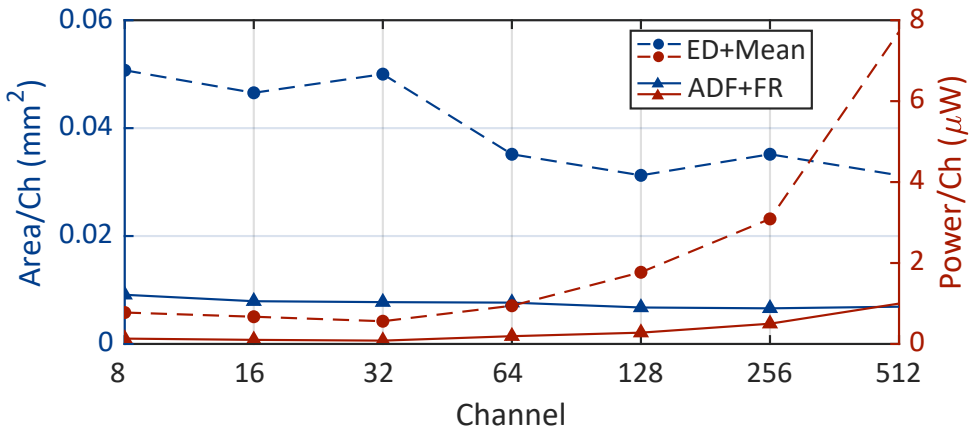

Figure 5.11: The area occupation and power consumption scaling with channel count for two different spike detector implementations in 0.18 µm technology.

Such an advantage comes from various aspects. On the one hand, since ADF requires no multiplication, it requires fewer logic cells, and the data bandwidth is maintained at 10 bits. In contrast, emphasisers using multiplications are more complex and require 20-bit output bandwidth. Reduced logic complexity and data width lead to less resource usage and more efficient placing and routing. On the other hand, the firing-rate-based thresholding of our algorithm does not require buffering of samples, reducing RAM usage. Additionally, it only requires one clock cycle for each sample. As a result, power and area can be significantly reduced.

5.6 Conclusion

In this chapter, firing rate information, for the first time, is used to set the threshold for on-implant real-time spike detection. Incorporating the ADF, the proposed firing-rate-based spike detection has achieved state-of-the-art spike detection performance, notable adaptiveness and significantly reduced hardware complexity. The main contribution of this chapter is summary below:

- The adaptive algorithm achieves 96% spike detection accuracy on a commonly used synthetic dataset, without the need for any prior training.
- The firing-rate-based spike detection is also capable of adapting in the long term, providing stable detection outcomes over 200 days, and can be used on different types of recordings collected by Utah arrays and Neuropixels probes without further calibration.
- The 128-channel ASIC design implemented in a 65 nm CMOS technology occupies 0.096 mm^2 silicon area and consumes 4.86 µW from a 1.2 V power supply.

The work in this chapter focuses on optimising the balance between spike detection performance and hardware complexity. With the firing-rate-based spike detection, we ultimately achieve notable performance on both sides. However, it is still worth noting that the goal of spike detection in a BMI system is to reduce the data bandwidth and provide informative features for neural decoding. Most spike detection algorithms are designed without considering such an ultimate goal. In the next chapter, we will show how the firing rate information can play a part in improving spike detection accuracy and reducing the data bandwidth.

Chapter 6

Spike detection and beyond: The relationship between spike detection and neural decoding

Continuing the journey of designing hardware-efficient adaptive spike detection algorithms, this chapter explores the broader implications beyond just spike detection. By investigating the relationship between spike detection and neural decoding, it becomes clear that setting spike detection parameters is not just about finding the optimal threshold to distinguish spikes from noise but rather a trade-off between data bandwidth and decoding performance. Based on this insight, new algorithms for firing rate compression and decoding are proposed with the aim of reducing data bandwidth while maintaining long-term decoding stability.

6.1 Introduction

In the previous chapters, we have shown the need of using spike detection to distil the features from the raw neural signals. Efficient on-implant feature extraction and compression can effectively reduce the data bandwidth and eventually allow brain activities to be transmitted wirelessly. We have introduced novel spike detection algorithms that not only enhance spike detection performance and ensure long-term detection stability but also demonstrate compatibility across various types of recordings. Importantly, these new algorithms have also been designed with an emphasis on reducing hardware complexity, a crucial step that

A portion of the content in this chapter has been published in [126], in which I contributed to the software/hardware algorithm design, result analysis, and draft manuscript preparation and revision.

further improves the efficiency and adaptiveness of the BMI systems.

Accurate detection of all spikes is required by studies of neural dynamics or brain activities. This criterion is not equally critical within the context of BMI applications. Our understanding of spike detection in wireless BMI systems has evolved to dispel a prevalent myth: The ultimate objective of a spike detection algorithm should extend beyond simply detecting all neural spikes accurately. Rather, its primary purpose is to enable effective neural decoding and ensure efficient bandwidth reduction for wireless transmission.

The reasons for this divergence are twofold. Firstly, the hardware used for accurate detection might surpass what is viable for wireless iBMI systems. Secondly, even if optimal parameters are established in spike detection to maximally differentiate spikes from noise, such parameters do not inherently guarantee the highest accuracy in neural decoding. Notably, what might initially be perceived as 'noise' could indeed be informative. Additionally, a frequent detection rate could inadvertently lead to a reduced compression ratio. Thus, it is crucial to maintain a balanced approach towards spike detection in BMI systems, taking into account the broader objectives of neural decoding and bandwidth efficiency.

Moving forward, we used the same dataset to decoding the subject hand movement using neural activities as in Section.5.5.4 and studied the relationship between spike detection and neural decoding, specifically through the use of a firing-rate-based spike detection algorithm, yielding the following insights:

1) Detecting only significant peaks causes less than 1% performance degradation. As fewer spikes are detected, less information needs to be transmitted. This can be the best trade-off between the data bandwidth and decoding performance, which is often preferable in the context of wireless brain machine interfaces.

2) Detecting small spike peaks around the noise floor can enhance the decoding performance. It reveals that there are neural activities far away that are informative relatively. Utilising this extra activity in decoding can improve decoding performance other than being detrimental despite detecting additional noise that is often considered to be detrimental to spike detection performance. Therefore, applications involves designing BMIs to maximise decoding performance or studying underlying science require undistorted neural information should follow this approach where the threshold should be set only a little above the noise floor, i.e. sensitivity is more important than specificity. A similar finding is also observed in [129, 132], we re-proofed such a finding in a practical setting.

In addition, we have designed a novel neural decoder that leverages the silence/active information provided by the firing-rate-based spike detection algorithm, resulting in a significant improvement in long-term decoding stability. Fig.6.1 illustrates the tasks and methods involved in this chapter.

Lastly, we have explored the opportunity for further reducing the data bandwidth through entropy encoding and demonstrated the benefits of using the firing-rate-based algorithm over

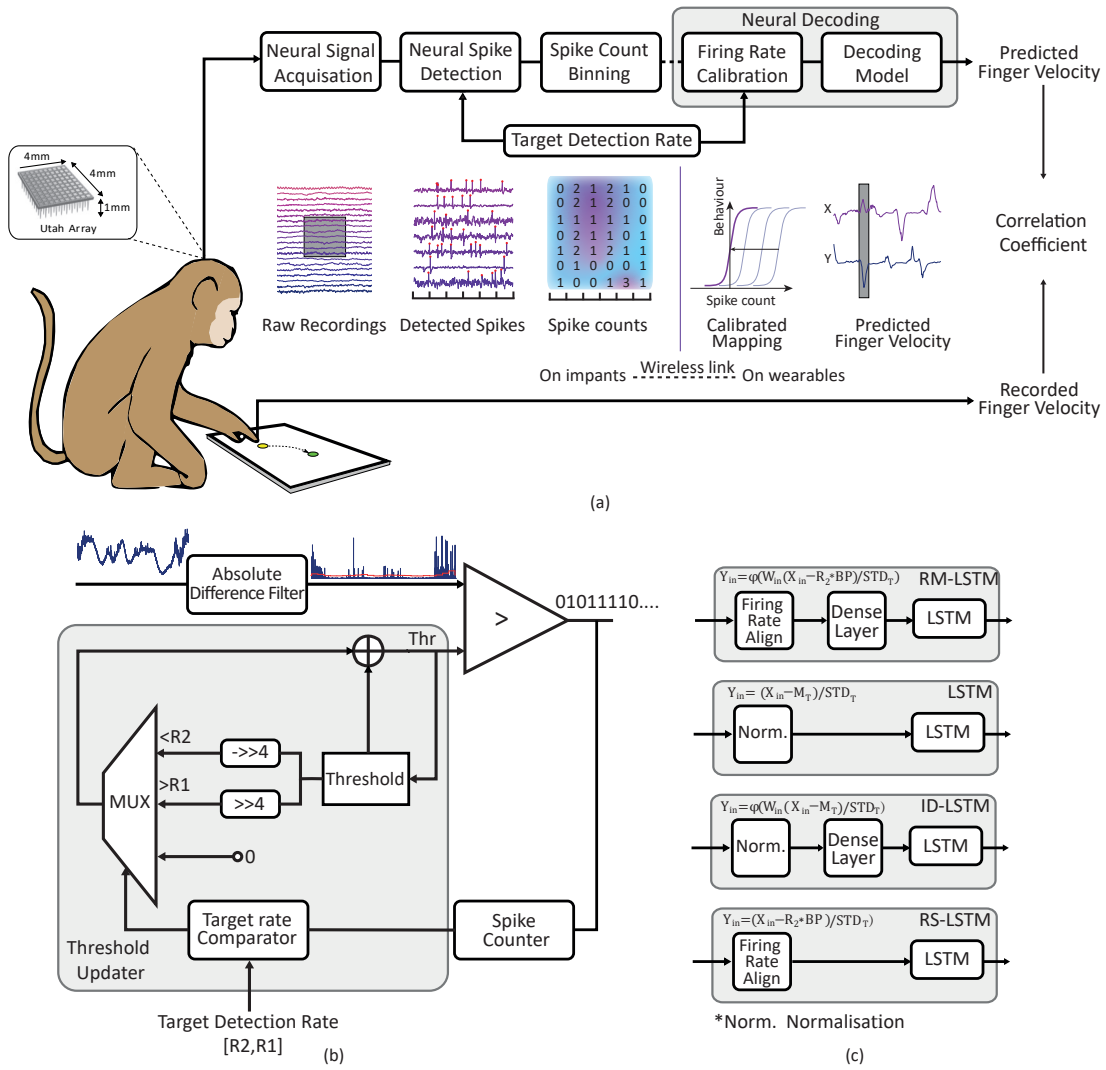

Figure 6.1: a) Spike detection - neural decoding system co-design. After digitising the intracortical signal, spikes are detected at the pre-set target detection rate. Detected spike counts are binned at a fixed period. Binned spike counts are then modulated using the target detection rate to align the feature representation over time. Finally, the decoding model predicts the target velocity, and its correlation coefficient to the real velocity is used to evaluate the system performance. b) Logic circuit of spike detection. The Absolute Difference Filter takes the input signal's derivative and absolute value to remove the LFPs. The spikes are detected if the filtered signal outnumbers the threshold. The threshold is updated duty-cycled according to the relationship between the target detection rate and the number of detections counted by Spike Counter. Repeated detections are counted only once. c) Flowcharts of the neural decoding modules. All modules use the LSTM decoder but have different input layers. The proposed Rate-Modulated LSTM (RM-LSTM) subtracts the target detection rate from the binned spike counts followed by a dense layer. Compared to the proposed architecture, the conventional LSTM subtracts the mean values calculated from the training dataset to perform a standard normalisation of the input feature. Input Dense LSTM (ID-LSTM) consists of a dense layer with the standard normalisation. Rate Subtracted LSTM (RS-LSTM) only modulates the input feature using the target rate without the dense layer.

conventional algorithms.

This chapter will introduce these three aspects, showing the advantages of employing the firing-rate-based spike detection algorithm in wireless iBMIs.

6.2 The relationship between spike detection and neural decoding

In order to find the relationship between the spike detection performance and neural decoding performance, we need a measure of the spike detection performance on the realistic dataset. It is challenging as there is no ground truth. However, with the FR-B, we can perceive the spike detection performance using the target detection rate. More specifically, if we set the target spike detection rate close to the local firing rate, the algorithm should provide the highest spike detection performance. To prove that, we used the Quiroga dataset (with an average spike rate of 60Hz), and tested different target detection rates R_1 by sweeping from 40 to 100Hz. The highest detection accuracy is expected to be achieved at around 60 Hz.

The detection accuracy, false detection rate, and sensitivity are shown in Figure.6.2b. The spike detection performance peaks at $R_1 = 56$Hz, close to the 60Hz real spike firing rate. It demonstrates that the proposed spike detection algorithm can provide the highest detection performance when the target detection rate is close to the actual firing rate. It can also be observed that when the target detection rate is above the actual firing rate, both sensitivity and false detection rate increase, which means more spikes are detected while more noise peaks are recognised as spikes and vice versa.

Moving on to the real dataset, we tried the Utah array dataset. Data collected from the Utah array, which has been spike sorted [43], shows 3 to 5 clusters per channel. Meanwhile, [132] suggests that, on average, there are 3.8 to 4 clusters per channel when no spikes are discarded. Therefore, a reasonable assumption of the target detection rate should be 30-40 Hz for a ticker control task (as motor neurons can fire at below 10 Hz when operating little-effort tasks). The actual interval can be even lower as the number of neurons can be fewer than the observed clusters and the subject did not operate the ticker continuously. With this assumption, we are able to set a target detection rate interval to get the most accurate spike detection results on a realistic dataset for the motor tasks. Varying the target detection rate allows us to change the spike detection performance and builds the relationship between detection and decoding.

Figure.6.2b is obtained after decoding the binned spike detection results with WCF and LSTM. 30-40 Hz should be the desired detection rate achieving the highest detection performance and is supposed to provide the best spike detection performance as well. However, the decoding performance of both methods peakss at 46Hz. Such unmatching suggests the

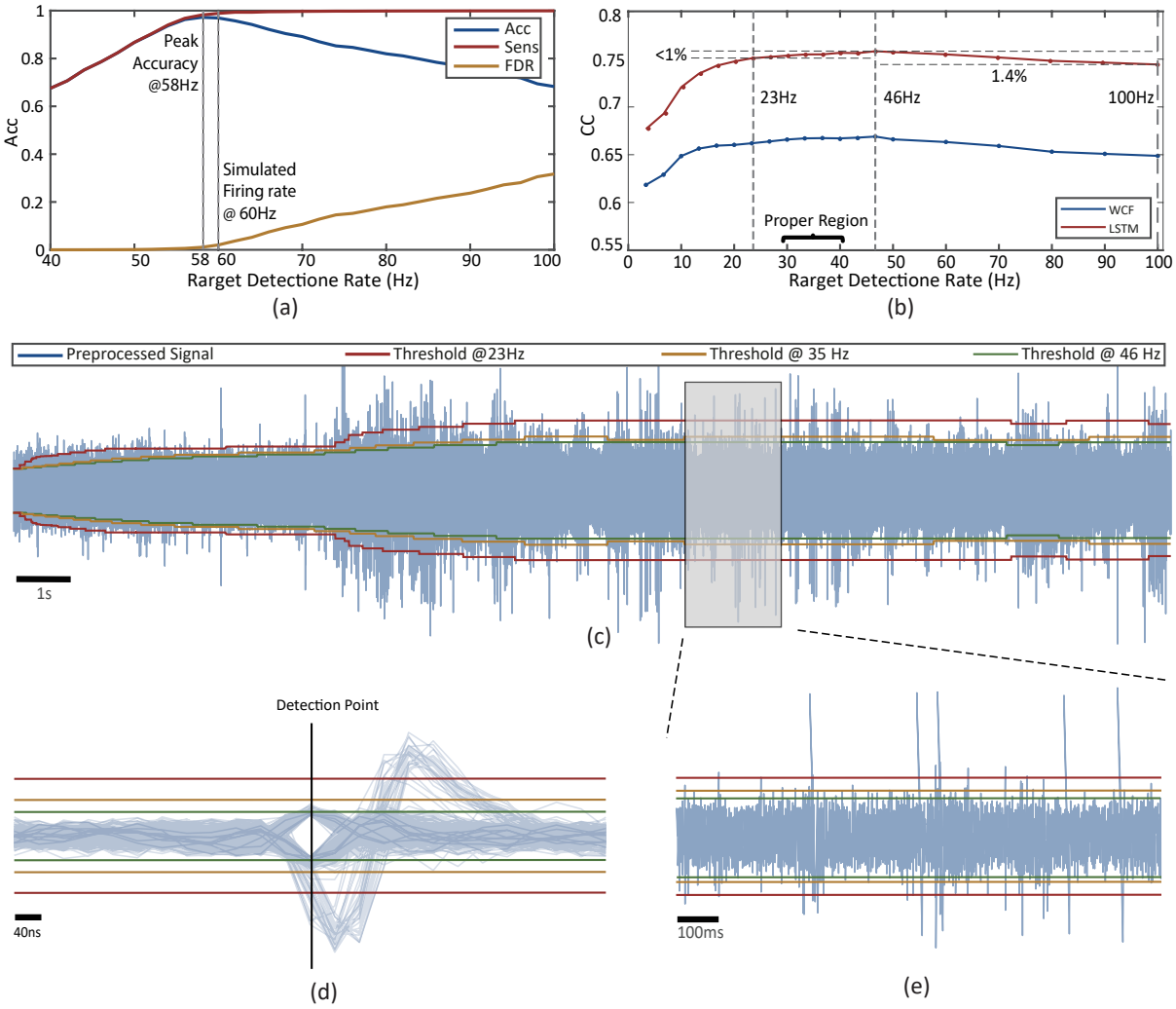

Figure 6.2: a) The detection accuracy (Acc) of the proposed algorithm at different target detection upper bounds for a simulated recording of spikes firing at 60Hz. The accuracy peaks at 58Hz. b) Decoding correlation coefficients (CC) using conventional LSTM and WCF v.s. target detection rate. CC peaks at 46Hz. However, if double the detection rate to 100 Hz, 1.4% performance can happen while only less than 1% degradation can happen if the detection rate is halved 23Hz. c) Preprocessed signal with three different thresholds: 23 Hz, under-detected, only significant spikes are detected; 35 Hz, properly detection; 46 Hz, over-detected, many noise-level peaks are detected. Note that in practice, the absolute value is used instead of dual thresholds, but we kept the sign here for better visualisation. d) The spikes and noise overlapped plot with the thresholds. The thresholds are the average threshold after the 7s at different detection rates. e) A zoomed-in plot of spike detection outcome.

nonlinear relationship between detection and decoding.

This desired detection rate interval might be inaccurate. To be more rigorous, we also visually observed the spike detection result in Figure.6.2c-e. They show a recording snapshot with the spike detection threshold generated with $R_1 = 23$ Hz, 35 Hz and 46 Hz in red, yellow, and green separately. The threshold at 46Hz is obviously adapted to the level only a little above the noise floor. Compared to the threshold at 35Hz, which is the best threshold to

discriminate spikes from the noise by visual inspection, this lower threshold detected more noise-level peaks. Both theoretically speaking and by inspection, detection at 46Hz is not an ideal spike detection setting but achieves the highest decoding performance.

This finding does not match what is expected intuitively - better spike detection performance and better decoding performance. It is also unmatched with the spike detection algorithm design objective that aims at increasing the sensitivity while reducing the false detection rate because the detection of these noise-level spikes can increase the false detection rate significantly. It suggests that some around-noise peaks are results in spikes actually firing far away. Detecting these spikes improves decoding performance despite also detecting more noise. This finding is also supported by the result in [129, 132, 134], suggesting that discarding the noise cluster is detrimental to the decoding performance in decoding neural signals using single-unit activities.

Another finding is that if the detection rate is doubled, to e.g. 100 Hz, which means about half of the spikes detected could be noise peaks, the performance is only degraded by about 1.4%. In addition to detecting noise, the low detection threshold borders the spatial detection region around the electrodes, and as a result, activities from regions further away with different functional tuning are more likely to be detected. This can also lead to degraded decoding performance [129]. However, this degradation is minor. This suggests that the decoding models, even the simple filter-based model (WCF), are robust to noise. We only investigated a simple Long-Short-Term Memory neural network. It would be expected that if more advanced deep learning models are used, even better noise resistance can be achieved.

Finally, if the detection rate is halved to 23 Hz, many of the spikes with lower peak amplitudes can be missed. However, the decoding performance is only degraded by less than 1%. This finding is significant for on-implant spike detection, allowing us to send fewer data without sacrificing much performance. The threshold crossing can be represented using a binary stream at 1 kbps. A lower detection rate means a more sparse signal, and the more sparse the signal is, the easier it is to be compressed, leading to less wireless transmission power.

Spike detection previously is an unsupervised problem in practice and highly depends on the results from synthetic data. The findings in this section are significant in providing a guide on setting the threshold. Especially for the brain machine interfaces with a limited power budget, the goal of spike detection would no longer be to find the best threshold level for the highest detection accuracy but the trade-off between the data bandwidth and decoding performance. With the firing-rate-based spike detection, the threshold level can be selected so it fulfils the system requirement on bandwidth and decoding accuracy without needing to consider the performance of the spike detection algorithm itself.

6.3 Firing-rate-modulated neural decoding

An effective spike detection algorithm can efficiently maintain the majority of the informative features inherent in raw neural signals while minimising data bandwidth. Upon evaluating the Neuropixels recording as discussed in the previous chapter, it became evident that the relative scale between the detection rate and target detection rate of the FR-B algorithm offers robust discrimination between silent and active behaviours.

Based on that assumption, we decided to integrate the target detection rate into neural decoding. This innovative approach resulted in a surprising improvement in long-term decoding accuracy by 10% after 80 days of model implementation. The specifics of the model implementation and its subsequent evaluation are detailed in the following sections.

6.3.1 Spike count binning

The output of the spike detection is a binary stream. It has been found in [41, 135] that binning the binary threshold crossings into MUA counts at a certain period can improve the decoding accuracy. However, a trade-off between decoding accuracy and temporal resolution has to be made. In this work, a 50 ms bin period is used as in [41, 135].

6.3.2 Neural decoding

The binned MUA counts from different days are split into ten sets for decoding model cross-validation. For each validation cycle, one set is used for testing; one set is used for validating the best model and the rest sets are used for training. The average score across 10-fold validation is used as the final score of this day.

As mentioned before, the proposed spike detection algorithm can provide extraction information of the potential silent and active periods. We have designed a special input layer for the LSTM to utilise such extra information to modulate the binned spike count. A Rate-modulated LSTM (RM-LSTM) is proposed to have the capability to keep the consistent feature representation even though the input feature statistics are changed over time as shown on top of Figure.6.1c. The input layer consists of a firing rate modulation step and a dense neural network layer which is formulated as Eq.6.1

$$\mathbf{Y_{in}} = \phi(\mathbf{W_{in}}(\mathbf{X_{in}} - R_2 * BP)/STD_T) \tag{6.1}$$

where $\mathbf{X_{in}} \in \mathbb{N}^N$ is the binned MUA signal, and N is the channel number. STD_T is the standard deviation of the training set, R_2 is the target detection count lower limit, and BP is the bin period in seconds used for firing rate alignment. $\mathbf{W_{in}} \in \mathbb{R}^{N \times N}$, $\phi(\cdot)$ and $\mathbf{Y_{in}} \in \mathbb{R}^N$ are the weights, Sigmoid nonlinear activation and the output of the dense layer,

Consistent feature representation means the input of the LSTM can be at a similar scale statistically regardless of the varying of the binned MUA count statistics. This is achieved using the firing rate modulation. Input feature consistency is critical to maintaining the long-term stability of the decoding model, as the SNR can degrade with implant ageing. $\mathbf{X_{in}} - R_2 * BP$ keeps the potential activate periods (large firing rate feature) positive while it shifts the less active periods (small firing rate feature) to be negative. This input-output relationship of the model is correctly mapped after shifting, as shown in Figure.6.1a firing rate modulation. This correctly mapped relationship is expected to be easier to learn by the decoding model compared to normalisation feature. At the same time, such mapping is input-feature-scale invariant. Even though the recording quality is degraded, the mapped feature statistic can stay unchanged. Moreover, such mapping is consistent for different target detection rates, and there is no need to retrain the decoding model after changing the target detection rate of the spike detection algorithms.

The nonlinear response adds the capability of the models to simulate the nonlinear relationship between the input firing rate and the output that estimates behaviour. We assume the effect of the input firing rate on the behaviour (finger movement velocity) should show a saturation trend when the input firing rate is very high or small. Such a relationship can be well simulated using the Sigmoid function, a widely used activation function in neural networks.

Compared to the conventional LSTM, RM-LSTM used different input ($\mathbf{X_{in}} - R_2 * BP$) and more parameters ($\mathbf{W_{in}}$). We built another two models: Rate-Subtracted LSTM (RS-LSTM), which only subtracts the rate from the input and Input-Dense LSTM (ID-LSTM), which only has the input dense layer and nonlinear activation. RS-LSTM controls the number of parameters to identify the improvement made from firing rate modulation over standard normalsation, and ID-LSTM controls the input feature statistics to identify the contribution of the increased model size. The architectures of these models are shown in Figure.6.1c.

6.3.3 Decoding performance

We used the target detection rate to bridge the neural decoding with spike detection. The improvement is shown in Figure.6.3a. Compared to the Conventional LSTM (Blue), over 1% improvement has been achieved using RM-LSTM (Red) at all different detection rates.

When the detection rate is above 40 Hz, more noise is likely to be detected. The detected noise will be added in MUA counts to the decoding model (Activity from further cells irrelevant to the targeted activity can also be regarded as noise). Because of such noise, we can observe a decaying trend to the tail of the conventional LSTM. RM-LSTM however contains the decay suggesting the firing rate modulation (keeping consistent feature representation) not only helps to improve the model capability but also improves the model's robustness to

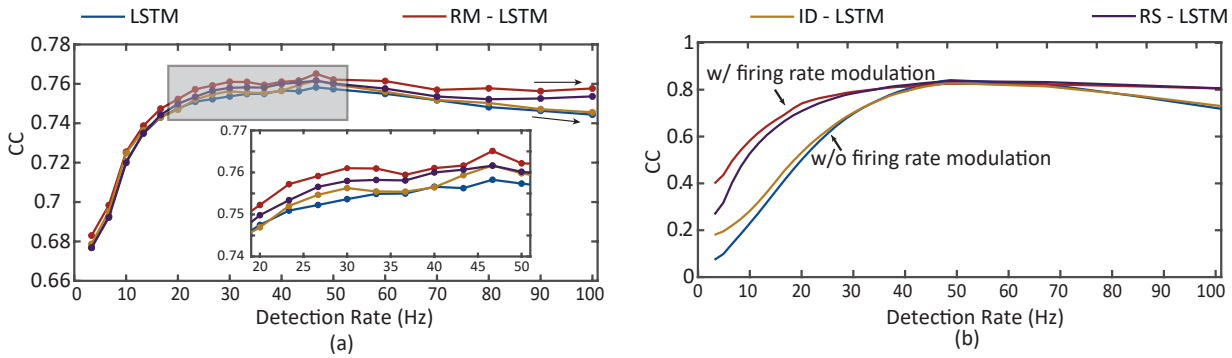

Figure 6.3: a) Decoding performance of different decoding models. RM-LSTM achieves the highest decoding performance among all models at all detection rates. b) The decoding performance when the model is trained with detections at 50 Hz and tested with detections at varying rates. The decoding performance is distinguishably improved when two rates are different by using the firing rate modulation.

the noise.

To further understand such improvement in decoding performance, we designed the ID-LSTM model, which has the same number of parameters as RM-LSTM but without the firing rate modulation as Figure.6.1c shows. The yellow curve in Figure.6.3a shows its performance. ID-LSTM provides better decoding performance than conventional LSTM but is not as good as RM-LSTM. Such observation suggests that the improvement of RM-LSTM is not solely from the increased number of neural network parameters, but the firing rate information also matters.

We mentioned that firing rate modulation could improve the decoding performance because such an operation can keep a consistent feature representation and zero-centre the mapping, which is easier to learn. However, data normalisation can have a similar effect. In order to show that using the rate information of the proposed spike detection is better than the normalisation, we designed the RS-LSTM shown in Figure.6.1c. It subtracts the target rate instead of the mean calculated from the training set. It can be observed in Figure.6.3a that with such a simple subtraction, RS-LSTM achieves (purple) better performance than the ID-LSTM model, which has 48190 more parameters. We can also observe that the decaying trend with increased detection rate is also contained, similar to RM-LSTM. Such observation suggests that using the firing rate modulation can more accurately map the high firing rates to the active behaviour while mapping the low firing rates to the inactive behaviour than using standard normalsation enhancing the learning procedure of the deep learning models.

We also tested how the performance of RM-LSTM can be affected when fewer or more spikes are detected (Changed target detection rate). We tried to train the model with the detection results in one target detection rate (50Hz) and tested at a different detection rate simulating more or fewer spikes being detected when the environment changes. We can observe the significant improvement in Figure.6.3b, especially when fewer spikes are detected

when the rate information is used, demonstrating the robustness of the system co-design.

Our results above not only demonstrates the improved performance of the proposed spike detection and neural decoding system co-design compared to the baseline models but also provide explanations as to where the improvement comes from, addressing the importance of consistent feature representation. It is essential that AI is not blindingly applied but rather its application is informed by some knowledge of the "system". We believe such a co-design system can thus provide better intuition in designing and optimising decodes performance than by simply applying complex deep learning architectures.

6.3.4 Long-term decoding stability

Maintaining long-term neural decoding performance is a challenging task in brain machine interfaces, because of the recording quality degrading with implant ageing. The growth of the scar tissue can push neurons further away from the electrodes weakening neural activities. As it has been shown in the previous chapter, the average threshold crossing rate of STD thresholding detection outcome can drop from 18 Hz to below 10 Hz due to the weakened signals as the yellow and grey curves show in Figure.6.4a. such decaying results in two factors which are detrimental the long-term neural decoding: information loss and feature statistical variation.

The proposed co-design methodology can address this issue. The firing-rate-based spike detection can update the threshold to lower levels over time, allowing for the detection of neural activity further from the electrodes and stabilizing the detection rate. Meanwhile, firing-rate-modulated decoding maintains zero-centered binned spike rate features, providing statistically consistent features in the long term, regardless of the noise levels and spike amplitude. A statistical-based approach cannot provide such a feature because the optimal threshold level is nonlinear to the noise statistics. While setting a new multiplier to the noise statistics can resolve this issue to some extent, manually tuning all channels to maintain a stable output stream is impractical.

Three different models have been implemented to evaluate the improved performance of the proposed co-design. In addition to the RM-LSTM and conventional LSTM, we implemented another model, Mean-Tracked LSTM (MT-LSTM). Its only difference from the conventional LSTM is that test data is normalised by the test data statistics rather than the values from the training set. This model simulates tracking of the input mean values, so-called mean-tracked.

The conventional LSTM and MT-LSTM use the threshold crossing of the adaptive STD threshold (It has been tested that decoding the adaptive STD threshold results performs better than using the static STD threshold), while RM-LSTM uses the results of the proposed algorithm detecting at 20 Hz and 50 Hz. The 20 Hz model keeps the threshold crossing

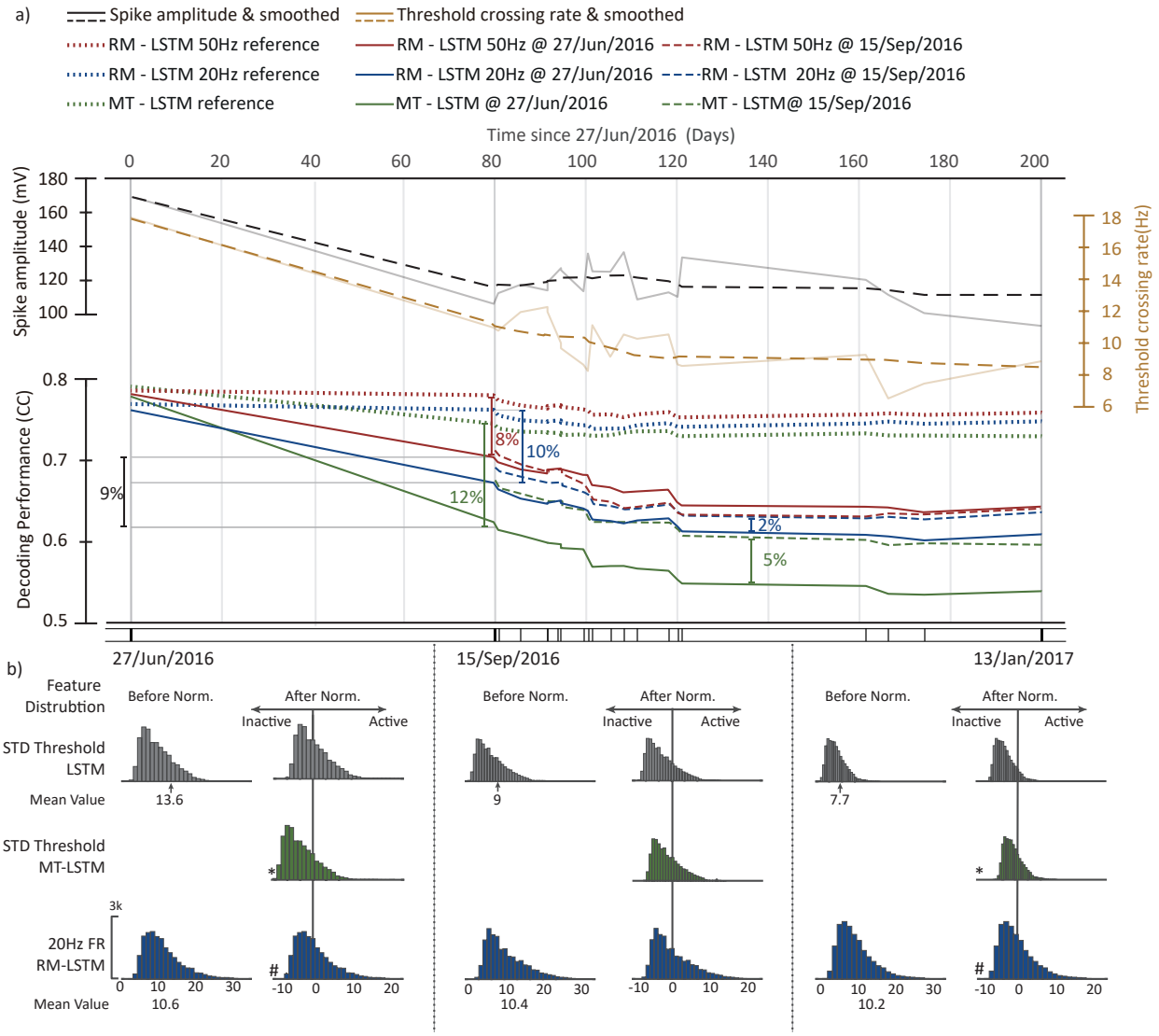

Figure 6.4: Firing-rate-modulated co-design long-term decoding performance and visitation illustration. a)The spike amplitude and threshold crossing rate (STD threshold detected) over time, the long-time decoding performance over 200 days. RM-LSTM and MT-LSTM are trained on the data recorded on 27/Jun/2016 and 15/Sep/2016, simulating severe and mild SNR degradation with electrode ageing after implementation. RM-LSTM is trained at a target detection rate of 20 Hz and 50 Hz, representing a compact setting with a threshold crossing rate close to MT-LSTM and a high-performance setting. Decoding performance curves are smoothed for better visualisation. The date of the experiments is indicated by the time axis on the bottom. b) Feature distribution before and after normalisation/firing-rate modulation when using STD or firing rate (FR) thresholding. Distributions are sampled on the date 27/Jun/2016, 15/Sep/2016 and 13/Jan/2017.

rate close to the STD-detected ones, while the 50 Hz model allows further cell activities to be detected to improve decoding performance. Comparing the decoding performance of conventional LSTM and MT-LSTM to RM-LSTM can demonstrate the need of using firing rate information in both spike detection and decoding to stabilise the features and improve the long-term decoding performance.

The decoding performance of different models is shown at the bottom of Figure.6.4a. We can clearly observe their decaying trends following the signal degrading. The conventional LSTM cannot provide any long-term decoding ability because of the inconsistent threshold crossing rate over time. CC is consistently below 0.1 and therefore not plotted. RM-LSTM-50Hz (Red), RM-LSTM-20Hz (Blue) and MT-LSTM (Green) are trained with the data from two different dates. The model trained on recordings collected on 15/Sep/2016 simulates the scenario when the electrode ageing is mild (dash curves), while the one trained on 27/Jun/2016 simulates the scenario of severe electrode ageing (solid curves). We also provide the results of the models trained and tested on the same day data as a reference observing how much the performance is degraded (dotted curves).

When the electrode ageing is mild, the nearly overlapped red and blue dash curves indicate that there is no significant difference using activities further away. Two RM-LSTMs outperform the MT-LSTM by about 2%. Such improvement is mainly contributed by the model capability instead of firing-rate modulation's contribution to long-term stability because the RM-LSTM reference already outperforms MT-LSTM reference for about 2% seen from the dotted reference curves. However, the improved performance of RM-LSTM is observed when the ageing is severe. After 80 to 100 days of implementation, compared to the reference model, 12% performance degradation can happen on MT-LSTM and 10% for RM-LSTM-20Hz. However, it is only 8% for RM-LSTM-50Hz.

Moreover, comparing the severe electrode ageing case to the mild ageing one, there is about a 5% difference for MT-LSTM. However, it is only a 2% difference for RM-LSTM-20Hz and even better performance using RM-LSTM-50Hz (Because the data quality of 27/Jun/2016 is better than that of 15/Sep/2016). Compared to the performance of the first day of implementation, after 100 days of implementation, MT-LSTM degrades for more than 20% while both RM-LSTMs degrade for only around 10%. The co-design improves the long-term decoding performance by nearly 10%.

To better understand the reason for improved long-term stability, we plotted the input feature distribution before and after normalization/firing rate modulation for different models in Figure 6.4b. The distribution of the firing rate feature obtained using STD thresholding changes significantly (becomes more left-tiled). After 200 days of implantation, the relationship between behaviour and firing rate feature was completely different using STD thresholding. Applying standard normalization (the grey row) to the testing data shifted the testing feature distribution mostly to the negative side. That is the reason for making the conventional LSTM fail over time. Although tracking the mean value of the features (the green row) can address this issue to some extent, the decoding model can still potentially suffer from the large difference between the training and testing feature distribution (comparing the two * distributions). However, when using firing-rate-based spike detection with firing-rate modulation, there is no significant difference in the distribution (comparing the

Table 6.1: The average decoding performance of the models tested on the date onward after being trained with the recording on 27/Jun/2016(Served ageing) or 15/Sep/2016 (Mild ageing).

	Served ageing		Mild ageing	
	20Hz	50Hz	20Hz	50Hz
RM-LSTM	0.632	0.669	0.653	0.662
MT-LSTM	0.497	0.633	0.663	0.668

two # distributions). Shifting the distribution with the target detection rate can effectively map high/low firing rates to active/inactive behaviour, which is always valid in these 200 days, unlike that using STD thresholding.

It is worth noting that using normalization for the firing-rate-based spike detection feature can lead to a similar mapping as firing-rate modulation does. In order to show how introducing the target detection rate information into decoding can improve the long-term decoding performance, getting rid of the contribution to FR-B spike detection. We tried to train MT-LSTMs on 27/Jun/2016(Served ageing) and 15/Sep/2016 (Mild ageing) detecting using FR-B at 20 Hz and 50 Hz. The average testing CC over the days onward is given in Table.6.1 (The trends are similar to what have been shown in Figure.6.4). It can be observed that RM-LSTM and MT-LSTM can achieve similar long-term decoding performance when the ageing is mild, while MT-LSTM is slightly better. However, over 3% improvement can be made when ageing is served, demonstrating the usefulness of such extra information in long-term stable neural decoding.

Based on the observations above, it appears that consistent features and activities from the further regions are both beneficial to improve long-term stability, especially when the signal ageing is severe. The proposed FR-B with RM-LSTM can take advantage of both, delivering improved long-term stability compared to other LSTM implementations with standard statistical-based spike detection.

6.4 Discussion

Our results suggest that the relationship between spike detection and neural decoding performance is non-linear. The spike peaks of high amplitude contain substantial information helping decoding while detection of spike peaks of amplitude comparable to the noise can improve the decoding performance despite more noise being detected. Such a finding brings opportunities for both brain machine interface engineering and neuroscience. Considering the proposed spike detection and neural decoding system co-design, we identify that main-

taining the input data consistency is one key to improving the decoding performance and robustness, especially in the long term, when recording quality is degraded gradually with electrode ageing.

6.4.1 Feature consistency

Inconsistent features can degrade the performance of the learning models. This inconsistency can come in three different forms. The first form is invalid features. For example, the model is fed with Motor Cortex recordings for behaviour decoding but tested with Visual Cortex recordings, which is technically incorrect. The second form is the incomplete feature set. For example, the model is trained on neural recordings for reaching tasks but tested to decode placing tasks. Such inconsistency can be overcome by designing more generalised models.

More importantly, the third form is the statistical difference, which we can meet in long-term neural decoding. The recorded amplitude of action potentials can gradually decrease over 37% within two months after implantation [136]. Some studies [41, 132, 136] suggest that threshold crossing can achieve better long-term stability than using spike waveforms or spike sorting results.

We need to know how machine learning or deep learning models are trained to understand such a reason. There is a normalisation step before feeding input features into models and guaranteeing the input is not statistically changed. However, tested datasets can only be normalised using training dataset statistics as the tested dataset should be unknown to the model. Normalising the test data using training statistics can be a problem in long-term decoding. The realistic mean of the input models can be lower than the features used in training. Normalisation can shift the testing features' centre to negative while it is expected to be at zero. That is why long-term degradation can happen (in addition to more noise being potentially contaminated).

Using spike waveform as the feature can be affected the most as it is most sensitive to such a normalisation error. Using SUA from spike sorting has worse long-term stability than using MUA. Besides fewer spikes appearing in each cluster, the inconsistent spike waveform can negatively impact the clustering algorithm. Such artefacts will propagate to the decoding model and even lower the decoding accuracy. Spike detection is only affected by the reduced number of detections.

Knowing the reason for long-term stability degradation, one key to resolving this issue is maintaining the consistent input feature of the decoding model. The proposed spike detection and neural decoding co-design system addresses that. This idea can potentially be explored in other domains, such as spike waveforms, single-unit activity, or local field potentials.

6.4.2 The opportunity for firing rate compression

In [42], It is suggested that relaxing some of the hardware requirements, for example, sampling frequency, amplifier performance, and ADC precision, can reduce the system power consumption by an order of magnitude without much degrading the decoding performance. It also shows that randomly missed spikes can only degrade decoding performance by a small amount. Until this work, there has not been reported any spike detection algorithm that can "miss" spikes heuristically.

Based on the finding of the relationship between the spike detection and neural decoding performance, we suggest relaxing the amount of data transmitted. Only transmitting the high-level peaks can reduce half of the data to be transmitted with less than 1% decoding performance degradation for behaviour decoding task.

The threshold crossings can be encoded into a binary stream or the number of spikes detected in a certain period. The reduced detected spikes rate can either create a more sparse binary stream or reduce the dynamic range of the possible spike count range.

In both cases, the bandwidth can be reduced. Taking the latter case as an example, we assume the detection stream is binned at τ sec, and the detection rate is λ Hz following Poisson distribution. The number of arrivals N of one bin is a Poisson process as Equ.6.2 shows.

$$P_\tau(k) = P[N(t+\tau) - N(t) = k] = \frac{e^{-\lambda\tau}(\lambda\tau)^k}{k!} \tag{6.2}$$

In practice, the spike count is saturated at a value S to limit the spike count data width. The probability of the spike count exceeding S (inclusive) is given in Equ.6.4 and it needs L^S bits for representing S values

$$P_\tau^S = P_\tau[N(t+\tau) - N(t) \geq S] = 1 - \sum_{k=0}^{S-1} P_\tau(k) \tag{6.3}$$

$$L^S = \lceil log_2(S) \rceil \tag{6.4}$$

If we saturate the spike count at $P_\tau(S) < 1‰$, when a 50Hz detection stream is binned at 0.01s, S is 6 with $P_{0.01}^6 \approx 0.12‰$. When the detection rate is reduced to 25Hz, S becomes 4 with $P_{0.01}^4 \approx 0.13‰$. Therefore 3 bits are required for 50Hz spike detection counts, while it is 2 bits when detection at 25Hz.

From an entropy perspective, the theoretical minimum entropy coding length L_τ that losslessly compresses such a stream can be calculated as Equ.6.5.

$$L_\tau = -\sum_{k=0}^{\infty} P_\tau(k) log_2(P_\tau(k)) \tag{6.5}$$

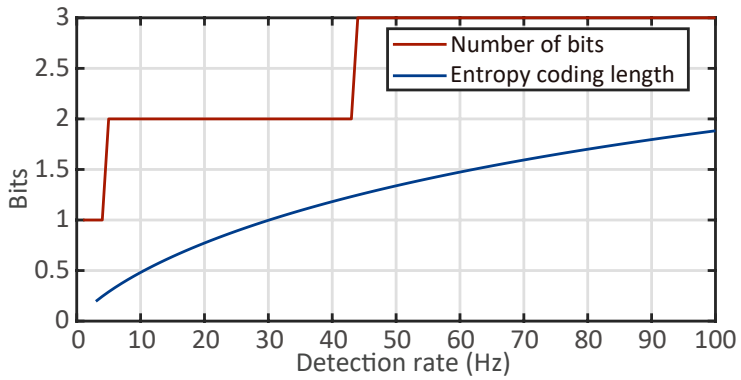

Figure 6.5: The number of bits and entropy encoding length for representing the spike counts

Figure.6.5 shows how the number of bits and minimum code length for representing the spike counts. The minimum coding length is 1.34 at 50Hz, while it becomes only 0.89 at 25Hz. 33% bandwidth can be reduced while it only introduces less than 1% decoding degradation.

Huffman encoding is a form of entropy encoding. To achieve optimal encoding performance with Huffman encoding, the distribution of the signal being encoded should match that of the signal used for training the encoders. In essence, the arrival frequency as outlined in Eq.6.2 should remain constant over time for optimal compression performance. Conventional statistical-based methods are unable to guarantee this; however, employing the firing-rate-based spike detection algorithm ensures stable, long-term spike detection outcomes. Furthermore, the firing-rate-based algorithm allows control of the output bit rate during the spike detection algorithm design phase, simplifying the estimation of transmission power, a critical factor in the wireless implant power budget.

In collaboration with Dr. Oscar W Savolainen, we have designed an event-driven data compression system using customized Huffman encoders. The idea is to transmit only the channel numbers of the active channels along with their spike detection count within a predefined bin period. This compression algorithm has proven particularly effective for compressing spike counts with bin periods under 10ms. Using this compression method, we are able to compress a 1 kbps binary spike detection stream into a 50 bps stream with almost no loss of information. It is estimated that the number of channels one implant can host will increase by factors ranging from 4.6 to 15, depending on the chip size. Dr. Savolainen and I jointly designed the algorithm, with my primary contribution being hardware optimization. Detailed information about the algorithm design and its evaluation can be found in the Appendix.C.

6.5 Conclusion

This chapter brings the current phase of the research in this thesis to a close. We have moved beyond just spike detection to show the nonlinear relationship between spike detection and neural decoding. Furthermore, we have demonstrated how the neural decoding accuracy can be improved and how the data bandwidth can be further reduced after spike detection. The key findings in this chapter are summarised below:

- The standard deep learning procedure on normalisation is not suitable for long-term decoding due to varying features over time. The firing-rate-modulated neural decoding is in part proposed to address this problem and it improves the long-term decoding performance by 10% after 80 days of implantation.
- Detecting small spike peaks near the noise floor can enhance decoding performance by capturing distant, informative neural activities, despite additional noise being detected. This approach benefits BMI design and scientific exploration requiring undistorted neural information.
- Detecting only significant peaks causes less than 1% performance degradation. As fewer spikes are detected, 30% less information needs to be transmitted. This really strikes an ideal balance between bandwidth usage and decoding accuracy.

This is the first study to establish a firm link between spike detection and decoding. We believe the significant improvement presented in this chapter can motivate hypothesis-led data science strategies to further develop brain machine interfaces.

Chapter 7

Conclusion and future directions

Brain Machine interfaces have provided new approaches for neurological disorder treatment, motor function impairment restoration and communication capability recovery. Translational efforts have intensified to develop products that cater to clinical or consumer needs, emphasising the need for safety and portability for life-long use. Designing wireless implantable brain machine interfaces is a trend aiming to fulfil these requirements, while the wireless transmission bandwidth and power consumption remain to be the bottleneck.

Facing these challenges, this thesis has developed novel neural signal processing and low-power hardware co-design, specifically focusing on bandwidth reduction of spike-based BMI system using spike detection techniques. Furthermore, the new opportunities for feature compression and enhancing long-term system performance, thereby paving the way for the next generation of high-channel-count brain machine interfaces

7.1 Original contributions

The main contributions of this thesis have been summarised below:

7.1.1 High performance, ultra-hardware-efficient spike detection co-design

Starting with the simplification of existing spike detection algorithms, several iterations have been collaboratively designed with hardware considerations in mind. The most refined version of this spike detection algorithm relies solely on a fixed-point representation, completely bypassing the need for any multiplication. Incorporating a time-sharing architecture, the system is capable of sharing most of its resources among channels, which significantly reduces resource usage and power consumption. In addition, this configuration promotes scalability, accommodating upwards of a thousand channels with ease.

The developed algorithm has been effectively implemented across various platforms including MATLAB, FPGA, and ASIC, and has undergone rigorous testing using both real and synthetic datasets. Remarkably, it achieved over 96% detection accuracy on the synthetic, positioning it among the most accurate and hardware-efficient spike detection algorithms currently available.

Our 65 nm ASIC 128-channel design demonstrates the algorithm's efficiency, consuming 38 nW per channel and occupying $7.51 \times 10^{-4}\,\mathrm{mm}^2$ per channel, which is the most power and area efficient spike detection design amongst the literature that we have examined.

7.1.2 Adaptive spike detection and decoding using firing rate information

Adaptive spike detection poses significant challenges in the realm of real-time spike detection. This thesis begins by examining the adaptiveness of the traditional spike detection algorithm through practical testing and theoretical modelling. It has been observed that the conventional statistical-based spike detection approaches tend to be less adaptive in the noisy environment of the brain, and they require frequency calibration to deliver optimal performance, particularly over the long term.

In response to the limitations of these traditional methods, we introduced firing rate information as a determining factor for setting the spike detection threshold. This innovative approach allows the algorithm to adaptively update the threshold, accounting for signal strength drift over time as scar tissue forms around the electrodes. Consequently, this algorithm delivers stable detection results over an extended period. It has led to a noteworthy 10% improvement in long-term neural decoding on the motor cortex following 80 days of implantation.

7.1.3 The relationship between spike detection and decoding

The design of most spike detection algorithms is primarily driven by the aim of enhancing spike detection performance. Yet, the relationship between spike detection and neural decoding remains largely unexplored. While a few studies have begun to delve into this area, there is still no clear understanding of how to set a threshold to achieve optimal system performance.

In this thesis, using the firing-rate-based spike detection algorithm, we have found this relationship to be non-linear. Interestingly, the optimal spike detection threshold does not correspond to the highest decoding accuracy. Instead, a threshold set at around the noise level does.

However, detecting only significant spikes seems to strike the best balance between neural

decoding and data bandwidth. By setting a higher detection rate to detect half as many spikes, it is possible to reduce the data bandwidth by at least 30%, while the decoding accuracy is marginally impacted, reducing by less than 1%. This finding illuminates a new direction for further optimising spike detection algorithms for BMI systems with limited transmission bandwidth.

7.2 Future directions

To further develop the BMI system building on this thesis, the future directions could be:

1. Further rigorous testing is necessary across a broader spectrum of recording types, tasks, brain areas, and subjects. In this thesis, a comprehensive evaluation of the co-design has been carried out exclusively on 200-day Utah array recordings from an NHP motor cortex during the execution of a ticker control task. To demonstrate the robustness and adaptiveness of the co-design and confirm our findings on setting the optimal threshold to be consistent, it is essential to extend performance testing to other recordings. Such a diversified testing strategy will contribute to a more universally applicable and reliable understanding of the system's efficiency and adaptability.

2. Extending the firing-rate-based spike detection and decoding into real-time BMI applications presents a promising direction for exploration. At present, many BMI systems utilise relatively basic spike detection algorithms coupled with standalone decoding models. However, the implementation of firing-rate-based spike detection and decoding offers potential benefits. The methods in this thesis could significantly enhance the long-term stability of system performance, a vital attribute for the practical utility of BMIs. Furthermore, this approach might also reveal new scientific findings, contributing to the advancement of the field.

3. Designing algorithm with hardware efficiency in mind. Applying this co-design philosophy more broadly to other types of signals or features, such as SUA or LFP, could lead to significant improvements. In this thesis, we have reduced power and area usage substantially, achieving orders-of-magnitude reduction. Techniques such as fixed-point representation, multiplication-free operation, and resource-sharing have proven valuable for minimizing hardware complexity. By applying these techniques in a customised manner to the targeted signals or features, we anticipate achieving a similar level of complexity reduction. This would greatly benefit applications extending beyond MUA-based BMIs.

7.3 Concluding remarks

During my PhD, I have had the opportunity to witness BMI progress to a translational stage as seen by the rise of numerous recent commercial efforts globally. Despite visible progress towards the translation of BMIs, the field still exists neuroscience unknowns and technical challenges, which need to be addressed before BMIs can truly become practical tools in the clinic or replace our phones as the next user interface.

Power and area constraints have always been challenges preventing new technologies from being real-world applications. This thesis aims to address these challenges specifically for future hardware design of wireless BMIs. The software-hardware co-design herein illustrates opportunities for reducing both power consumption and area occupation of the devices. Such advancements could provide feasible on-implant signal processing within hardware constraints, a critical step towards the development of wireless BMIs.

The technical advancements may reciprocally benefit neuroscience research by enabling previously unattainable experiments, such as all-day monitoring of free-moving subjects. With the multidisciplinary effort, we can push the boundaries of our current understanding and application of BMIs even further. The fusion of neuroscience, electrical engineering, computer science, and even psychology can expedite the translation of BMIs from research laboratories to clinical settings, and eventually, to everyday life. The impact of such work could be transformative, revolutionising not only how we treat neurological conditions, but also how we understand and interface with our own minds.

Embarking on an almost five-year journey to develop hardware-efficient spike detection algorithms has been both demanding and fulfilling. This journey began in November 2018, during my Master's degree, and involved a rigorous process of refining the algorithm again and again to achieve its optimal performance. As I sought to improve the accuracy, adaptiveness, and complexity of the algorithm, each new iteration presented its own set of challenges, leaving me to grapple with a fresh puzzle. Yet ultimately, I managed to strike a harmonious balance among these factors. I recall reaching a point after finalising the latest version of the algorithm where I was feeling no longer faced with a new puzzle. It was at this point that I decided to venture beyond spike detection. Although further improvements can still be made, I am confident that I am on the right path now.

This journey, while challenging, has been immensely rewarding and has significantly honed my critical thinking and problem-solving skills. As a student of electrical and electronic engineering, I delved into the realm of neuroscience to better interpret my data. And as an information engineer, I extended my skill base from MATLAB to embedded programming, Verilog, FPGA design, and eventually ASIC design. This multidisciplinary journey, spanning electronics to neuroscience and signal processing to chip design, will undoubtedly yield lifelong benefits.

Therefore, I present this thesis not only to demonstrate the contributions made during my doctoral studies but also to mark the learning curve I have experienced as a significant milestone reflecting my personal and academic growth.

Appendices

Appendix A

List of Publications

Publications based on materials in this thesis:

Calibration-free and hardware-efficient neural spike detection for brain machine interfaces
Zheng Zhang, Peilong Feng, Alexandru Oprea, and Timothy G Constandinou
IEEE Transactions on Biomedical Circuits and Systems, 17(4) 725-740, 2023.
https://doi.org/10.1109/TBCAS.2023.3278531

Firing-rate-modulated spike detection and neural decoding co-design
Zheng Zhang, Oscar W Savolainen, and Timothy G Constandinou
Journal of Neural Engineering, 20(3):036003, 2023
https://dx.doi.org/10.1088/1741-2552/accece

Algorithm and Hardware Considerations for Real-Time Neural Signal On-Implant Processing
Zheng Zhang, Oscar W Savolainen, and Timothy G Constandinou
Journal of Neural Engineering, 19(1):016029, 2022
https://dx.doi.org/10.1088/1741-2552/ac5268

Adaptive spike detection and hardware optimization towards autonomous, high-channel-count BMIs
Zheng Zhang and Timothy G Constandinou
Journal of Neuroscience Methods, 354:109103, 2021.
https://doi.org/10.1016/j.jneumeth.2021.109103

Selecting an effective amplitude threshold for neural spike detection
Zheng Zhang and Timothy G. Constandinou

44th Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC), pp.2328-2331, 2022
https://doi.org/10.1109/EMBC48229.2022.9871955

Hardware evaluation of spike detection algorithms towards wireless brain machine interfaces
Alexandru Oprea, **Zheng Zhang**, and Timothy G Constandinou
IEEE Biomedical Circuits and Systems Conference (BioCAS), pp.60-64, IEEE, 2022
https://doi.org/10.1109/BioCAS54905.2022.9948632

Ultra Low Power, Event-Driven Data Compression of Multi-Unit Activity
Oscar W Savolainen, **Zheng Zhang**, Timothy G Constandinou
bioRxiv, 2022
https://doi.org/10.1101/2022.11.24.517853

Other publications during the course of this research that are not included in this thesis:

A robust and automated algorithm that uses single-channel spike sorting to label multi-channel neuropixels data
Zheng Zhang and Timothy G Constandinou
10th International IEEE/EMBS Conference on Neural Engineering (NER), IEEE, pp.783-787, 2021
https://doi.org/10.1109/NER49283.2021.9441234

Hardware-Efficient Compression of Neural Multi-Unit Activity
Oscar W Savolainen[co], **Zheng Zhang**[co], Peilong Feng, Timothy G Constandinou
IEEE Access, 10:117515-29, 2022
https://doi.org/10.1109/ACCESS.2022.3219441

Development of an Ultra Low-Cost SSVEP-Based BCI Device for Real-Time On-Device Decoding
James Teversham, Steven S Wong, Bryan Hsieh, Adrien Rapeaux, Francesca Troiani, Oscar Savolainen, **Zheng Zhang**, Michal Maslik, Timothy G Constandinou
44th Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC), pp.208-213, 2022
https://doi.org/10.1109/EMBC48229.2022.9871064

Appendix B

Firing-rate-based spike detection (Complex version) hardware implementation and evaluation

B.1 Hardware implementation

B.1.1 FPGA implementation

Lattice ice40LP1K is an ultra-low-power FPGA board with 1280 logic cells (look-up-tables LUT + flip-flops FF) and sixteen 4kbit memory blocks (bRAMs). The program was developed with Mentor Modelsim Lattice Edition and iceCube2 2020.12. In order to minimise resource usage, we have designed a channel time-sharing architecture for the FPGA implementation, in which each channel shares the same processing circuits interchangeably. The block diagram in Fig. B.1. A shows the architecture. It has been divided into four main parts. 1) The clock Generator (CG) that generates the clock signal for different blocks. 2) The processing unit (PU) that is shared across channels for spike detection processing. 3) The memory unit (MU) that stores the temporary processing status of each channel and schedules the time-sharing across channels. 4) The control unit (CU) that controls and schedules the signal processing data flow. More details are given below.

The contents in this appendix has been published in [11], with my full contribution to the software/hardware algorithm design and result analysis

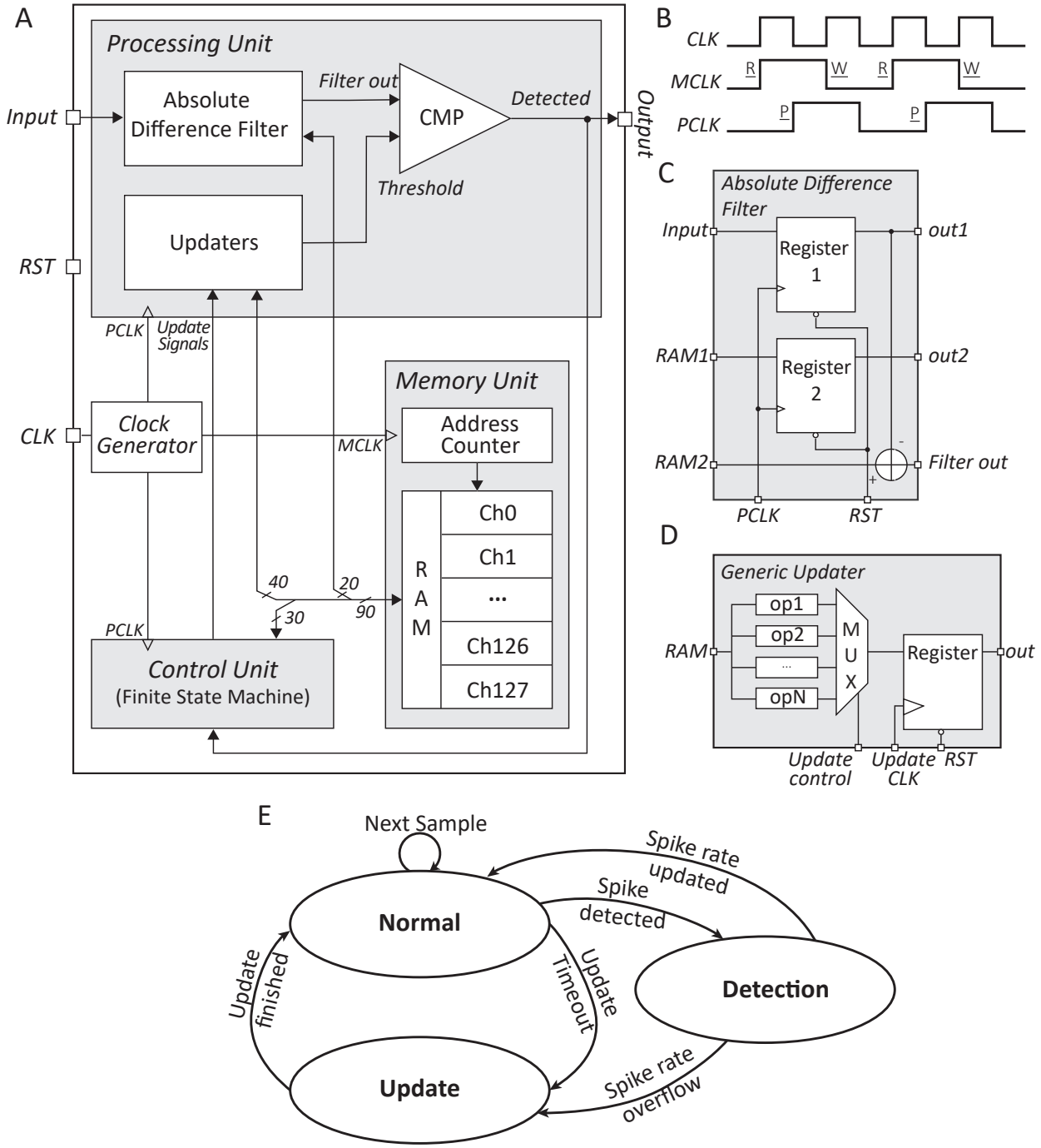

Figure B.1: A. Diagram of the FPGA implementation. Input is the current sample after ADC and output is a binary signal indicating detected or not. B. The clock signals from the clock unit. CLK: System clock. MCLK: Memory clock. PCLK: Processing clock for PU and CU. R: registers read from RAM. W: registers write to RAM. P: registers process the data that read from RAM. C. Multichannel absolute difference filter architecture. RAM1 and RAM2 are the value read from the RAM, which is the last input and the previous input before the last. Out1 and out2 are new values to be stored in RAM. D. A generic architecture of different updaters, including a threshold updater, a spike peak mean value updater and a max spike rate update. They share a similar architecture which stores the updated value according to the update signal from the control unit. E. The state transition diagram for the finite state machine.

B.1.1.1 Clock generator

When input arrives, the registers first read the data from RAM, then processing logic analyses the input with the status data from RAM to obtain the current status. These new values are, in the end, stored back to RAM. There are three operations involved: Read, Process, Write.

Instead of implementing a finite state machine to schedule the three operations, we have introduced two different clock signals to tick the Memory unit and the rest of the system, as shown in Fig. B.1.B. The processing clock was delayed for a quarter of the clock period. Using this dual clock system, we can maximise the data throughput and reduce the clock speed of registers and the RAM to save power.

B.1.1.2 Processing unit

This consists of three components: an absolute difference transformer, an updater and a compactor.

The absolute difference transformer is implemented as shown in Fig. B.1. C. RAM prepares the previous inputs of the current channel at the MCLK positive edge and two 10 bits registers are needed for storing current and last input. The adder calculates the outputs after the registers load the values from RAM at the positive edge of PCLK. RAM stores the outputs at the negative edge of MCLK.

The updater consists of a threshold updater, spike peak updater and max spike rate updater. They follow a similar structure as shown in Fig. B.1. D. Three 10 bits registers are required to store the values to be updated. The multiplexer can select the update values for these registers according to the control signal from CU. The updated values are calculated using multiplications/divisions operations and to avoid using such operations in hardware, we have replaced all multiplications/divisions with bit shifts.

B.1.1.3 Control Unit

A state transition diagram of this unit is shown in Fig. B.1. E. The program starts at the *Normal* state; when a spike is detected (when CMP in Fig. B.1. A issues a detect signal), the state transfers to the *Detection* state, in which the spike detection rate is updated. At the same time, a command will be issued to the spike peak updater to update the mean value of the spike peak. If the current spike rate fulfils the desired firing rate, the state will transit back to the *Normal* state. However, if the detection rate is outside of the acceptable range of the expected firing rate, the state will transit to the *Update* state for threshold and max spike rate updating according to the policies described in the main contain. When the update time is reached, the board will also go into the *Update* state and go back to *Normal*

Table B.1: Register bandwidth and descriptions

Register name	Width	Description
filter_1	10	Filter: The last input
filter_2	10	Filter: The input before last input
max_spike_rate	10	Updater: Max acceptable spike rate
spike_peak_mean	10	Updater: Current spike peak mean value
thr	10	Updater: Current threshold
spike_number	10	CU: Number of spikes has been detected in this duty cycle
detection	1	CU: Detection signal
spike_time_count	2	CU: Count during a spike time
update_status	1	CU: Update enable
update_command	3	CU: Update command to issue
update_count	13	CU: Count for updating
max	10	CU: Current maxima for spike peak

after finishing the update. The status control unit requires seven registers (40 bits in total) to record the current status to achieve the time-sharing architecture.

B.1.1.4 Memory unit

This consists of a RAM and an address counter.

The RAM unit is used to store the register values (90 bits in total) in PU and CU for different channels in order to resume in next duty cycle. These values are merged and split before writing into and reading from the RAM. The details of all registers are summarised in Table B.1. As the maximum data bandwidth of the bRAM is 16 bits, it requires six bRAM to ensure enough bandwidth. Therefore, six dual-port synchronous 256*16bit RAMs with positive read negative write clocks are used.

The address counter generates the address for RAM to read and write. It schedules the data I/O of the RAM to control the time-sharing task among different channels.

By storing the channel status into the RAM instead of preparing the registers for each channel separately, the logic cell usage can be highly reduced. This enables the implementation to be up-scaled to more channels in resource constrained targets.

B.1.2 Embedded implementation

The embedded implementation uses Freescale FRDM-KL05Z, a 32-bit ARM-cortex M0+ based MCU operating at 48 MHz. It has 32 KB flash and 4 KB RAM. One 12-bit ADC

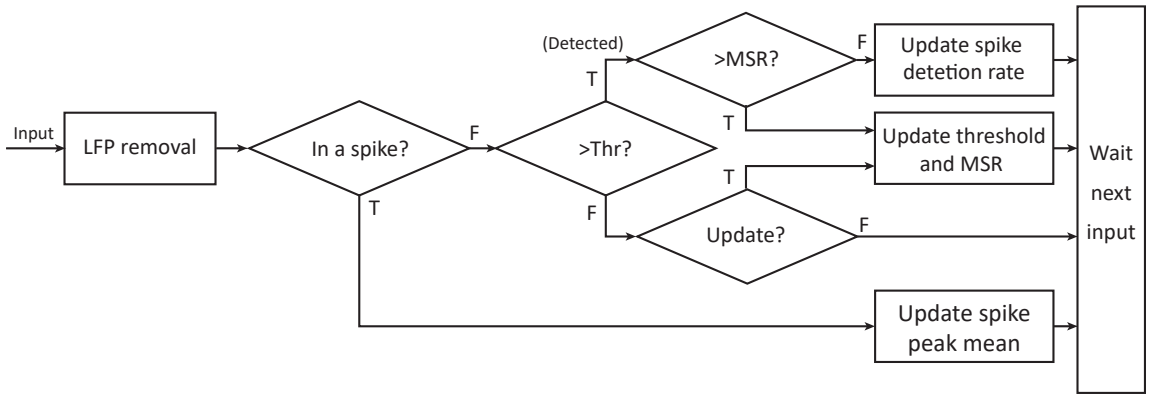

Figure B.2: A flowchart of the MCU implementation

and one 12-bit DAC are integrated onboard. The embedded implementation also uses a time-sharing architecture. A flowchart that describes the workflow of the embedded implementation is given in Fig. B.2.

The input sample is first processed by the absolute difference filter. The filtered value will be marked as a detected spike if it exceeds the current threshold and is above half the measured spike peak mean value. If a spike has already been detected, detection of this channel will be disabled for M samples to avoid re-detection, while the maximum amplitude within these samples will be used to update the spike peak mean value. M is sample rate dependent, and 5 @ 7 kHz were empirically found to be good values. The number of detections between the last update and the current time step will be recorded. If it is larger than the MSR, the threshold and MSR will be updated immediately. Otherwise, the processor will count steps until it reaches the time to update (1 s). The threshold and MSR will be updated according to the policies in the main contain. Multiplications/division of these updates are also achieved using bit shift. This process is duty-cycled and invoked at 896 kHz (128 channels, 7000 Hz each) by a ticker to process each channel interchangeably.

B.2 Results evaluation

In this section, we present the tested results demonstrating the operation of the algorithm and spike detection performance. We also report the measured hardware power consumption and resource occupation. These results show the intensive optimisation towards hardware efficiency we have made and the suitability of the proposed algorithm to be used in implantable BMI applications for real-time signal processing.

B.2.1 Spike detection performance

In order to test the proposed algorithm and compare it across different settings and algorithms, we implemented the single-channel version of it in MATLAB and tested it with a widely used synthetic dataset. We also evaluated its adaptiveness to different noise levels and its robustness to the setting of inaccurate initial parameters, showing that the proposed algorithm is able to practically adapt to different in-body environments.

B.2.1.1 Test dataset

The test dataset is from Quian Quiroga et al.. They have produced a synthetic dataset [45] that has been widely used in the literature [92, 97–99]. The dataset consists of 4 groups of signals (easy 1, 2; difficult 1, 2). The noise levels (the ratio between the noise standard deviation and the average amplitude) vary between 0.05 and 0.2 with an increment of 0.05. We selected 16 signal segments, all of 60 s duration and sampled at 24000 Hz, which we down-sampled to 7 kHz in our experiment as [96] showed is sufficient for spike detection. Each signal was assigned three different spike shapes with a mean firing rate of 20 Hz, where the inter-spike interval followed a Poisson distribution.

B.2.1.2 Detection performance and comparison

Sixteen pieces of the test signals were used for performance testing. A snapshot of the detection results is given in Fig. B.3. The initial spike peak mean was set as the maximum value in first one second, and initial threshold is set to be half of the initial spike peak mean and the initial max spike rate is set to be 50. One can notice that the threshold converges to a reasonable region within 1 or 2 updates, and that the tracked spike peak mean values also approach correct local spike levels despite initial offset.

This dataset has been used in various studies, and the algorithms in other studies tested all have sensitivities over 90%. We have chosen two of them as baseline methods for comparison for conciseness. The average performance is given in Table B.2. As shown in that table, we have achieved a sensitivity of 5% higher than the global thresholding in [45], even though the proposed algorithm is of limited precision and sampling frequency (Ours: fixed-point, ranging between ±300 @ 7 kHz; [45]: 32-bit float-point @ 24000Hz). Compared to our previous fixed-point implementation @ 7KHz in [92], its accuracy has been improved by nearly 10%.

The spike detection performance for different noise levels is given in Fig. B.4. One can notice that the proposed algorithm achieved the highest detection accuracy and sensitivity across all different noise levels. It is also capable of achieving nearly 95% sensitivity when the noise level is high, while keeping the FDR to around 5%.

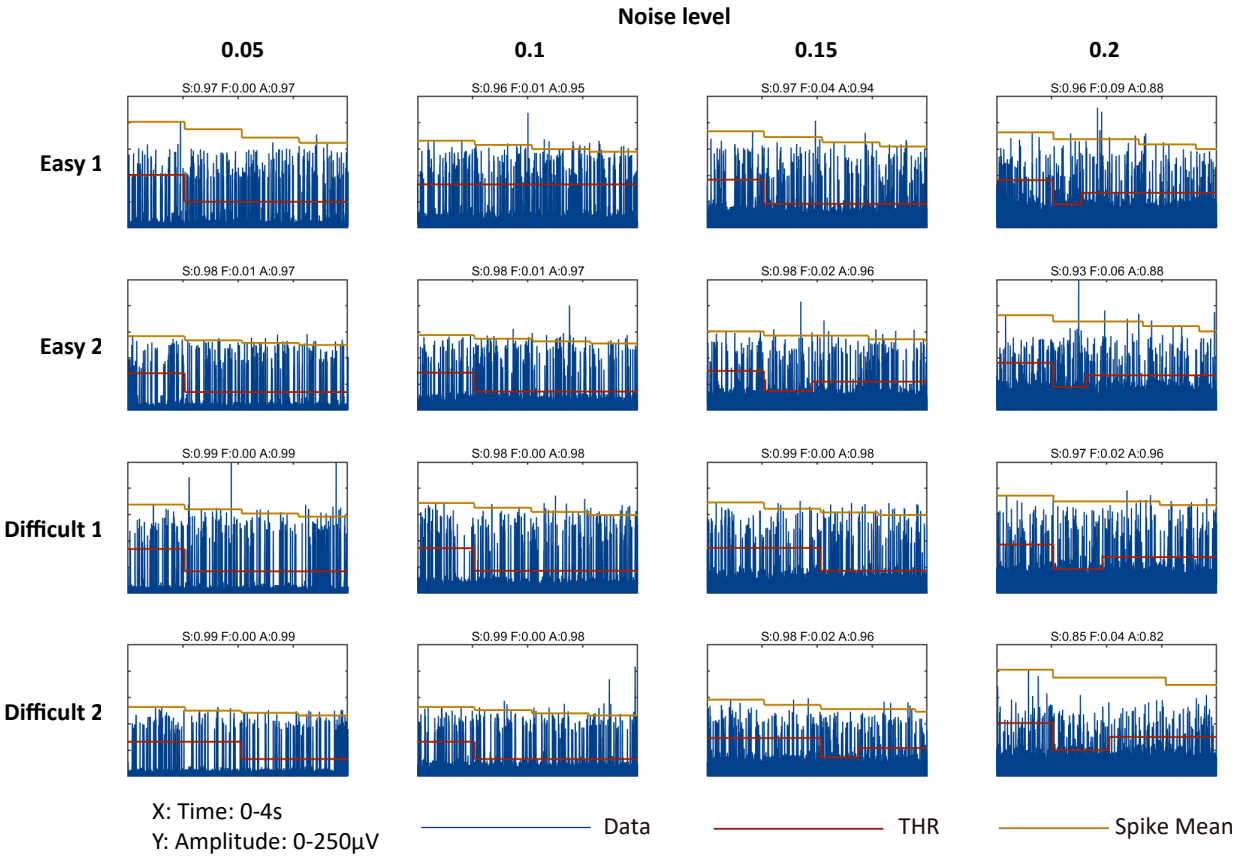

Figure B.3: A 4-second snapshot of the detection results with sixteen different signals. The threshold reaches a reasonable level within 1 or 2 updates. The estimated spike peak mean also gradually approaches the real spike peak mean. S: sensitivity, F: false detection rate, A: accuracy.

Our algorithm is unaffected by the fixed-point representation (with a less than 1% degradation in sensitivity). Conventional approaches derive the threshold from noise. They amplify the noise mean or standard deviation to a proper level for setting the threshold. One downside of this approach is that the limited precision of small numbers (noise) significantly affects the derived threshold value. A second is that the optimal threshold does not increase linearly with increases in noise levels. As a result, the conventional threshold can be suboptimal when the noise increases even though the same setting is optimal for a low noise level and vice versa. These two factors reduce the performance of the conventional approaches.

However, the threshold in our approach is guided by the spike peak values, which is less affected by the limited precision. Additionally, the threshold is not based on the noise level but aimed at detecting the spikes at a fixed rate. It is therefore not affected by the nonlinearity between the noise level and optimal threshold and is only affected by immense noise peaks that contaminate detected spike levels. In other words, the degradation in the low SNR condition of our algorithm is not from the inaccurate setting of the threshold but from the noisy signal itself, which is unavoidable.

Table B.2: The detection performance comparison among different algorithms. The results of our work show better sensitivity and accuracy.

	Acc	FDR	Sens
[45]	0.92	0.01	0.93
[92] Float-point	0.92	0.01	0.93
[92] Fixed-point	0.87	0.06	0.92
Our work Float-point	0.95	0.03	0.98
Our work Fixed-point	0.95	0.03	0.97

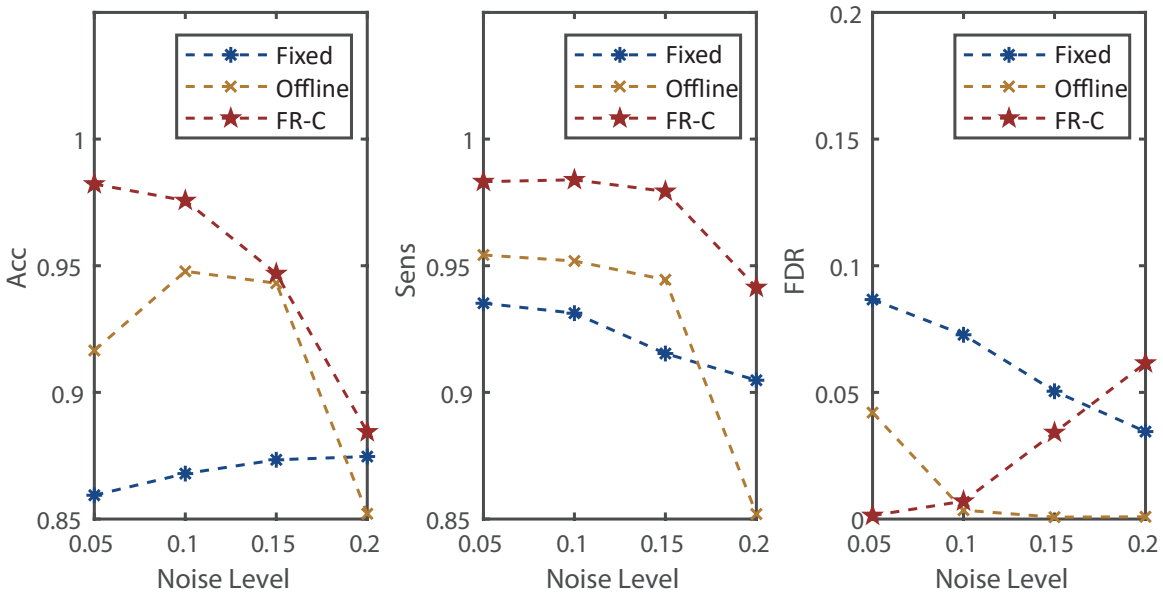

Figure B.4: The detection accuracy, sensitivity and false detection rate in the different noise levels of the proposed algorithm in comparison with the other two baseline algorithms. The plot of [92] is its fixed-point implementation

Table B.3: The algorithm detection performance for different settings. The first three rows are the results tested on the dataset described in Section B.2.1.1 and the last three rows are the results testing on the dataset with an extra LFP signal added.

	Acc	FDR	Sens
Improved algorithm	0.95	0.03	0.97
No absolute difference filter	0.85	0.08	0.91
No spike peak tracking	0.71	0.27	0.98
Absolute difference filter	0.94	0.03	0.97
Mean subtraction filter	0.91	0.03	0.93
2-order butterworth filter	0.81	0.13	0.91

B.2.1.3 Effectiveness of different operations in the proposed algorithm

A summary of how different operations influence the proposed algorithm detection performance is given in Table B.3. Clearly, the algorithm has been improved from a low complexity algorithm with only acceptable detection performance to an algorithm that achieves both high performance and low complexity. Analysis of different operations is given below:

a) The absolute difference filter can remove the LFP and enhance the spikes with only one subtraction operation. Though this operation highly distorts the spike shapes, it improves the detection accuracy by nearly 10%, which shows its capability of emphasising spikes. Since the test dataset does not include LFPs, we extracted a piece of LFP from a real recording and added it into our test dataset in order to demonstrate the algorithm's ability to remove large LFPs. Relative to the conventional float-point second-order IIR filter in [96] or mean subtraction filter in Section.3, the absolute difference filter improves the detection accuracy by 13% and 3% respectively. Such improvement is significant when considering that it only uses one subtraction in fixed-point representation. Based on the results discussed above, this LFP removal method can be regarded as one of the most efficient action potential signal pre-processors for the spike detection algorithm.

b) Spike peak mean tracking has been used to overcome the three conditions mentioned in Section 5.3.1. As seen from Table B.3, the FDR has been improved significantly (24%), while the sensitivity has only been marginally negatively affected (1% less). Fig. 5.2 (a) and (b) Con. 1 show that the number of falsely detected spikes has been reduced after comparison with the estimated noise floor, i.e. half the spike peak mean. Fig. 5.2 (b) Con. 2 also shows that when the real spike rate is lower than the pre-set max spike rate, our algorithm prevents the threshold from decreasing to the noise floor, which would lead to massive falsely detected spikes. Fig. 5.2 (c) Con. 3 shows that when the real spike rate is higher than the pre-set max spike rate, the algorithm prevents the threshold from increasing to close to the spike peak

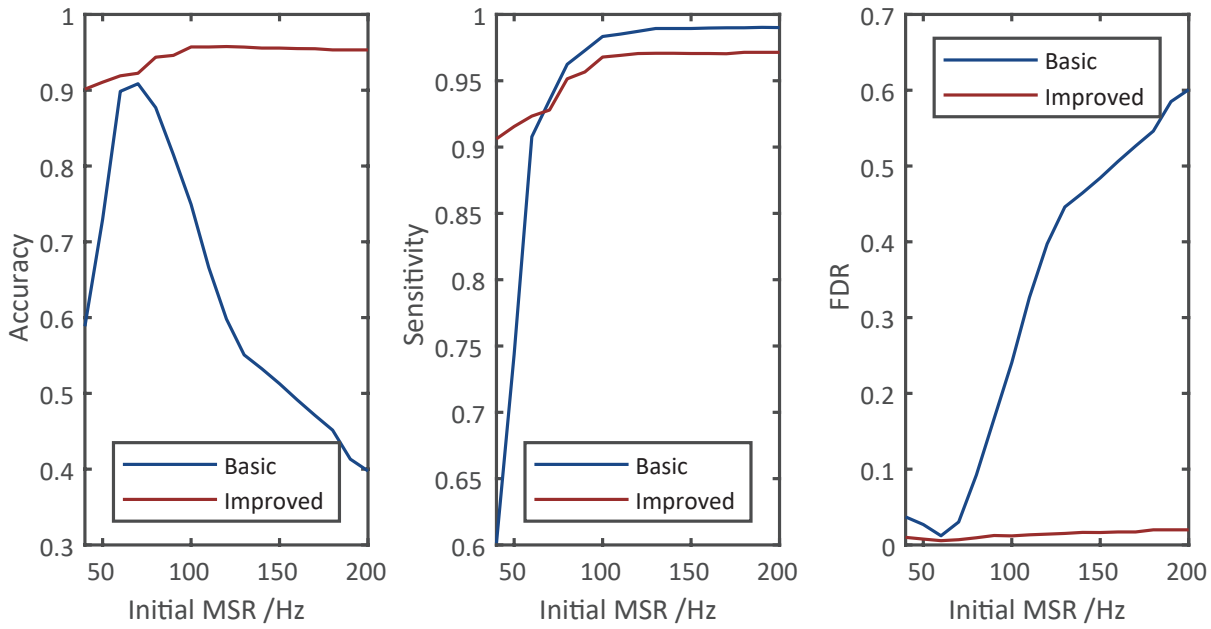

Figure B.5: The detection performance of the basic and improved algorithm for different initial max spike rate settings. The basic version is the detection algorithm without spike mean tracking and adaptive max spike rate

level, which avoids a decrease in missed true spikes. It should be noted that the recordings used in these figures are real recordings [44] and the labels are from [131]. This dataset is different from the data used elsewhere in this thesis. The usefulness of the spike peak mean tracking is easier to observe in these recordings, and we therefore posted these results to make it easier for readers to understand the algorithm.

Furthermore, the algorithm is also robust to inaccurate settings of the initial max spike rate. We tested the initial max spike rate from 40 to 200 Hz for the implementations with and without spike mean tracking, and the results are shown in Fig. B.5. The basic version can only get satisfying detection accuracy when the initial MSR is accurately set. The improved version is however more robust to the initial MSR. Because the spike peak levels are known, we can guarantee that the threshold is set to within a reasonable range. Taken together, the proposed algorithm can be regarded as having two modes. Mode I: when the MSR is inappropriate and the threshold tends to be unreliable, the threshold can still stay within the reliable region, and the MSR will update until it is also appropriate. Mode II: when the MSR is appropriate, the threshold will converge to a level so as to detect the spikes at the desired rate. These two modes operate in tandem and guarantee high performance.

c) There are however three weaknesses.

The first is that the difference filter can only remove LFPs that have a scale similar to the EAP signals. If the LFPs can vary as dramatically as the spikes, the resultant signal can be too noisy to be processed. However, this is generally not the case in practice, as LFP

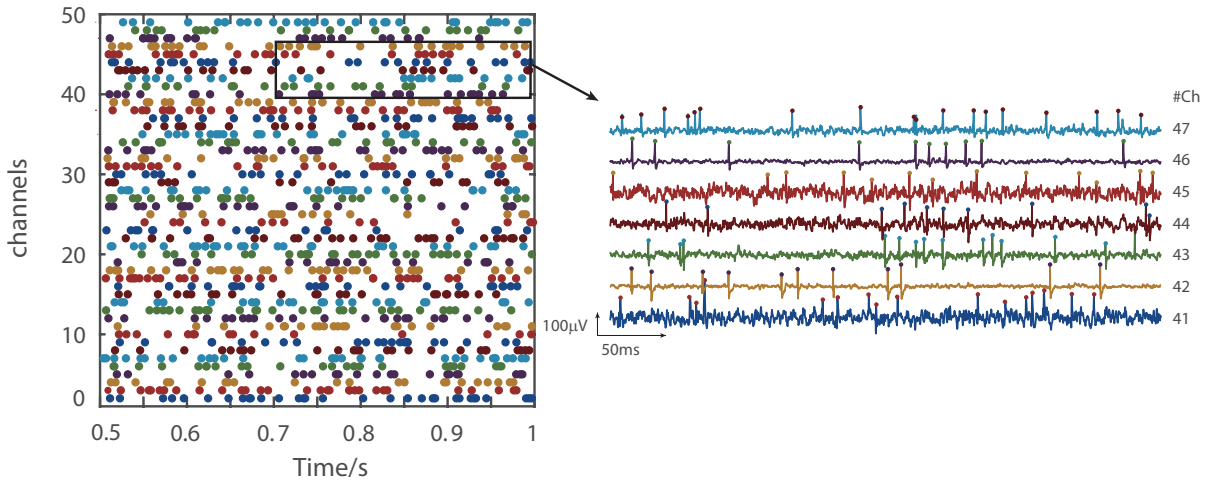

Figure B.6: The raster plot of the multichannel detection results in hardware. This is plotted after streaming the detection results from the hardware to the computer.

leakage past the front-end filters would not be sufficient to make this filter ineffective.

The second is that, as the spike mean value is updated according to the detected spikes, positive feedback can be introduced as successful detection is not always guaranteed. Specifically, false detection could lower the tracked spike mean value, leading to an even lower threshold. This fault could then even increase the false detection rate. However, this special case can only happen when the target signal is noisy, where many of the noise peaks are distributed around, or above half the value of, the true spike peaks. Based on our experience, the algorithm performance recovers suddenly when the noise reduces, and it never occurred in the test dataset we used here.

The third is that we have not introduced an optimal mechanism for updating the max spike rate when it is inaccurate. It may take some time to re-calibrate itself if the actual spike rate changes dramatically or the pre-set value deviates too much. However, the threshold is limited within a reasonable threshold region, and the detection accuracy can be guaranteed even if the MSR is inaccurate.

B.2.2 Real-time testing and hardware efficiency evaluation

To test the real-time suitableness and multichannel capability of the proposed algorithm, we have partitioned and distributed the test signal into 128 channels. This test signal is identical to the one used for detection performance evaluation and the distribution is random. We then stored it in a data RAM for the FPGA or gave it as input to the MCU using a waveform generator and the onboard ADC. A detected spike raster plot is given in Fig. B.6, and the detection performance was validated and found to be consistent with the results from the MATLAB tests.

Table B.4: The voltage across the resistor, core current supply, power consumption and average power consumption per channel of the FPGA implementation. Total is the normal algorithm running status, Static + input read is only when operating input memory read and write and spike detection is the difference between the former two, which is the extra dynamic power consumption from the spike detection algorithm circuits.

	Total	Static + input read	Spike detection
Voltage (μV)	312	268	44
Current (μA)	31.2	26.8	4.4
Power (μW)	37.4	32.2	5.2
power/ch (μW)	0.29	0.25	0.04

Conventional neural signal processing schemes can be computationally expensive and memory intensive. It is crucial to partition the processing flow among on-implant, on-node and off-node locations. A well-ordered workload distribution can balance the resource pressure and eliminate the power bottleneck, leading to maximised lifetime. The power consumption and resource occupation can therefore be an essential factor for deciding the process flow distribution. The power consumption and resource/memory occupation of the two targets (Lattice ice40LP1K FPGA and FRDM-KL05Z MCU) were measured, demonstrating the hardware-friendly nature of the proposed algorithm.

B.2.2.1 Power consumption and resource occupation of the FPGA implementation

The FPGA core voltage is 1.2 V and the sample frequency is designed to be at 7 kHz for each channel. The memory and processing clocks are therefore 0.896 MHz for a 128 channel system and the system clock is 1.792 MHz. Its power consumption is given in Table B.4. The spike detection dynamic power is obtained by subtracting the FPGA running power with the FPGA running power when only implementing the data RAM (which stores the input data). Only 5.2 μW power is consumed for a 128-channel spike detection algorithm, while the static power is 30 μW. The dynamic power is therefore an order of magnitude lower than the static power. This means that the dynamic power of our algorithm will only dominate when the channel count is in the thousands or more.

Spike detection starts at the posedge of the memory clock for loading current channel status from RAM to registers and ends at the next negedge of the memory clock for saving the current channel status. At 0.896 MHz this procedure takes about 0.56 us, and therefore only 2.9 pJ is consumed for one sample. The resource utilisation is given in Table B.5. Only 299 logic cells (2.33 per channel) are used because of the resource sharing schema. In the

Table B.5: FPGA resource utilisation of the proposed spike detection algorithm. Total is the total utilisation of the implementation, input read is the resources used by the data read RAM and the spike detection is the actual resource occupation of the algorithm.

	Total	Input Read	spike detection	Available	Utilisation rate
Logic cells	406	107	299	1280	23.3%
PLBs	66	20	46	160	28.8%
BRAMs	6	0	6	16	37.5%
I/Os	8	5	3	10	30.0%

Table B.6: The voltage across the resistor, core current supply, power consumption and average power consumption per channel of the MCU implementation. Running is the normal running status, Idle is executing empty operation, idle+ADC is when ADC samples at 0.896 MHz while running empty operation and spike detection is the difference between running and idle+ADC, which is the extra power consumption from the spike detection algorithm.

	Running	Idle	Idle+ADC	Spike detection
Voltage(mV)	72.2	57	60	12.2
Current(mA)	7.22	5.7	6	1.22
Power(mW)	23.83	18.81	19.8	4.03
power/ch(mW)	0.18	0.15	0.15	0.03

meantime, the RAM bandwidth is 90 bits, which only occupies six 16-bit block RAM.

B.2.2.2 Power consumption and resources occupation of MCU implementation

The MCU core voltage is 3.3 V, working at a 48 MHz clock frequency. When implemented with the fully functional 128 channel spike detection algorithm, the core supply current is 7.22 mA leading to 23.83 mW total power. As the ADC is used during the signal processing, to extract solely the spike detection power we ran a test where the ADC sampled at 0.896 MHz with the empty operation resulting in a current supply of 6 mA, which consumes 19.8 mW of power. The power consumed by the spike detection algorithm can therefore be as low as 4.03 mW and 31.5 μW per channel. The idle power can be regarded as the static power of the MCU. A summary of the power consumption is given in Table B.6. We have also implemented the basic implementation without spike peak mean tracking; the change in power consumption is invisible to the oscilloscope. Because of the fixed-point representation and the use of the simple absolute difference filter, we are able to run spike detection with

only 3KB RAM (4KB available onboard). On average, each channel only needs 24 Bytes.

B.2.2.3 Comparison to state-of-the-art

A comparison of our FPGA design and some other FPGA or ASIC designs is given in Table B.8 and B.7. We have reduced the power consumption by 10 or even 100 times. The dramatic power reduction is due to the simplicity of the hardware implementation. In [137], they used 16030 LUTs and FF for a 1024 channel spike detection system, while we only used 299 logic cells for a 128 channel system. On average, the implementation in [137] consumes 15.6 logic cells for each channel. By sharing the processing logic among channels, we required only 2.3 logic cells for each channel on average. In addition, they used an additional 17 DSP units that we do not have, and these units are power-hungry.

Table B.7: The FPGA resource utilisation comparison across three different algorithms

	[89]	[138]	Our work
FPGA Platform	Xilinx ZCU106	ZenyQ-700	Lattice ice40LP1K
Channels	128	1024	128
Logic cells	53607	16030	299
Logic cells/ch	419	16	2.9
BRAMs	7	3	6
DSPs	6	17	0

Table B.8: Comparison with other FPGA and ASIC hardware implementations.

	[106]	[139]	[140]	[141]	[137]	Our work
Detector	Absolute	Biphasic	EC-PC	MNEO	CA	ADT
Technology (µm)	0.6	0.5	0.13	0.065	FPGA	FPGA(0.04)
Sampling rate (kHz)	22	7.8	40	20	9.375	7
Bit/Sample	5	8	16	10	16	10
Channels	32	64	16	64	1024	128
Power (µW)	110	14400	1360	80	48000	37.4/5.2†
Power/Ch (µW)	3.44	225	85	1.25	46.8	0.29/0.04†

† Dynamic Power

Our MCU implementation is one of the most power-saving implementations among the literature we have seen. At the same time, the memory consumption has also been minimised to nearly as low a level as we can expect. A 16-point mean subtraction filter in [92] required

Table B.9: Comparison with other MCU implementations.

	[118]	[109]	[116]	[92]	Our work
Method	Absolute	Absolute	Static	ASO*	ADF†
Hardware	MSP430F	MSP430F	KL25Z	KL05Z	KL05Z
Sampling rate (kHz)	-	20	-	7	7
Channels	2	4	32	1	128
power (mW)	4.2	1.67	7.2	0.13	4.03
power/ch (mW)	2.1	0.4175	0.255	0.13	0.03

* Amplitude slope operator

† Absolute difference filter

38 Bytes, but we have used no more than 24 Byte/Ch for the whole spike detection algorithm while improving the performance by 4%.

The comparisons demonstrate the intensive optimisation we have made toward power efficiency and resource utilisation. More importantly, a 10-100 times reduction in power is significant because the power consumption bottleneck of other algorithms is their processing power, while the bottleneck in our system is the hardware itself.

Appendix C

Event-driven firing rate compression

C.1 Methods

All compression algorithm work was done in MATLAB R2021A. All hardware design and optimisation work was done using ModelSim Lattic Edition and IceCube2020.12.

C.1.1 Dataset and data formatting

Brain signal conditions can vary across subjects and tasks. In order to reduced the bias, we have used three different publicly available datasets [43, 142, 143], summarised in Table. C.1. For each dataset, the SUA data was intra-channel collated to MUA, then binned to the desired BP, where BP $\in [1, 5, 10, 20, 50, 100]$ ms. Based on the results in [135], we limited S to [2, 2, 2, 2, 3, 5] for BPs of [1, 5, 10, 20, 50, 100] ms respectively. The Supplemental Material includes results from S values $\in [\mathbb{Z}^+, 2, 3, ..., 14, 15]$.

The length of the recordings was largely irrelevant for the sake of this work, as all encoding was done without using a time derivative and the bandwidth was measured in [bps/channel], normalising for time. As such, a standard length of 100 s was set for each recording, long enough to gather a stable distribution for each channel for any of the tested BPs. To maximize the use of the data in observing the effect of channel count, recordings were split into consecutive 100 s segments and collated together. For example, a 400 s recording was represented as 4 parallel channels of 100 s long recordings, where x_o,...,$x_{N/4-1}$ became one channel, $x_{N/4}$,...,$x_{N/2-1}$ became the next, etc., where x_k is a single-channel MUA recording and $N = 400\,s/\text{BP}$ is the length of the recording in samples. A total of 79200 channels were available after splitting and collation.

The contents in this appendix has been submitted to Scientific Report, with my full contribution to the hardware design and joint effort on algorithm design and result analysis with Dr. Oscar Savolainen.

Table C.1: Dataset summaries.

Dataset	Neural data type	Species, electrode type and brain region	Details	Behaviour
Flint [142]	SUA	Rhesus macaque monkey Utah array M1	One subject 12 recordings across 5 days 96 channels Recording lengths (quartiles, s): 597, 604, 630	Free-reaching hand task Continuous data stored
Sabes [43]	SUA	Rhesus macaque monkeys Utah array M1 and S1	Subjects Indy and Loco 37 recordings for Indy across 10 months 10 recordings for Loco across a month 96-192 channels Recording lengths (quartiles, s): Indy: 472, 524, 816 Loco: 1771, 1928, 2384	Free-reaching hand task Continuous data stored
Brochier [143]	SUA	Rhesus macaque monkeys Utah array M1 and PMv/PMd	Subjects N and L Single session recordings 96 channels Recording lengths (s): N: 1003 L: 709	Hand reaching task Target stored

The data was then split into training and testing sets. This is because the encodings have parameters that need to be optimised, which was done on the training set. The testing of parameter-optimised encoding on the testing set can then provide an objective estimation of the encoding performance on the unseen data. The training-testing split was done by randomly selecting 30000 channels and placing them into the training set. The remaining 49200 were put into the testing set. For each encoding, BP, S and n combination, the training and testing results were each averaged across 5 training and testing runs. For each run, $n \in [10, 100, 1000, 10000, 30000]$ channels were selected randomly without replacement from amongst all training or testing channels.

C.1.2 Windowed encoding

The windowed encoding [135] was the first scheme to be implemented. It is called "windowed" because it sends out the number of MUA events in a non-overlapping window of length BP for each channel, regardless of whether an event occurs on a channel or not. It consists of representing the multi-channel MUA data as multiplexed data. Without lossless compression, the length of the data block is $n \times m$, where n is the number of channels and m

is the number of bits used to represent the MUA FR per BP on each channel. An example using a standard binary representation with m and $n = 3$:

$$001\ 111\ 000$$

would indicate that 1 neuron fired (001) on channel 1, 7 neuron firings occurred (111) on channel 2, and 0 (000) on channel 3. An advantage is that the channel ID is implicitly encoded in bit position, and so does not need to be explicitly encoded. E.g.,

$$c = ceil(t/m) \tag{C.1}$$

where $c \in [\mathbb{Z},\ 1 \leq c \leq n]$ is the channel ID, $t \in [\mathbb{Z},\ 1 \leq t \leq n \times m]$ is the bit position and *ceil* is the ceiling function. m is the number of bits required to represent all possible MUA FRs losslessly, and is generally set as $ceil(log_2(\texttt{max}(X) + 1))$, where X is the multi-channel MUA data. However, one can lossily compress the data by limiting the dynamic range at an S value, setting $X[X > (S-1)] \leftarrow (S-1)$. It requires

$$m' = ceil(log_2(S)) \tag{C.2}$$

bits to represent a range of 0 to $S-1$ MUA FRs. If $m' < m = ceil(log_2(\texttt{max}(X) + 1))$, this lossily reduces the required bandwidth. As such, the windowed encoding typically has a BR of:

$$\text{BR} = \frac{m'}{\text{BP}} \qquad \text{[bps/channel]} \tag{C.3}$$

However, as in [135] we integrated the windowed encoding with a SH encoder that compressed the FRs. The SH encoder was pre-trained on a decaying exponential which mimics the distribution of MUA FRs at BPs ≤ 100 ms [135], where smaller codewords are given to smaller FRs.

C.1.3 Explicit event-driven encoding

In the windowed architecture, the FR of each channel is encoded. The channel ID is implicitly encoded in bit position. To the best of the authors' knowledge, the following event-driven architectures are proposed for the first time in MUA compression. In these event-driven architectures, the channel ID is explicitly encoded and only active channels will be transmitted, i.e. the channel ID is only sent out if a non-zero FR occurs on that channel, followed by a binary codeword representing the rate. If a channel has a FR of 0, nothing gets communicated for that channel, and the offline decoder assumes the missing channels had FRs of 0. For sparse signals, i.e. where FRs not equal to 0 are rare, this can offer reduced bandwidth over the windowed paradigm, where even FRs of 0 need to be communicated for each channel.

As the channel IDs need to be explicitly encoded, the simplest method is to give the channels a standard binary codeword of length k_1, where:

$$k_1 = ceil(log_2(n)) \quad \text{[bits]} \quad \text{(C.4)}$$

Similarly, the MUA FR for each channel is encoded as a binary codeword of length m_2:

$$m_2 = ceil(log_2(S-1)) \quad \text{[bits]} \quad \text{(C.5)}$$

An exception occurs if $S = 2$, i.e. $m_2 = 1$ and i is limited to 0 and 1. In that case the firing rate codeword is unnecessary as the decoder can assume all received channel has the firing rate of 1. An example of the encoding without lossless compression, with $n = 4$, $k_1 = 2$ and $m_2 = 2$, is given by:

0001 0100 1011

This shows that channel 1 (00) had a FR of 2 events (01) in the bin, channel 2 (01) had FR = 1 (00), channel 3 (10) had FR = 4 (11), and channel 4 had a FR = 0 (absent). In this work, this encoding was named the Explicit Event-Driven (EED) encoding because the FR per channel is explicitly encoded in the m_2-length codeword that follows the channel ID.

To include the SH encoder, there was no good way to compress the channel IDs, since they are *a priori* equally likely to be active. Therefore, the EED encoding uses the same k_1 length codeword for the channel IDs, but uses varying length SH codewords for the FRs. As in the SH windowed implementation, the FR encoder was trained on a decaying exponential.

C.1.4 Delta event-driven encoding

The previous encoding suffers since the probability of each channel being active is *a priori* equal. As such, the channel IDs cannot be compressed as they are with SH encoders. However, the channel difference between two successive active channels has varying probability, and therefore Huffman encoding can be used. In other words, we can encode the delta-sampled channel IDs of active channels. Therefore, in this Delta Event-Driven (DED) encoding, if a channel has a FR above 0, it is given a Δ value by subtracting its ID, $j_{current}$, from the ID of the previous channel to have a FR above 0, $j_{previous}$. I.e., $\Delta = j_{current} - j_{previous}$.

Therefore, the FRs were compressed with the same SH encoder as in the EED encoding. However, the Δ-sampled channel IDs were compressed using a SH encoder trained on a decaying exponential. Significant memory optimisation was also done by setting a maximum Δ-sampled SH encoder size, using a form of run-length encoding. This reduced BR slightly but significantly reduced memory requirements. The details of this optimisation are extensively detailed in Sections 3 and 4 in the Supplemental Material.

C.1.5 Group event-driven encoding

In the Group Event-Driven (GED) encoding, one uses position and stop symbols to encode the FRs. As in the EED encoding, one explicitly encodes the channel ID, but here one encodes the FR per channel implicitly in channel ID position. For example, in decimal,

$$2\,4\,stop_2\,1\,6\,stop_3\,stop_4\,3$$

signifies that channels 2 and 4 had a FR of $i = 1$ in the given bin, channels 1 and 6 had a FR of $i = 2$, channel 3 had FR $i = 4$, and the rest of the channels had FR $i = 0$. For the codeword lengths, the simplest implementation is to give the stop symbols and the channel IDs a length of k_2 bits, where:

$$k_2 = ceil(log_2(n + S - 2)) \quad \text{[bits]} \quad \text{(C.6)}$$

E.g. $n = 2$, $S = 4$, means that there are 2 channels, FRs between 0 and 3 inclusive can be encoded, and $k_2 = 2$. Channel 1 gets a codeword of 00, channel 2 gets a codeword 01, and the stop symbols for $i = 2$ and $i = 3$ get codewords of 10 and 11 respectively. It essentially involves sorting the channels by FR, and using a form of run-length encoding.

There was no clear role for SH encoding, since the stop symbols and FR codewords need to come from the same encoder. Estimating these probabilities *a priori* is difficult. As such, no SH encoding for the group encoding was used.

C.2 Hardware implementation

The AH implementations require us to generate the Huffman codebook on implants, and assume perfect knowledge of the data to be compressed, which is not achievable in practice. Therefore, no hardware implementation was done for the AH encodings.

The windowed, explicit event-driven and delta-event-driven implementations share a similar hardware architecture. The SH delta-event-driven encoding is considered as the 'full version' architecture, while the explicit event-driven and windowed architectures are pruned versions. As such, for conciseness, we introduce the implementation of the delta-event-driven encoding and describe how the other architectures can be derived from these.

C.2.1 Delta-event-driven encoding

This compression architecture consists of 7 components. Four of them have been described in detail in previous work [135]. These four are the binner, histogram, mapper and encoder. As such, we only give a brief description of them.

- The binner is a counter that counts the number of detected spikes (FR) at the given BP.
- The histogram accumulates the frequency of different FRs to identify the most common FR for each channel. This is required for sorting and mapping.
- The mapper is the module responsible for assigning each FR to its codeword. The mapper maps the most frequent FR, as determined in the histogram, to the shortest codeword and maps the other FRs accordingly with a pre-defined combinatorial logic that defined the sorting. As detailed in Section 2.6 of the main manuscript, this mapping aligns the real-time FR distribution with the distribution that the Huffman encoder is trained on, which helps maximise the on-implant compression performance.
- The encoder is a LUT that encodes the mapped FR into a Huffman codeword.

In order to achieve the event-driven encoding and delta-event-driven encoding, two more components have been designed. The first component is the comparator. This compares the FR to be compressed with the most frequent FR as measured by the histogram. If no histogram/sorter/mapper is used, then the most common FR is taken to be 0. The comparator ensures that only the not-most-common FRs are sent out. The second component is responsible for the channel delta-sampling. It calculates the difference of the current channel to-be-transmitted to the last transmitted channel. ROM is used to store the original channel ID codeword instead of using a LUT implementation. This saves on logic cell resources.

For implementing multi-channel compression with minimum resource usage, we follow the time-sharing architecture as in [11,135]. All channels share the same processing components, and the signals of each channel are processed interchangeably. RAM is used to store the variables for each channel while the processing circuits are busy with another channel.

C.2.2 Windowed encoding

The basic implementation of windowed encoding transmits the binned raw FR of all channels with no further operation. Its SH implementation compresses the binned FR with a pre-trained Huffman encoder. It also has the option to use a histogram and mapper to enhance the compression performance.

C.2.3 Explicit event-driven encoding

The basic implementation of the explicit event-driven encoding uses the binner and comparator for obtaining non-zero FR and channel number for transmission. The SH implementation, similar to the windowed one, makes use of Huffman encoder to compress the FR. As with

the windowed encoding, a histogram and mapper can be used to better than chances that the most common FRs are given the shortest codewords.

C.2.4 Group event-driven encoding

The group event-driven encoding is different from the other three. The high-level summary is that the FR from each channel is read. The channel IDs with the same FR are then grouped. Once every channel has been read, the groupings/concatenations of channel IDs are sorted in order of FR with interleaved stop symbols to create the final bit stream.

In hardware, this consists of a sorter and a package generator. The sorter counts the FR in each channel continuously. Meanwhile, the multi-channel FRs are sorted in descending order. The package generator generates the bit stream based on the sorted FRs according to the policy described as section 1.4.

Sorting can be resource and time consuming, especially in hardware. Here it is implemented using a Finite State Machine (FSM). We sort the bin counts (FRs) in real time as they accumulate. Each time a spike is detected, only one channel's FR changes and it only increases by one. Swapping this FR with the furthest value in front which is smaller than it keeps the sorted FRs in order. To ensure that the FR and channel number are trackable after swapping, two arrays of registers are needed. The first stores the channel numbers of the sorted entry. The second stores the index number of each channel after swapping. Therefore, the temporal and spatial cost increases linearly with the channel number, i.e. $O(n)$, where n is the number of channels. This low complexity sorting is achieved by taking advantage of the nature of the binning where only one entry is increased by one at each spike detection. An example is give in fig.C.1, illustrating the sorting algorithm.

The package generator then scans the sorted FR while neglecting any zeros in the tails. The channel numbers are packaged/concatenated and the stop symbols are inserted whenever a new FR is scanned in the sorted FR sequence.

The group event-driven encoding however has more hardware complexity than the other three encodings. This is because the channel IDs must be placed in their correct location in the bit stream. This is the case even for the basic implementation. This is a significant hardware cost. Additionally, this sorted collation occurs at every BP and so adds a processing power burden, especially at larger channel counts as the channel ID codewords are longer. As such, although the group event-driven encoding generally outperformed the explicit event-driven encoding in terms of compression, its added processing power and hardware complexity may make it less practical as an encoding. It is however interesting that the entropic and basic implementations gave similar BRs, suggesting that the basic implementation gives almost ideal compression for the group event-driven encoding.

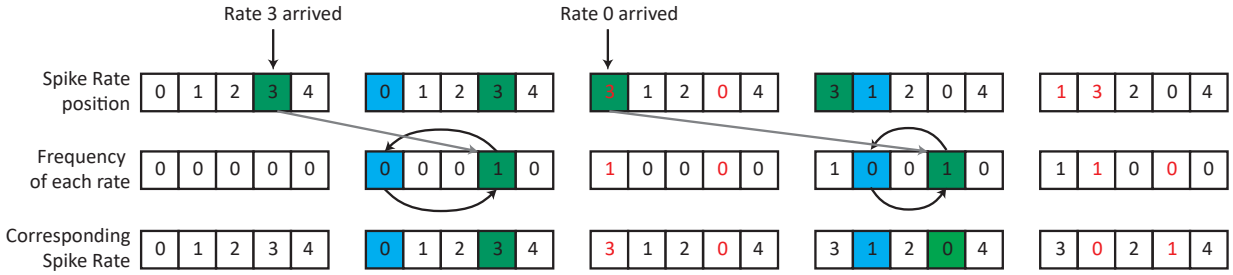

Figure C.1: An example of the sorting algorithm. Fequency of each channel: sorted spike rate of each channel. Corresponding channel number: the channel number corresponding to frequency above. Channel position: the index of the channel corresponding to spike rate of each channel array. The workflow goes as below: A spike is first detected in channel 3, Channel 3 spike rate then increases by one. The number to be swapped for sorting is the furthest smaller rate left to the rate just increased (the green 1 in B), which is the 0 in blue. Swapping then happens in all three arrays. Another spike is detection in Channel 0. According to array A, the spike rate of channel 0 is stored in index 3 of array B. The number to be swapped is the most left 0 in B and all corresponding values in A, B and C are swapped then.

C.3 Results

In Fig. C.2, we plotted the BRs of each encoding at different BPs and n. We can observe that the windowed and DED encodings perform best, with the best one depending on BP and n. From the hardware perspective, the windowed encoding is far more hardware efficient than the DED encoding. Whilst the amount of logic cells is similar for the two encodings, the DED encoding uses significantly more memory, and therefore the processing power is higher. Additionally, the windowed scheme performs the same independent of channel count. However, the DED scheme is affected by channel count. When n increases, the delta-sampled channel IDs can be larger, meaning longer codewords and higher BRs. As such, increasing n slightly increase the DED BR, but significantly less than for the EED and GED encodings.

As such, the optimal selection generally varies as a function of channel count and BP. Ultimately, we want the total power on-implant to be reduced. As such, Fig. C.3 shows the total dynamic power for the FPGA implementation, made up of the processing power for each encoding and the communication power, estimated as the BR multiplied by an estimated 20 nJ/bit communication energy.

C.3.1 Optimal encoding delection

From Fig. C.3 and our analysis of the hardware costs in the Supplemental Material, for each BP and n we selected the optimal compression system. These are given in Table C.2 (a). We then determined each selected system's performance on the test data, i.e. data that had hereto been untouched, and the BRs are given in (b). The associated required number of

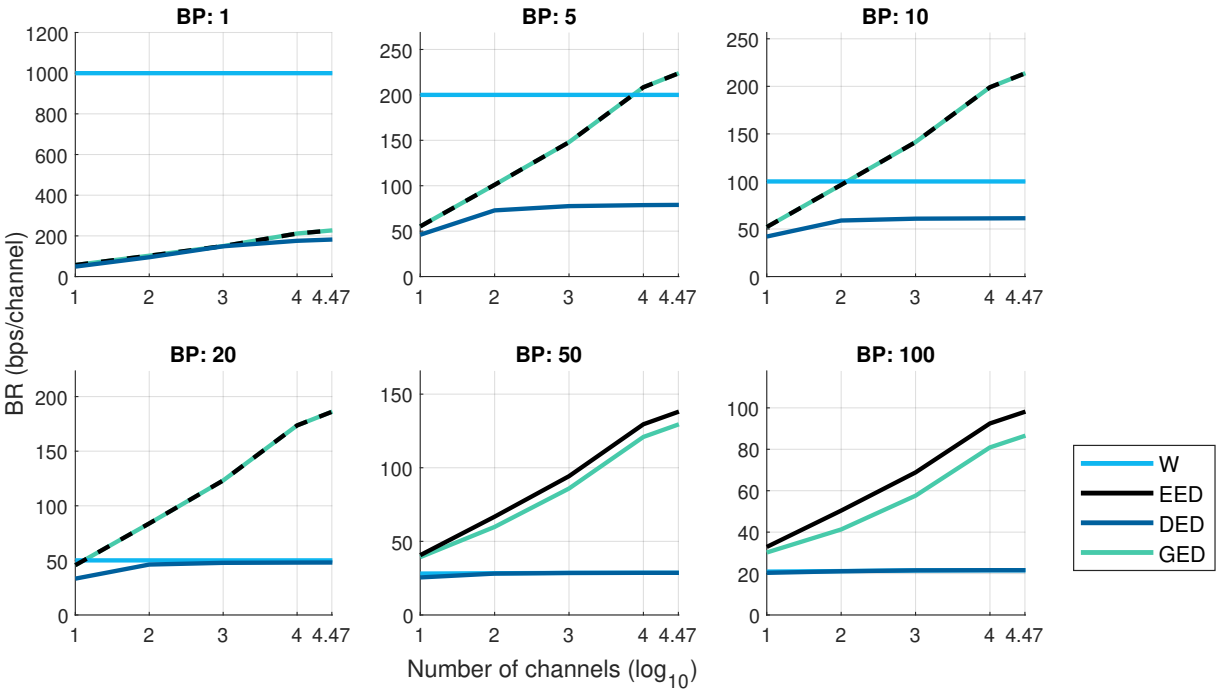

Figure C.2: Bit Rates for communication schemes at a BP $\in$ [1, 5, 10, 20, 50, 100] ms. For each BP, S was respectively fixed as [2, 2, 2, 2, 3, 5]. At $S = 2$, the explicit and GED encodings are mathematically identical, and so their BRs overlap for BPs $\leq$ 20 ms.

FPGA logic cells and memory are given respectively in Table C.2 (c) and (d). The total dynamic power on the Lattice ice40LP FPGA target, determined on the testing data, is given in (e). In summary, the DED encoding is suitable for short BPs (less than approx. 20 ms) while the windowed encoding is preferred when the channel count or BP is increased.

C.3.2 Impact of compression on channel counts in FPGA target

From this, knowing our BRs per BP and channel count, we can derive how many channels can be hosted on-implant, for different FPGA board dimensions and BPs. We assume:

- A 10 mW/cm^2 heat flux limit;
- An FPGA static power of 162 μW;
- A separate power and hardware budget for the front-end amplifiers, filters and ADCs;
- A 20 nJ/bit communication energy;
- A binner processing power of 0.96 μW, independent of BP [135].

Using the information from Table C.2 (e), we can derive how many channels can be hosted on-implant while staying within implant power limits. This is compared to the number of

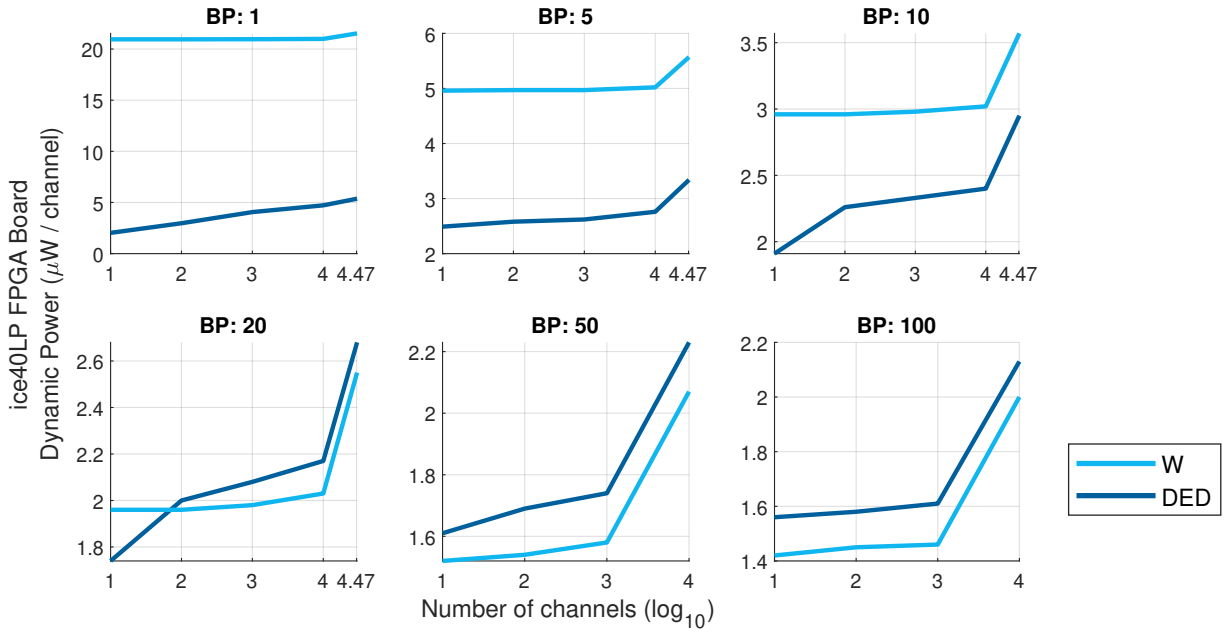

Figure C.3: Total communication + processing power, i.e. dynamic power, for the windowed and DED compression schemes at a BP $\in$ [1, 5, 10, 20, 50, 100] ms. For each BP, S was respectively fixed as [2, 2, 2, 2, 3, 5]. For BPs of 50 and 100 ms, the power is not given for 10000 and 30000 channels since the memory requirements exceeded the FPGA target memory budget.

channels that can be hosted given the standard uncompressed MUA representation at 1 ms BP, $S = 2$, that has a 1000 bps/channel BR.

The results are shown graphically in Fig. C.4. It can be observed that between 4.6 and 26 times more MUA channels can be fit on-implant with data compression, depending on BP $\in \{1, 5, 10, 20, 50, 100\}$ ms and FPGA size $\in \{1, 2.5, 5, 7.5\}$ mm. As such, one can observe that data compression allows one to fit many more MUA channels onto the same implant. Therefore, the compression schemes developed in this work are a useful addition to MUA-based WI-BMIs.

Table C.2: Suggested SH encodings and corresponding BRs and hardware resources for each BP and channel count. BR results from test data. 'W' represents the windowed encoding. 'x' entries are where the memory requirements exceeded those of the FPGA platform we selected. The FPGA Logic Cells and Memory are shared across all channels.

(a) Recommended Encoding

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	DED	DED	DED	DED	DED
5	**2**	DED	DED	DED	DED	DED
10	**2**	DED	DED	DED	DED	DED
20	**2**	DED	W	W	W	W
50	**3**	DED	W	W	W	W
100	**5**	W	W	W	W	W

(b) Bit Rates (bps / channel)

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	50	99	151	177	183
5	**2**	45	75	78	79	79
10	**2**	44	62	62	62	62
20	**2**	31	50	50	50	50
50	**3**	25	29	29	29	29
100	**5**	20	21	22	22	22

(c) FPGA Logic Cells

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	114	144	164	207	242
5	**2**	124	153	174	217	252
10	**2**	130	159	180	223	258
20	**2**	135	128	134	162	192
50	**3**	183	171	176	207	x
100	**5**	232	246	251	286	x

(d) FPGA Memory (bits)

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	60	648	2448	20448	60448
5	**2**	60	968	2768	20768	60768
10	**2**	60	968	2768	20768	60768
20	**2**	60	200	2000	20000	60000
50	**3**	140	400	4000	40000	x
100	**5**	60	600	6000	60000	x

(e) FPGA Dynamic Power (μW / channel)

		n				
BP	**S**	**10**	**10^2**	**10^3**	**10^4**	**3×10^4**
1	**2**	2.03	3.05	4.1	4.74	5.39
5	**2**	1.93	2.58	2.66	2.76	3.34
10	**2**	1.95	2.32	2.35	2.42	2.97
20	**2**	1.7	1.96	1.98	2.03	2.55
50	**3**	1.59	1.54	1.58	2.07	x
100	**5**	1.4	1.43	1.46	2	x

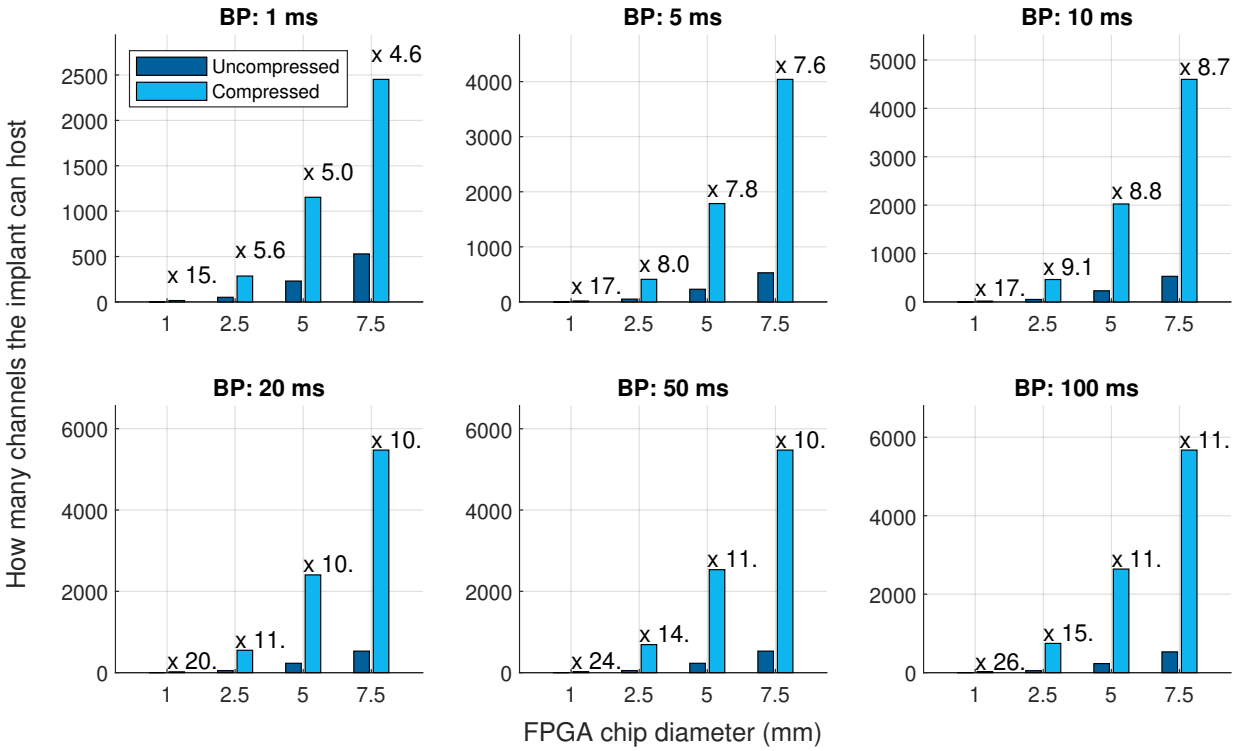

Figure C.4: Number of MUA channels we can host on a Lattice ice40LP FPGA of different dimensions for the selected communication schemes at a BP $\in$ [1, 5, 10, 20, 50, 100] ms while remaining within the power budget. This is compared to the number of channels that can be hosted given the standard uncompressed MUA representation at 1 ms BP, $S = 2$, that has a 1000 bps/channel BR. The required power for any front-end ADC and pre-amplifiers is ignored. The total dynamic power budget is given by the FPGA dimensions multiplied by a heat flux limit of 10 mW/cm^2, assuming heat flux from both faces, minus the FPGA static power. For each BP and FPGA dimension pair, the ratio of how many more channels can be fitted on-implant with compression is also given.

Appendix D

Statistic exploration

According to Eqs. 3.7 and 3.8, Threshold is can be written as Eq.D.1, which can be approximated as Eq.D.2. Assuming z follows half-normal distribution $f_Z(z) = \frac{\sqrt{\pi}}{\sigma\sqrt{2}} e^{-\frac{z^2}{2\sigma^2}}$, the probability at z = σ, $f_Z(z = \sigma) = \frac{\sqrt{\pi}}{\sigma\sqrt{2}} e^{-\frac{1}{2}}$. The expectation of half-normal distribution $E(Z) = \mu_z = \frac{\sigma\sqrt{2}}{\sqrt{\pi}}$. Therefore, $e^{-\frac{1}{2}} = f_Z(z = \sigma)\,\mu_z$ and the threshold can be expressed as in Eq.D.3. However, it is still unknown whether there is neuroscience or mathematical explanation behind or this equation is just a coincidence.

$$Thr = \frac{5}{8} * \sum_{i=n-64}^{n-1} z_i \tag{D.1}$$

$$Thr = e^{-\frac{1}{2}} * \sum_{i=n-64}^{n-1} z_i \tag{D.2}$$

$$Thr = 64\mu_z^2 f_Z(z = \sigma) \tag{D.3}$$

Bibliography

[1] Adrien B Rapeaux and Timothy G Constandinou. Implantable brain machine interfaces: first-in-human studies, technology challenges and trends. *Current Opinion in Biotechnology*, 72:102–111, 2021.

[2] Fei He and Yuan Yang. Nonlinear system identification of neural systems from neurophysiological signals. *Neuroscience*, 458:213–228, 2021.

[3] Michael J Young, David J Lin, and Leigh R Hochberg. Brain–computer interfaces in neurorecovery and neurorehabilitation. In *Seminars in Neurology*. Thieme Medical Publishers, Inc., 2021.

[4] Leigh R Hochberg, Mijail D Serruya, Gerhard M Friehs, Jon A Mukand, Maryam Saleh, Abraham H Caplan, Almut Branner, David Chen, Richard D Penn, and John P Donoghue. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–171, 2006.

[5] Francis R Willett et al. High-performance brain-to-text communication via handwriting. *Nature*, 593(7858):249–254, 2021.

[6] Guy H Wilson, Sergey D Stavisky, Francis R Willett, Donald T Avansino, Jessica N Kelemen, Leigh R Hochberg, Jaimie M Henderson, Shaul Druckmann, and Krishna V Shenoy. Decoding spoken english from intracortical electrode arrays in dorsal precentral gyrus. *Journal of Neural Engineering*, 17(6):066007, 2020.

[7] Francis R Willett, Erin Kunz, Chaofei Fan, Donald Avansino, Guy Wilson, Eun Young Choi, Foram Kamdar, Leigh R H Hochberg, Shaul Druckmann, Krishna Shenoy, et al. A high-performance speech neuroprosthesis. *bioRxiv*, pages 2023–01, 2023.

[8] Elon Musk et al. An integrated brain-machine interface platform with thousands of channels. *Journal of Medical Internet Research*, 21(10):e16194, 2019.

[9] Kate L Montgomery, Alexander J Yeh, John S Ho, Vivien Tsao, Shrivats Mohan Iyer, Logan Grosenick, Emily A Ferenczi, Yuji Tanabe, Karl Deisseroth, Scott L Delp, et al.

Wirelessly powered, fully internal optogenetics for brain, spinal and peripheral circuits in mice. *Nature Methods*, 12(10):969–974, 2015.

[10] Nur Ahmadi, Matthew L Cavuto, Peilong Feng, Lieuwe B Leene, Michal Maslik, Federico Mazza, Oscar Savolainen, Katarzyna M Szostak, Christos-Savvas Bouganis, Jinendra Ekanayake, et al. Towards a distributed, chronically-implantable neural interface. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 719–724. IEEE, 2019.

[11] Zheng Zhang, Oscar W Savolainen, and Timothy G Constandinou. Algorithm and hardware considerations for real-time neural signal on-implant processing. *Journal of Neural Engineering*, 19(1):016029, 2022.

[12] Eberhard E Fetz. Operant conditioning of cortical unit activity. *Science*, 163(3870):955–958, 1969.

[13] Apostolos P Georgopoulos, Andrew B Schwartz, and Ronald E Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

[14] John K Chapin, Karen A Moxon, Ronald S Markowitz, and Miguel AL Nicolelis. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nature Neuroscience*, 2(7):664–670, 1999.

[15] Leigh R Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y Masse, John D Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S Cash, Patrick Van Der Smagt, et al. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012.

[16] Alim Louis Benabid, Stephan Chabardes, John Mitrofanis, and Pierre Pollak. Deep brain stimulation of the subthalamic nucleus for the treatment of parkinson's disease. *The Lancet Neurology*, 8(1):67–81, 2009.

[17] A Bolu Ajiboye, Francis R Willett, Daniel R Young, William D Memberg, Brian A Murphy, Jonathan P Miller, Benjamin L Walter, Jennifer A Sweet, Harry A Hoyen, Michael W Keith, et al. Restoration of reaching and grasping movements through brain-controlled muscle stimulation in a person with tetraplegia: a proof-of-concept demonstration. *The Lancet*, 389(10081):1821–1830, 2017.

[18] Sharlene N Flesher, John E Downey, Jeffrey M Weiss, Christopher L Hughes, Angelica J Herrera, Elizabeth C Tyler-Kabara, Michael L Boninger, Jennifer L Collinger, and Robert A Gaunt. A brain-computer interface that evokes tactile sensations improves robotic arm control. *Science*, 372(6544):831–836, 2021.

[19] Paul Nuyujukian, Jonathan C Kao, Stephen I Ryu, and Krishna V Shenoy. A nonhuman primate brain–computer typing interface. *Proceedings of the IEEE*, 105(1):66–72, 2016.

[20] AG Rouse, JJ Williams, JJ Wheeler, and DW Moran. Spatial co-adaptation of cortical control columns in a micro-ECoG brain–computer interface. *Journal of Neural Engineering*, 13(5):056018, 2016.

[21] Gopala K Anumanchipalli, Josh Chartier, and Edward F Chang. Speech synthesis from neural decoding of spoken sentences. *Nature*, 568(7753):493–498, 2019.

[22] Paulina Kieliba, Danielle Clode, Roni O Maimon-Mor, and Tamar R Makin. Robotic hand augmentation drives changes in neural body representation. *Science Robotics*, 6(54), 2021.

[23] B Wodlinger, JE Downey, EC Tyler-Kabara, AB Schwartz, ML Boninger, and JL Collinger. Ten-dimensional anthropomorphic arm control in a human brain- machine interface: difficulties, solutions, and limitations. *Journal of Neural Engineering*, 12(1):016011, 2014.

[24] Lan Luan, Jacob T Robinson, Behnaam Aazhang, Taiyun Chi, Kaiyuan Yang, Xue Li, Haad Rathore, Amanda Singer, Sudha Yellapantula, Yingying Fan, et al. Recent advances in electrical neural interface engineering: Minimal invasiveness, longevity, and scalability. *Neuron*, 108(2):302–321, 2020.

[25] Samuel C Colachis, Collin F Dunlap, Nicholas V Annetta, Sanjay M Tamrakar, Marcia A Bockbrader, and David A Friedenberg. Long-term intracortical microelectrode array performance in a human: A 5 year retrospective analysis. *Journal of Neural Engineering*, 18(4):0460d7, 2021.

[26] James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232, 2017.

[27] Nicholas A Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539):eabf4588, 2021.

[28] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Charu Bai Reddy, Matteo Carandini, and Kenneth D Harris. Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, 364(6437):eaav7893, 2019.

[29] Nicholas A Steinmetz, Peter Zatka-Haas, Matteo Carandini, and Kenneth D Harris. Distributed coding of choice, action and engagement across the mouse brain. *Nature*, 576(7786):266–273, 2019.

[30] Michael Okun, Nicholas A Steinmetz, Lee Cossell, M Florencia Iacaruso, Ho Ko, Péter Barthó, Tirin Moore, Sonja B Hofer, Thomas D Mrsic-Flogel, Matteo Carandini, et al. Diverse coupling of neurons to populations in sensory cortex. *Nature*, 521(7553):511–515, 2015.

[31] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

[32] William E Allen, Michael Z Chen, Nandini Pichamoorthy, Rebecca H Tien, Marius Pachitariu, Liqun Luo, and Karl Deisseroth. Thirst regulates motivated behavior through modulation of brainwide neural population dynamics. *Science*, 364(6437):eaav3932, 2019.

[33] Kunal Sahasrabuddhe, Aamir A Khan, Aditya P Singh, Tyler M Stern, Yeena Ng, Aleksandar Tadić, Peter Orel, Chris LaReau, Daniel Pouzzner, Kurtis Nishimura, et al. The argo: a high channel count recording system for neural recording in vivo. *Journal of Neural Engineering*, 18(1):015002, 2021.

[34] Abdulmalik Obaid, Mina-Elraheb Hanna, Yu-Wei Wu, Mihaly Kollo, Romeo Racz, Matthew R Angle, Jan Müller, Nora Brackbill, William Wray, Felix Franke, et al. Massively parallel microwire arrays integrated with CMOS chips for neural recording. *Science Advances*, 6(12):eaay2789, 2020.

[35] Kevin Thurley, Stephen C Tovey, Gregor Moenke, Victoria L Prince, Abha Meena, Andrew P Thomas, Alexander Skupin, Colin W Taylor, and Martin Falcke. Reliable encoding of stimulus intensities within random sequences of intracellular Ca2+ spikes. *Science Signaling*, 7(331):ra59–ra59, 2014.

[36] Chong Xie, Ziliang Lin, Lindsey Hanson, Yi Cui, and Bianxiao Cui. Intracellular recording of action potentials by nanopillar electroporation. *Nature Nanotechnology*, 7(3):185–190, 2012.

[37] Frank S Werblin and John E Dowling. Organization of the retina of the mudpuppy, necturus maculosus. ii. intracellular recording. *Journal of Neurophysiology*, 32(3):339–355, 1969.

[38] Darrell A Henze, Zsolt Borhegyi, Jozsef Csicsvari, Akira Mamiya, Kenneth D Harris, and Gyorgy Buzsaki. Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *Journal of Neurophysiology*, 84(1):390–400, 2000.

[39] György Buzsáki and Erik W Schomburg. What does gamma coherence tell us about inter-regional neural communication? *Nature Neuroscience*, 18(4):484–489, 2015.

[40] Boris Rosin, Maya Slovik, Rea Mitelman, Michal Rivlin-Etzion, Suzanne N Haber, Zvi Israel, Eilon Vaadia, and Hagai Bergman. Closed-loop deep brain stimulation is superior in ameliorating parkinsonism. *Neuron*, 72(2):370–384, 2011.

[41] Nur Ahmadi, Timothy G Constandinou, and Christos-Savvas Bouganis. Robust and accurate decoding of hand kinematics from entire spiking activity using deep learning. *Journal of Neural Engineering*, 18(2):026011, 2021.

[42] Nir Even-Chen, Dante G. Muratore, Sergey D. Stavisky, Leigh R. Hochberg, Jaimie M. Henderson, Boris Murmann, and Krishna V. Shenoy. Power-saving design opportunities for wireless intracortical brain–computer interfaces. *Nature Biomedical Engineering*, 2020.

[43] Joseph E. O'Doherty, Mariana M. B. Cardoso, Joseph G. Makin, and Philip N. Sabes. Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology, May 2017.

[44] Nick Steinmetz, Matteo Carandini, and Kenneth D Harris. "Single Phase3" and "Dual Phase3" Neuropixels Datasets. 2019.

[45] R Quian Quiroga, Zoltan Nadasdy, and Yoram Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004.

[46] Daniel Valencia, Patrick P Mercier, and Amir Alimohammad. In vivo neural spike detection with adaptive noise estimation. *Journal of Neural Engineering*, 19(4):046018, 2022.

[47] Xiaorang Guo, MohammadAli Shaeri, and Mahsa Shoaran. An accurate and hardware-efficient dual spike detector for implantable neural interfaces. In *2022 Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2022.

[48] Gerardo Saggese and Antonio Giuseppe Maria Strollo. Low-power energy-based spike detector asic for implantable multichannel BMIs. *Electronics*, 11(18):2943, 2022.

[49] Yan Liu, Song Luan, Ian Williams, Adrien Rapeaux, and Timothy G Constandinou. A 64-channel versatile neural recording SoC with activity-dependent data throughput. *IEEE Transactions on Biomedical Circuits and Systems*, 11(6):1344–1355, 2017.

[50] Manuel Delgado-Restituto, Alberto Rodríguez-Pérez, Angela Darie, Cristina Soto-Sánchez, Eduardo Fernández-Jover, and Ángel Rodríguez-Vázquez. System-level design of a 64-channel low power neural spike recording sensor. *IEEE Transactions on Biomedical Circuits and Systems*, 11(2):420–433, 2017.

[51] James C Barrese, Naveen Rao, Kaivon Paroo, Corey Triebwasser, Carlos Vargas-Irwin, Lachlan Franquemont, and John P Donoghue. Failure mode analysis of silicon-based intracortical microelectrode arrays in non-human primates. *Journal of Neural Engineering*, 10(6):066014, 2013.

[52] Eric Drebitz, Marcus Haag, Iris Grothe, Sunita Mandon, and Andreas K Kreiter. Attention configures synchronization within local neuronal networks for processing of the behaviorally relevant stimulus. *Frontiers in Neural Circuits*, 12:71, 2018.

[53] James F Kaiser. On a simple algorithm to calculate the 'energy' of a signal. In *International Conference on Acoustics, Speech, and Signal processing*, pages 381–384. IEEE, 1990.

[54] Zhonglin Lin, Changshui Zhang, Wei Wu, and Xiaorong Gao. Frequency recognition based on canonical correlation analysis for SSVEP-based BCIs. *IEEE Transactions on Biomedical Engineering*, 53(12):2610–2614, 2006.

[55] Fernando J Chaure, Hernan G Rey, and Rodrigo Quian Quiroga. A novel and fully automatic spike-sorting implementation with variable number of features. *Journal of Neurophysiology*, 120(4):1859–1871, 2018.

[56] Xilin Liu, Milin Zhang, Andrew G Richardson, Timothy H Lucas, and Jan Van der Spiegel. Design of a closed-loop, bidirectional brain machine interface system with energy efficient neural feature extraction and pid control. *IEEE Transactions on Biomedical Circuits and Systems*, 11(4):729–742, 2016.

[57] Steven X Moffett, Sean M O''Malley, Shushuang Man, Dawei Hong, and Joseph V Martin. Dynamics of high frequency brain activity. *Scientific Reports*, 7(1):1–5, 2017.

[58] Zoran Nenadic and Joel W Burdick. Spike detection using the continuous wavelet transform. *IEEE Transactions on Biomedical Engineering*, 52(1):74–87, 2004.

[59] Florian Lieb et al. A stationary wavelet transform and a time-frequency based spike detection algorithm for extracellular recorded data. *Journal of Neural Engineering*, 14(3):036013, 2017.

[60] Karim Abdelhalim, Vadim Smolyakov, and Roman Genov. Phase-synchronization early epileptic seizure detector VLSI architecture. *IEEE Transactions on Biomedical Circuits and Systems*, 5(5):430–438, 2011.

[61] Dimitrios A Adamos, Efstratios K Kosmidis, and George Theophilidis. Performance evaluation of PCA-based spike sorting algorithms. *Computer Methods and Programs in Biomedicine*, 91(3):232–244, 2008.

[62] Nur Ahmadi, Timothy G Constandinou, and Christos-Savvas Bouganis. Estimation of neuronal firing rate using bayesian adaptive kernel smoother (BAKS). *PloS One*, 13(11):e0206794, 2018.

[63] Reid R Harrison. The design of integrated circuits to observe brain activity. *Proceedings of the IEEE*, 96(7):1203–1216, 2008.

[64] Sarah Gibson, Jack W Judy, and Dejan Marković. Spike sorting: The first step in decoding the brain: The first step in decoding the brain. *IEEE Signal Processing Magazine*, 29(1):124–143, 2011.

[65] Vaibhav Karkare, Sarah Gibson, and Dejan Marković. A 75-μW, 16-channel neural spike-sorting processor with unsupervised clustering. *IEEE Journal of Solid-State Circuits*, 48(9):2230–2238, 2013.

[66] Sivylla E Paraskevopoulou, Deren Y Barsakcioglu, Mohammed R Saberi, Amir Eftekhar, and Timothy G Constandinou. Feature extraction using first and second derivative extrema (FSDE) for real-time and hardware-efficient spike sorting. *Journal of Neuroscience Methods*, 215(1):29–37, 2013.

[67] Song Luan, Ian Williams, Michal Maslik, Yan Liu, Felipe De Carvalho, Andrew Jackson, Rodrigo Quian Quiroga, and Timothy G Constandinou. Compact standalone platform for neural recording with real-time spike sorting and data logging. *Journal of Neural Engineering*, 15(4):046014, 2018.

[68] Oscar W Savolainen and Timothy G Constandinou. Lossless compression of intracortical extracellular neural recordings using non-adaptive huffman encoding. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 4318–4321. IEEE, 2020.

[69] Robert D Flint, Zachary A Wright, Michael R Scheid, and Marc W Slutzky. Long term, stable brain machine interface performance using local field potentials and multiunit spikes. *Journal of Neural Engineering*, 10(5):056005, 2013.

[70] Sung-Phil Kim, John D Simeral, Leigh R Hochberg, John P Donoghue, and Michael J Black. Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *Journal of Neural Engineering*, 5(4):455, 2008.

[71] Zheng Li, Joseph E O'Doherty, Timothy L Hanson, Mikhail A Lebedev, Craig S Henriquez, and Miguel AL Nicolelis. Unscented kalman filter for brain-machine interfaces. *PloS one*, 4(7):e6243, 2009.

[72] Mitra Taghizadeh-Sarabi, Mohammad Reza Daliri, and Kavous Salehzadeh Niksirat. Decoding objects of basic categories from electroencephalographic signals using wavelet transform and support vector machines. *Brain Topography*, 28(1):33–46, 2015.

[73] Abdullah Caliskan, Mehmet Emin Yuksel, Hasan Badem, and Alper Basturk. A deep neural network classifier for decoding human brain activity based on magnetoencephalography. *Elektronika ir Elektrotechnika*, 23(2):63–67, 2017.

[74] Jin Zhang, Chungang Yan, and Xiaoliang Gong. Deep convolutional neural network for decoding motor imagery based brain computer interface. In *2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–5. IEEE, 2017.

[75] David Sussillo, Paul Nuyujukian, Joline M Fan, Jonathan C Kao, Sergey D Stavisky, Stephen Ryu, and Krishna Shenoy. A recurrent neural network for closed-loop intracortical brain–machine interface decoders. *Journal of Neural Engineering*, 9(2):026027, 2012.

[76] Stefano Tortora, Stefano Ghidoni, Carmelo Chisari, Silvestro Micera, and Fiorenzo Artoni. Deep learning-based bci for gait decoding from EEG with LSTM recurrent neural network. *Journal of Neural Engineering*, 17(4):046011, 2020.

[77] Dongxu Yang, Yadong Liu, Zongtan Zhou, Yang Yu, and Xinbin Liang. Decoding visual motions from eeg using attention-based rnn. *Applied Sciences*, 10(16):5662, 2020.

[78] J Blair Price, Aaron E Rusheen, Abhijeet S Barath, Juan M Rojas Cabrera, Hojin Shin, Su-Youne Chang, Christopher J Kimble, Kevin E Bennet, Charles D Blaha, Kendall H Lee, et al. Clinical applications of neurochemical and electrophysiological measurements for closed-loop neurostimulation. *Neurosurgical Focus*, 49(1):E6, 2020.

[79] Bingzhao Zhu, Masoud Farivar, and Mahsa Shoaran. Resot: Resource-efficient oblique trees for neural signal classification. *IEEE Transactions on Biomedical Circuits and Systems*, 14(4):692–704, 2020.

[80] Iñaki Iturrate, Michael Pereira, and José del R Millán. Closed-loop electrical neurostimulation: challenges and opportunities. *Current Opinion in Biomedical Engineering*, 8:28–37, 2018.

[81] Ian Schofield and Amirhossein Alimohammad. Parallel GPU-accelerated spike sorting. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–7. IEEE, 2019.

[82] Scott Stanslaski, Pedram Afshar, Peng Cong, Jon Giftakis, Paul Stypulkowski, Dave Carlson, Dave Linde, Dave Ullestad, Al-Thaddeus Avestruz, and Timothy Denison. Design and validation of a fully implantable, chronic, closed-loop neuromodulation device with concurrent sensing and stimulation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 20(4):410–421, 2012.

[83] Jeffrey Herron, Tim Denison, and Howard Jay Chizeck. Closed-loop DBS with movement intention. In *2015 7th international IEEE/EMBS conference on neural engineering (NER)*, pages 844–847. IEEE, 2015.

[84] Brady Houston, Zack Blumenfeld, Emma Quinn, Helen Bronte-Stewart, and Howard Chizeck. Long-term detection of parkinsonian tremor activity from subthalamic nucleus local field potentials. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3427–3431. IEEE, 2015.

[85] Elena Ryapolova-Webb, Pedram Afshar, Scott Stanslaski, Tim Denison, Coralie de Hemptinne, Krystof Bankiewicz, and Philip A Starr. Chronic cortical and electromyographic recordings from a fully implantable device: preclinical experience in a nonhuman primate. *Journal of Neural Engineering*, 11(1):016009, 2014.

[86] Scott Stanslaski, Jeffrey Herron, Tom Chouinard, Duane Bourget, Ben Isaacson, Vaclav Kremen, Enrico Opri, William Drew, Benjamin H Brinkmann, Aysegul Gunduz, et al. A chronically implantable neural coprocessor for investigating the treatment of neurological disorders. *IEEE Transactions on Biomedical Circuits and Systems*, 12(6):1230–1245, 2018.

[87] Felice T Sun and Martha J Morrell. The rns system: responsive cortical stimulation for the treatment of refractory partial epilepsy. *Expert Eeview of Medical Devices*, 11(6):563–572, 2014.

[88] László Schäffer, Zoltán Nagy, Zoltán Kineses, and Richárd Fiáth. FPGA-based neural probe positioning to improve spike sorting with osort algorithm. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017.

[89] Laszlo Schäffer, Zoltan Nagy, Zoltan Kincses, Richard Fiáth, and Istvan Ulbert. Spatial information based osort for real-time spike sorting using FPGA. *IEEE Transactions on Biomedical Engineering*, 68(1):99–108, 2020.

[90] Robert Toth, Mayela Zamora, Jon Ottaway, Tom Gillbe, Sean Martin, Moaad Benjaber, Guy Lamb, Tara Noone, Barry Taylor, Alceste Deli, et al. Dyneumo mk-2: An investigational circadian-locked neuromodulator with responsive stimulation for applied chronobiology. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3433–3440. IEEE, 2020.

[91] Mayela Zamora, Robert Toth, Jon Ottaway, Tom Gillbe, Sean Martin, Moaad Benjaber, Guy Lamb, Tara Noone, Zachary Nairac, Timothy G Constandinou, et al. Dyneumo mk-1: A fully-implantable, motion-adaptive neurostimulator with configurable response algorithms. *bioRxiv*, 2020.

[92] Zheng Zhang and Timothy G Constandinou. Adaptive spike detection and hardware optimization towards autonomous, high-channel-count BMIs. *Journal of Neuroscience Methods*, 354:109103, 2021.

[93] Joon Hwan Choi, Hae Kyung Jung, and Taejeong Kim. A new action potential detector using the MTEO and its effects on spike sorting systems at low signal-to-noise ratios. *IEEE Transactions on Biomedical Engineering*, 53(4):738–746, 2006.

[94] NXP Semiconductors. Freedom development platform for the kinetis KL05 and KL04 MCUs. https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-the-kinetis-kl05-and-kl04-mcus:FRDM-KL05Z. Accessed: 2023-07-10.

[95] NXP Semiconductors. Freedom development platform for kinetis K64, K63, and K24 MCUs. https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-k66-k65-and-k26-mcus:FRDM-K66F. Accessed: 2023-07-10.

[96] Joaquin Navajas, Deren Y Barsakcioglu, Amir Eftekhar, Andrew Jackson, Timothy G Constandinou, and Rodrigo Quian Quiroga. Minimum requirements for accurate and

efficient real-time on-chip spike sorting. *Journal of Neuroscience Methods*, 230:51–64, 2014.

[97] E. Yao, Y. Chen, and A. Basu. A 0.7 V, 40 nW compact, current-mode neural spike detector in 65 nm CMOS. *IEEE Transactions on Biomedical Circuits and Systems*, 10(2):309–318, 2016.

[98] A. Rodriguez-Perez, J. Ruiz-Amaya, M. Delgado-Restituto, and '. Rodriguez-Vazquez. A low-power programmable neural spike detection channel with embedded calibration and data compression. *IEEE Transactions on Biomedical Circuits and Systems*, 6(2):87–100, 2012.

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

[100] R Quian Quiroga. What is the real shape of extracellular spikes? *Journal of Neuroscience Methods*, 177(1):194–198, 2009.

[101] Deren Y Barsakcioglu, Yan Liu, Pooja Bhunjun, Joaquin Navajas, Amir Eftekhar, Andrew Jackson, Rodrigo Quian Quiroga, and Timothy G Constandinou. An analogue front-end model for developing neural spike sorting systems. *IEEE Transactions on Biomedical Circuits and Systems*, 8(2):216–227, 2014.

[102] Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78, 1998.

[103] Scott B Wilson and Ronald Emerson. Spike detection: a review and comparison of algorithms. *Clinical Neurophysiology*, 113(12):1873–1881, 2002.

[104] Iyad Obeid and Patrick D Wolf. Evaluation of spike-detection algorithms for a brain-machine interface application. *IEEE Transactions on Biomedical Engineering*, 51(6):905–911, 2004.

[105] Alex Zviagintsev, Yevgeny Perelman, and Ran Ginosar. Algorithms and architectures for low power spike detection and alignment. *Journal of Neural Engineering*, 3(1):35, 2006.

[106] Donghwi Kim, Milutin Stanacevic, Ridha Kamoua, and Zachary Mainen. A low-power low-data-rate neural recording system with adaptive spike detection. In *2008 51st Midwest Symposium on Circuits and Systems*, pages 822–825. IEEE, 2008.

[107] Yuning Yang, Awais Kamboh, and J Mason Andrew. Adaptive threshold spike detection using stationary wavelet transform for neural recording implants. In *2010 Biomedical Circuits and Systems Conference (BioCAS)*, pages 9–12. IEEE, 2010.

[108] Ermis Koutsos, Sivylla E Paraskevopoulou, and Timothy G Constandinou. A 1.5 μW NEO-based spike detector with adaptive-threshold for calibration-free multichannel neural interfaces. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 1922–1925. IEEE, 2013.

[109] Gabriel Gagnon-Turcotte, C-O Dufresne Camaro, and Benoit Gosselin. Comparison of low-power biopotential processors for on-the-fly spike detection. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 802–805. IEEE, 2015.

[110] Yuning Yang, C Sam Boling, Awais M Kamboh, and Andrew J Mason. Adaptive threshold neural spike detector using stationary wavelet transform in CMOS. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(6):946–955, 2015.

[111] Shashank Dwivedi and Anup K Gogoi. A novel adaptive real-time detection algorithm for an area-efficient CMOS spike detector circuit. *AEU-International Journal of Electronics and Communications*, 88:87–97, 2018.

[112] Sunghan Kim and James McNames. Automatic spike detection based on adaptive template matching for extracellular neural recordings. *Journal of Neuroscience Methods*, 165(2):165–174, 2007.

[113] H. Semmaoui, J. Drolet, A. Lakhssassi, and M. Sawan. Setting adaptive spike detection threshold for smoothed TEO based on robust statistics theory. *IEEE Transactions on Biomedical Engineering*, 59(2):474–482, 2012.

[114] Sudipta Mukhopadhyay and GC Ray. A new interpretation of nonlinear energy operator and its efficacy in spike detection. *IEEE Transactions on Biomedical Engineering*, 45(2):180–187, 1998.

[115] G. Gagnon-Turcotte, M. Sawan, and B. Gosselin. Low-power adaptive spike detector based on a sigma-delta control loop. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2167–2170, 2015.

[116] Deren Y Barsakcioglu and Timothy G Constandinou. A 32-channel MCU-based feature extraction and classification for scalable on-node spike sorting. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1310–1313. IEEE, 2016.

[117] Y. Yang and A. J. Mason. Hardware efficient automatic thresholding for NEO-based neural spike detection. *IEEE Transactions on Biomedical Engineering*, 64(4):826–833, 2017.

[118] G. G. Turcotte, C. . D. Camaro, A. A. Kisomi, R. Ameli, and B. Gosselin. A wireless multichannel optogenetic headstage with on-the-fly spike detection. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1758–1761, 2015.

[119] Zheng Zhang and Timothy G. Constandinou. Selecting an effective amplitude threshold for neural spike detection. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, pages 2328–2331, 2022.

[120] Alexandru Oprea, Zheng Zhang, and Timothy G Constandinou. Hardware evaluation of spike detection algorithms towards wireless brain machine interfaces. In *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 60–64. IEEE, 2022.

[121] Zheng Zhang, Peilong Feng, Alexandru Oprea, and Timothy G Constandinou. Calibration-free and hardware-efficient neural spike detection for brain machine interfaces. *IEEE Transactions on Biomedical Circuits and Systems*, 2023.

[122] Michael Rizk and Patrick D Wolf. Optimizing the automatic selection of spike detection thresholds using a multiple of the noise level. *Medical Biological Engineering Computing*, 47(9):955–966, 2009.

[123] Zhi Yang, Wentai Liu, Mohammad Reza Keshtkaran, Yin Zhou, Jian Xu, Victor Pikov, Cuntai Guan, and Yong Lian. A new EC–PC threshold estimation method for in vivo neural spike detection. *Journal of Neural Engineering*, 9(4):046017, 2012.

[124] Yang-Guo Li, Qingyun Ma, Mohammad Rafiqul Haider, and Yehia Massoud. Ultra-low-power high sensitivity spike detectors based on modified nonlinear energy operator. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 137–140. IEEE, 2013.

[125] Thomas M Hall, Kianoush Nazarpour, and Andrew Jackson. Real-time estimation and biofeedback of single-neuron firing rates using local field potentials. *Nature Communications*, 5:5462, 2014.

[126] Zheng Zhang and Timothy G Constandinou. Firing-rate-modulated spike detection and neural decoding co-design. *Journal of Neural Engineering*, 20(3):036003, 2023.

[127] VG Macefield, SC Gandevia, B Bigland-Ritchie, RB Gorman, and D Burke. The firing rates of human motoneurones voluntarily activated in the absence of muscle afferent feedback. *The Journal of Physiology*, 471(1):429–443, 1993.

[128] Hao Xu, Yan Han, Xiaoxia Han, Junyu Xu, Shen Lin, and Ray CC Cheung. Unsupervised and real-time spike sorting chip for neural signal processing in hippocampal prosthesis. *Journal of Neuroscience Methods*, 311:111–121, 2019.

[129] Emily R Oby, Sagi Perel, Patrick T Sadtler, Douglas A Ruff, Jessica L Mischel, David F Montez, Marlene R Cohen, Aaron P Batista, and Steven M Chase. Extracellular voltage threshold settings can be tuned for optimal encoding of movement and stimulus parameters. *Journal of Neural Engineering*, 13(3):036009, 2016.

[130] Alex H Barnett, Jeremy F Magland, and Leslie F Greengard. Validation of neural spike sorting algorithms without ground-truth information. *Journal of Neuroscience Methods*, 264:65–77, 2016.

[131] Zhang Zhang and Timothy G Constandinou. A robust and automated algorithm that uses single-channel spike sorting to label multi-channel neuropixels data. In *2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*. IEEE, 2021.

[132] Sonia Todorova, Patrick Sadtler, Aaron Batista, Steven Chase, and Valérie Ventura. To sort or not to sort: the impact of spike-sorting on neural decoding performance. *Journal of Neural Engineering*, 11(5):056005, 2014.

[133] Rafaella Fiorelli, Manuel Delgado-Restituto, and Ángel Rodríguez-Vázquez. Charge-redistribution based quadratic operators for neural feature extraction. *IEEE Transactions on Biomedical Circuits and Systems*, 14(3):606–619, 2020.

[134] D S Won, P H E Tiesinga, C S Henriquez, and P D Wolf. An analytical comparison of the information in sorted and non-sorted cosine-tuned spike activity. *Journal of Neural Engineering*, 4(3):322, 2007.

[135] Oscar W Savolainen, Zheng Zhang, Peilong Feng, and Timothy G Constandinou. Hardware-efficient compression of neural multi-unit activity. *IEEE Access*, 10:117515–117529, 2022.

[136] Cynthia A Chestek, Vikash Gilja, Paul Nuyujukian, Justin D Foster, Joline M Fan, Matthew T Kaufman, Mark M Churchland, Zuley Rivera-Alvidrez, John P Cunningham, Stephen I Ryu, et al. Long-term stability of neural prosthetic control signals from silicon cortical arrays in rhesus macaque motor cortex. *Journal of Neural Engineering*, 8(4):045005, 2011.

[137] Mattia Tambaro, Elia Arturo Vallicelli, David Tomasella, Andrea Baschirotto, Stefano Vassanelli, Marta Maschietto, and Marcello De Matteis. A 10 MSample/Sec digital neural spike detection for a 1024 pixels multi transistor array sensor. In *2019 26th*

IEEE International Conference on Electronics, Circuits and Systems (ICECS), pages 711–714. IEEE, 2019.

[138] Jongkil Park, Gookhwa Kim, and Sang-Don Jung. A 128-channel FPGA-based real-time spike-sorting bidirectional closed-loop neural interface system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(12):2227–2238, 2017.

[139] Amir M Sodagar, Gayatri E Perlin, Ying Yao, Khalil Najafi, and Kensall D Wise. An implantable 64-channel wireless microsystem for single-unit neural recording. *IEEE Journal of Solid-State Circuits*, 44(9):2591–2604, 2009.

[140] Tong Wu, Jian Xu, Yong Lian, Azam Khalili, Amir Rastegarnia, Cuntai Guan, and Zhi Yang. A 16-channel nonparametric spike detection asic based on EC-PC decomposition. *IEEE Transactions on Biomedical Circuits and Systems*, 10(1):3–17, 2015.

[141] William Biederman, Daniel J. Yeager, Nathan Narevsky, Jaclyn Leverett, Ryan Neely, Jose M. Carmena, Elad Alon, and Jan M. Rabaey. A 4.78 mm^2 fully-integrated neuromodulation SoC combining 64 acquisition channels with digital compression and simultaneous dual stimulation. *IEEE Journal of Solid-State Circuits*, 50(4):1038–1047, 2015.

[142] Robert D Flint, Eric W Lindberg, Luke R Jordan, Lee E Miller, and Marc W Slutzky. Accurate decoding of reaching movements from field potentials in the absence of spikes. *Journal of Neural Engineering*, 9(4):046006, 2012.

[143] Thomas Brochier, Lyuba Zehl, Yaoyao Hao, Margaux Duret, Julia Sprenger, Michael Denker, Sonja Grün, and Alexa Riehle. Massively parallel recordings in macaque motor cortex during an instructed delayed reach-to-grasp task. *Scientific Data*, 5(1):1–23, 2018.