UNIVERSITY OF CALIFORNIA SAN DIEGO

SAN DIEGO STATE UNIVERSITY

Towards Autonomous Brain-Computer Interfaces: Approaches, Design, and Implementation

A Dissertation submitted in partial satisfaction of the requirements
for the degree Doctor of Philosophy

in

Engineering Science (Electrical and Computer Engineering)

by

Daniel Valencia

2024

DEDICATION

To my family.

EPIGRAPH

No one can tell what goes on in between the person you were and the person you become.
There are no maps of the change. You just come out the other side.

Stephen King

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Starting out the Ph.D. five years ago I knew this would be a big undertaking. Completing the Ph.D. would not have been possible without the support group I had from various parts of my life. I'd like to thank my advisor Professor Amir Alimohammad for his constant guidance, not only during the Ph.D., but also during the end of my Bachelor's and throughout my Master's studies. Professor Alimohammad gave me an opportunity to get involved in research in the VLSI Lab at SDSU, and had I not taken up the offer I would not have grown as much as I have as both a scholar and a person. In truth, being the first of my family to attend University, I wasn't truly aware about higher levels of education and would not have believed it to be something I could achieve if it had not been for Professor Alimohammad's encouragement. I'd also like to thank Professor Patrick Mercier for accepting to be my co-advisor during the PhD and providing key insights for my publications, particularly on the analog circuit side of things. Special thanks go to the rest of my doctoral committee: Professor Gert Cauwenberghs, Professor Drew Hall, and Professor Hakan Toreyin. I appreciate the valuable questions and suggestions you have made toward my research and this final dissertation.

Part of my support group includes my friends, who helped keep me sane during stressful times, such as preparing for exams and making constant paper revisions late into the nights. In no particular order, Ezra, Jake, Jesus Leon, Jesus Fematt, Andrew, Abe, Carter, Nutdanai, Kristian, those games of Caverna and Terra Mystica (and the rare Agricola) meant more to me than you could know, I thank you for indulging me all those times. I certainly cannot forget those friend trips to the Grand Canyon, Sedona, and Yosemite, which seemed to come at just the right times to avoid burnout and get out a for a while. Former VLSI lab-mates and fellow students, your companionship was invaluable during my studies, special thanks go to Rushabh, Vishnu, Michael, Mauro, Mitchell, Xinqiao, Nick, Jose, Omar, and Raghu. I will always remember the illuminating discussions regarding Verilog and MATLAB programming, as well as our philosophical discussions over freshly brewed pour-over coffee.

Of course, none of this would have been possible without the support and encouragement

of my family. I appreciate all of the family trips, meals, and constant support as I worked toward not only the Ph.D., but other goals in my life. During rough times, I am sure my family believed in me more than I believed in myself. This dissertation, the publications, and the degree are not mine alone, but a mutual accomplishment we share.

This work was supported, in part, by the Center for Neurotechnology, a *National Science Foundation* (NSF) *Engineering Research Center* (EEC-1028725), the NSF Award #2007131, and the *San Diego State University Graduate Fellowship*.

This dissertation is based on various articles that have been published in or submitted to peer-reviewed journals, all of which are first-authored by the author of this dissertation.

Chapter 2 is primarily based on the paper by D. Valencia, P. P. Mercier, and A. Alimohammad, titled “In vivo neural spike detection with adaptive noise estimation,” published in the *Journal of Neural Engineering*, vol. 19, 046018, July 2022. The author of this dissertation is the lead investigator and primary author of this paper.

Chapter 3 is primarily based on the paper by D. Valencia and A. Alimohammad, titled “Neural spike sorting using binarized neural networks,” published in the *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 29, pp. 206 – 214, December 2020. The author of this dissertation is the lead investigator and primary author of this paper.

Chapter 4 is primarily based on the paper by D. Valencia and A. Alimohammad, titled “Partially binarized neural networks for efficient spike sorting,” published in the *Biomedical Engineering Letters*, vol. 13, pp. 73 – 83, February 2023. The author of this dissertation is the lead investigator and primary author of this paper.

Chapter 5 is primarily based on the paper by D. Valencia, P. P. Mercier, and A. Alimohammad, titled “Efficient In Vivo Neural Signal Compression Using an Autoencoder-based Neural Network,” published in the *IEEE Transactions on Biomedical Circuits and Systems*, vol. 18, pp. 691 – 701, January 2024. The author of this dissertation is the lead investigator and primary author of this paper.

Chapter 6 is primarily based on the paper by D. Valencia and A. Alimohammad, titled

"Towards in vivo neural decoding," published in the *Biomedical Engineering Letters*, vol. 12, pp. 185 – 195, May 2022. The author of this dissertation is the lead investigator and primary author of this paper.

Chapter 7 is primarily based on the paper by D. Valencia and A. Alimohammad, titled "A generalized hardware architecture for real-time spiking neural networks," published in *Neural Computing and Applications*, vol. 35, pp. 17821 – 17835, August 2023. The author of this dissertation is the lead investigator and primary author of this paper.

Chapter 8 is primarily based on the paper by D. Valencia, G. Leone, N. Keller, P. P. Mercier, and A. Alimohammad, titled "Power-efficient in vivo brain-machine interfaces via brain-state estimation," published in the *Journal of Neural Engineering*, vol. 20, 016032. The author of this dissertation is the lead investigator and primary author of this paper.

Chapter 9 is primarily based on the paper by D. Valencia, P. P. Mercier, and A. Alimohammad, titled "An efficient brain-switch for asynchronous brain-computer interfaces," published in the *IEEE Transactions on Biomedical Circuits and Systems*, May 2024. The author of this dissertation is the lead investigator and primary author of this paper.

Chapter 10 is primarily based on the paper by D. Valencia, P. P. Mercier, and A. Alimohammad, titled "A hybrid brain-computer interface for low-power and efficient neural decoding," submitted to the *IEEE Transactions on Biomedical Circuits and Systems*. The author of this dissertation is the lead investigator and primary author of this paper.

Daniel Valencia

December 2024

San Diego, CA

VITA

2016	Bachelor of Science, San Diego State University, San Diego
2018–2024	Instructional Student Assistant, Department of Electrical and Computer Engineering, San Diego State University
2018–2024	Research Assistant, VLSI Design and Test Lab San Diego State University, San Diego
2019	Masters of Science, San Diego State University, San Diego
2024	Doctor of Philosophy, University of California San Diego and San Diego State University, San Diego

ABSTRACT OF THE DISSERTATION

Towards Autonomous Brain-Computer Interfaces: Approaches, Design, and Implementation

by

Daniel Valencia

Doctor of Philosophy in Engineering Science (Electrical and Computer Engineering)

University of California San Diego, 2024

Professor Patrick Mercier, Co-Chair
Professor Amir Alimohammad, Co-Chair

Autonomous brain-computer interfaces (BCIs) are devices designed to record, process, and interpret neural activity, enabling individuals with neurodegenerative diseases or spinal cord injuries to regain their ability to interact with their environment without assistance. Over the past two decades, BCI technology has advanced remarkably, largely due to improvements in neural recording interfaces, such as high-density micro-electrode arrays (MEAs). These innovations allow researchers to collect more comprehensive neural data, enhancing our understanding of neural dynamics and their interaction with human physiology. However, BCI technology still faces three major challenges that hinder its progression toward practical applications.

1. **Data overload**: Modern MEAs, with their increased number of recording channels, generate vast amounts of data. Consequently, the power consumption required to acquire, process, and transmit this data becomes a significant barrier, especially given tissue-safe design constraints. This dissertation investigates three effective approaches to reduce data rates, and thus power consumption: spike detection, spike sorting, and neural signal compression, all of which leverage relevant signal features for specific applications.

2. **Neural decoding versatility**: No single neural decoding algorithm suits all applications or users. Therefore, a versatile BCI system must be capable of executing a range of neural decoding algorithms. This dissertation examines the efficient design and implementation of two processor architectures supporting various neural decoding schemes. One processor uses fine-grained sequential processing to implement arbitrary machine-learning-based neural decoding models, while the other employs biologically plausible neuron models to realize a spiking neural network.

3. **Autonomous user engagement**: Traditional BCIs require users to engage with the system during predefined time periods. This dissertation explores two methods for estimating a user's intention to engage with a BCI application: one using high-frequency neural spikes and the other using low-frequency local field potentials. A hybrid (multi-signal) asynchronous BCI is designed, implemented, and verified, combining both neural signal types to optimize intention estimation and improve neural decoding performance.

The approaches discussed in this dissertation present practical strategies to advance BCI technology by reducing power consumption, enabling flexible and robust neural decoding, and incorporating various neural signal features for efficient user intention estimation.

Chapter 1

Introduction

1.1 Introduction

The human brain is a remarkable product of human evolution. Within the brain, billions of cells, called neurons, interact with one another to enable a myriad of human behaviors, both autonomic– such as our ability to breathe without conscious effort– and voluntary, such as controlling our hands and feet to operate a vehicle. Neurons convey information to one another by firing action potentials through connections known as synapses [1]. The mechanism underlying the generation of action potentials relies on ionic current flow during the operation of the neuron. This shifting ionic balance causes variations in the voltage potential in the regions between neurons [2]. This voltage potential can be recorded by using penetrating electrodes, which initiates the process of interpreting neural activity. The concept of relating neural activity to physiological control began in the 1980s, with the pioneering work by Fetz et al. which studied how the activity of cortical motor neurons modulated muscle movement [3, 4]. These studies unlocked the idea of neural decoding, whereby brain activity can be mapped to a known variable or representation. For example, seminal work by Georgopoulos et al. proposed population vector coding, a concept suggesting that the direction of movement is represented as the weighted average of the preferred direction of a population of neurons [5]. Brain-computer interfaces (BCIs) have the potential to become foundational tools for aiding people suffering from spinal cord injury or other neuro-degenerative diseases. Studies have shown that, although a person may lose a limb due to an accident, the underlying neural circuitry that controls motor activity remains functional and active [6].

1.2 The State of the Art and Challenges of BCI

BCI technology has made significant progress over the past two decades. In particular, advances in neural recording technology now allow for simultaneous recording from thousands of sites [7, 8, 9]. This progress has enabled neuroscientists to deepen their understanding of the mechanisms underpinning various neurophysiological processes. Additionally, the rapid growth of machine learning algorithms has allowed researchers to develop relatively robust neural decoding models. By employing a data-driven approach, decoding performance has surpassed that of the hand-tuned models of the past [10]. For example, combining computer-aided assistance, such as typing auto-correction, with neural decoding has been shown to vastly improve communication speed in recent studies [11]. Similarly, artificial intelligence has been shown to enhance BCI control by improving movement-related tasks [12].

BCI applications have also extended beyond traditional kinematic decoding, including recent studies demonstrating the ability to decode speech from neural activity [13, 14]. It should be noted that the majority of these BCI studies focus on either non-invasive recording modalities, such as electroencephalography (EEG), or are limited to clinical or lab-based settings. In such settings, data transmission is typically performed through wired connections to ensure high-fidelity neural recordings. However, these wired connections pose challenges for long-term use, as they may not be safe or hygienic. As a result, there is significant ongoing research aimed at developing implantable BCIs with low-power wireless telemetry, such as backscatter communication [15] or near-field communication (NFC) for both wireless power and data transmission [16]. A key challenge in realizing wireless implantable BCIs is minimizing overall power consumption and energy dissipation, as increased in vivo processing can cause neural tissue temperatures to raise by more than 1° C, potentially leading to tissue damage [17]. To address this, extensive research efforts have focused on designing efficient implementations of various in vivo digital neural signal processing algorithms [18, 19, 20, 21, 22, 23, 24, 25]. However, even with all of these advancements, BCI technology remains in its early stages of development for practical daily use. This is due to three main challenges in BCI technology. First, the amount of data generated by modern recording circuitry is staggering. This can be partly attributed to two conflicting objectives in the development of modern neural recording interfaces. On one hand, increasing the total number of recording sites can be extremely useful for scientific inquiry. For example, recording electrodes employing high-density configurations are essential for resolving the activities of individual neurons over long periods of time [26]. However, using such electrodes not only increases the amount of data generated but also heightens operational demands. For instance, more recording sites require greater power for signal acquisition and conditioning. Additionally, the wireless transmission circuitry would require high-bandwidth telemetry to handle the increased data rate. On the other hand, studies have shown that for practical neural decoding purposes, such fine-grained high-density neural recordings are not required to achieve acceptable performance. For example, a recent study demonstrated how a state-of-the-art "thought-to-text" BCI was realized using a relatively simple 10x10 micro-electrode array (MEA) [11]. This dissertation thus covers three approaches to reduce the overall amount of data generated and transmitted for neural decoding, effectively lowering overall system power consumption to meet tissue-safe design constraints. The foundational approach for two of these methods is to focus neural recording on the activity of individual neurons, representing neural activity as a discrete sequence of action potential (spike) events. The first method involves performing spike detection, where the recorded neural activity is limited to that of nearby neurons instead of recording and transmitting the entire signal. The second method builds upon spike detection by providing finer granularity through spike sorting. In this approach, after spikes are detected, the activities of individual neurons are distinguished from one another. By targeting specific types of neurons, spike sorting further reduces the output data rate compared to spike detection. Both methods convert the continuous neural signals into discrete event-based signals, such that periods without spiking activity do not require wireless transmission. In certain applications, however, it may be beneficial to transmit continuous neural activity rather than reducing it to spike events, which might limit the recorded information due to the spatial selectivity of neurons [27, 28]. For example, the local field potential (LFP) represents the activities of entire neuron populations, extending beyond those near the recording electrode. The LFP has been shown to encode information in various forms, such as the power at different frequencies or specific variations in the continuous voltage waveforms [29]. A notable advantage of these signals is that their information exists at relatively low frequencies, typically up to 500 Hz. For these types of signals, signal compression can be employed. Rather than using conventional compression schemes such as rate coding or difference encoding, this dissertation explores a machine-learning-based compression scheme that leverages the nature of neural signals and the characteristics of multi-channel recording arrays.

Secondly, the large amount of inter-subject variability poses significant challenges in developing robust neural decoding algorithms. For example, a neural decoding algorithm tailored to one person may not be applicable to another. Additionally, due to the non-stationary nature of neural activity, a decoding algorithm designed for a specific individual may require periodic calibration or updates to maintain acceptable performance over extended periods of time. To tackle these challenges, this dissertation presents the efficient design and implementation of two processor architectures capable of executing a wide range of neural decoding algorithms. The first processor employs fine-grained sequential processing to implement various machine-learning-based decoding models, including convolutional neural networks and recurrent neural networks. The second processor utilizes biologically-plausible neuron models to realize a generalized spiking neural network.

Thirdly, traditional BCIs require users to engage with the application during predefined time periods. One vital step toward transitioning BCI technology to practical implementations is enabling a mechanism by which the user can control the BCI engagement. Not only would this reduce unnecessary processing, but it would also restore user autonomy to individuals who currently require the assistance of a caregiver or clinician. Thus, a critical step toward realizing autonomous BCI systems is estimating the user's intention to engage with the BCI application. This is also referred to as the asynchronous processing paradigm, where the user controls when the BCI application is active or inactive. However, the process of intention estimation necessitates constant monitoring of the neural activity for specific biomarkers that indicate user intention. This dissertation explores two methods for efficient and low-power user intention estimation. The first method employs spikes to detect user intention, while the second relies on the LFP. Specifically, this dissertation details the algorithms for intention estimation and the digital circuit optimizations required to achieve constant monitoring with minimal overhead.

Finally, a hybrid asynchronous BCI is proposed that combines both spikes and LFPs to optimize intention estimation while maintaining neural decoding performance.

Overall, the approaches and implementations discussed in this dissertation provide a foundation for enabling low-power, autonomous BCI systems. By optimizing the design of brain-computer interfaces to meet practical use constraints, these advancements serve to extend device lifespan and enhance the overall user experience.

Chapter 2

In Vivo Neural Spike Detection with Adaptive Noise Estimation

2.1 Introduction

Damage to the spinal cord can disrupt the pathway of signals sent between the brain and the body and may result in partial or complete loss of both motor and sensory functions. The loss of these functions can have devastating implications on the quality of one's life. Currently, most patients living with paralysis require around-the-clock assistance to fulfill their daily tasks, which can be cost-intensive and deprive their sense of independence. Extraction of motor intent directly from the brain using brain-machine interfaces (BMIs) have shown to be promising in generating control signals for external assistive devices. However, complex brain processes are reflected by the activity of large neural populations and that the study of a few neurons provide relatively limited information [30].

Biological neurons communicate information among each other via electrical pulses, called action potentials or spikes. Conventional micro-electrode arrays (MEAs), such as Utah Array, are able to record from hundreds of electrodes [31]. Each electrode records spikes from multiple neurons close to the electrode's tip. In fact, the neural signal on a recording electrode is the cumulative electrical activity of various nearby neurons contaminated with noise, which is referred to as multi-unit activity (MUA), offering an extracellular recording. In general the background noise is a non-Gaussian random process due to various issues including electrode drift during operation, tissue-electrode interface noise, electronics noise, variation in the spike shape, the presence of overlapping spikes, and correlations between spikes and local field potentials. Using conventional MEAs, it is not possible to precisely place electrodes to individually record from a single neuron. Extracellular electrodes detect changes in electrical potentials from a vicinity of a neuron (about 140 micrometers) where generally tens of neurons are present [32]. Researchers, however, often require single-neuron activity for the study of how neurons are correlated with each other for specific stimulus [33, 34]. Also, the algorithms employed for accurate neural decoding typically process spike trains, which represent the action potentials of individual neurons over time [35].

For detecting the spiking activity of neurons, the continuous recorded analog signal by the MEA is first amplified and then converted into a digital signal. The digitized signal is then passed through a band-pass filter typically between 300 Hz and 3000 Hz. Frequencies below 300 Hz are filtered to remove low-frequency activity and the upper cutoff frequency is set to diminish the noisy appearance of the spike shapes [36]. After filtering, spike detection attempts to separate the high amplitude spike signals from the low amplitude neural background noise. The background noise can be either fixed to a specific value or it can be estimated dynamically. The noise threshold is often set to a scaled version of the background noise. Detecting neural spikes is performed in two phases, the pre-processing to emphasize the spikes from the background noise and then applying the estimated threshold to the pre-emphasized signal. Spikes are interpreted as occurring when the pre-emphasized signal crosses the noise threshold. The spike detection process can thus return the spike waveform itself in addition to the time an action potential occurs.

This chapter focuses on the development of an accurate spike detection module toward implantable in vivo operation that can adapt to changes in channel statistics autonomously. The rest of this chapter is organized as follows. Section 2.2 presents the impact of various filtering methods on the performance of spike detection as well as the commonly employed signal pre-emphasis algorithms and alternative techniques for noise estimation. The computational complexity and feasibility of the pre-emphasis and noise estimation techniques for in vivo spike detection are discussed and compared. Section 2.3 quantifies the performance of various combinations of the pre-emphasis and noise estimation methods for spike detection using the widely-employed WaveClus synthetic datasets [37]. Section 2.4 presents the design and hardware implementation of our designed detection technique with adaptive noise estimation. Section 2.5 quantifies the reliability of our designed and implemented spike detection module and its application in neural decoding. Finally, Section 2.6 makes some concluding remarks.

2.2 Filtering, Pre-processing, and Noise Estimation Algorithms

After the amplification and analog-to-digital conversion of the recorded neural signals, spike detection, which consists of filtering, pre-emphasis, noise estimation, and thresholding, is applied. Filtering is used to remove unwanted frequency components from the recorded neural signals. For example, low frequency local field potential (LFP), with the frequency of about 250 Hz, are removed. Pre-emphasis involves processing of the filtered neural signals to discern the neural activity from ambient background noise. The ambient background noise is estimated dynamically, which is used to derive a threshold value that the pre-emphasized neural signal must exceed in order to be detected as spikes.

From the perspective of in vivo signal processing, low-order filters are preferable due to their lower computational complexity and memory requirements. Using Matlab's FilterDesigner toolbox, we designed five candidate band-pass causal filters, Equiripple FIR, Butterworth IIR, Chebyshev Types I and II IIR, and Elliptical IIR filters. Each of the filters had the following characteristics: Sampling rate of 24 kHz, high-pass frequency of 300 Hz, low-pass frequency of 3000 Hz, 60 dB of attenuation in both stop bands, unity gain and 1 dB of ripple in the pass band, and the filter orders between 4 and 10.

Figs. 2.1 (a) and (b) illustrate the impact of causal and non-causal filtering, respectively, on the spike waveforms with various sixth-order filters. Non-causal filtering was realized using MATLAB's `filtfilt` function. It can be seen that the causal filter realizations reduce the spike amplitude and more importantly, impose a phase delay to the signal. In the case of the Cheby2 realization, the signal is severely attenuated when using a relatively low order of six. Fig. 2.1 (b) shows the benefit of utilizing non-causal filtering, which ensures zero phase shift. Also, the amplitudes and shapes of the spike waveforms are better preserved compared to the causal filtering. Figs. 2.1 (c) and (d) show the bloxplots of the Euclidean distance between actual spike waveforms and the filtered spikes employing causal and non-causal filtering, respectively.

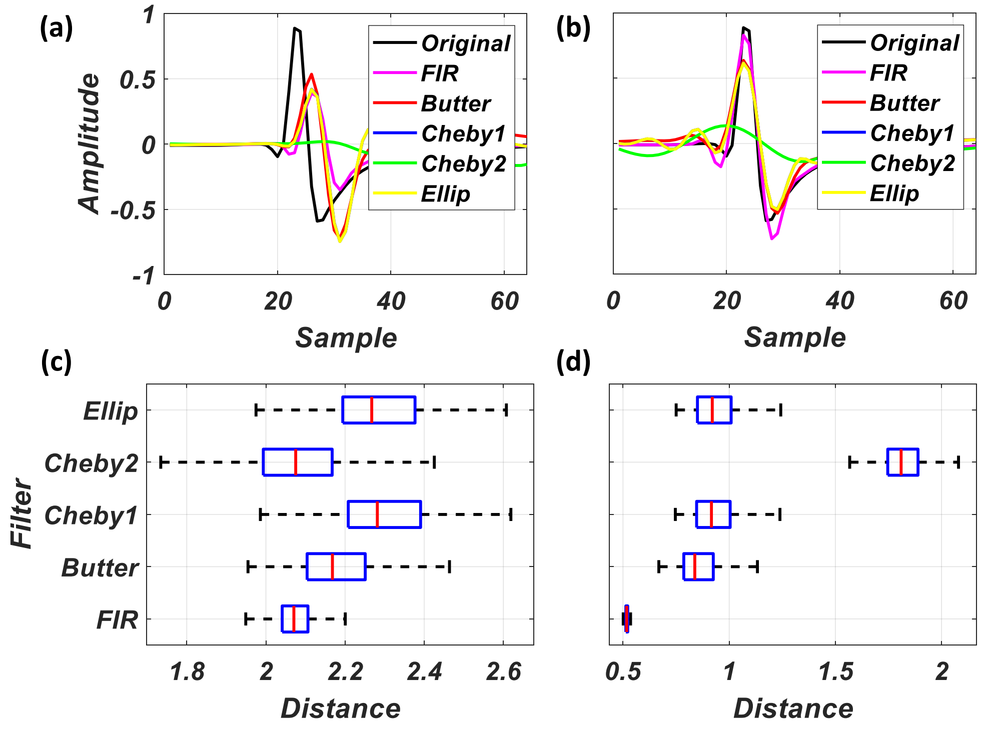


Figure 2.1. The impact of (a) causal filtering and (b) non-causal filtering on the spike waveform shapes. The boxplots (c) and (d) show the variations of the Euclidean dsitance between actual spike waveforms and the filtered spikes employing causal and non-causal filtering, respectively.

It is shown that the Equiripple FIR filter provides the least Euclidean distance among the various filter realizations.

In this work we consider two of the most commonly employed pre-emphasis methods, Non-linear energy operator (NEO) [38], and Absolute value (ABS) [37]. Energy-based signal pre-emphasis methods such as NEO accentuate the spikes by computing the energy of the signal. NEO is given as

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

where $\psi[n]$ denotes the energy of the signal $x[n]$, and is commonly employed as it amplifies spikes from background noise. The ABS pre-emphasis method is given as

$$\hat{x}[n] = |x[n]|, \tag{2.2}$$

where $\hat{x}[n]$ denotes the absolute value of $x[n]$. Depending on the type of pre-emphasis method, it may induce additional phase delay, and in some cases, no pre-emphasis is applied to the filtered neural signal. Figs. 2.2 (a) and (b) show the impact of causal and non-causal filtering, respectively, on the NEO pre-emphasized signal with various sixth order filter realizations. Figs. 2.2 (c) and (d) show the boxplots of the Euclidean distance between the actual and filtered energy waveforms utilizing causal and non-causal filtering, respectively. Note that in the energy domain, the causal filters perform similarly while the non-causal FIR filter retains the highest amplitude.

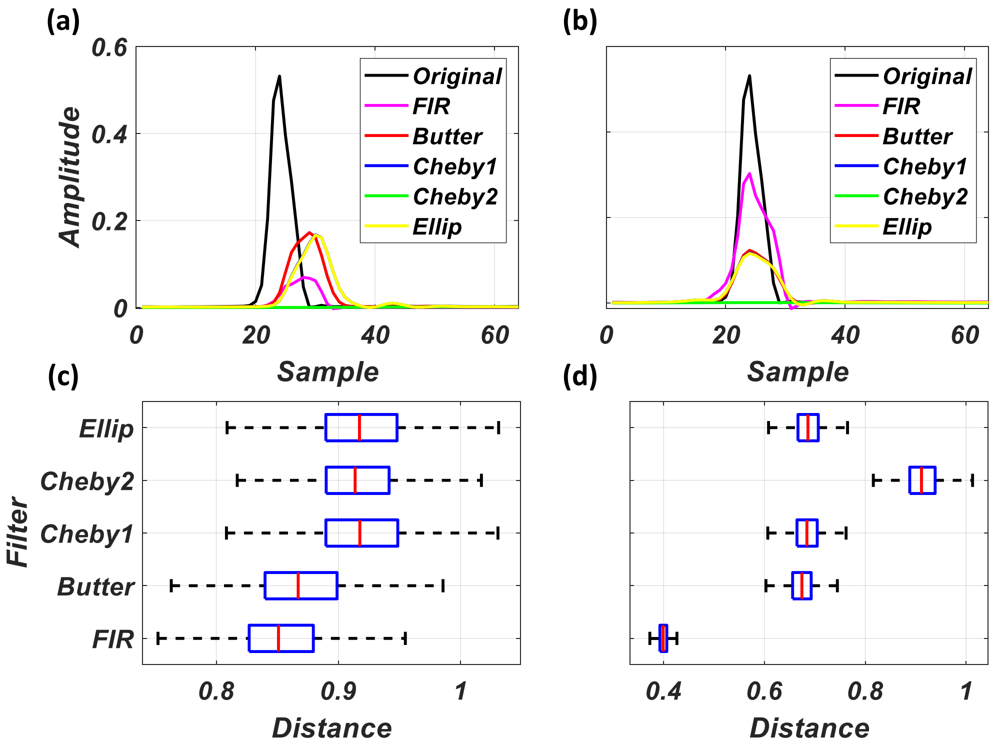


Figure 2.2. The impact of (a) causal filtering and (b) non-causal filtering on the spike energy waveform shapes. The boxplots (c) and (d) show the variations of the Euclidean distance between actual and the filtered energy waveforms employing causal and non-causal filtering, respectively.

While calculating the absolute value and the energy are trivial, estimation of the threshold is not. The scaling factor is often between 1.5 to 4 and is chosen empirically. In some applications, reducing the scaling factor to allow more noisy waveforms to be considered as spikes has resulted in a greater performance [39]. We used a constant scaling factor of 4, and the scaling factor was not tuned to optimize the performance of individual methods for a fair comparison among detection schemes. Thus, any process following the spike detection will interpret threshold crossings as if they were genuinely occurring spikes.

As threshold crossings are interpreted as spikes, the selection of the threshold is a crucial decision in the detection process for two main reasons: (i) the threshold should be set sufficiently high such that noisy or erratic behavior in the neural signal is not interpreted as genuine spiking activity; and (ii) the threshold should be accurate enough so that the threshold crossings behave similar to genuine spike trains.

In this chapter, we consider four alternative noise estimation methods, Median (MAD) [37], Root-mean square (RMS) [40], Ada-BandFlt (ABF) [41], and AdaFlt (AF-128) [41]. The Median (MAD) is a commonly employed algorithm for estimating the noise and is given as:

$$\sigma_e = \text{med}\big(\frac{|x|}{0.6745}\big),$$

where med denotes the median of the signal and 0.6745 denotes the 75th percentile of the standard normal distribution [37]. The median of the absolute value of the neural signal reduces the interference of spikes based on the assumption that spikes seldom appear in the signal. The median is often computed over a relatively long recording, on the order of one minute. However, storing relatively long neural recordings is infeasible for in vivo BMIs, in which the area- and energy dissipation constraints severely limit the amount of memory storage and the algorithms that can be realized.

An alternative noise estimation methods is the root mean square (RMS) [40], where the estimated noise is given as:

$$\sigma_e = \sqrt{\frac{1}{M}\sum_{i=0}^{M}\left[x_i^2\right]}, \tag{2.3}$$

and M denotes the number of samples of the band-pass filtered and pre-emphasized signal x. When estimating the noise using the RMS, it is a common practice to perform summation over one second. The spike threshold is typically considered as a scaled version of the estimated noise, such as $2\sigma_e$ or $4\sigma_e$. In practice, the square root operation can be avoided by squaring the pre-emphasized neural signal when compared to the estimated noise as $\hat{x}[n]^2 > \sigma_e^2$, where $\hat{x}[n]$ denotes the pre-emphasized neural signal.

The BandFlt algorithm [41] is similar to the RMS algorithm, but rather than a single window of one second, it utilizes 300 ten millisecond windows. The RMS is computed for each window and then sorted in ascending order. An improved version of the BandFlt algorithm is the Ada-BandFlt, where the input signal is split into 10 ms sample windows and the RMS is computed for each window [41]. The RMS values are then collected for 100 windows and an initial noise estimation is set as the 25th percentile of the distribution of RMS values of the 100 windows. After this initial noise estimation, the estimated noise is updated every following 250 ms as:

$$\sigma_e(n) = 0.8 \times \sigma_e(n-1) + 0.2 \times M_{0.25},$$

where $M_{0.25}$ denotes the 25th percentile of the RMS values.

The AdaFlt algorithm [41] works on 128 ten millisecond windows. The maximum and minimum values of each window are found and the 40th percentile of the maximum and minimum values are computed, and each of these are taken as initial estimates for positive and negative thresholds, respectively. Following the initial estimation, AdaFlt calculates one set of maximum and minimum values once for every ten millisecond windows. After 128 sets of maximum and minimum values are calculated, the noise estimates are given as:

$$\sigma_{p,e}(n) = 0.9 \times \sigma_{p,e}(n-1) + 0.1 \times M_{0.4},$$

$$\sigma_{n,e}(n) = 0.9 \times \sigma_{n,e}(n-1) + 0.1 \times m_{0.4},$$

where $M_{0.4}$ and $m_{0.4}$ denote the 40th percentile of maximum and minimum values, respectively.

Table 2.1 gives the computational complexity of four different noise estimation methods

Table 2.1. The computational complexity of various noise estimation methods over a one second window

Method	Adds./Mults. per window	Comparisons per window	Update frequency	Ops/s.
MAD	1	f_s^2	1 Hz	f_s^2+4
RMS	f_s-1	0	1 Hz	$6f_s+2$
ABF	$f_s/500$	$3f_s^2/10e3$	4 Hz	$(3f_s^2+60f_s)/2.5e3-14$
AF–128	$f_s/500$	$1.28f_s^2/10e3$	100 Hz	$(1.28f_s^2+60f_s)/100-580$

over a one second window. For algorithms that require sorting to compute the median, we assumed that sorting M values requires an average of M^2 comparisons [42]. To normalize the computational complexity of different operations, we estimated the complexity of addition and multiplication as 2 and 4 times the complexity of a comparison [43]. Note that while the RMS requires the most number of addition and multiplication operations, the lack of sorting significantly reduces the required number of operations per second. For example, for a sampling rate of $f_s = 10$ kHz, the MAD, RMS, ABF, and AF-128 algorithms require approximately 100e6, 60e3, 120e3, and 1.3e6 operations per second, respectively. However, we can measure the performance of the noise estimation method as the accuracy of the threshold crossings. However, the input to the noise estimation is the pre-emphasized signal. Thus, there are various combinations of signal pre-emphasis and noise estimation methods and we will evaluate their performance in the next section.

2.3 Performance of In Vivo Potential Spike Detection Techniques

To evaluate the performance of the candidate pre-emphasis and noise thresholding methods, we use the widely employed WaveClus datasets, which consists of twenty simulated neural recordings with three single-units in each recording. There are four difficulty levels: Easy1, Easy2, Difficult1, and Difficult2, which refers to the similarity of the three single-unit spike waveforms present in the recordings. The simulated recordings have varying levels of noise with a standard deviation between 0.05 and 0.20 (up to 0.40 for Easy1) relative to the amplitude of the spikes. The recordings are first simulated at 96 kHz and then down-sampled to 24 kHz. These datasets have been widely used as they also contain annotated spike timings and the classes of each spike, referred to as ground truth information.

The performance of the candidate spike detection methods can thus be given by comparing it's interpreted detected spike times to those present in the ground truth dataset. Using the ground truth information, we can evaluate whether a spike detected at time t_i is a genuine spike (i.e., a true positive TP), or an errant/noise spike (i.e., a false positive FP). Also, ground truth spikes that are missed by the detector can be denoted as missed spikes (i.e., a false negative FN). For our analysis, a spike is taken as a true positive if it is detected within two milliseconds of the ground truth spike time. Some commonly employed spike detection measures involve computing the probability of detection P_d, the probability of missed spikes P_m, and the probability of false alarms P_{fa}, which are defined as $P_d = \frac{TP}{TP+FN+FP}$, $P_{fa} = \frac{FP}{TP+FP}$, and $P_m = \frac{FN}{TP+FN}$, respectively. Some literature also uses the sensitivity metric $\frac{TP}{TP+FP}$, which quantifies the ratio of valid spikes given all detected spikes. A sensitivity value close to one represents that a large portion of the detected spikes are genuine. However, a high sensitivity value in conjunction with a high number of false positives may induce an artificially high sensitivity value. Another measure is the F-Score metric, defined as $F = \frac{TP}{TP+0.5(FN+FP)}$, which is a combination of the recall and precision of detection. For a fair comparison with the state-of-the-art work, the F-Score is our chosen metric in this chapter. For the first analysis, we employed a relatively-low order a fourth order Elliptical IIR filter is employed, following conventional low-order filtering methods in BMI systems [21, 11]. If alternative filters have relatively similar frequency and phase characteristics, then it follows that the performance differences among the alternative detection methods is primarily related to the employed detection methods. Table 2.2 gives the mean F-scores of the candidate detection schemes over the four WaveClus datasets. It is apparent that the combination of the NEO with the RMS or the ABF outperforms other candidate methods. It can also be seen that the NEO with RMS offers the lowest standard deviation across the datasets, offering a robust performance compared to the alternative methods. While the ABF noise estimation method

Table 2.2. The mean F-scores of the candidate detection methods over the WaveClus datasets

Detection method	Mean F-score	Standard deviation
NEO + MAD	0.45	0.012
NEO + RMS	0.92	0.001
NEO + ABF	0.94	0.002
ABS + MAD	0.91	0.034
ABS + RMS	0.89	0.104
ABS + ABF	0.91	0.075

performs well, it's requirement of sorting 100 RMS values makes it infeasible for efficient, real-time in vivo realization.

For the NEO and RMS-based detection, we also quantify the impact of alternative filters on spike detection performance by computing the F-Scores over the WaveClus datasets with ground-truth information. Figs. 2.3(a), (b), (c) and (d) show the performance of the NEO and RMS-based spike detection using different filters over the Easy1, Easy2, Difficult1, and Difficult2 datasets, respectively. It is shown that the FIR, Butterworth, Chebyshev Type I, and Elliptical filters outperform the Chebyshev Type II filter. Interestingly, all filters other than Chebyshev Type II do not necessarily see improved performance for increased filter order. While we found that non-causal filtering is preferred for retaining waveform shapes, such details are not relevant for applications that only require spike timings, such as BMIs that employ MUA threshold crossings rather than single-unit action potentials. While the NEO pre-emphasis reduces the amplitude of the spike waveforms, as shown in Fig. 2.2, it was also shown that causal filtering further reduces the amplitude of the spike waveforms. Figs. 2.4 (a) – (d) show the F-scores of employing fourth order causal and non-causal Equiripple FIR filters for various noise levels over the Easy1, Easy2, Difficult1, and Difficult2 datasets, respectively. It can be seen that the difference between the two filtering approaches is insignificant. The appeal of the low-order causal filtering is due to their relatively low computational complexity and memory requirements. Additionally, from the hardware implemenation perspective, the causal realization is applicable for real-time processing. Therefore, for the BMI applications where the integrity of waveform shapes is not a concert and only threshold crossing rates are required, using a relatively low-order causal FIR filter is adequare.

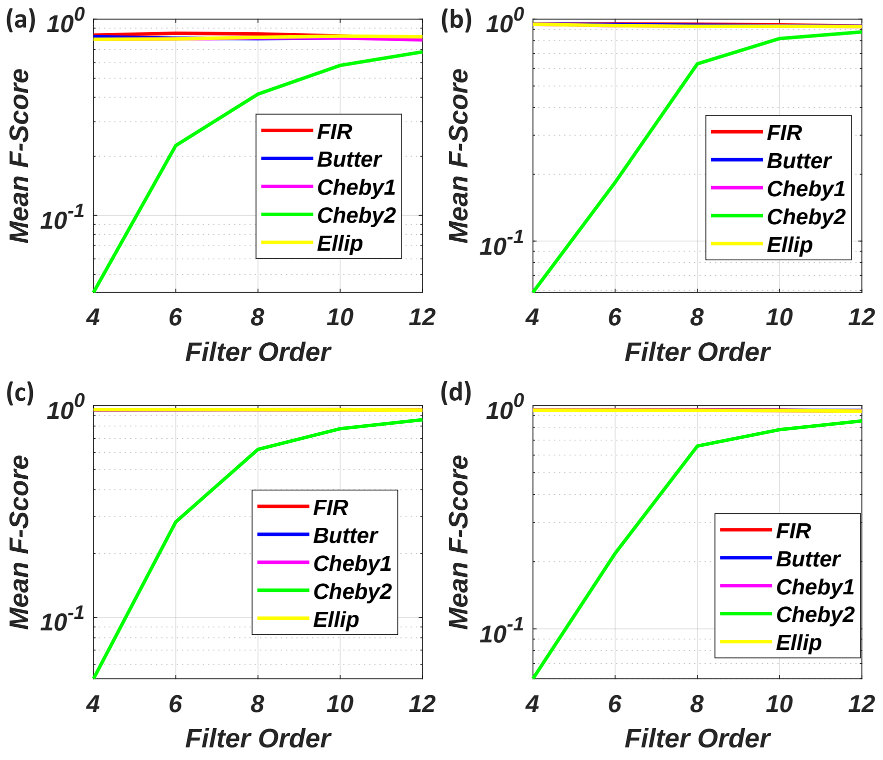


Figure 2.3. The F-scores of the NEO and RMS detection with various filters for the (a) Easy1, (b) Easy2, (c) Difficult1, and (d) Difficult2 datasets, respectively.

Note that the F-score metric measures the ratio of true spike detections over all detected spikes and is dependent on the application for which a value can be deemed acceptable. For example, in the case of spike sorting, acceptable values are 0.8 or higher, i.e., the majority of detected spikes are geniune. However, for neural decoding, it has been shown that voltage threshold settings can be tuned to optimize the encoding of movements' kinematics [39]. Thus, detecting more false positives as spikes and hence, a smaller F-score, may yield better a better decoding.

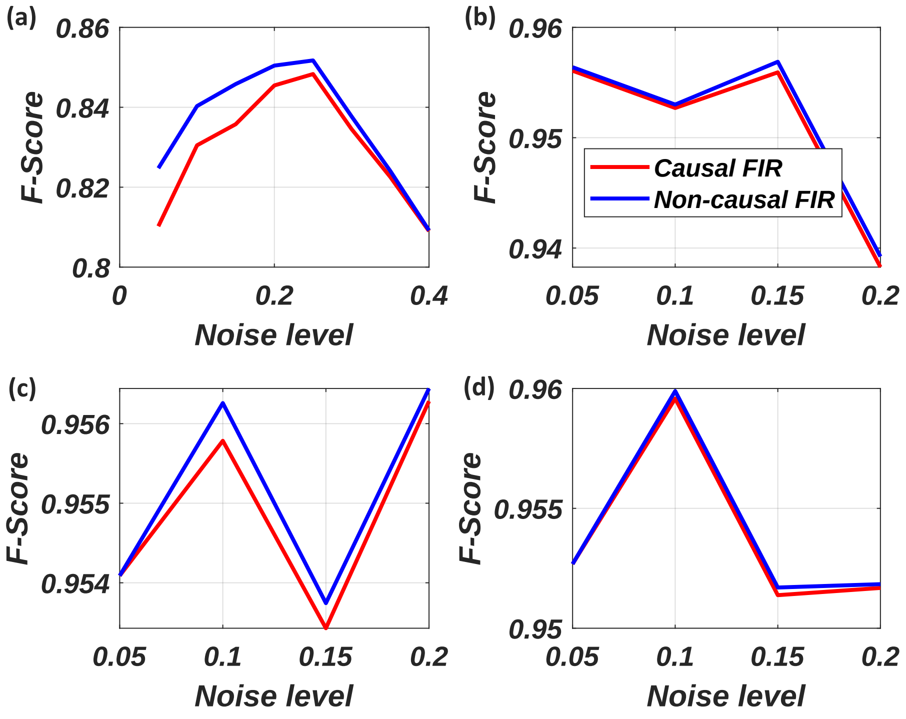


Figure 2.4. The F-Scores of the fourth-order causal and non-causal Equiripple FIR filter over the (a) Easy1, (b) Easy2, (c) Difficult1, and (d) Difficult2 datasets.

2.4 Hardware Realization of the Candidate Detector

Based on the discussed findings, we suggest that using the NEO pre-emphasis with the RMS thresholding provides the optimal detection performance.

In vivo spike detection must work with limited computational resources that are shared among hundreds to thousands of channels. For efficient hardware realization of the RMS thresholding, Equation (2.3) is simplified to avoid the division and square root operations. The division is replaced with an arithmetic shift operation and the square root is avoided by squaring the thresholding inequality $x[n]_{NEO} > \sigma_e$. The simplified noise estimation is then given as:

$$\sigma_e = \sum^{M}(x_{NEO}^2) >> \left(\log_2(\frac{M}{C^2})\right), \tag{2.4}$$

where M denotes the number of samples to accumulate before computing the estimated noise σ_e, C denotes the noise scalar, and $x >> y$ denotes the arithmetic right shift of x by y bit positions.

To employ the right shift operation, the values of M and C are constrained as powers of two. To quantify the impact of the applied simplification, we computed the F-Scores for the Easy1 and Difficult1 datasets over values of M ranging between 8 and 15, with the scalar $C = 4$. As shown in Figs. 2.5(a) and (b), the performance difference between the one second RMS window and the simplified Equation (2.4) is negligible for values of $M \geq 13$. In some cases, the performance of the simplified computation outperforms that of the fixed window. To demonstrate the noise adaptation over a real dataset, a single channel of data from the 'indy_20170131_02' dataset is passed to the NEO and RMS-spike detector using the Xilinx Vivado simulator. Fig. 2.6 shows the continuous update of the RMS noise threshold values in response to the signal's variations. For example, when the signal increases in amplitude, the noise threshold is increased to avoid detecting larger noise values as spikes. The noise threshold, shown as a sample-and-hold line, indicates that the RMS changes occur in fixed intervals (i.e., over 2^{13} samples). For clarity, a smoothed version of the noise adaptation using an eight-sample moving average filter is also shown.

The datapath of the designed multi-channel NEO and RMS-based spike detector is shown in Fig. 2.7. The datapath consists of two main components: the squaring NEO unit and the RMS estimator. The squaring NEO unit has an input signal shift register *CBuf*, radix-2 booth-encoded multipliers *B* to compute the NEO value, followed by a fixed-point multiplier to compute the square of the NEO signal. The depth of the shift register *CBuf* is equal to $2N$, where N denotes the number of supported channels. Rather than reusing the datapath by increasing its operating frequency, we instead choose to realize multiple datapath units that each process a relatively low number of channels. One caveat, however, is that implementing the squared NEO would be costly, especially when instantiated several times across datapath instances. We thus implement two of the three NEO multiplications with sequential radix-2 booth-encoded multipliers [44], as shown in Fig. 2.8. The RMS Estimator consists of a set of N registers for accumulating the squared NEO values of each channel, as well as a set of N latches for storing the RMS value once every M input samples, as per Equation (2.4). A control unit *CTRL* manages when the

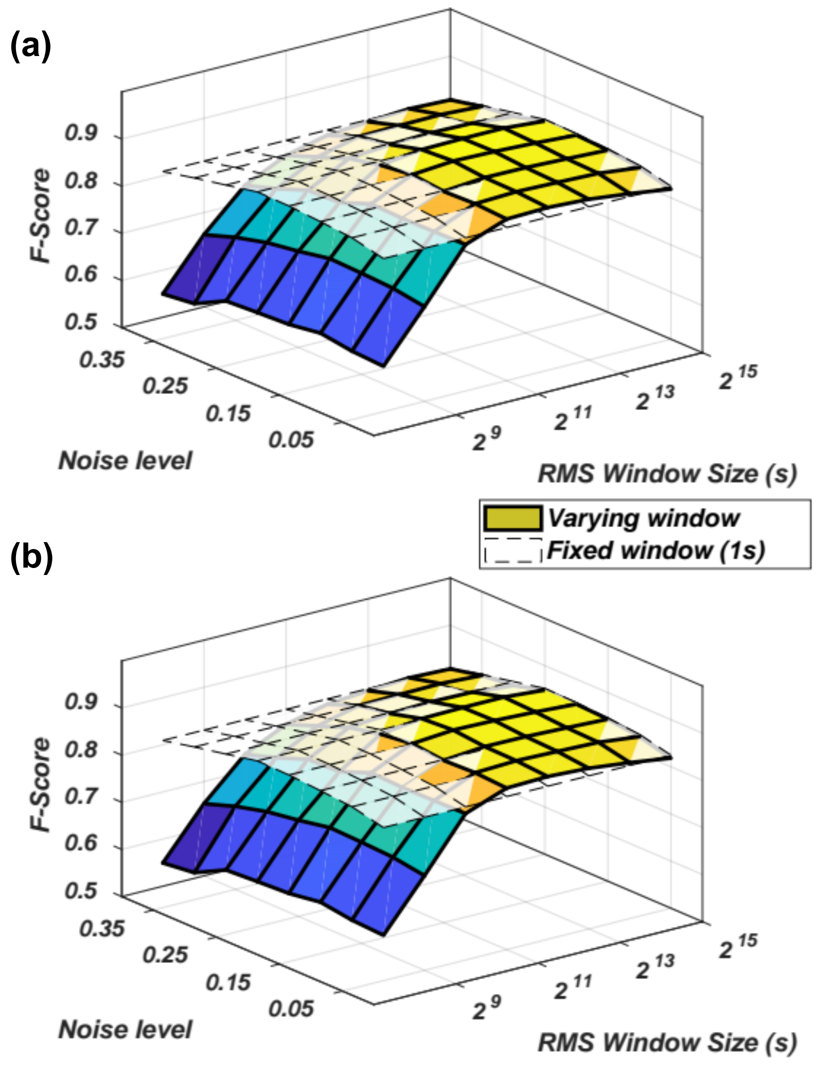


Figure 2.5. The F-Score of the NEO and RMS-based spike detector for the (a) Easy1 and (b) Difficult1 datasets over various noise levels and varying RMS window sizes using the simplified Equation (2.4).

booth-encoded multipliers, accumulator registers, and RMS value latches are enabled for writing.

We have designed and implemented the NEO and RMS-based spike detector in a standard 180-nm CMOS process with 16 datapaths processing 8 channels each, for a total of 128 channels. The ASIC layout, shown in Fig. 2.9, is estimated to consume 639 μW of power from a 1.8V supply when operating at 800 KHz and is estimated to occupy 3.44 mm^2 of silicon area. The implemented spike detector thus consumes 4.9 μW per channel and occupies 0.02 mm^2 of silicon area per channel. Note that the designed and implemented NEO and RMS-based spike

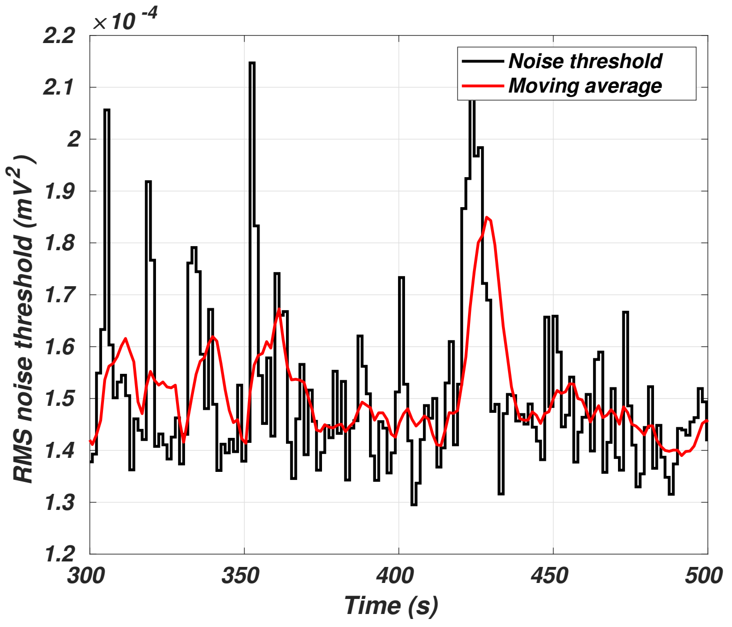


Figure 2.6. Adaptive noise threshold estimation.

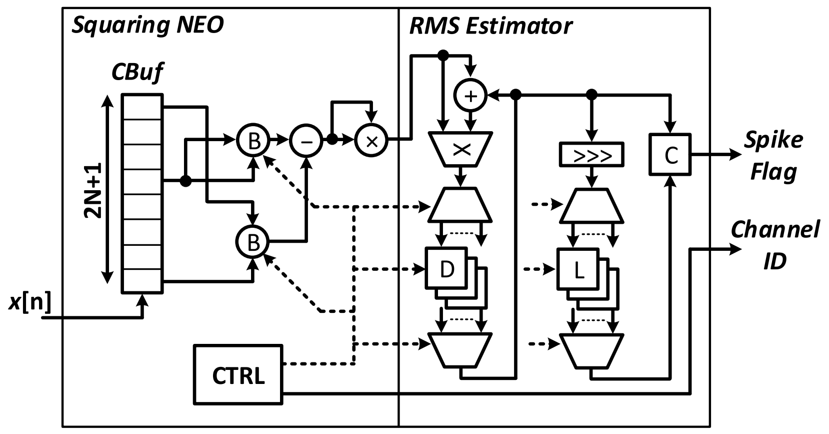


Figure 2.7. The datapath of the multi-channel NEO and RMS-based spike detector.

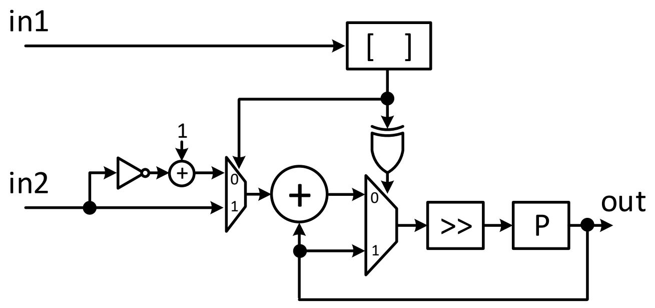


Figure 2.8. The datapath of the sequential radix-2 booth multiplier.

Table 2.3. The characteristics and implementation results of various spike detection ASICs

Work	Ours Booth	Ours Approx.	[45]	[18]	[46]	[19]
Technology (nm)	180	180	180	180	65	130
Implementation	Digital	Digital	Analog	Analog	Analog	Digital
Clock frequency (KHz)	800	80	–	24	20	800
Supply voltage (V)	1.8	1.8	–	1.8	0.8	1.2
Channels	128	128	16	1	64	64
Power per Ch. (μW)	4.9	0.64	4	1.5	1.21	3.04 – 4.54
Area per Ch. (mm^2)	0.02	0.02	–	0.03	0.0105	–
Pre-emphasis	NEO	NEO	None	NEO	NEO	None
Thresholding	RMS	RMS	Fixed	Peak	Fixed	Dual-Mean
Adaptive	Y	Y	N	Y	N	Y
Scaled power per Ch. (μW)†	4.9	0.64	4	1.5	4.67	7.53 – 11.26
Scaled area per Ch.†	0.02	0.02	–	0.03	0.05	–
Scaled energy per Ch. (pJ) †◇	6.24	0.8	–	1.87	5.83	9.4 – 14

† Scaled to a 180-nm process with a 1.8 V supply voltage, as described in [47].
◇ Normalized to a clock frequency of 800 kHz.

detector supports arbitrary window sizes of 2^B while the threshold is updated more frequently for smaller window sizes. Synthesis was performed with Synopsys Design Compiler and the place and route was performed with Cadence Innovus. After routing, the netlist is simulated to obtain a realistic switching activity for estimating the dynamic power consumption. The NEO and RMS-based spike detector has a power density of 18.58 mW/cm^2, which is within the tissue safe constraints (i.e., 40 mW/cm^2) [17]. We have also implemented an alternate version of the NEO and RMS spike detector in the same 180-nm CMOS process by replacing all multiplications with approximate log-based multipliers [48]. As opposed to the serial Booth-encoded multipliers, the log-based multipliers do not require an increased operation frequency and can operate at only 80 KHz, which significantly reduces the power consumption. The approximated NEO and RMS-based spike detector only consumes 0.64 μW of power per channel from a 1.8 V supply and occupies 0.02 mm^2 of silicon area. The power density of the approximated design is only 3.07 mW/cm^2, which is well within the tissue safe constraints.

Various hardware realizations of neural spike detection have been reported recently [45, 18, 46]. Table 3.3 lists the characteristics and implementation results of various spike detection ASICs. For a fair comparison, the implementation results have been scaled to a

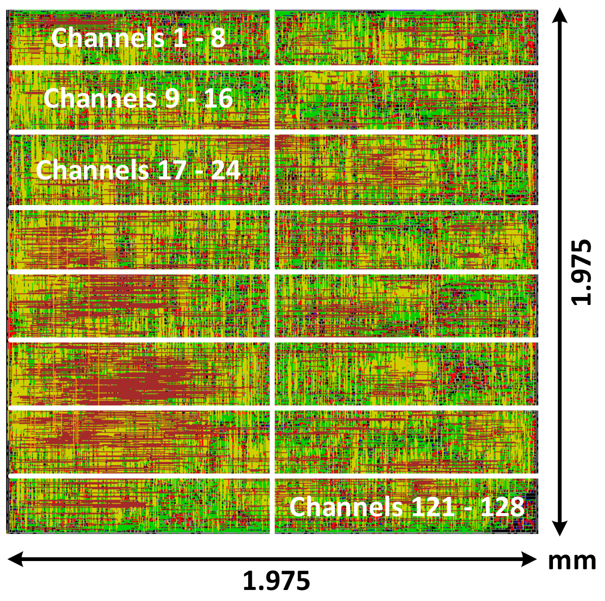


Figure 2.9. The ASIC layout of the designed 128-channel NEO and RMS-based spike detector.

180-nm CMOS process with a 1.8V supply voltage, as described in [47]. In [45], the authors present a 16-channel BMI with a digital implementation of window discriminator-based spike detection. Window discriminators involve two threshold values and detect a spike when an action potential crosses an upper and lower threshold, which correspond to the de-polarization and re-polarization of the spike waveform, respectively [49]. Unfortunately, the two threshold values are not adaptive to changes in the signal during run-time, and are programmed through a communication interface. In [18], the authors present an analog implementation of a NEO-based spike detector. The threshold is considered as the peak value of the NEO pre-emphasized signal, and it adapts to the signal if new values of the NEO signal exceed the current peak value. In [46], the authors present a 64-channel neural signal acquisition system-on-chip (SoC). Spikes are detected using a NEO-based pre-emphasis and a fixed threshold that is uploaded to the SoC, and unfortunately cannot adapt to real-time changes in channel statistics.

It can be seen in Table 3.3 that our designs are among the most power- and area-efficient spike detection circuits, while supporting adaptive noise estimation. Compared to the design in [18], which also employs the NEO pre-emphasis, our digital design naturally consumes more power, however, our design offers a slightly lower probability of detecting false positive spikes over the same WaveClus datasets. It can be seen in Table 3.3 that the NEO and RMS detector utilizing approximated multipliers dissipates less power than the detector using booth-encoded multipliers, and also consumes the least energy per channel among the state of the art spike detection ASICs. While power is a commonly employed metric for comparing ASICs, energy consumption is a vital metric for in-vivo BMIs, as it directly impacts the battery life for implantable circuits. For example, an in-vivo detector based on booth-encoded multipliers operating at 800 KHz, employing the SAFT LS14250 battery with a nominal capacity of 1200 mAh, would operate for approximately 433×10^3 hours, while the approximate-based NEO and RMS detector would operate for approximately 3.37×10^6 hours. The designs in [18, 46, 19] would operate for approximately 1.44×10^6 hours, 463×10^3, and 192×10^3 – 287×10^3 hours, respectively.

2.5 Spike Detection Performance Analysis

In a BMI system, the output of the spike detection is used by the subsequent neural signal processing modules. For example, spike sorting groups individual neurons' spikes into individual clusters [49]. Additionally, the dynamics of groups of neurons and behavioral data, such as motor movement [50], can be explored. Neural decoding translates spiking information into a quantifiable representation, such as movement kinematics for a robotic prosthesis [51]. We assess the performance of the candidate spike detection methods in a neural decoding task using trial-based behavioral recordings without ground truth information.

The advantage of synthetic datasets is that we can compute the F-Score as a quantitative measure for assessing the performance of the detection schemes due to the known times of action potentials, referred to as "ground truth" information. A real neural recording usually does not offer ground truth information and hence, it is subjective when comparing the performance among different spike detection methods. One approach to verify the spikes of real recordings is to use intra-cellular recordings, which senses the voltage inside the neuron itself [52]. When simultaneous intra- and extra-cellular recordings are available, the spike times detected in the extracellular recording can thus be verified using the intracellular recording [53]. Unfortunately, intracellular recording is challenging due to the lack of stability and long-term reliability of the recordings [54]. One useful method when employing high-density MEAs is to exploit the spatio-temporal correlations of spikes detected at different recording sites [55]. However, this is not applicable for a low-density clinical recording interface. Another method that can be used to quantify the performance of a spike detector is to employ a widely accepted toolset, such as WaveClus [37], as a “gold standard” [21]. The detected spike times from such tools could be used as a reference label to evaluate the probability of detection, missed spikes, and false alarms of alternative detection methods. One caveat, however, is that due to the nature of neural recordings, these tools cannot offer guaranteed true labeling of data since users need to set different parameters for different neural datasets. Due to the use of MUA spike trains for BMI applications, another method to quantify the performance of various spike detection methods is to model the separability and distinctness of spike trains for different source stimuli. For example, neurons in the motor cortex modulate their firing rates in response to specific movement direction [56]. Thus, a spike detection that can produce spike trains with activity more accurately correlated with the intended stimulus/action is preferable for BMI applications.

2.5.1 Spike Detection Performance Analysis Over Real Neural Recordings

A spike train can be considered as a stochastic point process that consists of binary events in time. The conditional intensity function (CIF) $\lambda(t|H_{t-})$ of a point process provides a complete probability model of the process and describes the instantaneous firing probability conditioned on its firing history. Following the approach in [57], we can compute the CIF for a channel of neural data over various repeated trials of a specific stimulus S_i. Then, we can compute the probability $P(N(t)|S_i)$ of observing a particular set of spike times $N(t)$ given the stimulus S_i.

For this analysis, we use the raw broadband data recorded from an adult male Rhesus macaque monkey in the Sabes Lab at the University of California, San Francisco [58]. The monkey was trained to perform self-paced reaches by controlling the position of a cursor on a screen. Data was recorded from area M1 of the primary motor cortex using a chronically implanted 96-channel Utah Array. Spikes were detected using the candidate spike detection methods described above. Using the relative starting and ending positions of the cursor target, we discretized the monkey's reach into four possible directions: left and upwards, left and downwards, right and upwards, and right and downwards. The detected spike times were also discretized as follows. The spike times were converted into spike trains $N(t)$ into K bins of width $\Delta = TK^{-1}$, where T denotes the total time of each reach, in our case aligned to the final 500 ms before the monkey finished its reach. The spiking activity during a reach is thus given as $N_{1:k} = [\Delta N_1, ..., \Delta N_k]$, where ΔN_k is one if there is a spike in the time interval $((k-1)\Delta, k\Delta)$ and zero otherwise. The discretized CIF $\lambda(k\Delta|N_{1:k-1})$ is given as the mean of the spike history $N_{1:k-1}$ divided by Δ. The probability of the observed spike train $N(t)$ conditioned on the presented stimulus S_i can be written as:

$$P(N(t)|S_i) = \exp\Big[\ \textstyle\sum_{k=0}^{K} log(\lambda(k\Delta|N_{1:k-1})\Delta)\Delta N_k \\ - \textstyle\sum_{k=0}^{K} \lambda(k\Delta|N_{1:k-1})\Delta\Big].$$

After computing the mean CIF for all four reach types, the probability of each trial conditioned on each of the four mean CIFs, i.e., $P(N(t)|S_1)$, $P(N(t)|S_2)$, $P(N(t)|S_3)$, and $P(N(t)|S_4)$ are computed, where $S_1 - S_4$ denotes reach types left and upwards, left and downwards, right and upwards, and right and downwards, respectively. We then compute a log likelihood feature vector given as $Y_{N(t)} = \log\left[P(N(t)|S_1), ..., P(N(t)|S_4)\right]$. $Y_{N(t)}$ can thus be considered as a projection of the spike train $N(t)$ onto the likelihood space of the four reach types. Thus, for each reach, there is a feature vector that represents a point in the likelihood space. A more useful spike detection yields a well-defined set of clusters for each type of reach. To quantify the resulting likelihood space clusters of projected spike trains, we employ the Calinski-Harabasz (CH) metric [59],

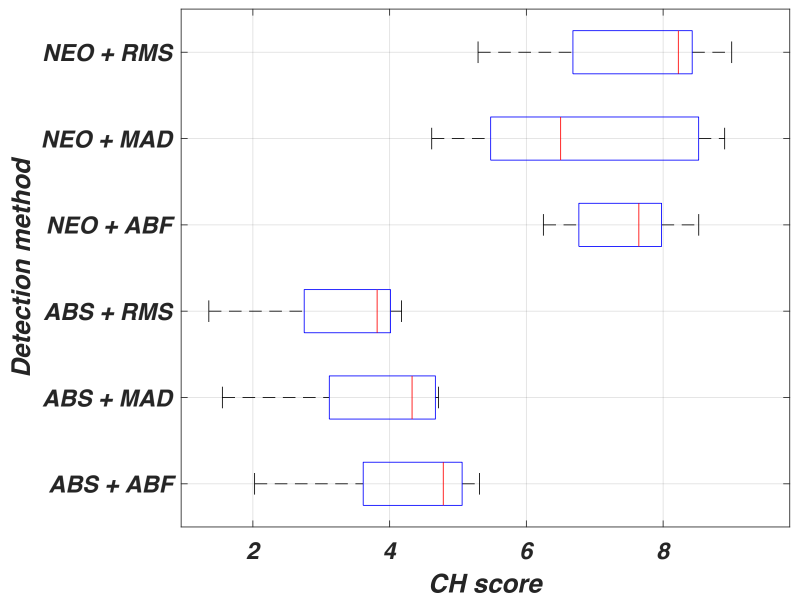


Figure 2.10. The CH score boxplots of the projected spike trains using various spike detection methods for the macaque monkey's reaching tasks.

which is defined as the ratio of the distances among clusters and the distances within a cluster. For the spike train analysis, we employed six spike detection methods using five 'indy' datasets, 20161220_02, 20170123_02, 20170124_01, 20170127_03, 20170131_02. The spike trains were partitioned into sets associated with the respective reach type for a particular movement. Fig. 2.10 shows the CH score boxplots of the projected spike trains generated by the candidate detection methods over the five datasets. It can be seen that the NEO-based methods outperform the ABS-based methods. Also, one can note that the combination of the NEO with RMS yields a higher median CH score. While the NEO and ABF method has a smaller variability, the computational complexity of the RMS is approximately half of the ABF's complexity, as given in Table. 2.1. This implies that the projected spike trains generated by the NEO and RMS detection produce better defined clusters, which may be perferable for neural decoding.

2.5.2 The Impact of Spike Detection on Neural Decoding Applications

To study the effect of alternative spike detection methods on the performance of neural decoding, we utilize the publicly available hc-2 dataset [60]. The dataset consists of the neural

Table 2.4. The validation and testing performance of the GRU-based RNN decoder using various potential in vivo spike detection methods

Detection method	Validation (R^2) score	Testing (R^2) score
ABS + MAD	0.73	0.75
ABS + RMS	0.79	0.76
NEO + MAD	0.73	0.74
NEO + RMS	0.85	0.86

recordings from the CA1 region of the hippocampus of freely moving rodents over various experiments. The rodents were provided with water or food as a reward at random locations throughout a platform. The positions of the rodent was determined using video footage by tracking the position of LEDs on the rodents' heads. The aim of decoding is to predict the location of the rodent based on the spiking activity of neurons in the hippocampus. The tip of each recording shank, which are the channels with the highest signal amplitude, were used for detecting spikes using the candidate methods. We designed and trained a gated recurrent unit (GRU)-based recurrent neural network (RNN) decoder to map the binned spike counts onto the rodent's positions. The training data contained 7772 samples, each representing 1.92 seconds of data over 75 bins (i.e., bin size = 25.6 ms). The training data was partitioned into 80% for training, 10% for validation, and 10% for testing. The validation data is used to evaluate the performance of the model on data not observed during training, while the testing data is used to evaluate the performance of the model after training. The RNN was trained using the Tensorflow framework for up to 250 epochs, using early stopping on the R^2 metric to avoid over-fitting to the training data. The R^2 metric, also known as the coefficient of determination, quantifies the amount of variance in the dataset that can be account for by the model, with a score of 1.0 being perfect. Table 2.4 gives the testing performance of the GRU-based RNN over various spike detection methods. One can see that the combination of NEO and RMS offers the highest performance for both the validation and testing sets.

2.6 Conclusion

This chapter investigated the efficiency of various potential spike detection techniques for in vivo implantation. It was found that the non-linear energy operator (NEO)-based pre-processing in combination with the root mean square (RMS) noise estimation outperforms other state-of-the-art spike detection methods. It was shown that the combination of NEO with RMS resulted in the highest F-Score when evaluated on the commonly employed WaveClus datasets. The design and implementation of a 128-channel NEO and RMS-based spike detector in a standard 180-nm CMOS process was presented. The synthesized NEO and RMS spike detector was estimated to occupy 0.2 mm^2 of silicon area and consume 4.9 μW of power from a 1.8 V supply while operating at 800 KHz. It was shown that our design is among the most power, area, and energy-efficient spike detectors. The designed and implemented spike detector was also employed for neural decoding and it was found that the NEO and RMS-based detection also offers the highest decoding performance across two different animal behavioral datasets.

2.7 Publications

The following article was published as an outcome of the research conducted in this chapter: D. Valencia, P. P. Mercier, and A. Alimohammad, "In vivo neural spike detection with adaptive noise estimation," *Journal of Neural Engineering*, vol. 19, 046018, July 2022.

Chapter 3

Neural Spike Sorting using Binarized Neural Networks

3.1 Introduction

The human brain is composed of billions of neurons, communicating with one another via electro-chemical processes. The brain is responsible for nearly all functional behavior of the human body, from controlling the movement of limbs to regulating one's breathing. Neuro-degenerative diseases and spinal cord injuries can impede the brain's natural ability to communicate with the limbs and/or organs and hence, imposing severe limitations on patients' bodily functions. Neuroscientists are keen on studying the behavior of neurons, how their electro-chemical activities are correlated with one another, and exploring how to form alternative neural pathways on a patient's brain.

The brain's neurons fire action potentials, also known as spikes, when their cumulative input activity exceeds some threshold. The brain can then decode the spiking activity to control the movement of a limb or muscles responsible for respiration. One approach to overcome the degeneration of neural connectivity is to use brain-machine interfaces (BMIs) to form alternative pathways for neural signals. In recordings preformed by large and dense multi-electrode arrays (MEAs), an electrode may record combined signals from multiple neurons nearby and background noise. Spike sorting is the process of separating the electrical activity of individual neurons from noisy recorded signals [61]. Conventionally, spike sorting is performed offline using a computer or a machine (i.e., in-silico). The recorded neural signal is first filtered between 300 and 3000 Hz to remove background noise. Spikes are then detected from ambient neural noise. Specific features of the spike waveforms are then extracted and used as inputs to a clustering algorithm, which groups or classifies similar spikes belonging to specific neurons.

The hardware realization of accurate spike sorting systems allows neuro-scientists to examine the spiking behavior of individual neurons. Recent hardware implementations of spike sorting aim to perform the required neural signal processing algorithms either as a supervised process, which requires some level of initial offline processing [62], or as a fully automated unsupervised process [63] in real-time [63, 62, 64, 65, 66, 67, 68, 23, 69, 24, 22]. The designs which require pre-processing aim to reduce the computational complexity and hence, silicon area and power consumption significantly, by performing a set of algorithms offline on a computer to estimate the design parameters for the real-time operation of the *in vivo* spike sorting system. To avoid heat-related tissue damage, the brain-implantable device is commonly realized using extremely low-power application-specific integrated circuits (ASICs) [70]. Computational acceleration of spike sorting using field-programmable gate arrays (FPGAs) have also been reported [62, 63, 71, 72, 73].

In this chapter we present an efficient design and implementation of binarized neural networks (BNNs) for real-time *in vivo* classification of neural spikes. In contrast to the conventional artificial neural networks (ANNs), in which the weights and activation functions are represented using real values, the BNNs utilize binarized weights and activation functions to significantly reduce the memory requirements and computational complexity of the conventional ANNs [74]. The learning process of the employed BNN is performed offline, which entails a partially supervised spike sorting. Most neural signal processing systems involve some offline processing [62, 63, 65, 23, 75, 76], which results in a more area- and power-efficient realization. While a trained BNN offers classification for waveforms similar to those seen during initial training, the BNNs can also offer generalization for spike sorting using alternative techniques such as noise injection [77] and Dropout [78], which act as a form of regularization and leads to low overfitting and greater generalization. This chapter introduces the application of BNNs for spike sorting in an effort to significantly reduce the computational complexity and memory requirement of the system, while utilizing an initial off-line parameter estimation.

The rest of this chapter is organized as follows. Section 3.2 briefly describes the spike sorting process and the employed neural signal processing algorithms. Section 3.3 discusses the fundamental operations of the BNNs and, for the first time, introduces their application for real-time *in vivo* spike sorting. Section 3.4 presents the hardware architecture design of the BNNs. The ASIC and FPGA characteristics and implementation results of the designed BNN-based spike sorting system are presented and compared with those of the state-of-the-art realizations.

The feasibility of the brain-implantable BNN-based spike sorting system for real-time processing of neural signals is also discussed. Finally, Section 3.5 makes some concluding remarks.

3.2 Spike Sorting

Fig. 3.1 shows the system-level block diagram of a BMI. BMIs translate (decode) neural signals recorded with a MEA [79] into commands for the direction of machines and a variety of prosthetic devices and hence, restoring impaired neural signal pathways. In conventional spike sorting systems, the spiking activity of neurons is first detected from the recorded, amplified, and filtered neural signals combined with ambient noise. Spike detection is usually done in two steps, pre-emphasis and thresholding. Pre-emphasis involves specific signal processing on the neural signals, such as computing the absolute value [80], the non-linear energy operator (NEO) [38], or the discrete wavelet transform (DWT) [81]. The thresholding step then compares the neural signal, or the pre-emphasized signal, to a threshold value. A spike is detected when the neural signal crosses an assumed threshold.

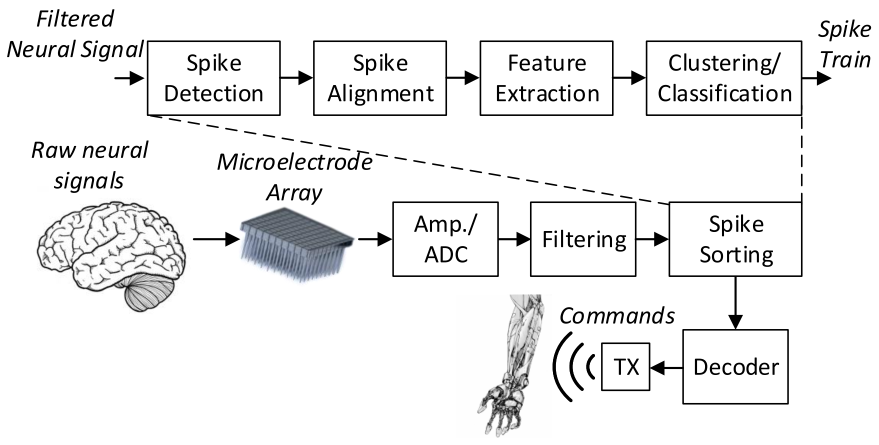


Figure 3.1. The system-level diagram of a BCI.

After spike detection, spike alignment is performed such that each detected spike has a common reference point (e.g., maximum amplitude or maximum energy) at the same sample number in the waveform [61]. For example, if a spike waveform is represented using 64 samples, the alignment index indicates which of the 64 samples holds the alignment feature.

Some commonly employed alignment features include the maximum slope, maximum absolute value, or maximum value of the spike waveform. In the alignment process, the distance between waveforms is commonly computed with the L_1-norm (Manhattan distance) or L_2-norm (Euclidean distance). The reliability of the distance computation can be improved by ensuring that the alignment feature is at the same sample index for all spikes. After spike alignment, feature extraction (FE) is utilized to reduce the dimensionality of the spike waveforms so that the characteristics of a spike waveform can be described using only a relatively small number of features. If FE is not used and the spikes are clustered directly, for example in a 64-dimensional space, a relatively large memory is required. However, it is far more efficient to interpret a two- or three-dimensional clustering space when representing the spike waveforms using a relatively small number of features, e.g., two or three, respectively.

The final step in the conventional spike sorting process is to group the extracted features into disjoint clusters. Feature extraction, clustering, and classification are closely tied together. The extracted features should be chosen such that the detected spikes originating from the same neuron create an individual cluster. Regardless of a particular FE approach, the classification process involves clustering of the features such that distinct features (and spike waveforms) create different groups (clusters). Early clustering methods involved manually identifying the clusters after extracting and plotting the features [61]. Two of the most commonly used clustering methods are the *k*-means [82] and OSort [83] algorithms. The *k*-means clustering algorithm begins by randomly defining *k* different cluster centroids. The data-points nearest to each cluster centroids are assigned to that cluster and the cluster centroid for each cluster is updated as the average of the cluster. One disadvantage of the *k*-means clustering algorithm, similar to the PCA algorithm, is that it cannot operate in real-time because it requires a relatively large number of data points to find the optimal cluster centroids. Even though recent improvements to the *k*-means algorithm involve computing relatively accurate approximations of the cluster centroids using a small number of data samples [84], the memory and computational requirements of processing high-dimensional data, such as neural spike waveforms, are still not adequate for area-

and power-constrained brain-implantable ASICs. Also, the number of clusters (NOCs) k is not known apriori and requires supervision. In contrast, Osort creates the clusters on the fly by means of comparing the new input feature vector to all existing cluster averages. If the distance of the new feature vector is beyond the assignment threshold, a new cluster is created for that feature vector. The clustering algorithm associates spike waveforms with the corresponding cluster that the feature vector is assigned to. The OSort approach is thus attractive for hardware realization [63] because it is both unsupervised (i.e., does not require pre-processing for the estimation of cluster averages/centroids) and can cluster the spikes in real-time as spike waveforms are detected and their features are extracted.

3.3 Binarized Neural Networks for Spike Sorting

ANNs mimic brain neurons' activities with two key distinctions to traditional Von Neumann architectures. Firstly, the computation is performed by a set of processing elements, also referred to as artificial neurons (ANs), interconnected by weighted synapses. The processing elements are loosely modeled after biological neurons, where the excitation of a neuron depends on the activity of its pre-synaptic neurons. Secondly, the synaptic weights among neurons are parameters that are obtained via a training process. Training an ANN mimics how a human brain learns by example. An ANN consists of an input layer, zero or more hidden layers, and one output layer with various number of ANs in each layer. For each AN in the hidden and output layers, the weighted outputs from pre-synaptic neurons are accumulated and a non-linear activation function is applied to the result. Commonly employed non-linear activation functions include the sigmoid function $f_{\text{sigmoid}}(z) = \frac{1}{1+e^{-z}}$, the hyperbolic tangent function $f_{\tanh}(z) = \frac{2}{1+e^{-2z}-1}$, and the rectified linear unit $f_{\text{ReLU}}(z) = z$ if $z \geq 0,$ else $z = 0$, where z denotes the accumulated weighted inputs to an AN [85]. Each AN may have a bias b, which essentially adds a lateral shift to the activation function and effectively tunes how easily or difficult it is to produce an excited output. The parameters of a neural network, including weights, biases, the number of hidden layers, and the number of neurons in each hidden layer, are adjusted during training. The numerical resolution of the weights and biases, as well as the number of neurons in the hidden and output layers, pose a constraint on the overall memory requirement of the ANN. The impacts of reducing the parameter resolution on the training and performance of ANNs have been studied in [86, 87].

Deep neural networks (DNNs) employ a relatively large number of hidden layers and have been used in a variety of machine learning applications, from image and pattern recognition, self-driving vehicles, to medical experimentations for diagnosing diseases. The challenge with the efficient hardware implementation of DNNs for extremely area- and power-constrained applications, such as brain-implantable devices, lies in the relatively large number of parameters for storage. For off-line realizations, this constraint is not crucial due to the availability of storage on conventional workstations.

We have previously employed ANNs for spike sorting [64]. This chapter aims to further reduce the overall logical resource and memory requirements of the implemented spike sorting system. In a notable effort to reduce the memory requirement of DNNs, a method for the binarization of weights and activation functions was presented in [74]. The binarization is given as:

$$k_b = \mathrm{sign}(k) = \begin{cases} +1 & \text{if } k \geq 0, \\ -1 & \text{if } k < 0. \end{cases} \tag{3.1}$$

where k_b denotes the binarized value of k and sign denotes the sign function. As opposed to the traditional ANNs that often employ biases in the activation function, BNNs do not add a bias to the weighted inputs since it will cancel the binarization. The binarization of the learned parameters and the activation functions dramatically reduces the amount of memory required for BNNs. For example, consider a network topology of 784–512–10 with 784 input neurons, 512 neurons in the hidden layer, and 10 output neurons. The number of weight parameters for this topology is 406,528. If the weight values for a traditional ANN are stored using 8 bits, the total memory requirement is 3.25 MB, whereas the memory requirement for a BNN with the same network topology is only 406 kbits. Even if the weights are stored using two-bit signed integers, i.e., 11_2 and 01_2 for -1 and +1, respectively, the memory requirement is 812 kbits, which is far less than that of the ANN. Moreover, the required storage for representing the neurons' outputs is considerably smaller for the BNNs.

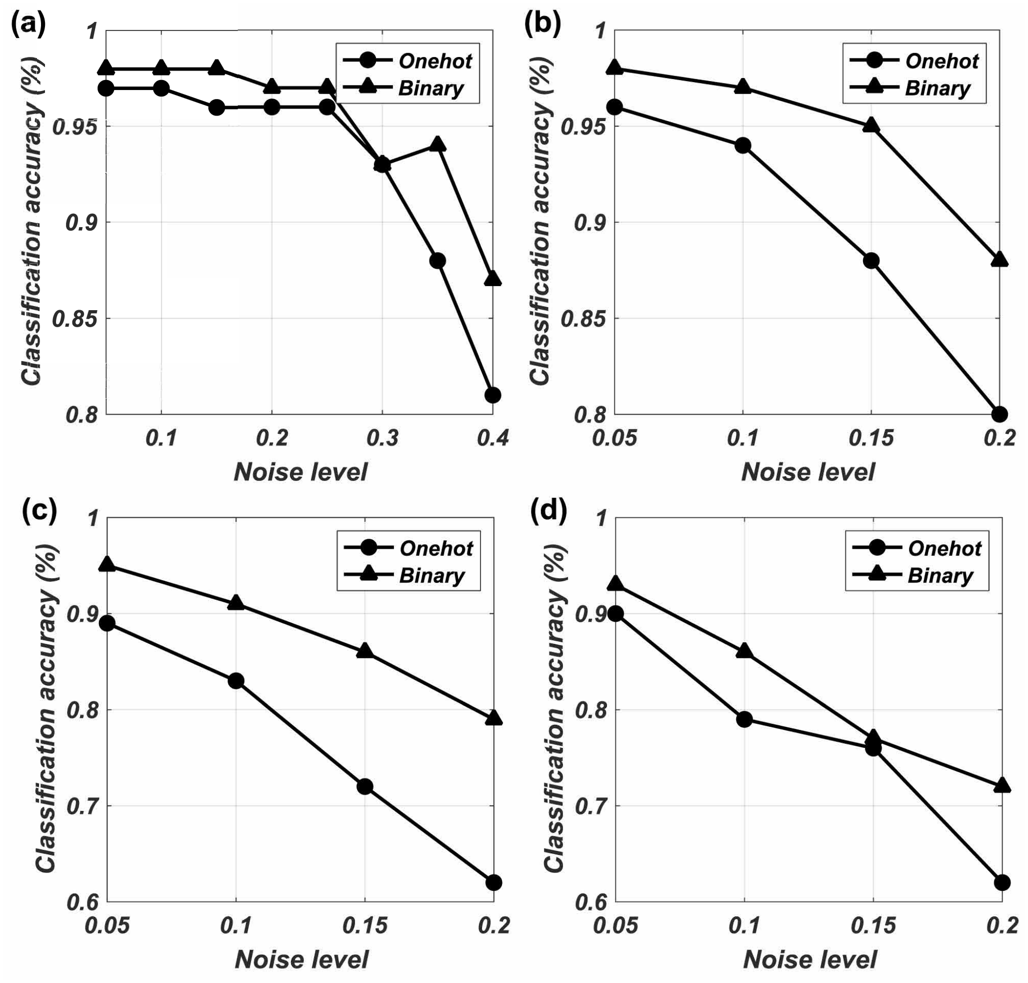


Figure 3.2. Classification accuracy of the BNN for the (a) Easy1, (b) Easy2, (c) Difficult1, and (d) Difficult2 datasets using the one-hot and binary encoding schemes.

The accuracy of various BNNs have been assessed using standard datasets, including MNIST, SVHN, and CIFAR-10, and it was shown that the accuracy of the BNNs is comparable with those of the state-of-the-art classifiers [88]. To train the BNNs for spike sorting, we use the datasets in [89] and quantify the accuracy of the sorting process. The datasets consist of simulated waveforms which differ based on the differentiality of the spike shapes (*Easy* and

Difficult) for various noise levels. Each dataset offers three distinct spike shapes. The spike shapes were randomly chosen from 594 spike shapes in a real neural signal database. Then, background noise was added to the spike waveforms with the noise standard deviation levels within [0.05, 0.2], and up to 0.4 for the Easy1 dataset, relative to the spike amplitude. The signal-to-noise ratios (SNRs) of the datasets are between -31.69 dB and -27.39 dB. Each dataset includes various ground truth information, such as the time of spike events, which indicates the particular spike shape (or spike class) that occurred at that time. The input to the BNN is a 64-sample aligned spike waveform and the three one-bit outputs of the BNN indicate the spike class that the input spike belongs to. For the binarized outputs of the network, two output encodings of one-hot and binary are utilized. The employed BNN topology is 64–5–n, i.e., 64 inputs, 5 neurons in a hidden layer, and n neurons in the output layer, where n is either equal to C for one-hot encoding or $\text{ceil}\left(\log_2 C\right)$ for the binary encoding, where C denotes the number of spike classes (three in our designed and implemented BNN classifier) and ceil denotes the ceiling operator.

The Python Larq framework [90] is used to train the BNNs. The Adam optimizer with the default parameter values of learning rate = 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, ε = 1e-07, is used to train the BNNs for 250 epochs. For the dataset in [89], the training data consists of 2460 spikes and their corresponding spike classes. The data is divided into 70% for training and 30% for testing. The data is distributed such that each of the spike classes in the dataset appears in equal numbers and hence, the network is trained using a balanced dataset. The spike classes, which are given as integers in the original dataset, are converted into row vectors indicating the target outputs for the given training spikes with the desired output encoding mode. For example, if the spike class is 3, the one-hot encoded row vector denotes 001_2, i.e., the third output neuron should produce a one. Similarly, the binary encoded output is given as 11_2. Because the sign function only results in outputs -1 and 1, the zeros are mapped to -1. So, a spike class of 3 is mapped onto row vectors [-1 -1 1] and [1 1] for the one-hot and binary encoding, respectively.

Figs. 3.2(a–d) show the classification accuracy of the trained BNN using the one-hot and binary encoding schemes, over the Easy1, Easy2, Difficult1, and Difficult2 datasets, respectively. One can see that for most cases, the binary output encoding outperforms the one-hot encoding. The classification accuracy degrades with increasing the noise level for both encoding schemes, however, utilizing the binary encoding, the accuracy of the BNN is less susceptible to increases in the noise level over all the datasets compared to utilizing one-hot encoding. The median classification accuracy of the one-hot encoded BNNs are 0.96, 0.91, 0.78, and 0.78, while for the binary encoded BNN are 0.98 0.96, 0.89, and 0.82, for the Easy1, Easy2, Difficult1, and Difficult2 datasets, respectively. Table 3.1 gives the average classification accuracy of the BNN employing two encoding schemes over four different datasets and various noise levels. One can see that for the employed datasets, the binary encoding provides greater average accuracy.

Table 3.1. The average classification accuracy of the BNN employing two encoding schemes over four different datasets and various noise levels.

Dataset	Encoding method	Classification accuracy (%)
C_Easy1	One-hot	92
	Binary	95.25
C_Easy2	One-hot	89.5
	Binary	94.5
C_Diff1	One-hot	76.5
	Binary	87.75
C_Diff2	One-hot	76.75
	Binary	82

We also trained the BNN using two other datasets, a multi-unit activity dataset [91] and a real human dataset [28]. A multi-unit activity is defined as the aggregate response of multiple single-units near the tip of an electrode. The multi-unit dataset was created to more accurately model the realistic conditions of recorded neural signals using a relatively dense MEA. The background noise was modeled by the superposition of the spiking activity of extracellular neurons. The five defined multi-unit datasets have varying amplitudes for the single-unit activities and have different spike firing rates. For classification, the datasets have three possible spike shapes: two distinct single-unit waveforms and a set of multi-unit waveforms. The average median classification accuracy for the one-hot encoded BNN is 0.76 and for the binary encoded

BNN is 0.81 over the multi-unit datasets 1 – 5. Again, the binary output encoding shows a better classification accuracy than the one-hot encoding. Because the binary encoding can cause output neurons to produce the same output value for different types of spike waveforms, this can be used as a form of regularization for the network during training.

The real human dataset is a recording from the temporal lobe of an epileptic patient. Because verified spike times and spike classes were not included in the real human dataset, we use the OSort algorithm to cluster a subset of the spike waveforms, which then converged to four distinct clusters. The cluster assignments were used to train the BNN using a subset of the spike waveforms. The BNN was then evaluated using the remainder of the spikes in the dataset. The classification accuracy of the BNN on the real human dataset was 0.72 and 0.88 using the one-hot encoding and the binary encoding schemes, respectively.

Some spike sorting systems employ feature extraction to reduce the dimensionality of the spike waveforms [65, 68, 23, 22]. To compare the sorting accuracy of the BNN-based classifier when using detected spike waveforms and a reduced feature space, we employed zero-crossing features (ZCFs) given as [92]:

$$\mathrm{ZC}_1 = \sum_{n=0}^{K_1-1} s(n), \quad \mathrm{ZC}_2 = \sum_{n=C}^{K_2-1} s(n),$$

where K_1 denotes the zero-crossing point in the spike waveform and K_2 denotes the number of samples in the spike waveform $s(n)$. ZC_1 and ZC_2 denote the areas under the positive and negative portions of the spike waveform, respectively. They reduce the dimensionality of the spike feature space by 96%. The ZCF would require two additional accumulators to compute ZC_1 and ZC_2 and the network topology becomes 2–5–3 for the one-hot encoding and 2–5–2 for the binary encoding, resulting in the weight memory requirements of only 25 bits and 20 bits, respectively. The dimensionality reduction of the network input and the weight memory requirements comes at the significant loss in classification accuracy (around 65% and 36% for the binary and one-hot encoded outputs, respectively). It is also shown that increasing the number of hidden layers and the number of ANs in a layer yielded negligible performance improvement. Therefore, we do not incorporate feature extraction in our designed spike sorting system.

3.4 BNN Hardware Architecture

Fig. 3.3 shows the top-level block diagram of the designed BNN-based spike sorting system using the NEO-based spike detector [62], the maximum amplitude spike alignment, and the BNN-based spike classifier. Because our goal is to reduce the silicon area and power consumption of the spike sorting circuit for real-time *in vivo* realization while achieving an acceptable level of sorting accuracy, we choose to use the NEO-based detection technique as it imposes less computational complexity than computing the DWT, yet providing improved accuracy over the absolute value method of pre-emphasis. This is primarily because the NEO algorithm derives its threshold value based on the probability of false alarm, whereas the threshold value in the absolute method is not adaptive to real-time characteristics of the neural signal. It was shown in [93] that in cases where the signal-to-noise ratio (SNR) was relatively high, the absolute value method is as accurate as NEO for spike detection. However, NEO outperforms the absolute value method when the SNR is low.

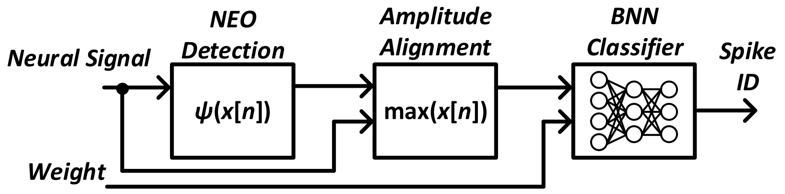


Figure 3.3. The top-level block diagram of the BNN-based spike sorting hardware.

The block diagram of the BNN-based spike classification module is shown in Fig. 3.4. The aligned spike waveform is stored in the shift register *SW ShiftReg*. To accommodate the aligned spikes, the depth of *SW ShiftReg* is set to 64 and each of the registers is 10 bits wide. The main processing units are the hidden layer processing units (*HPUs*) and the output layer processing units (*OPUs*). Once the aligned spike is received, the control unit begins to read values

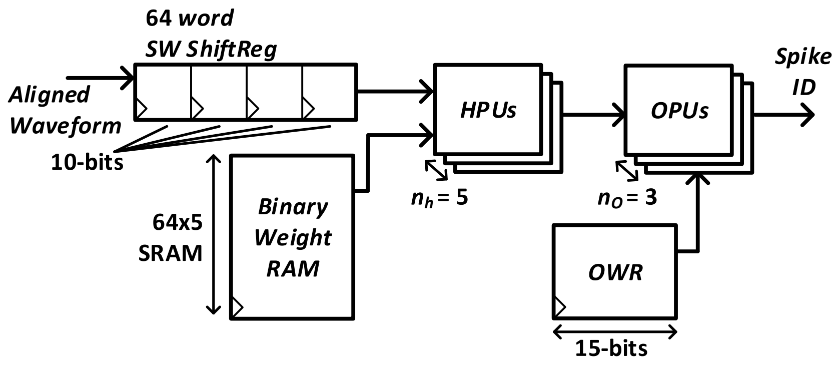


Figure 3.4. The block diagram of the designed BNN-based spike classification module.

from the weight memory *Binary Weight RAM* (*BWRAM*) into the *HPU*. To avoid using two bits to represent the values -1 and 1, a zero output of the sign function denotes a positive sign bit and correlates to a +1 output. Similarly, a one output of the sign function denotes a negative sign bit and correlates to a -1 output. Because the weights are constrained to -1 or +1 during the forward propagation, the accumulation of gradients is also constrained to the range between -1 and 1 during the training process. The width of the *BWRAM* is equal to the number of hidden layer nodes $n_H = 5$ and its depth is equal to the number of samples used to store the spike, i.e., 64. The *BWRAM* is implemented using the SRAM standard cells. The synaptic weights between the hidden layer and the output layer are stored in a 15-bit output weight register *OWR*. The number of bits stored in *OWR* is equal to the product of the number of hidden layer neurons n_h and the number of output layer neurons n_O. For our designed network topology with 64 inputs, 5 hidden layer neurons, and 3 output neurons, the synaptic weight memory is only 335 bits. Compared to our template matching (TM)-based spike classifier in [62], which requires 3072 bits of storage for three templates, the designed BNN-based spike classifier requires 89% less storage.

The block diagram of the *HPUs* is shown in Fig. 3.5. It consists of a two's complement unit, to negate the spike samples, and an accumulator. Since the spike inputs are real-valued and the weights associated with the synaptic connections may or may not change the signs of the spike samples, the hidden layer needs to accumulate non-binary input values. The weight values read from the *BWR* determine whether the value that is accumulated is the negated spike sample value or the original value for the weight 1 or 0, respectively. The control signal *AccRST* is used to reset the accumulator registers. The *WI.WF* denotes the integer and fraction lengths of the accumulation registers, respectively. The sign function is implemented by passing the most significant bit of the accumulating registers to the output. To process all n_h hidden layer neurons in one clock cycle, the *HPU* consists of n_h sets of multiplexers and accumulators, producing n_h bits at its output.

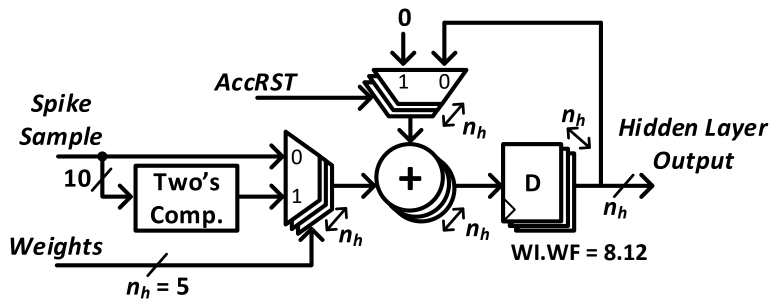


Figure 3.5. The block diagram of the hidden layer processing units.

Fig. 3.6 shows the block diagram of the output layer processing units. It consists of an array of n_O XOR gates, majority functions $m(X)$, where X denotes the outputs of the XOR arrays, and a set of inverters. The XOR arrays are effectively used to perform the bitwise multiplication between the outputs of the *HPU* and the weight values stored in the *OWR*. The output of each XOR array is n_h bits wide and the majority function $m(X)$ is used to determine whether the n_h bits contain more ones than zeros. For our designed BNN, since $n_n = 5$, $m(x) = 1$ when at least three of the bits in X are one. The set of inverters is only used when the one-hot encoding is employed, in which case the value of each output bit corresponds to the presence or lack of a particular spike class at that time.

To assess the accuracy of the designed BNN classifier over varying number of clusters, we employed the datasets from [94], which contains 95 simulations of neural signals with three to twenty clusters. The classification accuracy in Fig. 3.7 shows that the designed BNN with only five ANs in the hidden layer can reliably sort the activity of up to 12 single units with the binary encoding scheme. When utilizing the binary encoding scheme, the NOC is limited to a

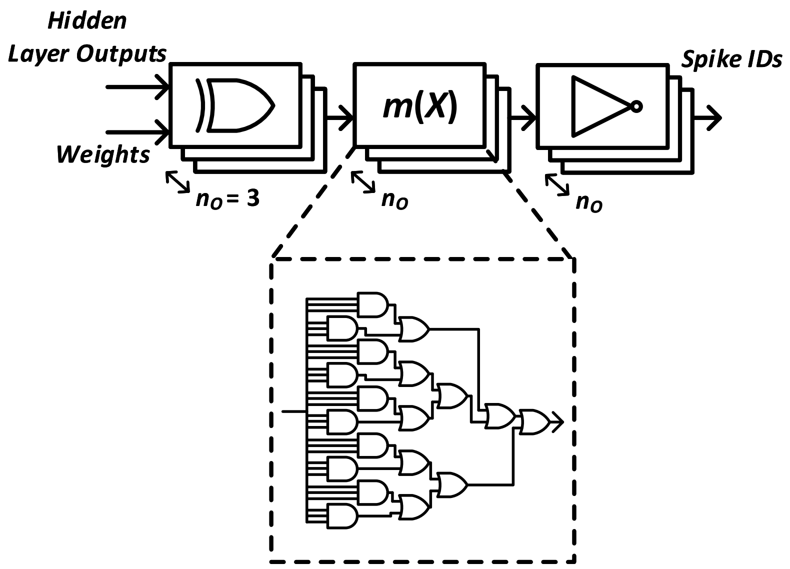


Figure 3.6. The block diagram of the output layer processing units (*OPUs*).

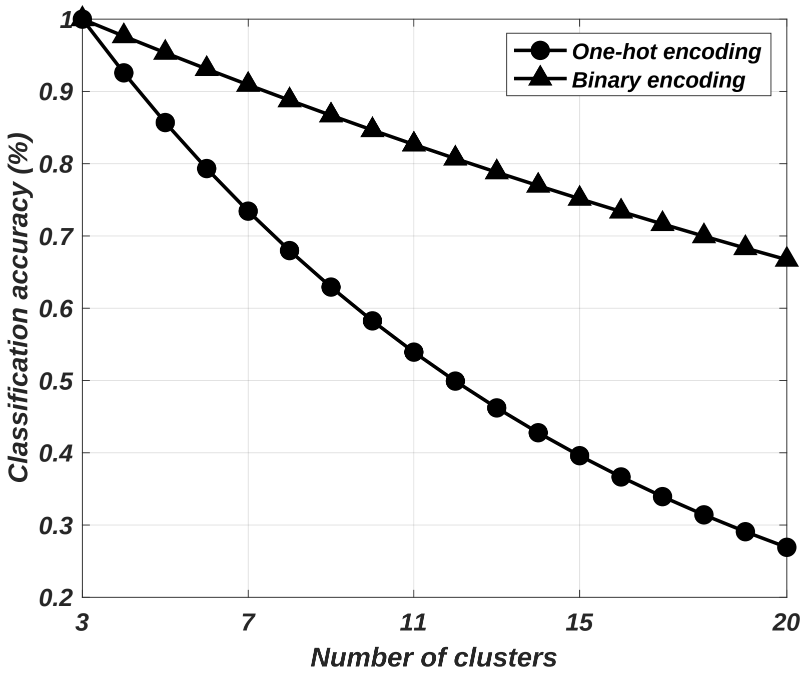


Figure 3.7. Classification accuracy of the BNN utilizing the one-hot and binary encoding schemes for varying numbers of clusters.

maximum of $2^{n_o}-1$, where n_o denotes the number of ANs in the output layer. Thus, the designed BNN-based spike sorting system with $n_o = 3$ can sort up to seven clusters. The memory size of the one-hot encoded BNN can be given as $(64 \times n_h) + (n_h \times NOC)$, where 64 is the number of

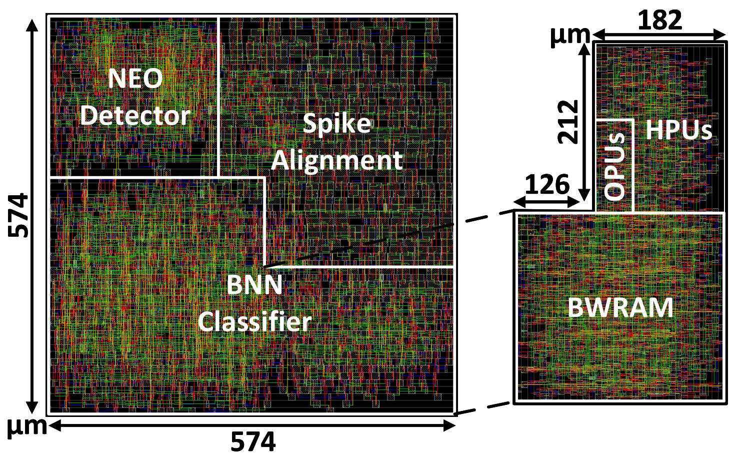


Figure 3.8. The chip layout of the designed BNN-based spike sorting system.

Table 3.2. The ASIC characteristics and implementation results of various spike sorting systems

Design	Ours	[63]	[62]	[64]	[65]	[66]	[67]	[68]	[23]	[69]	[24]	[22]
Algorithm	BNN	OSort	TM	ANN	FE	OSort	OSort	FE	FE	TM	FE	FE*
Number of clusters per channel	3–7	3	3	3	–	3	3	3	4	3	3	6
Classification accuracy	0.91–0.90	0.87	0.90	0.98	0.77	0.75	0.93	0.84	0.85	0.93	0.86	0.92
Data rate reduction	2000×	1600×	3200×	1866×	11×	240×	278×	240×	–	–	257×	–
Supervised	Y	N	Y	Y	Y	N	N	N	Y	N	N	N
Technology (nm)	180	32	45	32	90	65	45	45	130	40	65	65
Core voltage (V)	1.8	1.16	0.25	0.7	0.55	0.27	–	1.1	1.2	1.1	0.54	1
Operating frequency (kHz)	24	24	24	20	4000	480	56	960	160	500	3200	30
Area per channel (mm^2)	0.33	2.57	0.30	0.009	0.06	0.07	0.07	2.7	0.023	0.0175	0.003	0.09
Power per channel (μW)	2.02	2.78	0.064	1.04	2	4.68	10.3	20	0.75	19	0.175	0.181
Scaled area per channel (mm^2)*	0.33	102.8	5.7	0.36	0.396	0.84	1.33	51.3	0.066	0.473	0.036	1.08
Scaled power per channel (μW)*	2.02	8.849	6.04	15.04	56.89	671.61	–	57.99	1.86	56.8	6.28	1.06

* Scaled to a 180-nm CMOS technology with a 1.8V supply voltage, as defined in [47]. Our implementation results are highlighted.

samples in a spike waveform and n_h denotes the number of ANs in the hidden layer. The memory size of the binary encoded BNN can be given as $(64 \times n_h) + (n_h \times \text{ceil}[\log_2(NOC)])$. Because the binary-encoded BNN is able to represent a wider range of output values using fewer ANs in the output layer, it lowers the number of hidden-to-output layer weights. Moreover, the binary encoded BNN offers a greater accuracy for different number of clusters and over varying noise levels. Compared to the other state-of-the-art spike sorting systems, offering up to six clusters per channel, the designed BNN-based spike sorting system can classify up to 20 clusters.

The training time for large-scale and deep neural network models could be considerable, however we found that since our model is relatively small, increasing the NOCs, which increases the number of ANs in the output layer, has a relatively small impact on the training time. However, the training time for the estimation of the binary weights is directly related to the number of training examples, the number of training epochs, and the number of training iterations. The off-chip training time of our *in vivo* BNN-based spike sorting system ranges between 250 ms and 400 ms for 10 iterations of training with 50 epochs per iteration.

We have implemented our BNN-based spike sorting system in a standard 180-nm TSMC CMOS process. The chip layout of the BNN-based sorting system is shown in Fig. 4.8, which is estimated to occupy 0.33 mm^2 of silicon area. We previously designed and implemented the NEO-based spike detection and spike alignment units for our TM-based spike sorting system and the parallel OSort-based real-time spike sorting system in [62] and [63], respectively. Synthesis was performed using Synopsys Design Compiler and the place-and-routing was performed using Cadence Innovus. After routing and static timing analysis, the power was estimated by simulating the layout with the datasets from [89]. A variable change dump file was used to more accurately estimate the switching activity in Innovus. It was estimated that the BNN-based spike sorting chip will consume 2.02 μW from a 1.8 V supply while operating at 24 kHz.

A commonly used metric for quantifying the classification accuracy of spike sorting realization is the F-score metric [95] and is given as $F = \frac{2T_P}{2T_P+F_P+F_N}$, where T_P, F_P, and F_N denote the number of true positive, false positive, and false negative classifications, respectively. True positives are defined as spikes that are detected, classified, and exist in the reference dataset. A true positive is verified by finding the spike waveform in the reference dataset as well as the time of detection. A false positive is a spike that is detected and classified, but does not exist in the reference dataset. A false negative is a spike that exists in the reference dataset, but is not classified by the spike sorting system. Our ASIC implementation of the BNN-based spike sorting system achieve the average F-Scores of 0.96, 0.94, 0.86, and 0.86 for the Easy1, Easy2, Difficult1, and Difficult2 datasets, respectively. The F-scores for the multi-unit datasets 1–5 are given as 0.94, 0.83, 0.89, 0.84, and 0.96, respectively. The F-scores for the datasets from [94] with three to seven NOCs are given as 1.0, 0.98, 0.97, 0.96, and 0.95, respectively. The

F-score cannot be computed for the real human dataset due to the lack of the ground truth. The designed ASIC classifier is a bit-true realization of the fixed-point software model and hence, their F-scores are equivalent.

Table 3.3. The characteristics and implementation results of various spike sorting systems on FPGAs

Work	Algorithm	Device	WL.WF	Regs.	LUTs.	BRAMs	DSP48s.	Max. Freq. (MHz)	Clustering Latency	Sorting Accuracy
Ours	BNN	Artix-7	16.11	267 (0.1%)	314 (0.3%)	2 (0.27%)	0 (0%)	125	0.51 μs	93%
[63]	OSort	Virtex-6	16.11	8444 (2%)	16472 (9%)	29 (4%)	130 (9%)	123	0.26 μs	87%
[71]	OSort	Virtex-5	16.8	16245 (27%)	23567 (40%)	63 (25%)	29 (4%)	100	11.1 ms	–
[62]	TM	Virtex-6	16.11	4880 (1%)	6635 (3%)	0 (0%)	5 (0.6%)	122	0.55 μs	90%
[72]	TM	Virtex-6	20.0	29000 (6%)	190000 (83%)	24 (6%)	–	–	2.65 ms	–
Ours*	BNN	Zynq-7000	16.11	124 (0.05%)	147 (0.12%)	0 (0%)	0 (0%)	263	0.24 μs	–
[73]	OSort	Zynq-7000	16.0	12150 (11%)	14037 (16%)	102 (72%)	120 (54%)	101	179.4 μs	–

Our implementation results are highlighted.

Various ASIC implementations of spike sorting systems have been previously reported [63, 62, 64, 65, 66, 67, 68, 23, 69, 24]. Table 4.3 gives the characteristics and implementation results of various state-of-the-art spike sorting systems. Our designed and implemented BNN-based spike sorting system is able to classify single units of up to seven clusters. In [63] we designed and implemented a parallel OSort-based spike sorting system in a standard 32-nm CMOS process. In [62] we implemented the TM-based spike sorting ASIC using three templates in a 45-nm CMOS process. In [64] we implemented an ANN-based spike sorting system. The ANN consists of one hidden layer neuron and three output layer neurons, all of which utilize the ReLU activation function. The work in [65] performs NEO spike detection, aligns detected spikes to maximum derivative, and implements FE via discrete derivatives. Their design consists of four 16-channel modules which produce either aligned spikes or feature vectors, but does not perform real-time classification of the detected spikes. The work in [66] uses the absolute value detection scheme and implements the OSort clustering algorithm for a single 16-channel module. Similar to [66], in [67] spikes are first detected using a voltage threshold. Detected spikes are then aligned to a maximum absolute amplitude and OSort-based clustering was utilized. The work in [68] performs single-channel spike sorting using the NEO-based detection, maximum amplitude alignment, and FE using discrete derivatives. It supports an unsupervised learning process, similar to the OSort-based systems. The work in [23] presents a multi-channel spike sorting ASIC based on FE. The design in [69] presents a multi-channel TM-based spike sorting ASIC with a built-in OSort learning system. The work in [24] presents a multi-channel spike sorting processor based on the integer coefficient FE and clustering. Finally, the work in [22] presents the ASIC implementation of a dictionary learning-based feature extraction. Similar to the BNN approach, the dictionary values are constrained to the ternary set of [-1, 0, +1] and no multiplications are required. Note that the authors only list the ASIC implementation results for the feature extraction module, which are denoted as "FE*" in Table 4.3. The spike sorting systems, which employ unsupervised learning, support real-time clustering/classification of detected spike waveforms. However, the TM-based and BNN-based sorting systems that require pre-processing of the neural recording in order to generate the template waveforms and BNN weights, respectively, require significantly less storage and reduced computational requirements, while achieving comparable sorting accuracy. The ASIC designs in [65, 66, 23, 69, 24] are multi-channel systems and, for a fair comparison, the area and power consumption results for single-channel sorting systems are given in Table 4.3. During spike sorting operation, the sampling rate is 24 kSamp/sec. Each sample is represented using 10 bits, which results in an input bitrate of 240 Kbps. With an average neuron spiking rate of 40 spikes per second [96] and representing the BNN output classification with 3 bits (one bit per neuron in the output layer), the output bit rate is reduced to 120 bps. This results in a 2000 times data rate reduction compared to the input sampling rate. Since the energy required to transmit one bit of data is approximately 3 nJ [97], this results in a wireless transmission power of about 360 nW. Therefore, the total power of our ASIC chip is 2.36 μW with the power density of 7.15 $\mu\text{W/mm}^2$, which satisfies the tissue-safe requirements for brain implantable devices [98].

For a fair comparison of the ASIC implementation results across different CMOS technologies and supply voltages, we have also reported the power consumption and area utilization of the previously published work scaled to 180-nm technology as presented in [47]. As given in Table 4.3, the BNN-based spike sorting system requires a smaller silicon area and consumes less power, while providing comparable sorting accuracy. Our TM-based spike sorting ASIC [62] and the OSort-based clustering system [63], as well as the work in [67, 68, 69, 24], have utilized the same reference datasets from [89] to quantify the accuracy of their sorting system. The work in [65] uses both synthetic data and real data, but the synthetic data differs from the one used in [89]. The work in [66] and [23] both use real neural data for evaluating the accuracy of their spike sorting systems.

Compared to the OSort clustering, in which the algorithm can adapt to signal variations in real time, such as electrode drift or increases in noise levels, the BNN-based clustering does not offer run-time adaptability. Nevertheless, compared to the other supervised approaches that require pre-processing for parameter estimation, such as TM-based spike sorting, the BNN can employ various optimization schemes during training to create a generalized and robust network model that reduces overfitting [99]. For example, the training dataset can be augmented by additional data with increased noise levels to improve the robustness of the BNN model [77]. An alternative approach to diminish overfitting is to introduce Dropout among layers of the BNN by removing some synapses during training and hence, forcing the network model to "space out" it's weighting of parameters [78].

Our BNN-based spike sorting system is portable and designed for FPGA and ASIC realizations. Both ASIC and FPGA implementations were tested and achieved the same sorting accuracy. Table 3.3 gives the characteristics and implementation results of various state-of-the-art spike sorting systems implemented on FPGAs. In Table 3.3, for a fair comparison, we list the implementation results for the designed BNN-based spike sorting system using similar FPGA devices. One can see that the unsupervised Osort-based implementation requires considerably more reconfigurable resources than that of the TM-based and that of the supervised BNN-based sorting systems. The design in [71] presents an FPGA implementation of OSort, which was designed for high-performance processing of recorded neural data on a workstation. The reported latency of 11 ms was assumed based on a 24 kHz sampling frequency, which implies the worst case sorting latency of 266 clock cycles. The work in [72] presents the hardware implementation of a bayes-optimal TM-based spike sorting system. While the maximum operating frequency of their design was not reported, their sorting latency was given as 53 clock cycles, which is slightly less than that of our design at 64 clock cycles. Since the work in [73], which uses the OSort algorithm for real-time clustering of spikes, only presents the FPGA implementation results for the clustering and classification modules, for a fair comparison the FPGA implementation results of the designed BNN classification module are given separately and denoted by "Ours*" in Table 3.3. Note that the classification latency of the OSort-based implementation in [73] is 18,127 clock cycles while the classification latency of our BNN-based classification module is only 64 clock cycles.

3.5 Conclusion

This chapter presented that the binarized neural networks (BNNs) can be efficiently used for real-time *in vivo* spike sorting, while significantly reducing the memory requirement and computational complexity of the system compared to the other state-of-the-art realizations. Our ASIC implementation results confirmed that a relatively small BNN can be used for brain-implantation due to its small silicon area and its low power consumption, while providing reliable sorting accuracy for varying numbers of clusters.

3.6 Publications

The following article was published as an outcome of the research conducted in this chapter: D. Valencia and A. Alimohammad, "Neural spike sorting using binarized neural networks," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 29, pp. 206 – 214, December 2020.

Chapter 4

Partially Binarized Neural Networks for Efficient Spike Sorting

4.1 Introduction

The ability to efficiently record and decode neural signals is of vital importance towards the rehabilitation of patients with various neurodegenerative diseases, including Alzheimer's and Parkinson's. A brain-machine interface (BMI) translates neural activities into commands for controlling external devices. For example, in [11] neural activity associated with imagined handwriting is used to convert the brain's activity into text. Also, in [100], neural signals were used to enable patients to synthesize speech directly from their thoughts. The algorithms employed for accurate neural decoding typically process spike trains, which represent the action potentials (or spikes) of individual neurons over time [10].

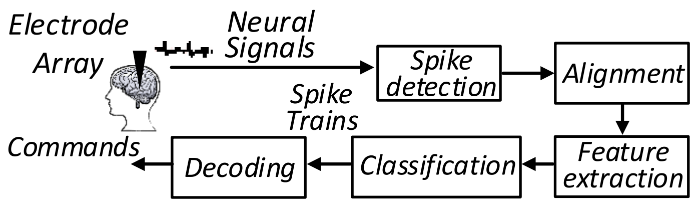


Figure 4.1. The block diagram of a BMI system employing spike sorting.

For greater spatial resolution and decoding performance, neural activities are first recorded by a multi-electrode array (MEA), which consists of intracortical electrodes capable of recording neural spikes from a relatively large number of neurons, in the order of hundreds, simultaneously. The recorded raw signals are then amplified and filtered into the frequency bands of interest. While each electrode records voltage signals fired by a neuron, or single-unit activities (SUAs), it also records spikes from neighboring neurons, or multi-unit activities (MUAs). Spike sorting, which is the process of associating recorded spikes to individual neurons is of prime importance for reliable neural decoding [61]. Spike sorting can be viewed as a clustering process where action potentials fired from a particular neuron with similar waveforms are grouped together. Spike sorting is conventionally performed over four steps, as shown in Fig. 4.1: (i) spike detection, (ii) spike alignment, (iii) feature extraction, and (iv) clustering. Spike detection involves detecting the spiking activity of an ensemble of neurons from the background noise.

Alignment is the process of ensuring that each detected spike waveform is aligned to a particular metric, such as the maximum amplitude. Feature extraction (FE) is optionally used to reduce the dimensionality of spike waveforms by describing them using a relatively small set of features. Finally, clustering involves grouping similar spike waveforms and creating disjoint clusters of spike features, which identifies spikes originating from individual neurons. The clustering process also yields spike timings for each neuron that are subsequently used by the spike decoder to translate the generated spike trains into commands for controlling or communication with an external device [101].

While some designs implement spike sorting algorithms offline on a computer [65], some realizations based on template matching [62], neural networks [64, 102], and decision trees [23] employ offline estimation of parameters for in vivo sorting. Some techniques instead implement unsupervised learning algorithms in vivo to sort the detected spikes into clusters in real-time. For example, OSort creates and manages clusters as new spikes/features are made available [63, 67, 68, 69, 66], while *k*-means clustering iteratively updates cluster centroids using a set of training spikes/features. Due to advances in digital signal processing, the state-of-the-art brain-implantable spike sorting systems employ on-chip unsupervised learning, which inevitably leads to increased circuit area and power consumption. In our earlier work in [102], we reduced the power consumption and the silicon area of the spike sorting circuitry by designing a binarized neural network (BNN) for classifying spike waveforms using only bit-wise operations and employs an offline training for weight estimation. Unfortunately, the BNN-sorted spike waveforms are more susceptible to noise due to their higher dimension compared to the extracted features of spike waveforms. In this chapter, feature extraction is first applied to spike waveforms to reduce the dimensionality to only a few fundamental features. To improve classification performance compared to BNNs, we propose a partially binarized neural network (PBNN), which is less susceptible to noise while requiring fewer overall parameters for sorting. To the best of our knowledge, this is the first work employing a PBNN for neural network-based classification of feature vectors.

The rest of this chapter is organized as follows. Section 4.2 discusses the efficiency of various candidate feature extraction algorithms for hardware realization. Section 4.3 reviews the operation of PBNNs, presents their application in classifying extracted feature vectors, and discusses the accuracy of the proposed PBNN-based spike sorting system. The characteristics and implementation results of the designed spike sorting system are presented and compared with those of the state-of-the-art realizations in Section 4.4. Finally, Section 4.5 makes some concluding remarks.

4.2 Efficiency of the Feature Extraction Methods

The principal component analysis (PCA) [61] and independent component analysis (ICA) [103] are two commonly employed feature extraction benchmark algorithms. Usually the first few principal components are the ones with the highest variations and hence, the dimensionality of a spike waveform can be reduced significantly. However, PCA has two fundamental limitations that hinders its feasibility for the in vivo implementations. Firstly, PCA requires a relatively large number of detected spikes to find the orthogonal basis vectors, which prevents its real-time realization. Secondly, the computational complexity of the PCA makes it impractical for extremely low-power brain-implantable devices [104]. Even though ICA outperforms PCA for signals with a relatively high noise level [105], ICA is also computationally-intensive and is unrealistic for in vivo implementations.

A number of more computationally-efficient feature extraction algorithms have been reported. For example, zero-crossing features (ZCFs) [92] are given as:

$$ZC_1 = \sum_{n=0}^{K_1-1} s[n], \quad ZC_2 = \sum_{n=K_1}^{K_2-1} s[n],$$

where $s[n]$ denotes the detected spike waveform, K_1 and K_2 denote the zero-crossing point and number of samples in the waveform, respectively, and ZC_1 and ZC_2 denote the accumulated energy of the neural signal before and after the zero-crossing point, respectively. Similar to the zero-crossing features, the integral transform (IT) algorithm [106] has found applications in neural spike feature extraction. The IT features are given as:

$$I_A = \frac{1}{K_A} \sum_{n=n_A}^{n_A+K_A-1} s[n], \;\; I_B = \frac{1}{K_B} \sum_{n=n_B}^{n_B+K_B-1} s[n],$$

where n_A and n_B denote the first sample of the positive and negative phases, respectively, and K_A and K_B denote the total number of samples in the positive and negative phases, respectively. One notable weakness of the IT algorithm is that the indeces n_A and n_B are not known a priori and finding them requires analysis of the spike waveforms, which imposes additional latency and is undesirable for real-time spike sorting. The minimum delimitation (MD) feature extraction algorithm [107] is relatively computationally-efficient and is given as:

$$MD_1 = \sum_{n=1}^{M} s[n], \quad MD_2 = \sum_{n=M+1}^{K} s[n],$$

where M denotes the index position of the minimum value and K denotes the number of samples in the spike waveform. The ZCF, IT, and MD feature extraction algorithms commonly represent the spike waveforms using two features and hence, significantly reduce the memory requirement of the subsequent clustering module at the cost of a relatively small FE hardware. Some realizations may utilize more than only two features by including more details of the spike itself, such as the index of the minimum or maximum point.

Another class of the FE algorithms are based on the discrete derivatives (DD), which can be considered as a simplified discrete wavelet transform [108]. The DD algorithm computes the slope of the spike waveforms at different scaling factors δ and can be written as:

$$DD_\delta[n] = s[n] - s[n-\delta].$$

Various combinations of different scaling factors can be utilized for constructing the feature space. For different scaling factors, some key features of the spike waveform, such as the positive peaks, negative peaks, and peak-to-peak amplitudes are accentuated. For neural spike sorting, four variations of the DD algorithm have been studied: the maximum difference test (DD-MDT) [93], the first and second derivatives (DD-FS) [109], DD with uniform sampling (DD-US) [65], and DD with two-extrema sampling (DD-2Ex) [110]. The DD-MDT, which is considered a simplified version of the Lillefors Test [111], extracts the multimodal coefficients of each scaling factor. The DD-FS algorithm computes the first and second derivatives of the spike waveform to accentuate its geometric characteristics. Specifically, the first derivative can be used to interpret the variations of the spike waveform's gradient, while the second derivative emphasizes its low frequency characteristics. The DD-US algorithm involves computing the DD with three different scaling factors and downsampling of the DD waveforms at even intervals. The DD-2Ex algorithm creates the DD waveforms with two different scaling factors and extracts the minimum and maximum values for the two DD waveforms, representing a spike waveform using four features. Among the FE algorithms that are feasible for efficient online implementation, the notable algorithms are the MD, DD, and ZCF. It has been shown in [110] that DD-2Ex is a good candidate for feature extraction due to its immunity to noise and its tolerance for similar-shaped spike waveforms, the DD-2Ex with scaling factors $\delta = 3, 7$ performs the best based on its computational complexity, performance, and 16 times dimensionality reduction from $m = 64$ spike waveform samples to $n = 4$ features.

4.3 Partially Binarized Neural Networks

To map the newly extracted spike's feature vector to a particular spike class in real-time, we propose to utilize a neural network-based classifier. The computation of neural networks is performed by a set of processing elements, so-called artificial neurons (ANs), which interact with one another through weighted connections (synapses). The synaptic weights control how the network responds to specific input stimuli. The accumulated weighted input activities, either from network input or from pre-synaptic ANs, is passed to a non-linear activation function to produce the AN's output [85].

The size of networks and the numerical resolution of the synaptic weights and activation functions pose a strict limitation on the types of networks that can be used for extremely area- and power-constrained brain-implantable applications. Binarized neural networks (BNNs) have been introduced to significantly reduce the computational complexity and memory requirements by performing simple binary operations [74]. The binarization kernel is given as $k_b = sign(k)$, where k_b will be 1 for positive values of k and -1 otherwise.

It has been shown that using the binarization kernel at the input and output layers reduce the model's accuracy compared to binarizing only the hidden layers [88]. By performing feature extraction and representing the spike waveform with only 4 data points, it will be inevitably more difficult for the network to learn additional information from fewer data points. Moreover, as the spikes have already undergone feature extraction, binarizing the input layer of the BNN may further reduce the accuracy of the sorting process. In the proposed PBNN, the input layer's synaptic weights remain binarized, however, contrary to a BNN, the PBNN employs one of four different quantization modes: Mode 00, Mode 01, Mode 10, and Mode 11, which denote where the binarization kernel k_b is employed. Mode 00 does not employ k_b, Mode 01 applies it only to the output layer's weights, Mode 10 applies it only to the outputs from the previous layer, and Mode 11, as in a BNN, applies it to both the previous layer's outputs and the output layer's weights. For modes other than 11, we apply the sigmoid activation function to the network output. For hardware implementation, the complexity is reduced by using a quantized sigmoid function QSigmoid(z,b) that will be equal to 1 if the accumulated weighted activity z is greater than or equal to the neuron's bias b. Because the sigmoid function is symmetrical around the y-axis at input $z = 0$ with $f_{\text{sigmoid}}(0) = 0.5$, QSigmoid$(z,b)$ can be considered as applying a threshold of 0.5 to the output of the sigmoid after adding the bias b. Note that the Qsigmoid activation function is only applied during the forward propagation of signals at the output layer during PBNN inference. Employing this approximation during the training phase makes it challenging to generate useful gradients for approximating the optimal parameters of the network.

Table 4.1. The classification accuracy of the PBNN over various datasets and quantization modes

Dataset	Mode 00	Mode 01	Mode 10	Mode 11	*k*-Med	Dataset	Mode 00	Mode 01	Mode 10	Mode 11	*k*-Med
Easy1 0.05	93.88	93.17	56.54	33.49	93.59	Easy2 0.15	90.11	60.61	31.039	34.69	88.43
Easy1 0.10	95.46	93.26	63.82	52.69	95.46	Easy2 0.20	82.93	61.1	36.048	32.57	80.31
Easy1 0.15	94.25	67.09	51.79	30.17	93.67	Difficult1 0.05	92.76	85.81	36.041	33.53	92.9
Easy1 0.20	93.23	76.33	56.90	35.10	93.09	Difficult1 0.10	91.95	72.02	32.318	33.33	91.01
Easy1 0.25	92.04	68.56	56.81	43.71	91.81	Difficult1 0.15	83.38	63.81	36.834	35.97	81.72
Easy1 0.30	87.55	67.69	53.02	40.79	88.77	Difficult1 0.20	72.25	61.93	38.506	31.62	68.81
Easy1 0.35	83.87	72.4	49.08	34.79	85.71	Difficult2 0.05	93.90	67.75	33.135	34.91	94.20
Easy1 0.40	81.78	63.71	50.81	35.39	79.0	Difficult2 0.10	90.62	64.93	49.206	34.92	90.04
Easy2 0.05	95.89	62.39	31.96	32.40	95.74	Difficult2 0.15	80.88	64.24	64.82	32.26	78.77
Easy2 0.10	93.11	61.43	36.36	33.87	93.18	Difficult2 0.20	72.10	65.52	57.36	34.7	72.67

4.3.1 Evaluation of BNN and PBNN for Feature Vector Classification

We employ a PBNN as a feature vector classifier in our designed spike sorting system. In order to evaluate the performance of the spike sorting system, we employ the widely used WaveClus datasets [89]. The WaveClus datasets consist of 20 simulations of neural recordings with four levels of difficulty, Easy1, Easy2, Difficult1, and Difficult2. The difficulty level denotes the similarity of the spike waveforms between three single units present in the recording. The simulated recordings have varying levels of noise with a standard deviation between 0.05 and 0.40 relative to the amplitude of the spikes. Figs. 4.2(a) and (b) show the similarity of the spike waveforms for different classes in the Easy1 and Difficult1 datasets, respectively. One can see that the Difficult1 classes are harder to distinguish compared to the Easy1 dataset.

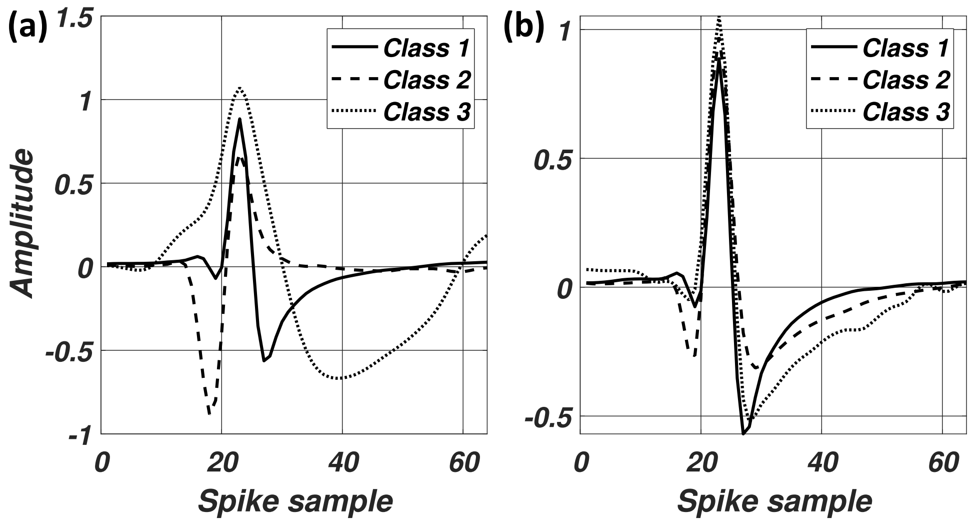


Figure 4.2. The mean spike waveforms for the (a) Easy1 and (b) Difficult1 WaveClus datasets.

First, DD-2Ex feature vectors are extracted from detected spike waveforms. The employed network consists of four input units, one per feature, three hidden layer neurons, and three output layer neurons, one per spike class. The chosen network topology of 4–3–3 provided the highest classification accuracy over different datasets and noise levels and increasing the number and size of the hidden layers yielded a negligible increase in classification accuracy. The 4–3–3 network topology allows the PBNN to sort the spikes of up to three neurons using a one-hot encoded output. Clustering was performed first offline on a subset of the feature vectors using the k-medoids algorithm. The resulting clusters were then assigned identifiers 1–3, matching the spike classes present in the dataset. The PBNN is then trained to learn the mapping between feature vectors and spike classes defined by the k-medoids algorithm. Because the PBNN learns the mappings produced by the k-medoids algorithms, the performance of the clustering algorithm should be verified prior to training. After the initial clustering, the PBNN was trained on feature vectors extracted from the spike waveforms given in the WaveClus dataset using the Python Larq framework [90] extension for Tensorflow. For optimizing the weight and bias parameters, we employ the RMSProp algorithm and a modified version of L2 regularization [112], which encourages the binarized weights toward values of -1 and +1. Layers of a PBNN that are not fully binarized employ normal L2 regularization rather than the modified L2. We train for 250 epochs, and use early stopping on the validation loss (mean squared error) to prevent overfitting of the model to the training set. Additionally, we utilize ten-fold cross-validation to estimate the performance of the model accurately. The data is split in ratios of 80%, 10%, 10% for training, validation, and testing sets, respectively. For example, the Easy_0.10 dataset consists of 3522 spike waveforms, where the training, validation, and testing subsets are comprised of 2818, 352, and 352 feature vectors, respectively.

Table 4.1 gives the classification accuracy of the neural network utilizing the WaveClus datasets and four quantization modes. The k-Med column indicates the performance of the k-medoids algorithm by assigning the test subset features to the nearest cluster centroid. It is apparent that Mode 00 consistently outperforms other quantization modes. Another commonly employed metric for quantifying the performance of a classifier is the F-Score $F = \frac{2TP}{2TP+FP+FN}$, where TP denotes the number of true positive classifications, FP denotes the number of false positive classifications, and FN denotes the number of false negative classifications. Over all Wave_Clus datasets and noise levels, our model achieves a median F-Score of 0.91, ranging from 0.95 to 0.71 with the standard deviation of 0.072. The median F-Score is computed using the testing data subsets, which the model has not observed during training, for each of the ten cross-validation sets. Using the *k*-medoids clustering and the trained network, the performance of the BNN classifying spike waveforms is verified over the Wave_Clus datasets. The median classification accuracy over the low SNR datasets Easy1 0.30, Easy1 0.35, and Easy1 0.40 was 0.86, 0.78, and 0.67, respectively. Based on the results given in Table 4.1, the PBNN classification of feature vectors offers more robust accuracy for low SNR data.

While the weight values for Mode 00 were not constrained to +1 or -1 with the binarization kernel, we found that representing the output layer's weights with only 4 bits (2 bits for each of the integer and fractional parts) has a negligible impact of less than 0.9% on the classification accuracy. This implies the robustness of the PBNN for classifying feature vectors compared to the classification of spike waveforms using a BNN. Moreover, when classifying feature vectors, applying the binarization kernel to a layer's output imposes a greater degradation than binarizing only the weight values. For example, one can note a significant performance degradation from Mode 00 to Mode 10. Thus, when sorting feature vectors, we employ the Mode 00 PBNN over the BNN (Mode 11). Note that all following analyses are performed using a PBNN in Mode 00.

While the results in Table 4.1 show that the PBNN performs well for classifying the activity of three neurons, it is often not known how many neurons' spikes are present in the neural signal apriori. To study the effect of MUAs, the PBNN is trained to sort spikes of up to 20 different neurons [94]. The dataset of synthetic spikes is generated similarly to the WaveClus dataset, but provides information about the number of neurons that can be distinguished. Fig. 4.3 shows the classification accuracy of the PBNN for increasing number of neurons in the neural recording. It can be seen that the PBNN can provide relatively accurate sorting for up to 7 units with 73% accuracy. Beyond this, it may be beneficial to increase the number of hidden units to allow the network more degrees of freedom for mapping features to cluster.

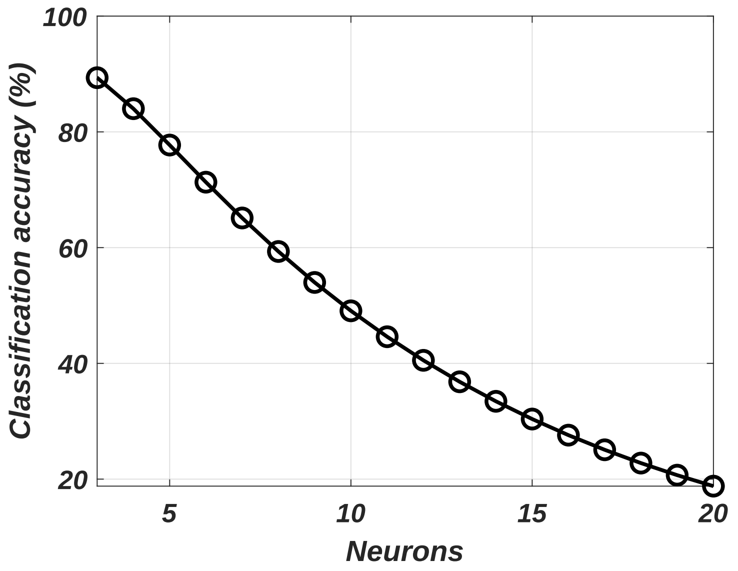


Figure 4.3. The classification accuracy of the PBNN over varying number of neurons.

4.3.2 Evaluation of BNN and PBNN-based Sorting Over Real Datasets

The ground truth information of the synthetic datasets offers a common benchmark for various sorting algorithms. Unfortunately, real neural recordings do not offer such ground truth information. We employ a two step approach for evaluating the performance of the spike sorting systems using real recorded data. Following spike detection, DD-2Ex features are extracted and a subset of them (80%) are clustered using the *k*-means algorithm. Since no ground truth information is available, we use the *k*-means cluster centroids as reference clusters, and assess the BNN and PBNN's performance by how well these networks classify spikes to match to their reference clusters. As given in Table 4.1, the conventional BNN does not perform well for classifying feature vectors.

To evaluate the BNN- and PBNN-based sorting schemes, we use a total of four neural recordings from two pigtail macaques (two each), aliases J and K, at the Washington National Primate Research Center. For training the BNN and PBNN, the same approach described in

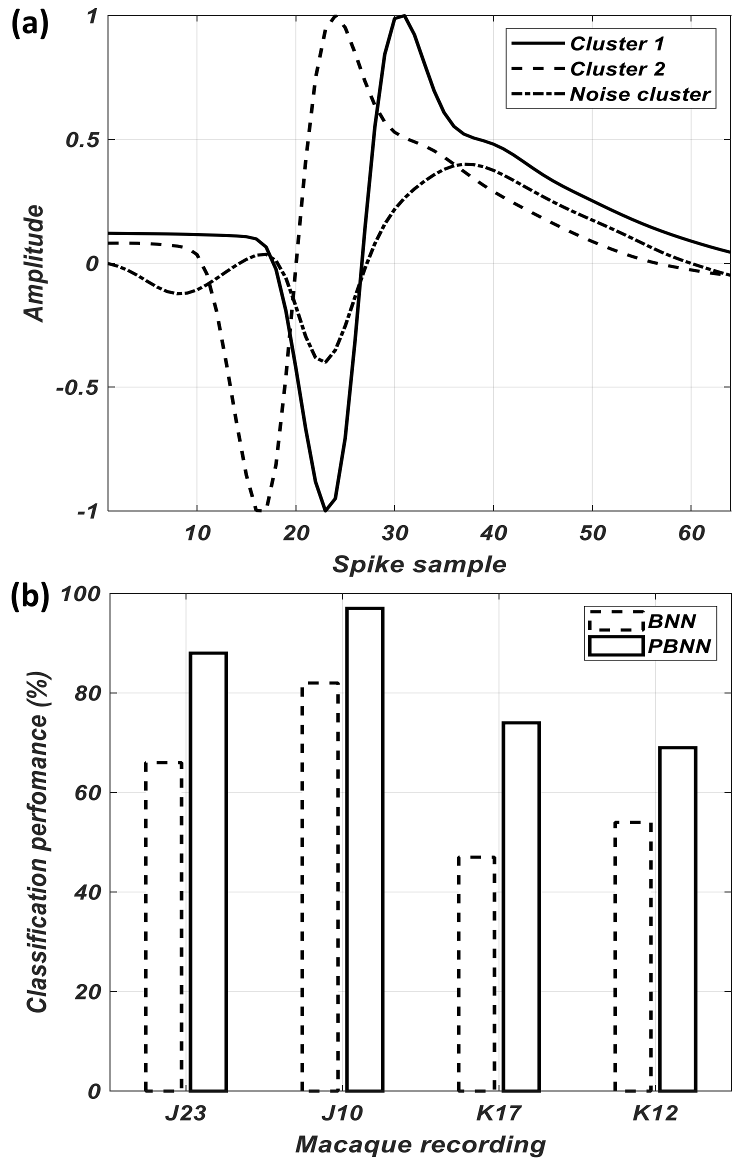


Figure 4.4. (a) Example cluster waveforms of the J23 macaque recording and (b) the classification performance of the BNN and PBNN-based clustering over four macaque recordings.

Section 4.3.B is employed. The performance of the BNN and PBNN was assessed using the 20% testing subset. Fig. 4.4 (a) shows an example of the mean *k*-means cluster waveforms and (b) shows the performance that the classification performance of the PBNN outperforms that of the BNN over four macaque recordings.

4.4 Hardware Architecture and Implementation of the PBNN-based Classifier

In our designed and implemented PBNN-based spike sorting system, the spike waveforms are first detected using the non-linear energy operator (NEO) algorithm [38] and aligned to the maximum amplitude, as described in our earlier work in [62] and [63]. In our design, we implement the NEO unit using log-based approximate multipliers to conserve circuit area and power [48], which we refer to approximated NEO. To estimate a spike detection threshold, the noise of the approximated NEO signal is computed using the root-mean square (RMS) method [40]. The spike detection threshold is then set to a scaled value of the estimated noise, given as $4\sigma_e$, where σ_e denotes the RMS of the estimated noise. The hardware implementation details are given in [113]. Once a detected spike is aligned, the waveform is sent serially to the FE module designed based on the DD-2Ex algorithm, as shown in Fig. 4.5. The FE module consists of a

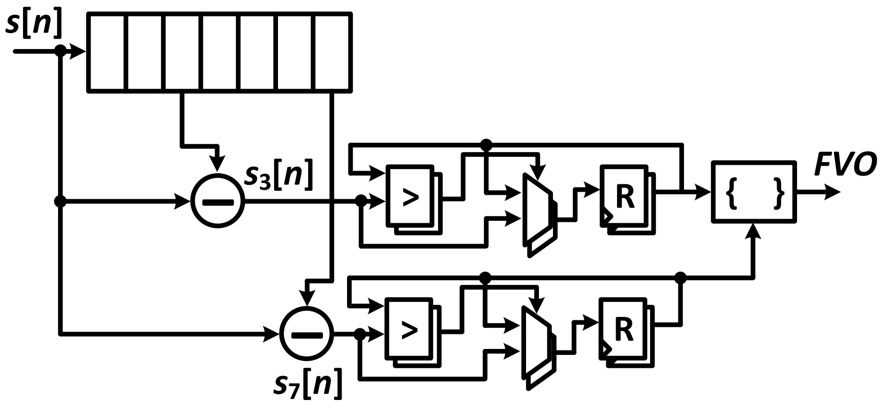


Figure 4.5. The block diagram of the DD-2Ex-based FE module.

7-word shift register for the scaling factors $\delta = 3$ and $\delta = 7$, and four sets of comparators and registers for finding the minimum and the maximum samples of $DD_{\delta=3}[n]$ and $DD_{\delta=7}[n]$. The registers *R* are updated only when the current value of the spike waveform is larger or smaller than the value currently stored in *R* for the maximum and the minimum features, respectively. After 64 clock cycles, the FE module will have computed the maximum and the minimum values of the $DD_{\delta=3}$ and $DD_{\delta=7}$ waveforms. These values are concatenated and passed on to the PBNN module via the *FVO* feature vector output port.

The top-level block diagram of the designed PBNN is shown in Fig. 4.6. The datapath classifies feature vectors into their corresponding spike IDs. It consists of parallel-input serial-output *PISO* shift registers, hidden-layer processing units *HPUs*, the output layer processing units *OPUs*, and the SRAM-based memory units. The 4-word feature vector outputs *FVOs* from the FE module are passed on to the PBNN datapath, which are then stored in the 4-word *PISO* shift registers. Each word in the *PISO* is passed on to the *HPUs* serially to compute the hidden layer outputs. The *Binary Weight SRAM* stores the binarized weights of the hidden layer and consists of four words of three bits, one word per input feature. The output of the *HPUs* is stored in the 3-word *PISO* for serial processing by the *OPUs*. Because the binarization kernel is not applied to the output layer weights, they are represented in a 10-bit signed fixed-point format with 4 and 6 bits for the integer and the fractional parts, respectively.

The *HPUs* and *OPUs* shown in Figs. 4.7(a) and 4.7(b), respectively, are similar modules which compute the accumulated input activity of the previous layer. The *HPUs* accept the binarized weight values, which are used as the select lines of the multiplexers to choose either the sample of the feature vectors or its negated value in two's complement format. Because the binarization kernel is not applied to the output of the hidden layer, the *HPUs* accumulate the weighted input feature vectors using 19-bit accumulators. Each *OPU* accepts 10-bit signed weights and computes the product of the weights and the *HPUs*' outputs. To represent the accumulated input activity of the *HPUs* with sufficient resolution, the *OPUs* use 32-bit accumulators. The accumulated value is then passed on to the Qsigmoid activation function and is compared against the bias values for each AN stored in the *Output Weight SRAM* shown in Fig. 4.6. The Qsigmoid function will then generate spike IDs.

We have implemented the PBNN-based spike sorting system in a standard 180-nm CMOS process. The chip layout, shown in Fig. 4.8, is estimated to consume 2.5 μW of power from a 1.8-V supply while operating at 24 kHz and to occupy 0.34 mm^2 of silicon area. The NEO and RMS-based spike detection unit occupies 0.16 mm^2 of silicon area and consists of

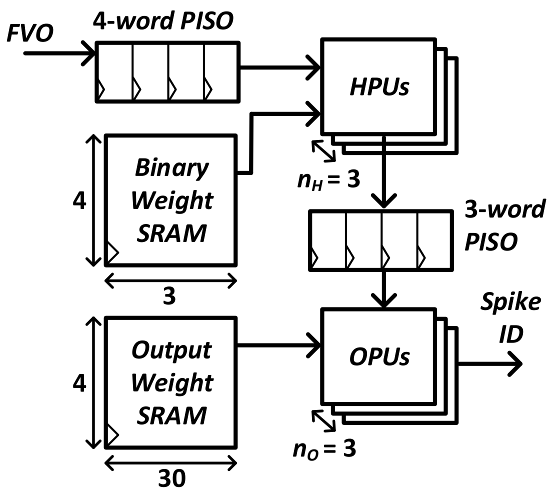


Figure 4.6. The block diagram of the PBNN.

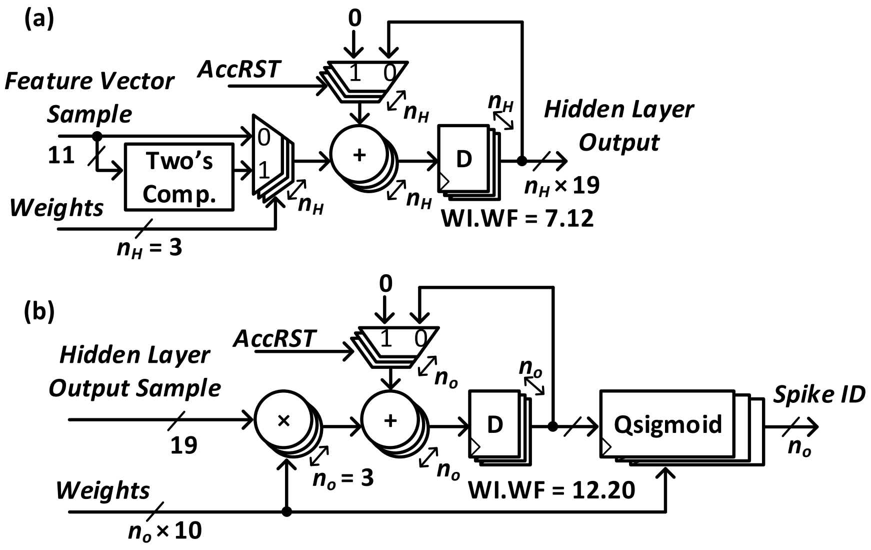


Figure 4.7. The block diagram of (a) the hidden layer processing units (*HPUs*) and (b) the output layer processing units (*OPUs*).

the approximated NEO unit, the adaptive RMS noise estimation unit, and the spike waveform alignment unit. The DD-2Ex feature extraction unit occupies 0.02 mm^2 of silicon area. The PBNN classifier occupies 0.15 mm^2 and consists of the Qsigmoid activation function for the output layer, an SRAM unit *SRAMFP*, which stores the output layer's weights in the fixed-point format, and the SRAM unit *SRAMB*, which stores the binarized weights for the input layer. The spike sorting system was described using Verilog HDL and the synthesis was performed using

Synopsys DC Compiler. After synthesis, the placement and routing was done with Cadence Innovus. For estimating the power consumption, the post place and route synthesized netlist was used and the switching activity of the ASIC was modeled using the WaveClus datasets. Table 4.2 gives the power consumption of the various ASIC modules, assuming a mean spike firing rate of 10 Hz. The frequency column denotes the rate at which the power is dissipated, i.e., for the spike detection it is dissipated on each input sample and for the subsequent units the power is dissipated on every spike event. It can be seen that the spike detection and noise estimation circuitries dissipate the most power and energy. It is also clear that the DD-2Ex feature extraction provides a very power and energy efficient operation. The energy shown in Table 4.2 is over one second of operation, and the total energy consumption is approximately 683.66 nJ.

Table 4.2. The power consumption of each ASIC module

Module	Power (μW)	Frequency (Hz)	Energy (nJ)
Spike Detection	0.68	24000	672
Spike Alignment	0.26	$\sim$10	8.77
Feature Extraction	0.02		0.52
PBNN Classifier	0.57		2.37

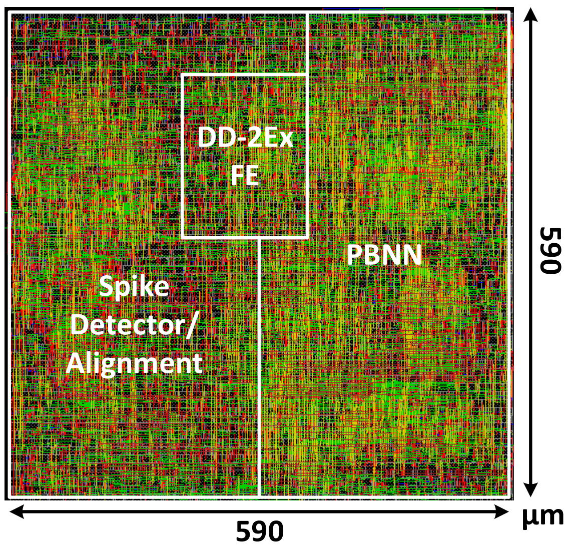


Figure 4.8. The layout of the designed PBNN-based spike sorting system.

Based on the results given in Table 4.1, although the performance of the *k*-medoids clustering is similar to that of the PBNN in Mode 00, to compare their hardware efficiency, we consider their computational complexities and their memory requirements based on three design parameters: the wordlength of the feature vectors w_k, the wordlength of the PBNN output layer parameters w_o, and the number of clusters N. The computational complexity of the PBNN and k-medoids can be compared based on the number of their operations. Given N clusters and four samples per feature vector, computing the Euclidean distance metric $\eta = \sqrt{\sum_{i=1}^{4} \left[x_i - y_i\right]^2}$ in the k-medoids algorithm, where x and y denote the template feature vector and the new feature vector, respectively, requires $7N$ additions and $4N$ multiplications, not considering the square root, which is unnecessary for comparisons. Comparing N values to one another would also require $N(N-1)/2$ comparisons, not accounting for the priority logic required to assign the feature vectors to only one cluster. Assuming that the complexity of an addition and a multiplication can be estimated as 2 and 4 times of a comparison, respectively [43], the computational complexity of the k-medoids algorithm is $30N + N(N-1)/2$ operations. For the PBNN, the multiplication with -1 at the input layer is realized using a two's complement operation, which requires 21 additions. The output layer's multiplication of a (1×3) vector and a $(3 \times N)$ weight matrix requires $3N$ multiplications and $2N$ additions. The QSigmoid function also requires N comparisons. The normalized computational complexity of the PBNN is thus $17N + 42$, which makes the PBNN algorithm computationally more efficient $N > 3$ clusters. The parameter memory stores the cluster centroids and weight matrices for the k-medoids and the PBNN algorithms, respectively. For the k-medoids algorithm, N template feature vectors are stored for comparison with the newly detected spike feature vectors, requiring a total of $4Nw_k$ bits. For the PBNN of size $4 \times 3 \times N$, the binarized input layer requires 12 bits and the output layer in Mode 00 requires $(N+1)w_o$ bits, a total of $(N+1)w_o + 12$ bits. Thus, the k-medoids algorithm will be more memory-efficient only if $w_k < (w_o(N+1)+12)/4N$. Given that the analog-to-digital converters in BMIs are typically 10 to 16 bits and w_o ranges between 4 and 10 bits, the PBNN is also more memory-efficient than k-medoids.

Table 4.3 gives the ASIC characteristics and implementation results of various state-

Table 4.3. The ASIC characteristics and implementation results of various spike sorting systems

Design	Ours	[64]	[102]	[63]	[62]	[65]	[66]	[67]	[68]	[23]	[69]	[24]	[22]
Algorithm	PBNN	ANN	BNN	OSort	TM	FE	OSort	OSort	FE	FE	TM	FE	FE
Median accuracy	0.91	0.98	0.91	0.87	0.90	0.77	0.75	0.93	0.84	0.85	0.93	0.86	0.92
Data rate reduction	2000×	1866×	2000×	1600×	3200×	11×	240×	278×	240×	–	–	257×	–
Adaptive spike detection	Y	N	N	N	N	N	Y	N	Y	Y	N	N	N
Supervised	Y	Y	Y	N	Y	Y	N	N	N	Y	N	N	N
Technology (nm)	180	32	180	32	45	90	65	45	45	130	40	65	65
Core voltage (V)	1.8	0.7	1.8	1.16	0.25	0.55	0.27	–	1.1	1.2	1.1	0.54	1
Frequency (kHz)	24	20	24	24	24	4000	480	56	960	160	500	3200	30
Norm. area/ch (mm^2)†	0.34	0.36	0.33	102.8	5.7	0.396	0.84	1.33	51.3	0.066	0.473	0.036	1.08
Norm. power/ch (μW)†	2.5	7.81	2.02	4.57	1.35	27.67	152.4	–	41.33	0.96	42.92	2.49	0.54
Norm. energy/ch (pJ) $^{\dagger\ \ddagger}$	104.16	468.6	84.16	190.41	56.25	1152.9	6350	–	1722	40	1788	103.7	22.5

† Scaled to a 180-nm process with a 1.8 V supply voltage, as described in [47].
‡ Normalized to a clock frequency of 24 kHz.

of-the-art spike sorting systems. In [64] we implemented an ANN-based spike sorting system composed of one hidden layer neuron and three output layer neurons utilizing the ReLU activation function. In [102] we implemented a BNN-based spike sorting system to classify spike waveforms in a standard 180-nm CMOS process. In [63] we designed and implemented an OSort-based spike sorting system in a standard 32-nm CMOS process. In [62] we designed and implemented a TM-based spike sorting ASIC in a 45-nm CMOS process. The work in [65] performs the NEO-based spike detection, aligns detected spikes to maximum derivative, and implements FE via discrete derivatives. Their design consists of four 16-channel modules which produce either aligned spikes or feature vectors, but does not perform real-time sorting. The work in [66] uses the absolute value detection scheme and implements the OSort clustering algorithm for 16 channels. In [67], spikes are first detected using a voltage threshold and aligned to maximum absolute amplitude clustered using OSort. The work in [68] performs spike sorting using NEO-based detection, maximum amplitude alignment, and FE using discrete derivatives. It supports an unsupervised learning process, similar to the OSort-based systems. The work in [23] presents a multi-channel spike sorting ASIC based on the Haar wavelet FE algorithm. The design in [69] presents a multi-channel TM-based spike sorting ASIC with a built-in OSort learning system. The work in [24] presents a multi-channel spike sorting processor based on the integer coefficient FE and clustering. Since the ASIC designs in [65, 66, 23, 69, 24] are multi-channel systems, for a fair comparison, their equivalent area and power consumption results for the single-channel sorting are given in Table 4.3. Finally, the work in [22] presents the ASIC implementation of a dictionary learning-based FE unit, in which the dictionary values are constrained to -1, 0, or 1 and no multiplications are used. However, the ASIC implementation is for the FE module only. We have also reported the normalized power consumption and area of the designs listed in Table 4.3 to 180-nm technology with a 1.8 V supply voltage, following the scheme presented in [47]. The spike sorting systems, which employ unsupervised learning, support real-time classification of detected spike waveforms. However, the TM-based and the neural network-based spike sorting systems that require pre-processing of the neural recordings in order to generate the template waveforms and synaptic weights, respectively, need a significantly smaller storage with a lower computational complexity, while providing comparable classification accuracy. Compared to the TM-based sorting, in which a distance metric is used to quantify the similarity between two spike waveforms, the neural network-based schemes can offer a more robust classification due to the non-linearity of the network model. Note that the work in [23] presents the most compact design, however, its median classification accuracy is relatively low at about 70% for low SNR neural data, while that of the PBNN-based classifier is 83%. It is shown that the designed PBNN-based spike sorting system not only offers similar accuracy to those of the state-of-the-art systems, also as opposed to the other designs for which performance degrades with increasing noise levels, it provides a robust classification accuracy over various noise levels and datasets.

Among the neural network-based architectures, one can see the advantage of reducing the computational complexity of the PBNN. Compared to our ANN classifier in [64], our BNN classifier in [102] reduced the area slightly, from 0.36 mm^2 to 0.33 mm^2, but reduces the power consumption by over two-fold. By employing a PBNN classifier, the size of the network and the number of parameters can be reduced. At the cost of slightly increased area (only 0.01 mm^2 larger than our BNN ASIC in [102]), the power of the classifier is reduced by a factor of 3.42. Assuming a 24 kHz sample rate with 10 bits per sample, the input rate 240 kbps. With an average neuron spiking rate of 40 spikes per second [96], representing the PBNN's classified outputs with three bits reduces the data rate to 120 bps. This results in a 99.9% reduction in the output data rate compared to the input sampling rate. Since the energy required to transmit one bit of data is approximately 3 nJ [97], the power consumption for the wireless transmission of the spike IDs is about 360 nW. The total power dissipation of our synthesized ASIC design is thus 2.8 μW with the power density of 8.23 $\mu\text{W}/\text{mm}^2$, which satisfies the tissue-safe requirement for the brain implantable devices [98].

4.5 Conclusion

This chapter presented the efficient hardware design and implementation of a spike sorting system utilizing a partially binarized neural network (PBNN) classifier. Among various efficient feature extraction algorithms for hardware realization, the discrete derivatives-based feature extraction algorithm was chosen and employed to reduce the dimensionality of neural spike waveforms and decrease the memory requirement of the neural network. It was shown that the designed spike sorting system not only offers similar accuracy to those of the state-of-the-art systems, also as opposed to the other designs for which performance degrades with increasing noise levels, it provides a robust classification accuracy over various noise levels. The implemented PBNN-based spike sorting system meets the strict power dissipation constraints of implantable devices, making it applicable in brain-machine interface systems.

4.6 Publications

The following article was published as an outcome of the research conducted in this chapter: D. Valencia and A. Alimohammad, “Partially binarized neural networks for efficient spike sorting,” Biomedical Engineering Letters, vol. 13, pp. 73 – 83, February 2023.

Chapter 5

Efficient In Vivo Neural Signal Compression Using an Autoencoder-based Neural Network

5.1 Introduction

The field of intra-cortical brain-computer interfaces (BCIs) has been rapidly evolving over the past decade. BCIs effectively translate (decode) recorded neural signals into a quantifiable representation for augmenting or enhancing the user's working experience. The input to a neural decoding algorithm is a specific representation of the neural activity and the output is either a continuous variable or a discrete selection. For example, the former can represent the kinematic variables to control a computer cursor or a robotic limb [114, 115], while the latter may represent the particular mental state of a user (e.g., sleeping or alert) [116] or the end-goal of a planned movement [117, 118]. The application of novel machine learning (ML)-based decoding algorithms has enabled increasingly complex BCI applications, such as thought-to-text [11] and thought-to-speech synthesis [119].

Intracortical neural recording systems have continuously advanced from the widely-employed Utah Array [120], supporting up to a hundred recording sites, to high-density recording electrodes, such as Neuralink [8] and Neuropixels [9], supporting hundreds of recording sites per implantable recording shank and hence, thousands of recording channels. An increasing number of recording channels will inevitably impose a higher data rate requirement. For example, a neural recording system with 1000 recording electrodes, sampled at 20 kS/s with 16-bit resolution, would require a data rate of 320 Mbps. The state-of-the-art wireless transmission of neural data has a mean energy dissipation of 6.7 pJ/bit [121, 122, 123, 124, 125], which would imply 2.14 mW of power for wireless transmission alone. In accordance with the the Food and Drug Administration [126], considering neural tissue-specific absorption rate, the limit for a safe wireless power transfer is 7.7 mW. Thus, transmitting raw neural signals would consume over 27% of the available power budget. By employing in vivo neural signal processing and compression, the data rate requirement can be drastically reduced. For example, as shown in Fig. 5.1, if in vivo compression is able to reduce the data rate by a factor of 20, the wireless transmission requires only 0.1 mW of power, i.e., approximately 1.4% of the available power budget. Therefore, efficient realization of in vivo signal compression becomes crucial for BCIs employing high-density microelectrode arrays (MEAs).

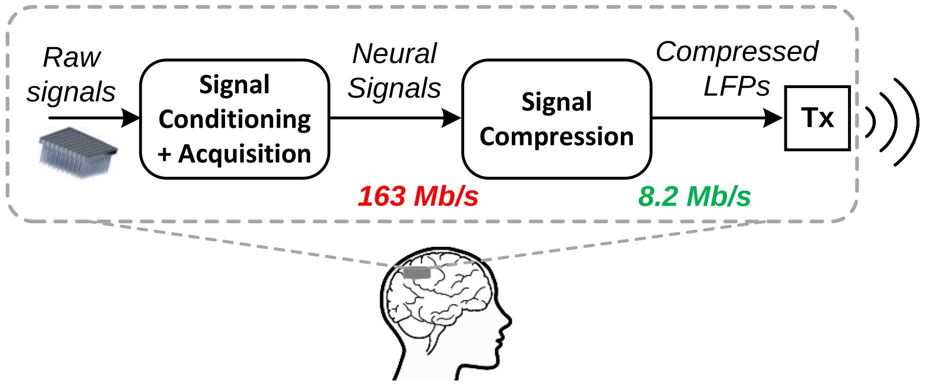


Figure 5.1. Intracortical recording and wireless transmission of compressed neural signals, employing 1000 recording channels.

Compression schemes are generally divided into two categories. Lossless methods involve reducing the dynamic range of the neural signals and encoding the signals as variable bit-rate data streams [127, 128, 129]. One of the commonly-employed lossless compression methods for LFPs is to exploit the temporal redundancy and spatial correlation of LFPs. The temporal difference $x_d[n] = x[n] - x[n-1]$ reduces the dynamic range of the signals and consequently, the required number of bits per sample to about a half [127]. With a reduced numerical range, neural signals can then be represented using Huffman coding [130], which encodes more commonly occurring samples using fewer bits. The combination of temporal difference and Huffman coding is considered as a lossless compression scheme, which generally offers a compression rate on the order of 2 to 5 [127, 128, 129].

While lossless methods provide perfect reconstruction, lossy methods can significantly increase the data compression rate by employing spatial downsampling. The underlying principle for employing lossy methods in the context of neural signals is the relatively large amount of spatial correlation among neural recording channels. Considering that the state-of-the-art neural decoding algorithms are relatively robust to noise and signal perturbations [131, 132], by tolerating relatively small signal errors, employing a lossy compression scheme might be a more viable approach for BCI applications. Sources of noise include instrumentation perturbations due to electrode micro-motions, bit errors during wireless transmission, and quantization noise caused by finite numerical resolutions. Compressed sensing (CS) [133] is a lossy scheme used for neural signals [134, 135, 136], where signal $\mathbf{x} \in \mathbb{R}^n$ is multiplied with a sensing matrix $\Phi \in \mathbb{R}^{m \times n}$ to produce $\mathbf{y} \in \mathbb{R}^m$, effectively compressing the signal by a factor of m/n. A variety of CS-based algorithms are employed in silico to reconstruct the original signal $\mathbf{x}$ from the compressed (encoded) signal $\mathbf{y}$ [134, 135, 136].

In this chapter, we propose the novel application of a ML-based compression scheme based on autoencoders (AEs). Compared to the state-of-the-art neural signal compression circuits, the designed AE-based compression scheme offers a greater compression rate, higher signal-to-noise and distortion ratio, smallest silicon area, and lowest power consumption. The rest of this chapter is organized as follows. Section 5.2 discusses the motivation toward employing the local field potentials over neural action potentials. Section 5.3 discusses the algorithm and performance of the designed autoencoder-based compression scheme. For an efficient realization of the in vivo autoencoder, various algorithmic and architectural optimization techniques are employed and presented in Section 5.4. The architecture of the designed compression hardware is presented and discussed in Section 5.5 and its implementation characteristics are compared against the relevant compression circuits. Finally, Section 5.6 makes some concluding remarks.

5.2 Local Field Potential-based BCIs

Compared to the non-invasive electro-encephalography (EEG), the invasive intra-cortical recording modality offers the highest temporal and spatial resolutions in which the neural activity can be represented, as either the excitation of individual neurons, called single-unit activities (SUAs), or an ensemble of neurons, called multi-unit activities (MUAs). Applications employing SUAs or MUAs conventionally perform in vivo spike detection [113] to reduce the wireless data rate requirements. Neurons are known to fire relatively infrequently with respect to the sampling rate, on the order of 40 Hz [137]. Additionally, due to the physiological refractory period of neurons, spiking activity is usually represented at the millisecond level with the required bandwidth of at most one kHz per channel. SUAs can be obtained by spike sorting, which can be viewed as a clustering process where spikes fired from the same neurons are grouped together [138]. Some BCIs employ in vivo circuitry to classify spike waveforms and transmit only 2 – 3 bits per spike class [62, 63, 102, 139], while others avoid in vivo spike sorting and instead transmit the MUA spike waveforms, which requires about 2 to 3 milliseconds of data per spike waveforms. By transmitting only spiking activity, BCIs employing in vivo spike classification have a compression rate of 1000 – 6000 [62, 63, 102, 139] at the expense of spike sorting computations, while BCIs that transmit the entire spike waveforms have compression rates of 2 – 44 [140] at the cost of greater data transmissions. Compared to the SUA-based neural signal processing, MUA-based decoding, however, does not require computationally-intensive spike sorting [62, 63, 102, 139]. It has been shown that the overall decoding performance degradation is negligible when employing MUAs [141]. In addition, transmitting only MUA events (e.g., spike counts) drastically reduces wireless transmission rates. For example, if neural signals are sampled at 10 – 30 kS/s with a 10 – 16 bit resolution, the required wireless data rate is 100 – 480 Kbps per recording channel. With a 96-channel Utah Array, the staggering transmission rate is 9.6 – 46 Mbps. However, MUA features are commonly represented as spike counts over an interval of 1 – 25 milliseconds. Most implementations of spike detection impose a biologically plausible spike refractory period of one millisecond [131, 1] and hence, one millisecond spike bins would require the data rate of only one Kbps per recording channel, yielding a data rate of at most 96 Kbps for a 96-channel Utah Array, resulting in at least a 100 times data rate reduction. Therefore, MUAs have been widely employed in both clinical trials and therapeutic applications of BCIs.

Neural activities can alternatively be represented by the local field potentials (LFPs), which are formed by the aggregate synaptic activities of populations of neurons. Compared to the SUAs and MUAs, LFPs represent slower variations in the neural signal's voltage and have lower spatial resolutions (LFPs: 0.5 mm, MUAs: 0.1 mm, and SUAs: 0.05 mm) [142, 29, 143]. Due to recording variations and instabilities, such as electrode drift and neuron drop-out [144], as well as potential scarring between the electrode-tissue interface over a relatively long period of time [145], LFPs are considered more stable compared to the SUAs and MUAs. While many of the MUA-based BCIs focus on decoding the neural activity in the motor cortex, LFP-based BCIs decode neural activities of the cognitive regions of the brain, such as the posterior parietal cortex, which allows higher-level cognitive decoding than that of the lower-level continuous motor control [146, 118]. Various, studies have shown that reliable neural decoding can be performed using LFP signals [147, 146, 118]. A study reported in [147] found that various movement intentions, such as the imagined end-point, kinematic trajectory, and type of movement, can be predicted reliably from the LFP signals.

From the perspective of implantable integrated circuits for neural recording, the acquisition and processing of LFPs offers an opportunity for significantly reducing the in vivo power consumption. For example, neural signals are often sampled at a rate of 10 – 30 kS/s to provide the necessary temporal resolution for detecting action potentials (spikes) [131]. The acquisition of neural signals consists of low-noise amplifiers and analog-to-digital converters (ADCs), most of which employ 10 – 16 bit resolutions and have a nominal power consumption of 0.25 μW – 7.3 μW per recording channel. Because the LFP frequencies of interest are often up to a few hundred hertz (i.e., 0.1 – 300 Hz), the sampling rate can be reduced significantly compared to that required for spike-based processing, typically between 1 – 2 kS/s. A five fold reduction in the sampling rate and signal bandwidth would reduce the power consumption of the in vivo signal acquisition and conditioning [131]. Also, to extract SUAs or MUAs from the recorded neural signals, additional in vivo neural signal processing, such as adaptive threshold estimation and spike detection, which consumes between 0.6 and 1.78 μW of power per recording channel, is required [132, 148]. Thus, a reasonable estimate for processing MUAs is approximately 8 μW of power per channel. LFPs require no threshold estimation or additional processing aside from signal filtering to discard signal components above 300 Hz. While the in vivo neural signal processing for LFPs is more power efficient than that for SUAs/MUAs, the data rate requirements, however, are considerably higher. For example, considering LFPs sampled at 2 kS/s with a 16-bit resolution, its wireless transmission would require 32 Kbps/channel, while the spike counts over one millisecond bins would require only one Kbps/channel. Therefore, the direct transmission of continuous LFPs would require over 3 Mbps for a 96-channel electrode array. The data rate would increase as higher density electrodes are implanted for emerging BCI applications.

In practical systems, various algorithmic and architectural-level schemes can be employed to efficiently manage the in vivo BCI power consumption. For instance, in an asynchronous (self-paced) BCI paradigm, the user's desire to engage in the BCI task can be detected with an intention estimation logic. When the user is not actively involved in the BCI task, a significant portion of signal acquisition and processing can be powered down, leading to substantial power savings [132]. While the topic of user's intention estimation is beyond the scope of this work, its potential to effectively disable the majority of in vivo circuitry, along with the designed signal compression scheme, could significantly reduce the energy dissipation of LFP-based BCIs.

5.3 Autoencoder-based Compression

LFPs are formed by the accumulated synaptic activity of populations of neurons and hence, they can be readily detected by recording channels that are relatively close to one another. Fig. 5.2 shows the inter-channel correlation over 30 seconds of a neural recording from a non-human primate performing self-paced reaching tasks [149]. One can see that various sub-regions within the recording array are highly correlated. Most lossy compression methods aim to exploit this inherent spatial correlation to perform a spatial downsampling of the LFPs. By exploiting the spatial correlation, the LFPs can be represented using either a subset of the channels or a linear transformation of the original signal.

An autoencoder (AE) is a neural network, consisting of an encoder network, which reduces the spatial dimension of the input data, and a decoder network, which is trained to

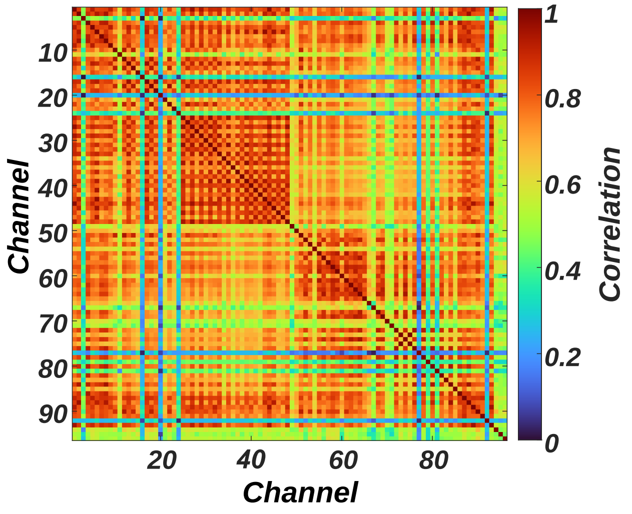


Figure 5.2. The inter-channel correlation over 30 seconds of a neural recording.

reconstruct the original input. Conventional AEs employ mirrored network architectures such that the encoder and decoder have the same number and type of layers, but in the reverse order. Due to their data-driven approach to learn an optimal low-dimensional representation of the input signal, AEs have been previously employed for spike waveform feature extraction [150, 151] and for compressing spike waveforms [140].

The block diagram of the designed and implemented AE-based compression architecture is shown in Fig. 5.3. The LFPs are derived from the recorded neural signals using a second-order Butterworth low-pass filter with a cutoff frequency at 300 Hz. The encoder network is relatively small and is feasible to be realized as an implantable circuit in vivo for compressing filtered neural signals before wireless transmission, while the decoder is implemented in silico to reconstruct the neural signals for subsequent processing. The encoder network consists of a single dense layer DL_E and the decoder consists of a recurrent layer RL_D and a dense layer DL_D. The encoder layer reduces the input dimension from N to F, where N denotes the number of channels in the recorded signal and F denotes the number of units in the dense layer. The recurrent layer RL_D accepts the F-dimensional outputs of the encoder, learns temporal information within the encoded signals, and provides an R-dimensional spatially upsampled output. The decoder's dense layer DL_D performs the final spatial upsampling of R to N and reconstructs the input signals. The units in the DL_E employ Tanh activation functions, while the DL_D performs a linear regression to reconstruct the channel activity. The compression rate of the designed model is $CR = Nw_i/Fw_e$, where w_i and w_e denote the resolution of the input data and the output resolution of the DL_D, respectively. For example, for a 96-channel Utah Array with $F = 10$ units, 16-bit data samples and 10-bit outputs, the compression rate is 15.36. Note that conventional digital acquisition (DAQ) systems typically employ 16 bits of resolution for sampling neural signals. In practice, the lower-order bits of the digitized neural signals may be discarded if they fall below the noise floor of the amplifier [129]. For instance, with a given signal amplifier featuring V_1 μV_{RMS} input referred noise and an ADC resolution of V_2 μV per bit, the least significant $\log_2(V_1/V_2)$ bits can be discarded. It is important to note that our analyses do not involve such resolution reduction, as the focus is on the design and implementation of a parameterizable auto-encoder-based compression scheme that can be synthesized onto a custom resolution, independent of the specification of the employed DAQ system.

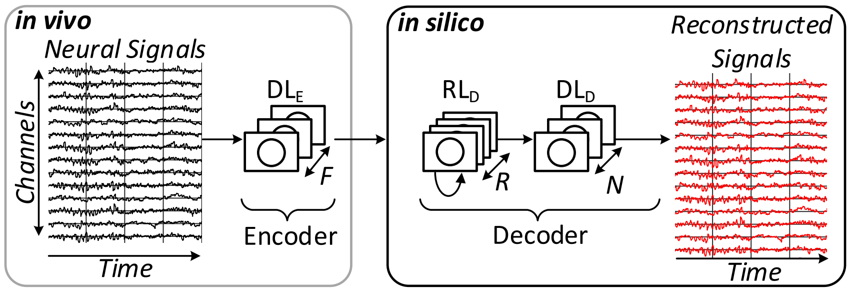


Figure 5.3. The block diagram of the designed autoencoder-based compression scheme.

The designed AE-based network is trained with the Python Tensorflow framework using two publicly available datasets. Dataset I [149] consists of neural recordings from a Macaque monkey while performing a self-paced point-to-point reaching task. Dataset II [152] consists of neural recordings from two monkeys K and L while performing an object reach and grasp task. Dataset I is sampled at $f_s = 24.4$ kS/s and is filtered using an anti-aliasing low-pass filter at 7.5 kHz, built into the recording system. Dataset II is sampled at $f_s = 30$ kS/s and is filtered using a high-pass filter at 0.3 Hz and a low-pass filter at 7.5 kHz. Dataset I consists of 30 recordings acquired over 7 months and Dataset II consists of two recordings (one for each monkey) during a single recording session. Both datasets are downsampled to 2 kS/s by applying a low-pass filter with the cutoff frequency at 1 kHz followed by decimation. The model was trained using the Adam optimizer and the mean absolute error between the input signals and the reconstructed signals was considered as the loss function.

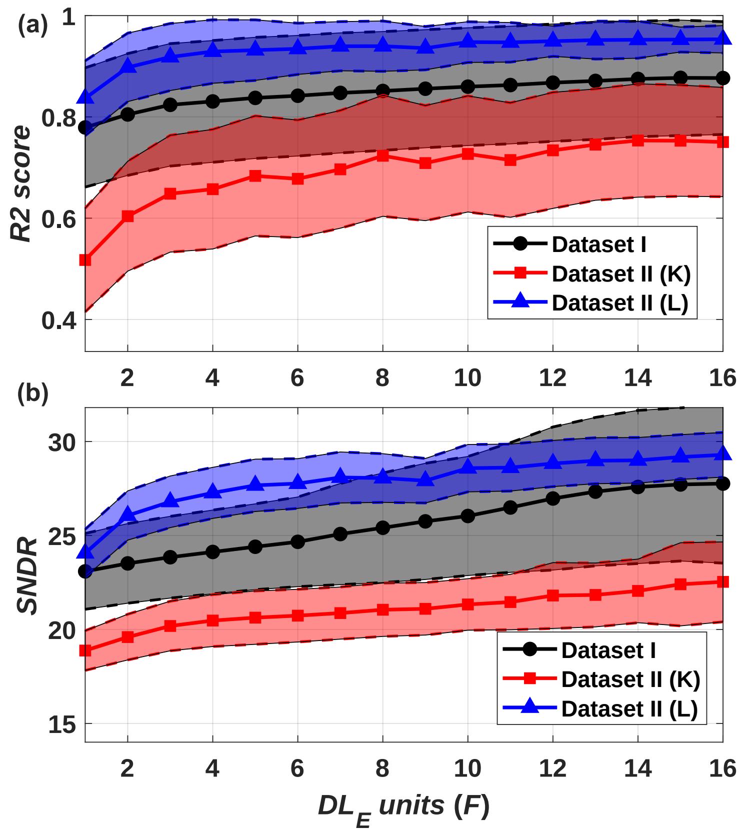


Figure 5.4. The mean and standard deviation of (a) R2 scores and (b) SNDRs for the reconstructed LFP signals analyzed over various numbers of DL_E units F.

Lossy compression methods often report reconstruction error using the signal-to-noise and distortion ratio (SNDR) metric, defined as $\mathrm{SNDR} = 20\log_{10}\frac{||\mathbf{x}||_2}{||\mathbf{x}-\hat{\mathbf{x}}||_2}$, where $\mathbf{x}$ and $\hat{\mathbf{x}}$ denote the original and reconstructed neural signals, respectively, and $||\cdot||_2$ denotes the L2-norm [153]. The performance of the designed model was evaluated using the R2 score by analyzing the similarity between the true and reconstructed LFPs. The R2 score describes the ability of the reconstruction model to capture the variance present in the data and is given as R2$= 1 - \frac{\sum_i(\hat{y}_i - \bar{y})^2}{\sum_i(y_i - \bar{y})^2}$, where $\hat{y}_i$ and

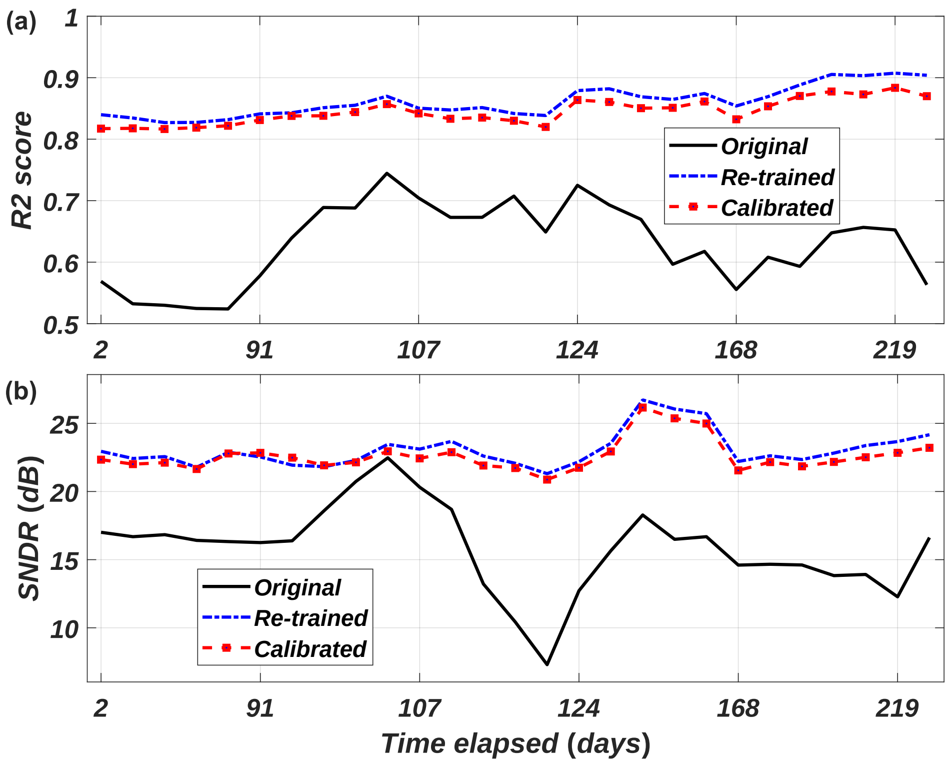


Figure 5.5. The mean of the (a) R2 scores and (b) SNDRs of the original, re-trained, and calibrated models over all recordings of Dataset I, respectively.

y_i denote the i-th reconstructed and true outputs, respectively, and $\bar{y}$ denotes the mean. While the compression rate increases with a smaller F, the increased spatial downsampling may adversely impact the reconstruction quality. Using the first recording from Dataset I and both recordings from Dataset II, the model was trained by varying F between 16 and 1 and R between 256 and 64 with 16 evenly spaced intervals of 13. This approach was employed to ensure compensation between smaller values of F and larger values of R in the decoder. As the recordings of the two datasets are from three different animals (Dataset I with one animal, and Dataset II with two different animals), the model was trained separately on each dataset. The training was performed on the first 80% of each recording session, with the next 10% used for validation and the final 10% for testing. The training, validation, and testing sets were then split into time spans of 160 ms, which corresponds to 320 time steps with $f_s = 2$ kS/s. The training was performed for up to 1000 epochs using early stopping on the validation set to prevent overfitting. Fig. 5.4(a) shows the mean R2 scores over various number of units F for the first recording session of Dataset I and both recordings of Dataset II, and the shaded areas show the standard deviation across the recording channels. It was found that the model with $F = 8$ and $R = 128$ provides a median R2 score of 0.852 ± 0.04 trained on each of the 30 recordings in Dataset I, 0.72 ± 0.23 and 0.93 ± 0.09 over the Monkey K and L recordings of Dataset II, respectively. It was found that the encoder dimensionality F has a stronger impact on the overall performance of the model compared to that of the decoder dimensionality R. While the decoder dimensionality could be further increased, values of R beyond 128 did not yield considerable performance gains. Fig. 5.4(b) shows the mean of SNDR. Similarly to those of the R2 score, an increasing number of units F shows a small increase in SNDR. The median SNDR over each of the 30 recordings in Dataset I was 22.79 ± 2.19, and 21.19 ± 2.89 and 28.89 ± 2.66 over the Monkey K and Monkey L recordings, respectively.

As shown in Fig. 5.4, the performance of the model is data-dependent, as is in general for the ML-based algorithms. One important characteristic of the ML-based compression models is their ability to generalize to future data. Fortunately, Dataset I contains several months of recordings, which allowed the generalization of the baseline model trained on the first recording session to be tested on all subsequent recordings. Figs. 5.5(a) and (b) show that the designed model provides a variable performance, with the R2 scores between 0.5 and 0.73 and the SNDR between 5.26 and 22.93, which is due to the changes in the statistics of the neural signals per recording channel. For a relatively steady performance, the original model is either re-trained or calibrated using a small amount of new data. Figs. 5.5(a) and (b) show a negligible difference between the re-trained model and the calibrated version of the original model, however, the calibrated model only requires 10% of the new data whereas the re-trained model requires 80% of the new data for training. Additionally, the calibrated model retains the weights of the encoder network and only adjusts the weights of the decoder. In addition to the R2 score, SNDR is a more commonly employed metric to quantify the performance of lossy compression algorithms. The range of acceptable R2 scores depends on the underlying application (i.e., neural decoding, which is outside the scope of this work). However, modern ML-based decoding algorithms for spike-based BCIs are robust to about 10% of input spiking errors [132, 131]. Although the signal modality of our design is LFPs, it can be inferred that relatively high R2 scores, where the model can account for at least 70% of the variance of the LFP signals, may be sufficient for reliable neural decoding.

One important consideration is that the employed calibration requires the acquisition of uncompressed LFP signals to create the new ground truth data. Implantable BCI systems generally operate in one of the two modes. In the “active” state, the compression circuitry is enabled to reduce the transmitted data rate while the user is engaged in the BCI task. In the “training” state, there is no need for real-time transmission of compressed neural signals. During the “training” state, raw LFP waveforms can be recorded and transmitted for the calibration or re-training of the AE decoder network. Recently employed BCIs, such as NeuraLink [8] and miniaturized implantable recording motes [154, 155, 156], utilize near-field communication technology to establish communication with in silico microprocessors. This allows for higher communication bandwidths and long-term data storage during the “training” phase.

5.4 In Vivo Encoder Optimizations

Because the encoder network will be realized in vivo, it is imperative to investigate potential architectural optimizations for reducing the memory requirements, computational complexity, and data resolution of the encoder output. Since the DL_E requires one weight value per output node per input channel, the total memory for the DL_E is $F(N+1)$. One method for lowering the total number of parameters is to reduce F, however, as discussed in Section 5.3, this will result in degraded reconstruction quality. An alternative is to lower the number of input channels N, however, instead of selecting a subset of channels to process, the spatial correlation among the channels is exploited to reduce the input dimension of the network. For an efficient signal conditioning, we propose the staggered spatio-temporal downsampling (SSTD) recording configuration, as shown in Fig. 6. Conventional analog front-end (AFE) circuitry for neural recording typically employs a shared ADC among various recording sites along with low-noise amplifiers (LNAs) [157]. To accurately sample each signal, the switching frequency of the multiplexer f_c is toggled at the rate of Nf_s, where N denotes the number of recording sites sharing the ADC, and f_s denotes the sampling rate of the underlying neural signal. Due to the relatively high inter-channel correlation within recorded neural signals, each channel is temporally downsampled by a factor α. This downsampling is achieved by reducing the switching frequency f_c by a factor α, providing a temporally downsampled data stream for each recording site. Lowering the input dimension of the network by a folding factor of α reduces the total memory requirement to $F(N/\alpha+1)$. The decoder network is still trained to reproduce the original input dimension N and predicts the samples in between the temporally downsampled data along each input channel, i.e., the samples for times $2\Delta s$ to $(\alpha-1)\Delta s$ for each channel.

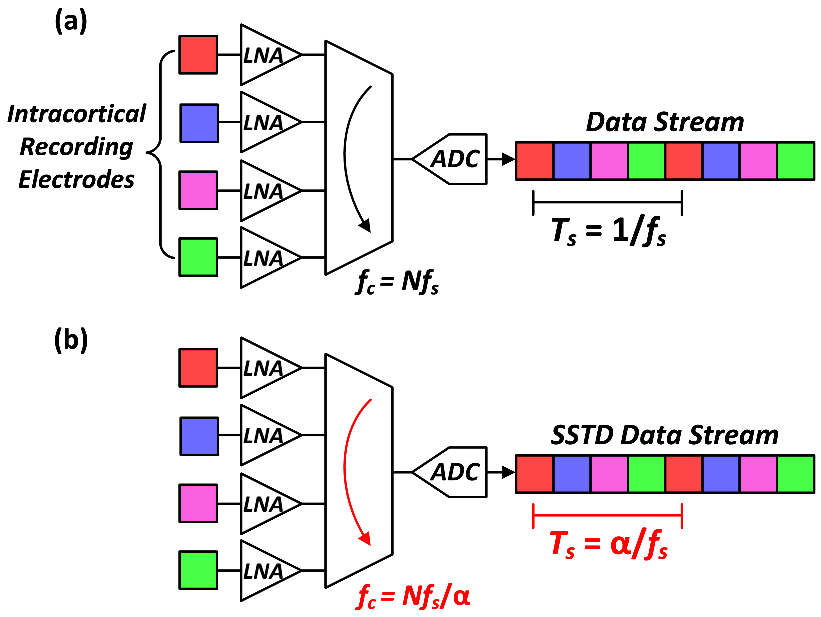


Figure 5.6. The analog front-end configuration for realizing staggered spatio-temporal downsampling (SSTD).

Figs. 5.7(a) and (b) shows the mean R2 scores for the recording session I of Dataset I and the two recordings of Dataset II. It can be seen that both the R2 performance and SNDR are data dependent. Further, the selection of α depends on the dataset for achieving a particular range of performance metrics. It should be noted, however, that the variation of performance metrics with respect to α is relatively minor, and the dataset itself has a larger impact on the performance variation. Since the performance of the ML-based algorithms is generally data dependent, employing the SSTD configuration introduces a larger statistical variation on Dataset I compared to that of Dataset II, and the change is relatively linear with respect to the folding factor. We chose to employ a folding factor of $\alpha = 12$, which reduces the memory requirement of the in vivo encoder from 776 to only 72 parameters, a 90% reduction. Lowering the input dimension of the network by a folding factor $\alpha = 12$ also reduces the computational complexity of the model. The original input consists of a multiplication of an input vector $\mathbf{x} \in \mathbb{R}^{1\times N}$ and a weight matrix $\mathbf{W} \in \mathbb{R}^{N\times F}$, requiring $F \times N$ multiplications and $F \times (N-1)$ additions, while lowering the input dimension by a factor of 12, the total number of multiplications and additions will be dropped to $F \times N/12$ and $F \times (N/12-1)$, respectively.

Another opportunity for optimizing the in vivo encoder is the realization of the Tanh activation function, defined as:

$$y = \frac{e^{-z} - e^{z}}{e^{-z} + e^{z}}, \tag{5.1}$$

where z denotes the accumulated weighted input to the activation function. To avoid implementing the exponential and division operators directly, the piecewise linear approximation is commonly employed [85]. An alternative approach that does not require linear approximation parameters is employing the HardTanh function, defined as:

$$y = \begin{cases} -1 & \text{if } z \le -1, \\ z & \text{if } -1 < z < 1, \\ 1 & \text{if } z \ge 1. \end{cases} \tag{5.2}$$

, where z denotes the weighted input to the activation function, and requires only two comparators to determine whether the output of the function should saturate at +1 or -1.

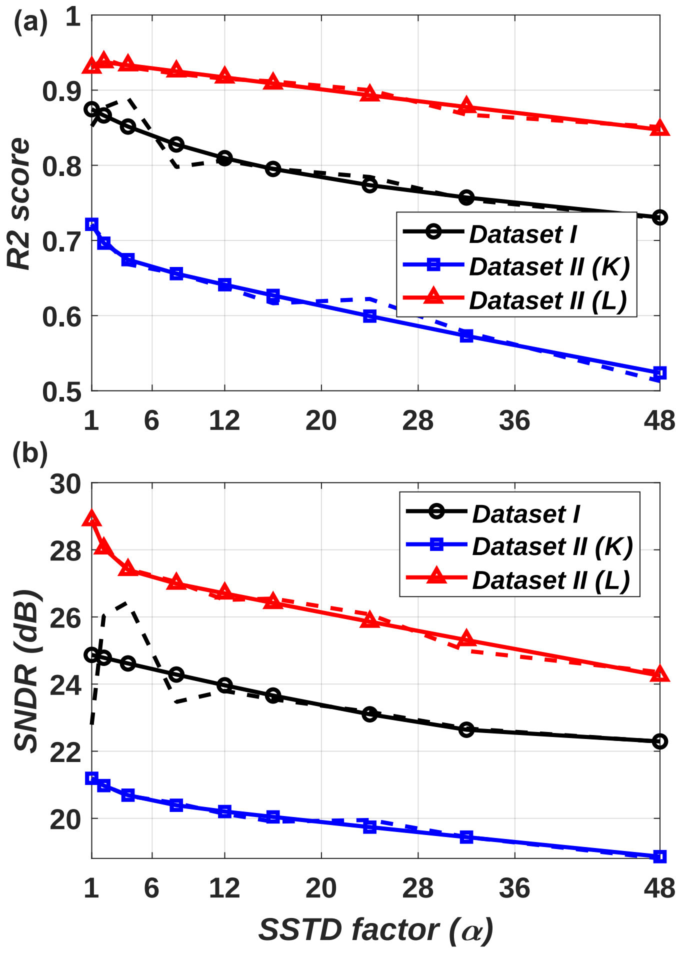


Figure 5.7. The variations of (a) R2 scores and (b) SNDR for the first recording of Dataset I, the Monkey K recording of Dataset II, and the Monkey L recording of Dataset II, over various SSTD factors. The dashed lines show the metrics for raw data, while the marked-up lines depict the trend.

Another consideration for efficient in vivo hardware realization is that the encoder network parameters and activations are represented in the fixed-point format. The baseline model is first trained employing the SSTD configuration and the HardTanh activation function. After an initial training, the encoder and decoder networks are split into two separate networks. During the initial training, the encoder network's parameters are constrained to [-1, 1) and hence, are represented in the fixed-point format Q(1.WF_K), where 1 and WF_K denote the number of integer and fraction bits, respectively. The output of the HardTanh function is quantized into Q(2.WF_A) format. The decoder network is then re-trained to account for the slight variations introduced by the encoder quantization. Fig. 5.8 shows the variations in the R2 score and SNDR over values of WF_K and WF_A for Datasets I and both recordings of Dataset II. It can be seen that the HardTanh output resolution WF_A has a stronger impact on the performance variation compared to that of the encoder paramter resolution WF_K. It was found that WF_K=4 and WF_A=8 results in a 6.9% variation in the median of the R2 score values for Dataset I, a negligible 0.03% variation for the Monkey K recording of Dataset II, and a 0.4% variation for the Monkey L recording of Dataset II compared to the SSTD model with $\alpha = 12$. Thus, the chosen numerical quantization have a negligible impact on the reconstruction quality of the model. It's worth noting that, as shown in Fig. 5.8, smaller values for WF_A and WF_K result in a relatively lower performance variation, but these values may be viable when primarily focusing on a single dataset and might be unviable for a generalized realization.

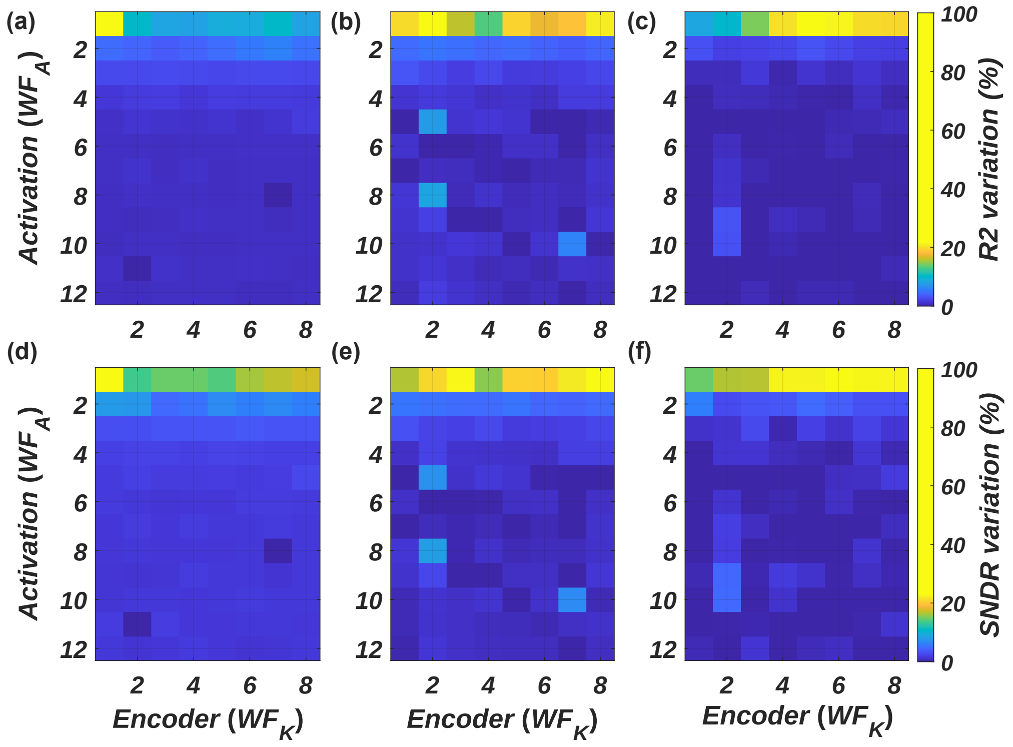


Figure 5.8. The variations of the R2 score for (a) Dataset I, (b) Dataset II (K), and (c) Dataset II (L), along with the variations of the SNDR for (d) Dataset I, (e) Dataset II (K), and (f) Dataset II (L) over different encoder resolutions WF_K and HardTanh output resolutions WF_A.

Figs. 5.9(a) and (b) shows the mean of the R2 score and SNDR values, respectively, of the original model, the re-trained model, and the calibrated model, over the 30 recordings of Dataset I with the encoder quantization and SSTD. The performance of the SSTD configuration without quantization is also shown. It is shown that the calibration with 10% of the training data achieves a performance comparable to that of the re-trained model as well as that of the model without employing quantization. One can see that the employed in vivo optimizations have a negligible impact on the performance degradation of the model.

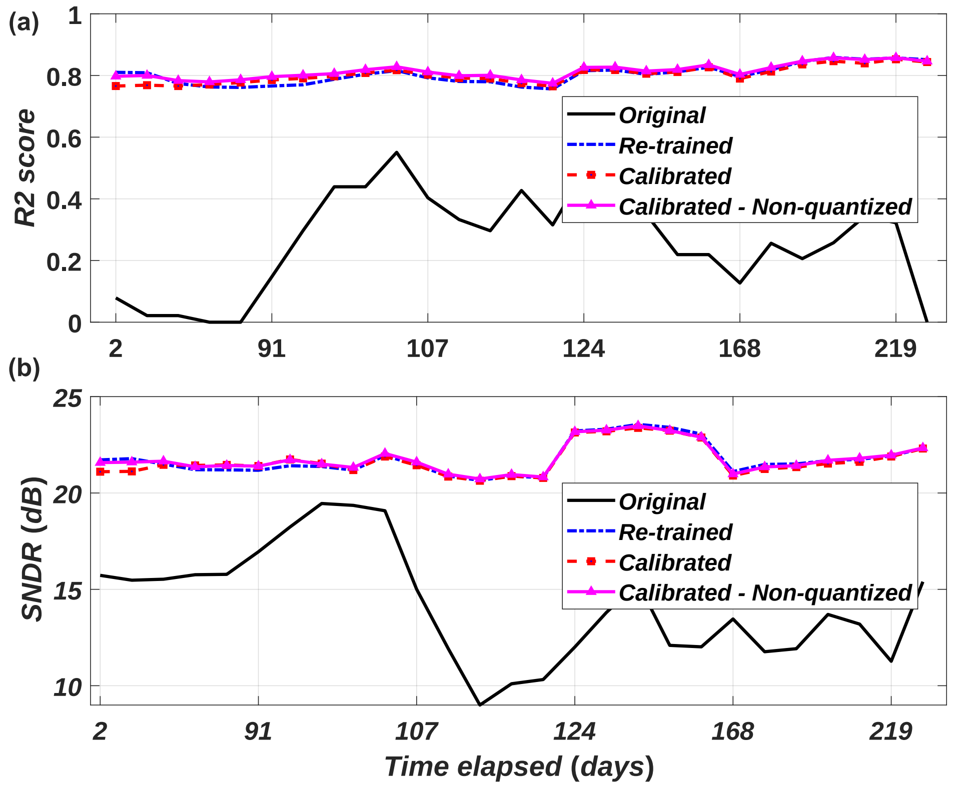


Figure 5.9. The mean (a) R2 scores and (b) SNDR of the original, re-trained, and calibrated models using encoder quantization and SSTD over all recordings of Dataset I, respectively. The performance of the calibrated SSTD configuration without quantization is also shown.

5.5 Hardware Architecture and Implementation Results

Fig. 5.10 shows the top-level block diagram of the designed and implemented in vivo encoder network. It consists of an array of normalization units, multipliers, adder trees, and HardTanh activation function units. The normalization units are used to normalize the input data based on the training set. The number of normalization units *D* defines the number of input channels to process simultaneously and is equal to the number of electrode channels divided by the folding factor α. Each input channel requires a set of normalization parameters *min* and *range* prior to applying the DL_E weights, where *min* and *range* denote the channel's minimum and the range of amplitude values stored in the normalization parameter memory *nRAM*, respectively. The model parameters are stored in the weight parameter memory *wRAM*. Each bank of weight multipliers and its associated adder tree computes the accumulation of the weighted inputs. The weighted sums are biased with the adders and then the HardTanh activation function is applied.

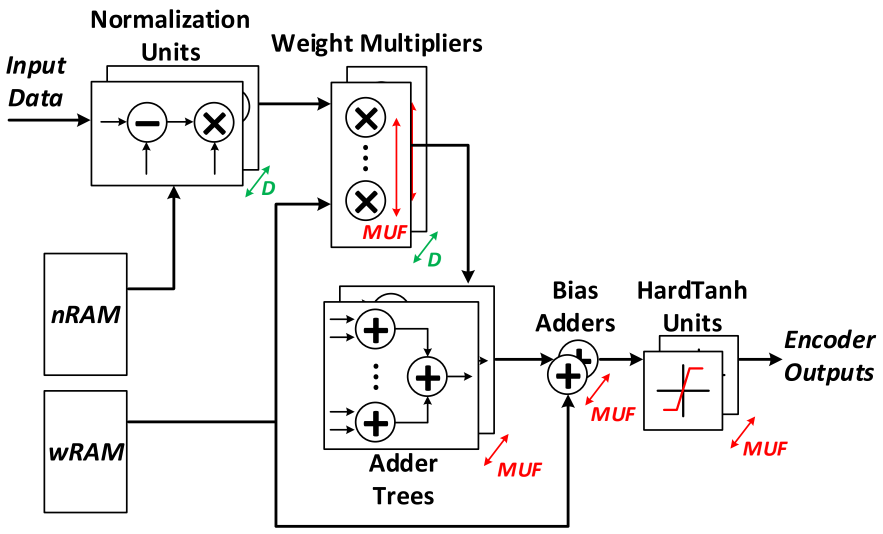


Figure 5.10. The top-level block diagram of the designed encoder network.

The multiplier unfolding factor parameter *MUF* denotes the number of weighted sums computed per clock cycle. For example, with $MUF = 1$, it would take eight clock cycles to compute the eight outputs of the in vivo encoder. For a throughput equal to the sampling rate f_s, a clock frequency equal to $8f_s$ is required. To find an optimal value for *MUF*, the encoder network is synthesized with α=12 for the *MUF* values of 1, 2, 4, and 8. The logic synthesis was performed using Synopsys Design Compiler and the place and route was performed with Cadence Innovus in a standard 180-nm CMOS process. To estimate the power consumption of the design, the post-routed netlist was simulated using Synopsys Verilog Compiler Simulator (VCS) by applying a testing subset of the Dataset I to the post-routed netlist. Table 5.1 gives the dynamic power consumption and area utilization of the designed in vivo encoder over various values of MUF. Reducing the operating frequency to f_s would naturally reduce the dynamic power consumption, however, it was found that $\mathit{MUF} = 4$ consumes the least power due to requiring half the number of adder trees and hardware of those for $\mathit{MUF} = 8$. The ASIC layout of the synthesized in vivo encoder with $\mathit{MUF} = 4$ is shown in Fig. 10.10 and is estimated to occupy 1.94 mm^2 of silicon area in a standard 180-nm CMOS process.

Table 5.1. The ASIC characteristics of the designed and synthesized in vivo encoder over various values of MUF

MUF	Freq. (kHz)	Area (mm^2)	Power (μW)
1	16	1.25	10
2	8	1.37	7.6
4	4	1.94	7.35
8	2	2.11	10.4

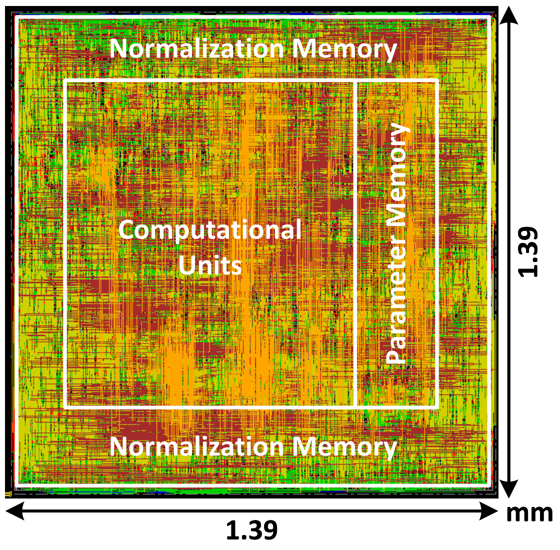


Figure 5.11. The 1.94 mm^2 ASIC layout of the synthesized in vivo encoder in a standard 180-nm CMOS process.

Table 5.2 gives the ASIC characteristics and implementation results of various previously published intra-cortical neural signal compression designs. Both lossless and lossy compression modalities have been reported. Since the main focus is on the efficient design and implementation

Table 5.2. The characteristics and implementation results of in vivo neural signal compression ASICs

Work	Algorithm	Tech. (nm)	Supply (V)	Area/ch. (mm^2)†	Power/ch. (μW)†	CR	SNDR (dB)
[127]	DRR, Huffman coding (Lossless)	180	1.8	0.039	3.57	4.58	–
[128]	DRR (Lossless)	130	1.2	0.008	15.8	2	–
[135]	CS (Lossy)	180	1.2	0.008	3.55	4	$\simeq 14$
[158]	CS (Lossy)	130	1.2	0.087	31.003	10	–
[159]	CS (Lossy)	180	–	–	4.8	8	9.78
Ours	AE (Lossy)	180	1.8	0.002	0.076	19.2	Dataset I: 15 ± 3 Dataset II: 19 ± 3

† Normalized to a 180-nm CMOS process with a 1.8 V supply as described in [47], accounting only for the in vivo digital compression circuitry.

of the compression circuits, the ASIC characteristics and implementation results provided in Table II exclusively take into account the silicon area and power consumption of the in vivo compression circuits to ensure a fair comparison. While there are several techniques for lossless compression, the most commonly employed methods for compressing neural signals involves the dynamic range reduction (DRR) and variable wordlength encoding. In [127], the spatial and temporal correlation of neural signals were exploited to reduce the dynamic range of LFPs. First, a temporal difference was applied to the signal followed by removing the spatial average of groups of 16 recording channels. Then, Huffman encoding was applied to compress the LFPs to 2 or 3 bits prior to transmission. With the input LFPs represented using 11 bits, this resulted in an average compression rate of 4.58. The reported design compressed temporally-sparse spike waveforms by performing spike detection and transmitting only spike events rather than the continuous waveforms. For a fair comparison, our design is compared with the digital LFP compression hardware, which was reported to consume 3.59 μW of power per channel from a 1.8 V supply operating at 2 kHz. A similar lossless compression scheme was reported in [128] where the dynamic range of the signals was reduced by applying common-average referencing and subtracting this baseline from all channels prior to transmission. However, rather than encoding the signals, the baseline was transmitted for in silico reconstruction, resulting in an average compression rate of 2. The compression circuit was estimated to consume 6.4 μW of power per channel from a 1.2 V supply when operating at 400 kHz. In [135], an analog-based realization of a CS-based compression scheme was reported. The sensing matrix consists of samples uniformly selected from the Bernoulli distribution and achieves a compression rate of up to 16. The design was implemented in a 180-nm technology operating at 4 kHz and consuming 0.95 μW of power from a 1.2 V supply. In [158], another CS-based compression ASIC was presented, employing a novel Manhattan distance cluster-based sensing matrix for compressing multi-channel neural signals, achieving the compression rate of 10. The design was implemented in a 130-nm CMOS process and the simulated power consumption was 12.5 μW per channel from a 1.2 supply voltage. The CS-based compression system-on-chip reported in [159] was estimated to consume 4.8 μW per channel and to achieve a compression rate of 8.

As given in Table 5.2, the designed and implemented autoencoder-based compression scheme consumes the lowest power per channel, while achieving a significantly higher compression rate than the previously reported designs. Compared to the compressed sensing methods for spatial dimensionality reduction, the designed and implemented autoencoder-based compression scheme offers reduction in power due to the spatio-temporal downsampling that drastically reduces the complexity of the encoder network. Additionally, the units in the encoder network learn to share parameters among different input channels, which has a small impact on its reconstruction quality, as was shown in Section 5.4. For practical BCI applications in which subsequent decoding and processing may tolerate reconstruction and wireless transmission errors reasonably well, the lossy methods with a greater compression ratio offer a more viable approach. Nevertheless, even though the lossless methods do not achieve relatively high compression ratios, their true signal reconstruction capability is a valuable attribute for carefully analyzing neural signals in silico.

Table 5.3 gives the power consumption of various neural signal processing ASICs employing alternative signal modalities. In [113] MUAs were obtained by performing spike detection. In [102] the data rate was further reduced by obtaining SUAs using spike detection and in vivo spike sorting. In [140] SUAs were obtained by compressing detected spike waveforms using an in vivo AE. Waveform reconstruction and spike sorting were performed in silico. Evidently, employing more in vivo neural signal processing would consume more power per recording channel. The analog front end (AFE) of the neural recording circuitry, which consists of low-noise amplifiers and analog-to-digital converters, typically consume an average of 7.4 μW of power per channel [160]. The power consumption of the LNAs depend on various factors, such as the acceptable input-referred noise and whether chopper-stabilization is required to mitigate dominant flicker noise [161]. Additionally, the power consumption of the amplifier is also proportional to the width of the frequency band of interest [131], which for LFPs is about five times smaller than that of spikes. Conventional AFEs often employ successive approximation register (SAR) ADCs, whose power consumption is relatively linearly proportional to the sampling rate f_s [162, 163]. Therefore, it is reasonable to assume that reducing both the amplifier bandwidth and ADC sampling rate by a factor of five would yield a similar reduction in power consumption, approximately $7.4/5 = 1.5$ μW per channel. Therefore, the compression of LFPs would provide a more scalable recording scheme for BCIs employing high-density MEAs.

Table 5.3. The power consumption of the in vivo neural signal processing ASICs employing MUA, SUA, and LFP neural signals

Work	**Neural signal**	**Method**	**Power/ch. (μW)[†] in vivo digital signal processing**	**Power/ch. (μW) AFE**
[113]	MUAs	Spike detection	0.64	7.4
[102]	SUAs	Spike sorting	2.02	
[140]			4.09	
Ours	LFPs	AE-based compression	0.076	1.4[‡]

† Normalized to a 180-nm CMOS process with a 1.8 V supply.
‡ Assuming the reduced sample rate and bandwidth requirements for LFPs over spike-based recordings [131], accounting only for the in vivo digital signal processing circuitry.

Intracortical BCIs must operate within strict low-power constraints to prevent heating of the brain tissues beyond 1° C, as outlined by the Food and Drug Administration. A maximum power budget of 7.7 mW has been reported for neural implants with wireless power transfer, based on 15-mm receiving antennas implanted on the cortical surface of the brain with an RF operating frequency of 2.02 GHz. Further analyses of different antenna sizes at various brain depths and their maximum power budgets are discussed in [126]. Fig. 5.12(a) shows the total power consumption for detecting MUAs and compressing LFPs with the designed AE-based digital circuit (including that of the AFE and wireless transmission), and the implant's power budget. Given a wireless transmission power of 158 pJ/bit [15], it becomes evident that the proposed compression would enable simultaneous recording from over 3000 channels, a significant increased compared to spike-based counterparts. Fig. 5.12(b) shows the supported data rates for transmitting MUAs, LFPs, and compressed LFPs over various number of channels. Considering the highest Bluetooth data rate of 2 Mbps, it can be seen that the transmission of raw LFPs is only feasible for fewer than 100 recording channels. The proposed AE-based compression scheme would make it possible to significantly increase the number of recording channels, approaching that of MUAs. Accounting for both the 7.7 mW implant's power budget and also the maximum data rate of 2 Mbps, the maximum number of channels to transmit raw LFPs is only 62, whereas that for MUAs with one millisecond bins is 878. By employing the designed AE-based compression, however, the maximum number of recording channels can be increased to 1152, an 18 times improvement over that for raw LFPs and a 1.3 times improvement over that for MUAs. In the above comparison, we assumed the conventional independent processing of multi-channel signals. To enhanced the overall density of intracortical electrodes, modern realizations of recording ICs decrease the electrode pitch by suppressing the sensitivity around the recording electrodes to avoid multiple detections of the same action potential. This isolation effectively reduces the overall processing of the neuronal data and in vivo power consumption. [148, 164].

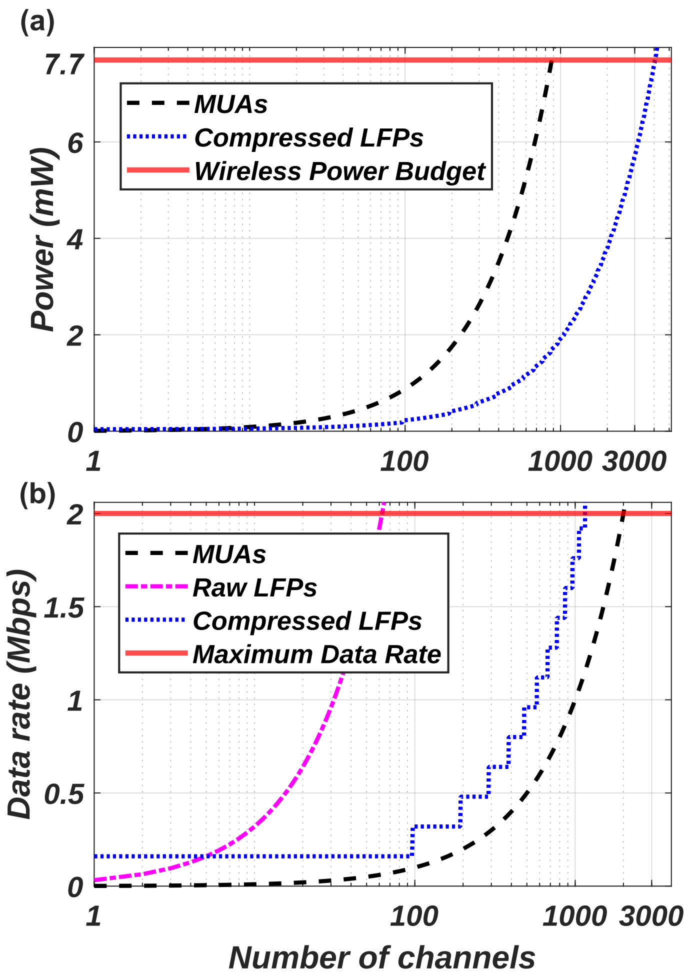


Figure 5.12. (a) The total power consumption for detecting MUAs and compressed LFPs (including that of the AFE and wireless transmission) and (b) the supported data rate for MUAs, raw LFPs, and compressed LFPs over various number of channels.

5.6 Conclusion

This chapter presents the design and implementation of an autoencoder-based compression scheme for in vivo compression of neural signals. Various optimization schemes were employed for the efficient hardware realization of the designed circuits. Using two widely-employed neural datasets, we discussed and composed the reconstruction performance of the designed compression scheme against other state-of-the-art designs. We presented an application-specific integrated circuit of the designed in vivo encoder in a standard 180-nm CMOS process, estimated to occupy 0.02 mm^2 per channel and consume 0.076 μW of power per channel from a

1.8 V supply. Compared to the recently reported compression integrated circuits, the designed and synthesized compression architecture occupies the least silicon area, consumes the least power, offers the highest compression rate of over 19 times, and achieves a mean reconstruction quality of 17 dB over two datasets.

5.7 Publications

The following article was published as an outcome of the research conducted in this chapter: D. Valencia, P. P. Mercier, and A. Alimohammad, "Efficient In Vivo Neural Signal Compression Using an Autoencoder-based Neural Network," IEEE Transactions on Biomedical Circuits and Systems, vol. 18, pp. 691 – 701, January 2024.

Chapter 6

Towards In Vivo Neural Decoding

6.1 Introduction

Patients suffering from various neurological disorders and amputations may lose their ability to control some of their normal bodily functions or to take care of themselves entirely. A brain-machine interface (BMI) creates a link between the brain's neural activity and the control of an external assistive device by bypassing dysfunctional neural pathways. Invasive BMIs record neural signals directly from the motor cortex using brain-implanted electrodes. By understanding how brain activity relates to behavior, the encoded motor intentions in the recorded neural signals can potentially be decoded into meaningful control commands to restore a patients' ability to interact with their surroundings, via improved communication or actuating an assistive device such as computers and prostheses.

Conventional acquisition systems employ multi-electrode arrays (MEAs), which allow simultaneous recording of hundreds of channels. The wireless transmission of a large amount of recorded data can impose high power consumption and long delays, which restricts real-time and in vivo operation of BMIs. For example, the data rate for a Utah Array [165] with 100 recording channels sampled at 20 kHz with a 10-bit resolution is 20 Mbps. For this transmission rate, an implantable chip requires several hours of continuous operation and a few hours for recharging the battery [166]. The commonly employed acquisition systems are unable to precisely place electrodes to individually record from a single neuron. Hence, each electrode records the neuronal activities of a region, where generally tens of neurons are present and thus, providing the acquisition of multi-unit activity (MUA). Moreover, the fixed geometry of the conventional MEAs, such as the Utah Array, constrains the populations of neurons that can be accessed. They are made from rigid metals or semiconductors, which can limit their application and longevity.

It has been shown that complex brain processes are reflected by the activity of large neural populations and that the study of a few neurons provide relatively limited information. Therefore, advances in BMI systems rely on the ability to record from large populations of neurons in the order of thousands. Recent high-density micro-electrode array (HD-MEA) technology developed by Neuralink [8] use flexible polymer probes to provide greater bio-compatibility and longevity, as well as an unprecedented high-bandwidth and spatio-temporal resolution for long-term intracellular neural recording. An array of up to 3,072 electrodes can be implanted individually by a neurosurgical robot offering single-neuron (single unit) resolution. Neural recording MEAs with neuron-level precision open a fascinating opportunity to investigate how large cell populations encode sensory behavior and motor function. Biological neurons communicate by means of firing electric pulses, called action potentials (APs) or spikes. Neural activity is usually sparse, and since the firing rates of neurons, which is the number of spikes emitted by a neuron over a given time interval, rarely exceeds 10 Hz, the data transmission rate will be reduced drastically by detecting spikes on an implanted chip and only transmitting the spiking information. For example, employing a MEA with 100 channels sampling at 20 kHz with a resolution of 10 bits per sample, would require 20 Mbps for transmitting raw signals. If only spiking activity is transmitted, the data transmission rate can be reduced to only 2 to 3 kbps, assuming that each electrode records the activity of two to three neurons firing at an average rate of 10 Hz.

Researchers often require single-neuron activity for the study of how neurons are correlated with each other for specific stimulus or for decoding motor intentions. Since in a conventional extracellular recording an implanted electrode can record the neural activity of neurons from a distance of up to about 140 micrometers [32], the neural signal recorded by an electrode consists of the cumulative neural activities of multiple adjacent neurons and the neural activity of relatively far away neurons contaminated by noise and technical artifacts, such as electrode drift, tissue-electrode noise, and electronics noise. Neurons in a local area often fire spikes of similar shape and amplitude, however, relative to the distance of an electrode to a neuron, the shape of spike waveforms will differ for various neurons [167]. This key fact allows the spiking activity of individual neurons to be distinguished and separated through a process, called spike sorting [138], which may assist neural decoding for certain applications. Spike sorting is the process of grouping detected spikes into clusters based on the similarity of their shapes. The resulting clusters thus correspond to the spiking activity of different putative neurons.

Transmitting detected spike waveform shapes, assuming a spike waveform is represented by only 48 samples and that the recording channel detects two to three neurons firing at a rate of 10 Hz, would require data rates between 960 Kbps and 1.44 Mbps. Increases in channel counts, spiking frequency, or the number of neurons in the vicinity of the electrode can raise this data rate dramatically. An alternative approach is to perform neural signal processing in vivo and only transmitting decoded neural activity directly to an external assistive device. Neural decoding aims to translate the neural activity of the brain into quantifiable commands for controlling or communicating with an external device. When decoding neural activity into a discrete set of commands, the output data rate of decoding can be given as $R_d = \frac{\lceil \log_2(N_c) \rceil}{\Delta t N_\Delta}$, where $\lceil \cdot \rceil$ denotes the ceiling operator, N_c denotes the number of discrete commands, Δt denotes the time bin size over which spikes are counted, and N_Δ denotes the number of time bins collected prior to performing decoding. For example, simple cursor control can be decomposed into a discrete set of movements $\{$left, right, up, down, none$\}$. If detected spikes are binned every 10 ms and 10 bins are collected prior to decoding, the output data rate is $R_d = 30$ bps. Controlling a prosthetic arm with multiple degrees of freedom (DOF) increases the data rate due to the numerical resolution of kinematic variables. The output data rate can be given as $R_d = \frac{V_b N_v}{\Delta t N_\Delta}$, where V_b denotes the number of variables and N_v denotes the total number of variables. For example, consider a robotic arm with seven DOF (i.e., α, β, and γ endpoint; x, y, and z angles; and grip velocity) with 8 bits of resolution per variable. If spikes are binned every 10 ms and 10 bins are collected prior to decoding, the output data rate is $R_d = 560$ bps. Note that this is considerably less than that for transmitting raw data, spike events only, or spike waveform outputs [64]. Moreover, in vivo decoding decreases the time required to transmit data and the offline processing time. Minimizing the processing latency between biological signal generation and intended motor function is important. It has been shown that in non-human primate trials, a maximum latency of about 150 ms between signal generation and the robotic arm movement results in natural-looking and smooth motion [168].

In this chapter, we discuss the feasibility of realizing in vivo neural decoding with and without spike sorting. More specifically, in Section 6.2, we first discuss how much information is contained in the two alternative spike train analysis methods. Section 6.3 discusses the impact of spike sorting on decoding performance. The design and training of a temporal convolutional neural network decoder is presented. Section 6.4 presents the design and implementation of a programmable neural network-based decoding processor. Various challenges for realizing in vivo neural decoding are discussed. Section 6.5 makes some concluding remarks.

6.2 Estimating Information in Neural Spike Trains

Cortical neurons use spike trains to communicate with other neurons. The output spike train of each neuron is a stochastic function of its input signals from other neurons. Understanding what information is encoded in its output spike train and what information is discarded, as well as quantifying how much information a single neuron transfers (or loses) from the input it receives (i.e., a sensory stimuli) to the activity of neuron (i.e., the neural code), are important for efficient decoding of the brain activity.

It is generally assumed that all the information contained in the spike train can be represented by the spike times. The simplest representation of a neuron's response can be given by its firing rate. In addition to the firing rate of neurons, the relative timing between spikes also encodes information, which may be useful for decoding. Rate coding and temporal coding are two techniques for quantifying the information encoded in a neural spike train [169, 170]. Rate coding implies that the firing rate of a neuron represents the information encoded in the spike train, while temporal coding implies that the information is not only based on the firing rate, but also the relative timing of spikes. Rate coding can be divided into two categories: neuron rate coding and population rate coding. Population or ensemble rate coding is a measure of the firing rate of an ensemble of neurons rather than individual neurons.

A visual representation of the two rate coding methods over a two second snippet of a neural spike train is shown in Figs. 6.1(a) and (b). Fig. 6.1(a) shows the grey boxes horizontally,

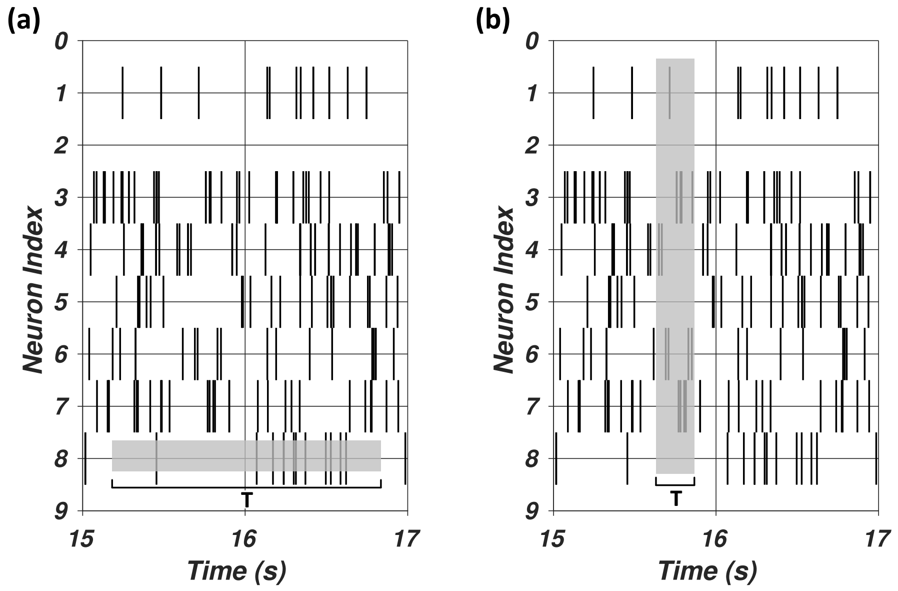


Figure 6.1. (a) Neuron rate coding and (b) ensemble rate coding.

which represents the firing rate over the time interval T as a temporal average. Fig. 6.1(b) shows the grey boxes vertically, representing the firing rate as a spatial average over all neurons over a time window T. Neuron firing rates are computed by counting the number of spike events occurring during a time window T, while ensemble firing rates are computed by counting the number of spike events occurring across an ensemble of neurons during a time window T. Both measures are then normalized by T, which results in units of Hz. It has been shown that the ensemble rate coding more closely resembles the brain's natural coding scheme, by which ensembles of neurons communicate their accumulated activity with other neural populations [1]. This can be considered as a spatial average of the neural activity, whereas the neuron firing rates represent a temporal average over a longer time.

The entropy $S = -\sum_i \left[P_i \log_2 P_i\right]$, where P_i denotes the probability of a particular pattern of spike events, such as firing rate, denotes the average amount of "information" or "uncertainty" inherent to the random variable. Random variables with a more uniform distribution have a higher entropy, while distributions with a more localized peak have lower entropy. In other words, if a probability distribution is more biased toward specific outcomes, then there is less information encoded in the random variable compared to the one with higher entropy [171]. A coding scheme with a higher entropy can be interpreted as having a more uniform distribution and hence, a greater amount of information regarding the stimulus that caused the neural response.

Table 6.1. The entropy rates for different neurons

Neuron index	Entropy rate (bits/sec)	Neuron index	Entropy rate (bits/sec)
1	18.17	10	17.12
2	10.97	11	27.01
3	45.41	12	42.98
4	44.92	13	41.01
5	33.32	14	35.41
6	30.96	15	42.83
7	35.51	16	43.68
8	34.69	Ensemble Rate	38.25
9	38.94		

For a spike train, we employ an estimation of entropy, in which the random variable is represented by the spike counts over small bins of $\Delta\tau$ seconds over time windows of size T [172]. An intuitive interpretation of spike train entropy is that if a spike train is completely deterministic, it will have low variability in the time between spike events and by implication, its firing rates. This implies that a deterministic neuron encodes less information than a randomly firing neuron. To compare the entropy between the two rate coding schemes, we utilize the publicly available hc-2 dataset from the collaborative research on computational neuroscience [60]. The dataset consists of neural recordings from the CA1 region of the hippocampus of freely moving rodents over various experiments. The rodents were provided with water or food as a reward at random locations throughout a platform. For the recording session ec013.527, the entropy of the spike train and the calculated ensemble rate are given in Table 6.1. One can see that some neurons have low entropy, while others have relatively high entropy compared to the ensemble rate. The ensemble rate has a relatively high entropy and is not far from the mean entropy of all the neurons, (i.e., 33.93 bits/sec). Therefore, it can be assumed that using single unit firing rates would yield more information for decoding. In order to test this hypothesis, we divide the data into 125 ms intervals, which is equivalent to 5 samples of video frames to align the rodent positions to the physiology. A feed-forward neural network (FNN)-based decoder is trained to map firing rates (rate and ensemble coding) to rodent positions. The testing dataset contains 7797 samples, 80% of which are utilized for training, while the remaining 20% are used for validation and testing. Each sample in the dataset consists of binned neural data over 1.28 seconds with bins of size 25.6 ms, i.e., 50 time steps. The employed FNN has three hidden layers with tanh activation functions. The input dimension for the neuron-rate coding was 16, while the input dimension for the population-rate coding was one. The FNN was trained using the TensorFlow machine learning framework for Python for up to 1000 epochs, using early stopping to avoid overfitting to training data. The Adam optimizer with default parameters $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1\mathrm{e}-07$ was used for training. The input to the neural network was scaled (z-scored) as $x_z = (x - \tilde{x})/(x_\sigma)$, where x denotes the training data and $\tilde{x}$ and x_σ denote the mean and standard deviation of the training data, respectively, to have a zero-mean and unit variance distribution, which is useful for training. Additional statistical enhancements can be made by applying temporal smoothing of the input spike firing rates. One of the commonly employed methods is the Gaussian smoothing kernel [173], which convolves the input spike counts with a Gaussian curve of a pre-defined width. Applying Gaussian smoothing makes the distribution of the input data resemble a normal distribution more closely, which makes it easier to apply machine-learning algorithms to the data. The training dataset denotes the subset of the data used for optimizing the parameters of the network, while the validation dataset is used to assess the performance of the model during training. The testing dataset is used to evaluate the performance of the network, as it contains examples not observed during training.

Fig. 6.2 shows the training and validation losses for the neuron and ensemble rate coding schemes. The testing loss for the neuron rate coding was 42.8 while the population rate loss was 43.29. The y-axis is plotted on a log scale to view the differences between the two rate coding schemes more easily. One can see that the neuron rate coding outperforms the ensemble rate decoding slightly due to the higher entropy for certain neurons and also due to the higher

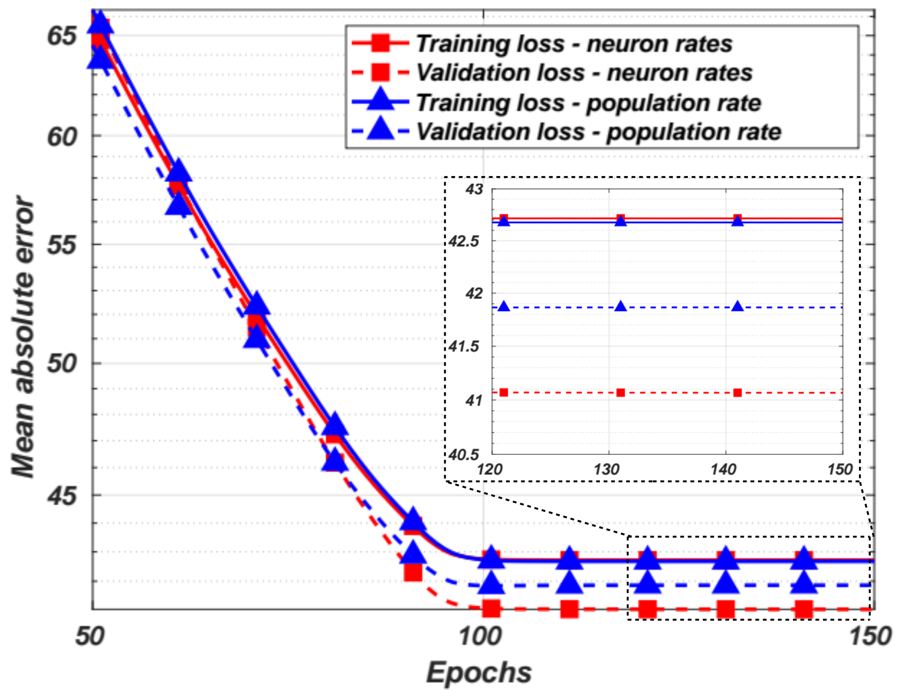


Figure 6.2. The training and validation loss for neuron rate coding and ensemble rate coding over various training epochs.

dimensional space to learn from compared to the single dimensional input of the ensemble rate coding scheme. Thus, we conclude that in the context of neural decoding, neuron-rate coding may outperform the population-rate coding.

6.3 Neural Decoding and Spike Sorting

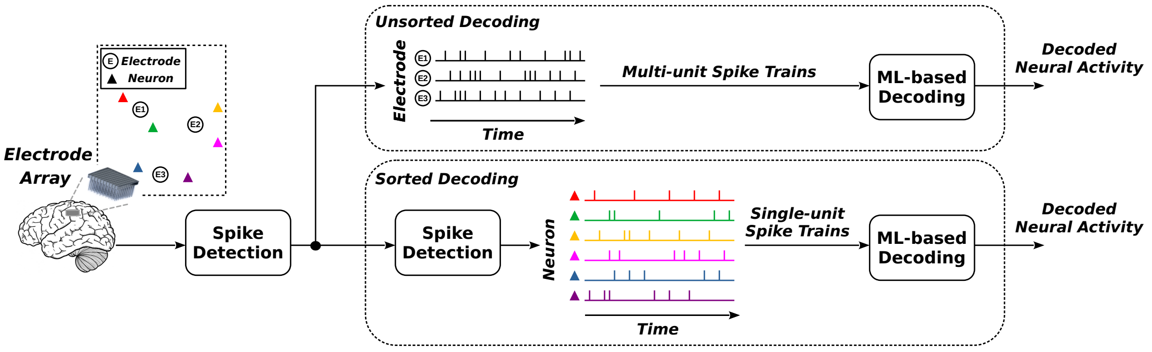


Figure 6.3. The in vivo neural decoding paradigms.

6.3.1 Investigating Sorted and Unsorted Decoding Paradigms

Spike sorting has been employed to extract the activity of individual single units influencing an electrode from the recorded MUA. Conventionally, spike sorting consists of several processing steps. First, spikes are detected from the background noise. Common detection techniques involve estimating the noise in the recorded and filtered signal and comparing it with a threshold. From detected spike waveforms, specific features are then extracted. These extracted features are then clustered into distinct groups, in which similar features and thus, similar spikes, are grouped together. A commonly employed clustering algorithm is *k*-means clustering [174], in which the feature vectors are clustered into *k* distinct groups, indicating up to *k* different single units found in the neural recording.

A desirable approach toward in vivo decoding could be avoiding spike sorting altogether, thus reducing the amount of in vivo computation and/or wireless data transmission. Without spike sorting, all threshold crossings of the voltage waveform on an electrode are treated as firing from one putative neuron. While this approach ignores the kinematic information provided by individual neurons recorded on the same electrode, this loss of information can potentially be avoided due to the significant enhancement in the recording technologies at the single neuron level [8]. We thus explore two possible decoding paradigms: (i) unsorted decoding and (ii) sorted decoding, as shown in Fig. 6.3. First, spikes are detected from the neural signals recorded by an implanted MEA. The unsorted decoding paradigm will consider detected spikes from all neurons recorded by each of the electrodes to create a multi-unit spike train (MUST). The sorted decoding paradigm employs spike sorting to cluster the activities of individual neurons detected by the spike detection unit and create a single-unit spike train (SUST). The raster plots in Fig. 6.3 are created based on the detected spikes from each electrode and sorted spikes, respectively, where the *x* and *y* axes represent time and the spiking activity of different neurons, respectively. The spike train in the unsorted decoding paradigm shows the spikes detected on each of the *N* recording electrodes. The spike train in the sorted decoding paradigm shows the sorted spiking activity of *M* single units. The MUST or SUST is then passed to the decoding unit, which can employ a variety of decoding algorithms. Traditionally, the decoding models of interest are FNNs and recurrent neural networks (RNNs). RNNs can encode temporal information in their hidden layers [175] while FNNs cannot due to the lack of temporal memory. We propose to use the temporal convolutional network (TCN) [176]. The TCN, shown in Fig. 6.4 (a), encodes temporal information in the input sequence with dilated convolutions at deeper layers and thus spanning the entire input sequence. The distinct network architecture of the TCN addresses many of the limitations of RNNs with learning temporal features, as the TCN does not propagate gradients over time, but rather through convolutional layers. The output of the TCN is connected to a layer to linearly map TCN outputs to the rodent positions. The convolutions are causal, which means that the TCN is appropriate for real-time neural decoding.

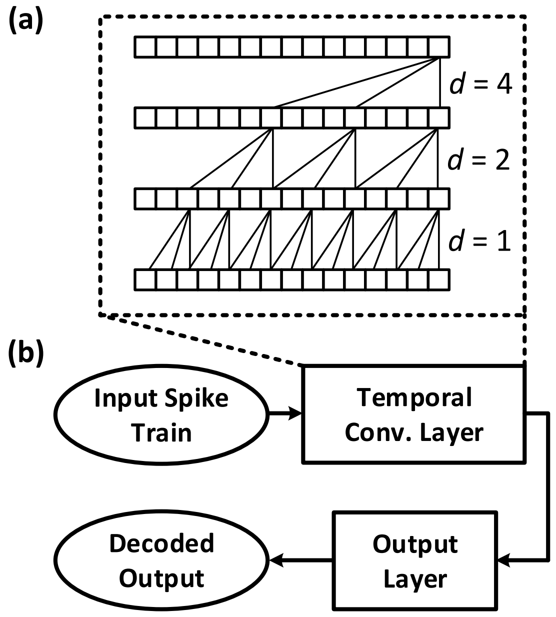


Figure 6.4. (a) The block diagram of a TCN layer and (b) the TCN-based decoder.

Accurate spike sorting to retrieve the individual neuronal activities from multi-unit activity involves computationally- and memory-intensive algorithms and is often performed offline. Moreover, the requirement to sort spikes becomes more imposing with increasing the number of implanted recording electrodes, especially for spike sorting in vivo. Interestingly, some earlier work reported that decoding the spike detector's outputs directly will make the system more robust for long-term use [177]. Also, several studies have suggested recently that spike sorting might not be critical for estimating intended movements in the motor BMI applications [178, 51]. In those ensemble decoding studies, the rate at which the filtered neural electrical signals crossed a pre-determined voltage threshold provided nearly an equivalent amount of information about the discrete direction of intended movements as did the spike rates of isolated, single neurons, irrespective of whether the threshold crossings detected on each electrode came from one or multiple neurons. Threshold crossing rates have successfully been used for real-time neural control in animals and humans, even years after the implantation of micro electrode arrays. Finally, the application of HD-MEA with neuron-level precision may eliminate the need for spike sorting altogether.

To study the effect of spike sorting on the performance of neural decoding, we utilize the publicly available hc-2 dataset [60]. The aim is to predict the location of the rodent based on the spiking activity of neurons in the hippocampus, which is the region of the brain responsible for cognitive processes related to spatial navigation and memory. For recording neural signals, four and eight channel tetrodes were used to improve spike sorting performance. The tip of each recording shank, which are the channels with the highest signal amplitude, were used for detecting spikes. A 32-sample waveform was created from each same-shank tetrode and hence, each spike is represented by an 8×32 dimensional matrix. Dimensionality reduction was performed with discrete derivatives [110] to reduce the dimensionality to $8 \times 3 = 24$ samples per waveform. *k*-means clustering was then performed to group feature vectors into 16 clusters, as done in [60].

Fig. 6.5(a) shows the mean absolute error of the training of the unsorted and sorted TCN-based decoding paradigms using the training and validation sets over 100 training epochs. An important observation is that the sorted decoding paradigm has the advantage of a relatively faster convergence. However, as shown in Fig. 6.5(b), both models converge to close R^2 scores for the validation set. Moreover, both paradigm perform well for the testing set, with the mean $R^2 = 0.9$

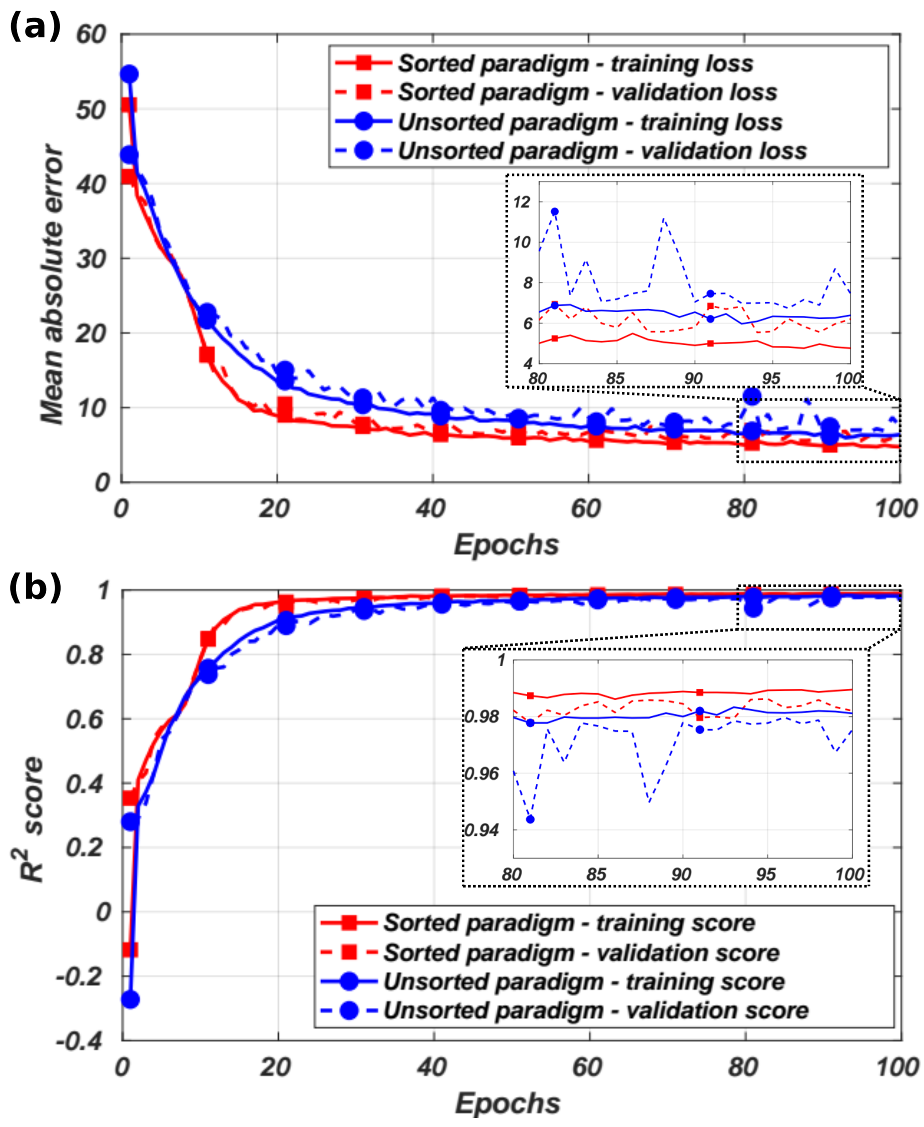


Figure 6.5. (a) The training performance and (b) the R^2 score of the TCN-based decoder for both decoding paradigms.

and $R^2 = 0.9$ for the sorted and unsorted paradigms, respectively. The unsorted paradigm reduces the dimension of the model's input from the number of neurons to the number of recording shanks. While increasing the input's dimension may increase the amount of information provided to the model, it has been shown that constraining the overall number of parameters in a model reduces overfitting [179]. We found that the TCN-based decoder offers robust decoding performance across a variety of spike detection algorithms, with the mean $R^2 = 0.97$ for both sorted and unsorted paradigms. Similarly, for this particular decoding task, it was found that a temporal sequence length of 75 time steps was optimal for time-series neural decoding with the mean $R^2 = 0.98$, compared to the feed-forward network with 50 time steps. Therefore, we employ the unsorted decoding paradigm for the real-time in vivo BMI realization.

6.3.2 Training and Calibration

The training and calibration time of the neural decoder must be reasonable for a satisfying user experience. For example, it has been shown that for motor decoding applications, individual neurons often encode a preferred direction or movement [168]. Additionally, for single unit recordings, the same single units may not be present in the recorded neural signals across different sessions over multiple days or weeks. The process of re-training, or calibrating, neural network-based algorithms involves re-training a pre-trained model on a subset of the original training data in addition to the new data [180]. A common problem in network calibration, so called catastrophic forgetting [181], is that the model forgets data observations from earlier training sessions. In addition to retaining some of the previous data used for training, specific layers of the model may fix their weight values, similar to a transfer learning approach [182]. While the size of the updated training dataset or the total number of weight updates may be significantly smaller than the first training session, the time it takes to train the model remains relevant. A decoder for neural-controlled robotic arms reported in [183] has a calibration phase performed as follows. First, the subject is asked to carefully observe the automated movement of the arm for a given task. This is due to the modulation of neural firing patterns given the observation of the intended motion. The spike trains recorded during this process are then used to train the decoder. The observation-based decoder is then calibrated by allowing the subject to control the robotic arm with some amount of automated movement aid. The aid is intended to reinforce the subject to modulate their neural firing patterns that best correspond to the desired movement. Over several training sessions, the amount of automated movement aid is reduced and the subject learns to control the robotic arm on their own. The TCN-based decoder, for example, is better suited for real-time decoding and calibration due to the fact that there is no internal hidden state to compute over time. This allows TCNs to be trained significantly faster than RNN-based models of similar size. In our tests on a Desktop PC with an 8-core processor and 16 GB of RAM, we found that the TCN-based decoder took 25 minutes to train, while RNN models took over three hours to train. The training and calibration process could take longer if they are performed on a mobile device, which has considerably more processing limitations compared to a high-performance computer.

6.4 Towards an Implantable Decoding Processor

While relatively small decoding architectures and algorithms can be employed for high levels of decoding performance, conventional BMIs still require an external device for processing neural signals. This imposes some major limitations: (i) for wireless BMIs, the amount and speed of data transfer are limited by the bandwidth of the communication medium; (ii) for wired BMIs, cables may limit the patient's mobility and cause a poor user experience; (iii) the requirement of an external computing device means that the user must rely on additional devices for proper decoding operation. These limitations motivate the design and implementation of an in vivo machine learning-based decoding processor.

Fig. 6.6 (a) shows the conventional approach for wireless BMIs. Neural signals are recorded and processed by a spike count unit, which detects spikes from ambient noise and stores the spike count over a predefined amount of time before transmitting it to an external computing device. For example, the NeuraLink BMI transmits spike counts to a mobile device or a computer in 25 millisecond time bins for cursor control tasks [184]. Fig. 6.6 (b) shows the proposed approach for a fully-implantable wireless BMI. An in vivo processor would be able to directly process input spike trains and perform a variety of decoding tasks, such as controlling a prosthetic limb. An implantable processor is first programmed using a computer or mobile device. The processor is then able to interact directly with the external assistive device by performing

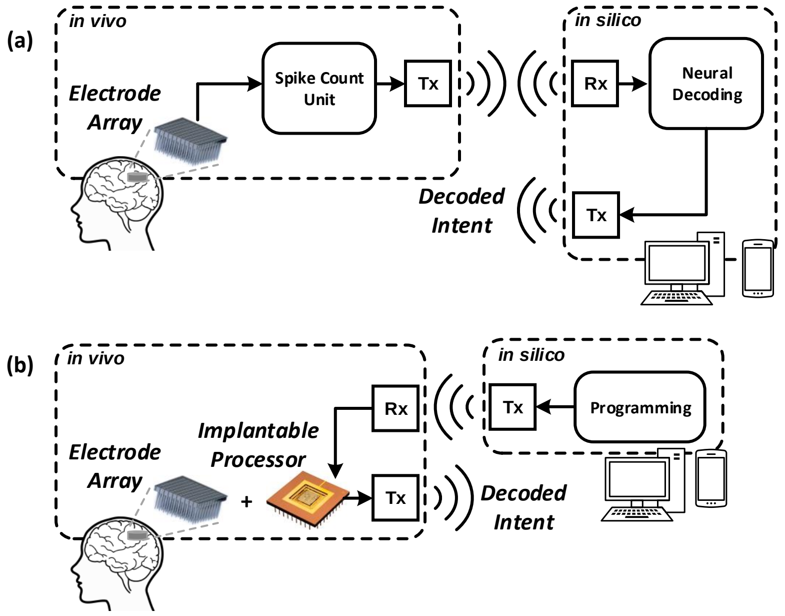


Figure 6.6. The system-level diagrams of (a) conventional and (b) the fully-implantable wireless BMIs.

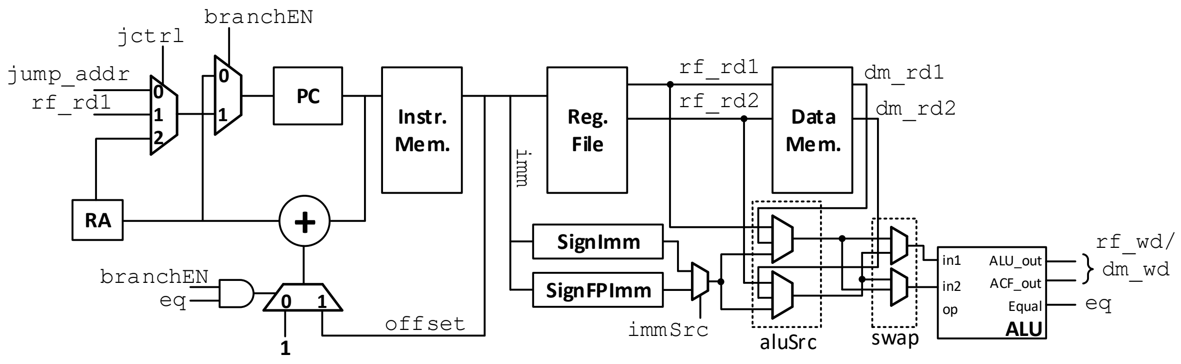


Figure 6.7. The top-level block diagram of the designed and implemented neural network processor.

neural decoding in vivo.

Due to the rapid advances in the field of machine learning, the processor would need to be programmable such that alternative decoding algorithms can be readily deployed using the same hardware. To this end, we have designed and implemented a custom neural network processor architecture for implementing arbitrary network operations. Our previously designed embedded processor architecture in [85] is enhanced to support arbitrary network operations, such as matrix multiplication, convolution, as well as the instructions required for implementing both feed-forward and temporal neural network architectures. The top-level datapath of the processor is shown in Fig. 6.7. The processor is a single-cycle reduced instruction-set computer (RISC) that executes an instruction over one clock cycle. Note that the relatively slow speed of biological signal generation and processing alleviates the need for high-speed and large-scale vector processing units that are typically found in conventional neural network accelerators. The designed processor has three memory units: the instruction memory *Instr. Mem.* used to store the program to execute; the register file *Reg. File* used to store program variables and memory addresses; and the data memory *Data Mem.*, which is used to store neural network input data, weight parameters, and any necessary intermediate network computation results. The *Instr. Mem.* has a single address port that supports asynchronous read and synchronous write. The *Reg. File* is a dual-port memory unit with asynchronous read and synchronous write. The two outputs of the *Reg. File*, `rf_rd1` and `rf_rd2`, are used to address the *Data Mem.*, which is also a two-port memory unit with asynchronous read and synchronous write. Since the *Data Mem.* is addressed by values stored in the *Reg. File*, the write address of the *Data Mem.* is also addressed by `rf_rd1`. The computations of a decoding algorithm are performed in a sequential manner by the arithmetic logic unit *ALU*, which supports integer computations, as well as fixed-point operations. The ALU also has a dedicated activation function unit that implements the sigmoid, hyperbolic tangent (tanh), and rectified linear (ReLU) functions for hidden units of a neural network. The detailed implementation of the activation function unit is given in [85].

The instruction set for the designed neural network processor is given in Table 6.2. Each instruction is 29 bits long, with an instruction operation code (opcode) of 5 bits, and the remaining 24 bits are dedicated for defining register addresses or immediate values. For example, for the `add_r` instruction, three 8-bit fields denote the register source for the two ALU inputs as well as the write address to store the result in the register file. The instruction set consists of four types of instructions: integer computation, fixed-point computations, setup, and control instructions. The integer computations are mainly used to define program variables, such as loop counters. The fixed-point computations are used to perform neural network computations, such

Table 6.2. The instruction set of the designed neural network processor.

Instruction type	Instruction	Function
Integer computation	`add_r $s1 $s2 $dst`	`RF[$dst] = RF[$s1] + RF[$s2]`
	`sub_r $s1 $s2 $dst`	`RF[$dst] = RF[$s1] - RF[$s2]`
	`mult_r $s1 $s2 $dst`	`RF[$dst] = RF[$s1] × RF[$s2]`
	`addi_r $s1 Simm $dst`	`RF[$dst] = RF[$s1] + SignExt[imm]`
	`subi_r $s1 Simm $dst`	`RF[$dst] = RF[$s1] - SignExt[imm]`
	`muli_r $s1 Simm $dst`	`RF[$dst] = RF[$s1] × SignExt[imm]`
Fixed-point computation	`add_e $s1 $s2`	`DM[$s1] + DM[$s2]`
	`sub_e $s1 $s2`	`DM[$s1] - DM[$s2]`
	`mult_e $s1 $s2`	`DM[$s1] × DM[$s2]`
	`addi_e $s1 $s2`	`DM[$s1] + FPSignExt[imm]`
	`subi_e $s1 $s2`	`DM[$s1] - FPSignExt[imm]`
	`muli_e $s1 $s2`	`DM[$s1] × FPSignExt[imm]`
	`wr_alu $s1`	`DM[$s1] = alu_result`
	`wr_acf1 $s1 $s2`	`DM[$s1] = DM[$s1] = ACF(alu_result, DM[$s2])`
	`wr_acf2 $s1 $s2`	`DM[$s1] = DM[$s1] = ACF(acc_result, DM[$s2])`
	`wr_acc $s1 $s2`	`DM[$s1] = DM[$s1] = acc_result`
	`rst_acc`	Reset ALU accumulator to zero
Setup	`iset_acf code`	Set activation function to sigmoid, tanh, or ReLU
	`sw_imm`	Swap ALU inputs
	`done`	Assert processor done flag for 1 clock cycle
	`rd_dm $s1`	Read `DM[$s1]` and write to processor output port
Control	`beq $s1 $s2 offset`	Branch to `PC+offset` if `RF[$s1] == RF[$s2]`
	`bneq $s1 $s2 offset`	Branch to `PC+offset` if `RF[$s1] != RF[$s2]`
	`jump jump_address`	`PC = jump_address`
	`jump_reg $s1`	`RA = PC+1; PC = RF[$s1]`
	`jump_return`	`PC = RA`
	`nop`	No operation

as matrix multiplication, element-wise addition/multiplication, and convolution. The fixed-point format consists of 16 bits for the integer part and 16 bits for the fractional part. Note that due to the relatively large amount of data required for neural networks, the *Data Mem.* is addressed by values stored within the register file. For example, the `mult_e` instruction performs the fixed-point multiplication of the value stored in the *Data Mem.* as indexed by the value stored in the register file at address `$s1`. For immediate instructions, the secondary input source to the ALU is either an 8-bit sign-extended immediate (`SignExt[imm]`) for integer computations, or a sign-extended and zero-padded 8-bit immediate, denoted as `FPSignExt[imm]`. The ALU has an internal accumulation unit that is enabled during fixed-point operations, which is useful for performing matrix multiplication, dot products, and convolutions. The setup instructions are used for miscellaneous operations, such as defining which activation function to use, swapping the order of ALU inputs, asserting the processor's flag `done`, and reading values from *Data Mem.* to the processor's output port. The control instructions are used for implementing loop constructs. Fig. 6.8 shows the program snippet for computing the dilated convolution. This code segment is used iteratively for implementing the filter convolution operations.

```
2DCompLoop: beq 0x80 0x1e 0x0c //branch to 2D_CompDone
mult_e 0x20 0x22 0x00 // .*
addi_r 0x80 0x80 0x01 //Increment compCounter
addi_r 0x20 0x20 0x01 //Increment filter read address
addi_r 0x22 0x22 0x01 //Increment data_read_address
beq 0x81 0x31 0x03 //Branch to dilation increment
addi_r 0x81 0x81 0x01 //Increment input dim counter
jump 0x00 0x0093 //Jump to 2DCompLoop
inputDilationOffset: addi_r 0x00 0x81 0x01
add_r 0x22 0x30 0x22 //Increment data_read_address by dilation offset
jump 0x00 0x0093 //Jump to 2DCompLoop
2D_CompDone: jump_return 0x00 0x00 0x00 //jumps to return address
```

Figure 6.8. The program snippet for computing the dilated convolution.

The synthesized ASIC layout of the designed neural network processor in a standard 180-nm CMOS process is shown in Fig. 6.9. Synthesis was performed with Synopsys Design Compiler, while place-and-route was performed with Cadence Innovus. The ASIC was syn-

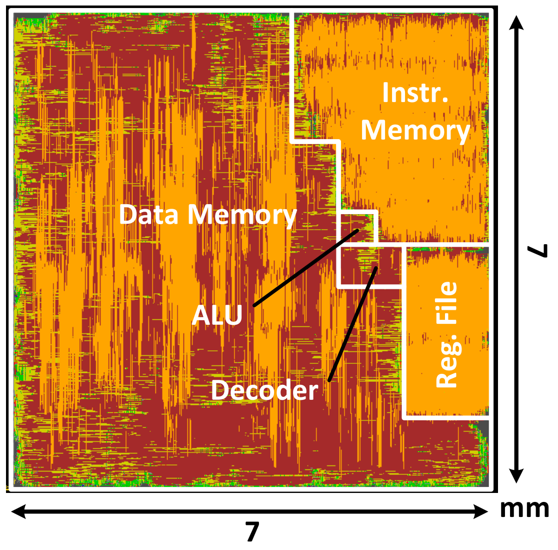


Figure 6.9. The ASIC layout of the designed and implemented neural network processor.

thesized with a maximum data memory depth of 4096 elements. The ASIC is estimated to occupy 49 mm^2 of silicon area and consumes 12 mW of power while operating at 500 kHz. The power density of the ASIC is 24.4 $\frac{\text{mW}}{\text{cm}^2}$, which is within the tissue-safe limitation of about 40 $\frac{\text{mW}}{\text{cm}^2}$ [17]. The operating speed of the ASIC and the spike count bin size directly limit the size and complexity of models that can be implemented for real-time decoding. Consider the processor operating at f Hz, and the spike counts are accumulated over time bins of size Δt seconds. The number of clock cycles λ for which to complete one time step of the real-time decoding can be given as $\lambda < \Delta t \times f$. For example, given Δt = 25 ms, the model should complete computations within 12,500 clock cycles with an ASIC operating at 500 kHz. Table 6.3 gives an estimated number of clock cycles required for different types of network operations. As an example, consider that spike counts are generated for 96 input channels and the decoded output dimension is 2 (e.g., x and y coordinates of a computer cursor). For a TCN model with 3 layers, 8 filters per layer, and a temporal filter length of 3, the TCN requires 16,408 instructions. One can note that the constraint $\lambda < \Delta t \times f$ may not be needed for non-continuous decoding schemes for which processing time is not vital. For example, if an output is only required after the entire temporal sequence is given, the processor can begin operating only when the final input spike

Table 6.3. The estimated number of instructions for various network operations

Operation	Matrix 1 size	Matrix 2 size	Number of instructions
Matrix Mults.	$M_1 \times N_1$	$N_1 \times N_2$	$(M_1 N_2) \times (3N_1 + 2)$
Convolutions	$M \times N$		$4(MN) + 1$
Element-wise +/–/×	$M \times N$		$6MN$

counts are received. Alternatively, meeting the constraint for λ will result in significantly faster inference time, as each temporal iteration of the decoder is computed immediately after the set of spike counts for the current time step is received.

6.5 Conclusion

This chapter explored the feasibility of the in vivo neural decoding with and without employing spike sorting. It was found that the neuron-rate coding outperforms the population-rate coding, both in terms of the amount of entropy inherent to the coding scheme as well as the decoding performance. It was shown that the temporal convolutional network (TCN)-based decoder provides relatively accurate decoding while being trained and calibrated faster than alternative machine learning-based models. Moreover, it was also found that the TCN-based decoder provides robust decoding without the need for spike sorting, thus reducing the computational and memory requirements for the in vivo processing. The design and implementation of a programmable decoder processor with a custom instruction set architecture for executing neural network operations in a standard 180-nm CMOS process was presented. The ASIC layout was estimated to consume 49 mm^2 of silicon area and to dissipate 12 mW of power from a 1.8 V supply, which is within the tissue-safe limitation of 40 mW/cm^2.

6.6 Publications

The following article was published as an outcome of the research conducted in this chapter: D. Valencia and A. Alimohammad, "Towards in vivo neural decoding," Biomedical Engineering Letters, vol. 12, pp. 185 – 195, May 2022.

Chapter 7

A Generalized Hardware Architecture for Real-time Spiking Neural Networks

7.1 Introduction

In the United States alone, over 17000 cases of spinal cord injury (SCI) or other neurodegenerative diseases (NDDs) are reported per year [185]. SCI and NDDs can affect the ability of biological brains to communicate and/or interface with other parts of the body. Brain-machine interfaces (BMIs) have become a viable solution for creating alternative communication pathways between the human brain and external devices, such as a prosthetic arm or a computer.

Over the past decade, artificial neural networks (ANNs) have been employed in a wide range of applications. Feed-forward neural networks (FNNs) and multi-layer perceptrons (MLPs) process data in a single direction from input to output, while recurrent neural networks (RNNs) utilize feedback connections among artificial neurons (ANs) to learn about the temporal information within the input data. Convolutional neural networks (CNNs) are primarily used in image and video processing. They can decompose images with a variety of filter kernels and can be combined with FNNs to perform accurate image classification, segmentation, and recognition tasks, such as detecting and segmenting medical images to assist with disease diagnosis [186]. Temporal convolutional networks (TCNs) employ causal convolutions and strided dilations to adapt to sequential data with a receptive field larger than conventional RNNs [176]. Transformer neural networks [187] employ a self-attention mechanism to weigh the relevance of particular points in sequential data. The mechanism of self-attention has proven invaluable for natural language processing tasks, such as machine translation [188].

The early study of neural networks led to the development of more biologically-plausible neural networks. Spiking neural networks (SNNs) [189], often referred to as the third generation of neural networks, employ more accurate mathematical models for describing the behavior of brain neurons. The neurons in SNNs are connected to one another using synapses, and the weights associated with the synapses model the impact of one neuron's spike on its post-synaptic neighboring neurons. The values of the synaptic weights are obtained through the training process, in which weights are fine tuned so that the employed SNN can mimic desired spiking behaviors, such as responding to input stimuli at specific times. While neurons in an ANN communicate by sending discrete values to one another, spiking neurons communicate with one another via exhibiting spiking behavior, similar to those of a biological brain, through the emulation of the membrane potentials of the neuron.

Due to their biological plausibility, which is not inherent in the relatively simple mathematical models of neurons in the conventional ANNs, SNNs have become the prime candidate for interfacing with living biological brain neurons [190, 191]. Various experiments have shown that interfacing a SNN for stimulating the main respiratory muscles, such as the diaphragm, has helped patients regain ventilatory control [192]. Recently, it has been shown that information transfer from SNNs to biological neural networks is possible [193], and hence SNNs can potentially be realized as replacement neural micro-circuits. For example, in [194], one hundred synthetic spiking neurons' parameters are tuned to match the behavior of biological neurons for restoring communication between two neuronal populations separated by a focal lesion. Another effort has shown that SNNs can potentially be used as replacements for damaged biological networks in the hippocampus of behaving rats to restore and improve memory functionality [195, 196].

The real-time operation of the SNNs, however, imposes strict limitations on the feasible computational complexity of the mathematical models for SNNs and hence, on the silicon area, power consumption, and latency of the realized hardware circuits. Various neuron models have been proposed, ranging from the simple integrate and fire (I&F) model [197] to the relatively complex Hodgkin and Huxley (HH) model [198]. The Izhikevich neuron model [199], however, can reproduce the spiking dynamics of cortical neurons reliably while also being computationally more efficient than the other state-of-the-art SNN models [200].

This chapter focuses on the area- and power-efficient design and implementation of a generalized hardware architecture for moderately-sized SNNs, in the range of tens of thousands of neurons, based on the Izhikevich neuron model. While most of the previously-published work have elaborated on fixed dedicated hardware architectures, supporting a predefined number of neurons and synaptic interconnect topologies [201, 202, 203, 204], the proposed architecture aims to exploit the reconfigurable nature of the field-programmable gate arrays (FPGAs) to support an arbitrary number of spiking neurons and diverse network interconnect structures, while offering an adjustable time resolution. In addition to FPGA [201, 202, 203, 204, 205, 206] and programmable processor [207] realizations of SNNs, application-specific integrated circuit (ASIC) implementations of SNNs have also been reported [208, 209, 210, 211, 212, 213]. While FPGA and processor realizations offer greater flexibility in realizing diverse SNNs, ASIC implementations can consume significantly lower power and can be implemented in a smaller form factor and hence, are practical for the in-vivo brain implantations. In this work, we propose to implement a tightly integrated SNN core with embedded memory and computational elements. In contrast to the state-of-the-art ASIC realizations, our design's power consumption and real-time processing are not affected by the spiking rate of the emulated SNN. Our design supports various spiking rates based on the number of neurons and allows an adjustable degree of parallelism and time step resolution. State-of-the-art ASIC implementations, such as Loihi [213], SpiNNaker [207], and Tianjic [214], aim for high-performance computing by exploiting massively-parallel architectures or distributing computations over a large amount of processing cores. While our approach is that of a generalized real-time SNN, our goal is to integrate all of the required elements of the SNN core, such as processing elements and memory, into a single unified chip. The relatively small silicon area and low power consumption of the implemented generalized SNN hardware architecture make it suitable for single-chip realizations, while emulating SNNs with various sizes and interconnect topologies in real-time. Our proposed design is scalable and additional cores can be readily utilized at the expense of increased silicon area.

The rest of this chapter is organized as follows. Section 7.2 briefly reviews the main computational requirements of the Izhikevich-based SNN model. Section 7.3 provides a detailed explanation of the designed generalized hardware architecture for SNNs. A memory partitioning scheme for parallel computation of the membrane potentials is presented. Section 7.4 discusses the limitations of the real-time SNN realizations, quantifies the implementation characteristics of the proposed design, and provides comparisons between the FPGA and ASIC implementation results of our generalized SNN architecture and those of the state-of-the-art work. Finally, Section 7.5 makes some concluding remarks.

7.2 A Computationally-Efficient SNN Model

Izhikevich's neuron model iteratively reproduces the spiking dynamics of biological neurons using the following mathematical expressions [199]:

$$v' = (t \cdot (0.04v^2 + 5v + 140 - u + I)) + v, \tag{7.1}$$

$$u' = a(bv' - u), \tag{7.2}$$

$$\text{if } v' \geq \text{ 35 mV}, \begin{cases} v \leftarrow c \\ u \leftarrow u' + d \end{cases}, \tag{7.3}$$

where v denotes the membrane potential in terms of mV, u denotes the recovery variable, I denotes the accumulated input activity to the neuron, and t denotes the resolution of the time step. The spiking dynamics of the neurons can be controlled by the parameters a–d, where a denotes the decay rate of the recovery variable u, b denotes the coupling between v and u, which can result in sub-spiking oscillations, c denotes the value of the membrane potential after a spike, and d denotes the reset of the recovery variable u after a spike. Given a unit-step input current, Fig. 7.1 shows the membrane potential voltages for four different types of neurons. The regular spiking (RS) neurons exhibits the spiking behavior shown in Fig. 7.1(a) with parameters $a = 0.02$, $b = 0.2$, $c = -65$, and $d = 8$, and are the most common type of neurons in the mammalian cortex [215]. The parameters $a = 0.02$, $b = 0.2$, $c = -50$, and $d = 2$ define the chattering (CH) neurons, which fire bursts of spikes following an excitation, as shown in Fig. 7.1(b). Fig. 7.1(c) shows a low-threshold spiking (LTS) neuron, which can fire high-frequency spikes with adaptation (i.e., slowing down of firing rate) using $a = 0.02$, $b = 0.25$, $c = -65$, and $d = 2$. Fig. 7.1(d) shows a fast-spiking (FS) neuron, which can fire rapid spikes without adaptation, with $a = 0.1$, $b = 0.2$, $c = -65$, and $d = 2$. Both RS and CH neurons are examples of excitatory cells, while the LTS and FS neurons are inhibitory cells.

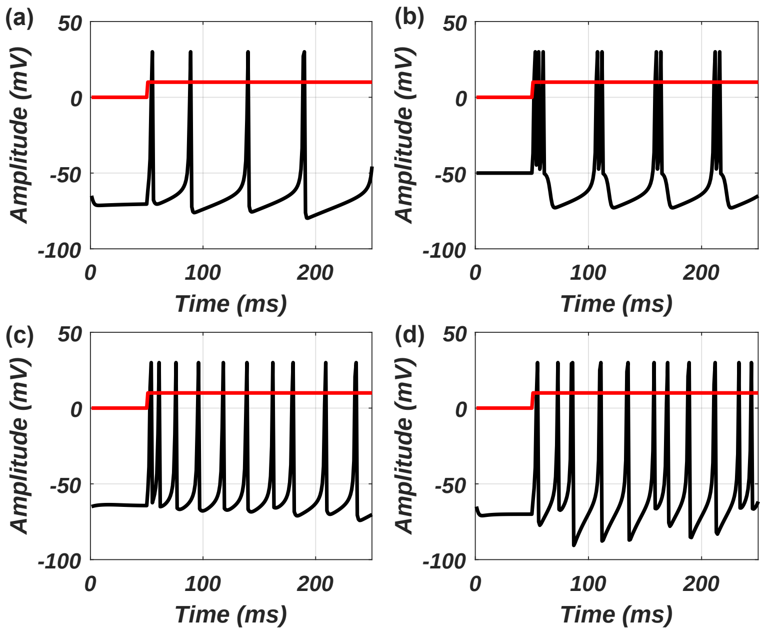


Figure 7.1. Membrane potential waveforms for (a) regular spiking, (b) chattering, (c) low-threshold spiking, and (d) fast-spiking neurons.

An iteration of the Izhikevich-based SNNs using the neuron state Equations (7.1) – (7.3) can be executed in three steps: (i) receiving network inputs; (ii) performing synaptic weight accumulation; and (iii) computing the membrane potentials. In the first step, the network inputs are buffered. The second step poses the main computational bottleneck for realizing the SNN model. In traditional ANNs, the neuron's communicate by passing discrete values to one another; however, SNNs operate differently. The output of a spiking neuron is the excitation or lack thereof and hence, can be represented as a binary value, denoting whether the neuron emitted (fired) a spike. A neuron fires a spike if the membrane potential crosses a predefined threshold of

35 mV, which results in resetting the membrane potential v to the predefined neuron's potential value c. After a neuron fires a spike, the neuron cannot fire another spike for a period of time, known as the refractory period. When a neuron fires, the synaptic weights of all pre-synaptic firing neurons are accumulated and added to the current input stimuli for every connected post-synaptic neuron. Note that since Equation (7.1) is on the millisecond scale, more precise spike timings can be modeled using finer temporal resolutions. This requires more iterative computations of Equation (7.1) to complete one millisecond of emulation. For example, if the time resolution $t = 0.5$ ms, the membrane potential v' has to be computed twice (with $v = v'$ for the second iteration). The third and final step is to update the membrane potentials using the neuron state Equations (7.1) – (7.3).

The biologically-inspired models attempt to replicate the behavior of biological neural systems, but not necessarily in a biologically-plausible manner. For example, the I&F model employs a relatively simple biologically-inspired spiking neuron model. Even though this model is less biologically realistic, it produces enough behavioral resemblance to the biological neurons to be useful in spiking neural systems. The leaky I&F model adds a leak term to the simple I&F model, which causes the potential on a neuron to decay over time. The leaky I&F model updates the membrane potential v as:

$$v' = I + a - bv,$$

$$\text{if } \mathrm{v}' \geq \mathrm{v}_{\text{thresh}} : v \leftarrow c,$$

where I denotes the input current to the neuron and a, b, c, and v_{thresh} are the neuron parameters. Because the I&F model has only a single variable, it cannot exhibit all 20 neuro-computational features and hence, is not considered a biologically-plausible neuron model. An alteration to the I&F model decreases the frequency of spiking and is referred to as spike-frequency adaption. An extension to this model was the Adaptive Exponential (AdExp) model which changes the input current injection to conductance-based injection. These extensions represent a better reflection of the changes seen by cortical neurons. The biologically-plausible models, however, explicitly model the types of behavior that are seen in biological neural systems. The most popular biologically-plausible neuron model is the HH model, which is relatively computationally-intensive and takes 1200 floating-point operations to evaluate one millisecond of time, which can be limiting for real-time SNN emulation. The Izhikevich spiking neuron model (IZH) was developed to produce similar bursting and spiking behaviors as can be elicited from the HH model, but do so with a much simpler computation.

7.3 A Generalized Reconfigurable Hardware Architecture for SNNs

For network topologies which follow the classical, layer-based structure, network inputs are only applied to the input layer. However, for realizing a generalized hardware architecture, the inputs are given to every neuron in the network. The designed scalable architecture utilizes the degree of parallelism parameter *PDeg*, which controls the number of input accumulations to be computed in parallel. Utilizing folding transformations [216], computations can be reduced to *PDeg* SNs at a time, resulting in a hybrid sequential-parallel hardware architecture. By effectively folding the computations down to *PDeg* neurons at a time, the overall resource utilization can be significantly reduced. This in effect allows more complex systems to be integrated onto a single chip or logic resources can be allocated for additional neural signal processing or computational functions.

For example, the brain-computer interface in [101] employs spike sorting in conjunction with emulated spiking neural networks to manipulate a prosthetic device. While their system utilizes multiple CPU cores for real-time SNN emulation, our proposed hardware architecture can accommodate several neural signal processing modules onto a single chip for emulation of the real-time moderately-sized SNNs. By updating *PDeg* SNs at a time, the hardware architecture can be readily scaled such that a millisecond of SNN emulation is completed within a predefined number of clock cycles. Compared to the software realizations, the total number of neurons N, and the degree of parallelism *PDeg*, supported by our hardware architecture are limited by the number of reconfigurable resources available on the target FPGA device or the silicon area and power consumption of the ASIC implementation.

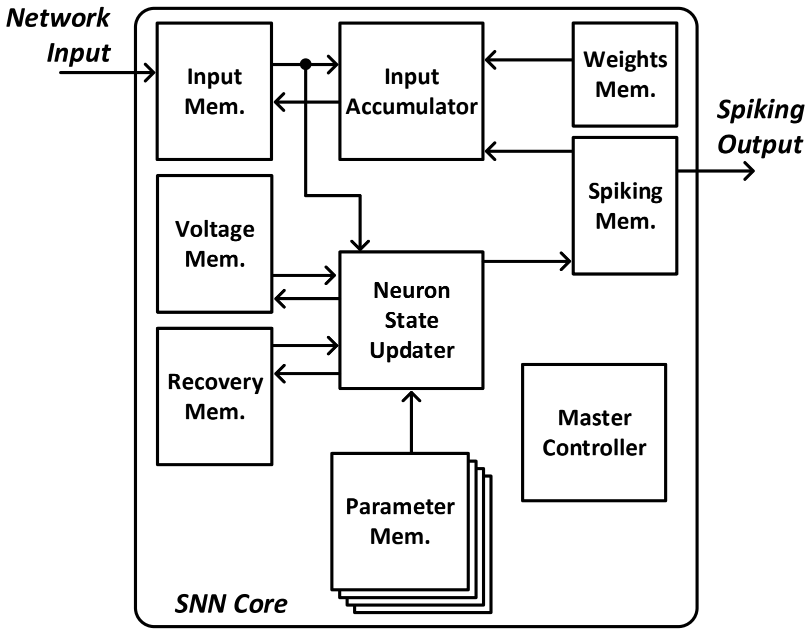


Figure 7.2. The top-level block diagram of the designed and implemented SNN core.

7.3.1 SNN Core Memory

The top-level block diagram of the designed SNN core is shown in Fig. 7.2. The SNN architecture consists of several memory units for storage of: (i) the network's inputs *Input Mem.*; (ii) the neurons' membrane potential voltage values *Voltage Mem.*; (iii) the neurons' recovery variables *Recovery. Mem.*, (iv) the SN's outputs *Spiking Mem.*; (v) the synaptic weights *Weights Mem.*; (vi) and the parameter memory *Parameter Mem.* to store the Izhikevich parameters $a - d$ per neuron. In order to support an arbitrary interconnect structure among the SNs, the *Weights Mem.* stores N^2 synaptic weights. By changing the synaptic weights stored in the *Weights Mem.*, various SNN configurations can be realized. The memory units shown in Fig. 7.2 are implemented using the FPGA's block RAM (BRAM) resources. The BRAMs are specialized on-chip storage resources on an FPGA, where their maximum width and depth varies among different device families. For example, the Xilinx Artix-7 FPGAs can store up to 36 kB of data in BRAMs, with the BRAMs' width being programmable up to 72 bits.

In order to process *PDeg* neuron computations per clock cycle, all memory units are partitioned in such a way that there are no conflicting data accesses. This avoids implementing a queue for managing memory read/write requests. Each row of the designed partitioned memory modules stores the indices of *PDeg* SNs, and thus, N/PDeg rows are required to store all neurons' indeces. The neuron's index stored at row i and column j of the partitioned memory module is given as $i+(j \times N/\mathit{PDeg})$. The synaptic weight memory has N^2/PDeg rows and each row stores *PDeg* weights, each of which is denoted with a unique identifier $\lambda\rho$, representing the synaptic weight from neuron λ to neuron ρ. Moreover, a bitwise OR operation is performed on the read data from the SN output memory. When all of the *PDeg* bits in a row of the spiking output memory are zeros, unnecessary accumulations from the neurons represented by that row to all other post-synaptic neighboring neurons are avoided, which in turn saves a significant number of clock cycles, memory read operations, and redundant accumulations.

7.3.2 Neuron Input Accumulation

Due to the generalized architecture, each neuron can have a large fanout of N, where N denotes the number of total neurons supported by the target device. The synaptic input accumulation thus poses the largest computational bottleneck of the proposed design. Fig. 7.3 shows the datapath of the *Input Accumulator* module. The network inputs *Net. Input* for the current emulation time are passed to the accumulation register *R* through the 2-to-1 multiplexers. The *iRST* control signal enables the accumulation registers to be enabled, thus resetting the accumulated input to the first *Net. Input* value. The synaptic weights *Syn. Weight* and *Spiking Output* are read from the *Weights Mem.* and *Spiking Mem.*, respectively. If a *Spiking Output* bit is high, then the register is enabled, and the synaptic weight from that source neuron is added to the net input of the neuron.

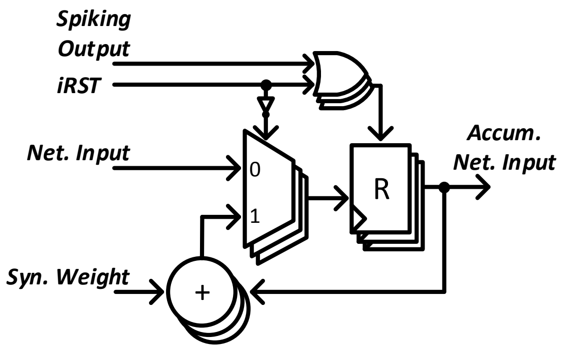


Figure 7.3. The datapath of the input accumulator module.

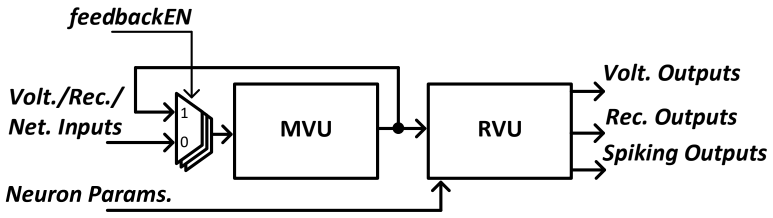


Figure 7.4. The block diagram of the neuron state updater.

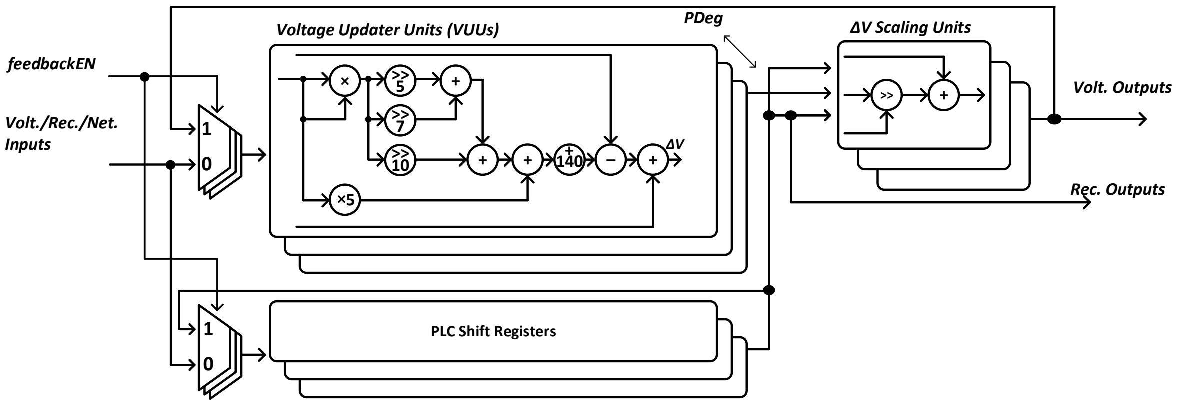


Figure 7.5. The block diagram of the membrane voltage updater (MVU).

7.3.3 Neuron State Computation

The module *Neuron State Updater* performs the main SNN computations described by the neuron state Equations (7.1) – (7.3). Fig. 7.4 shows the high-level block diagram of the neuron state updater. The datapath consists of two pipelined units, the membrane voltage updater *MVU* and the recovery variable updater *RVU*, which are responsible for computing the neuron state Equations (7.1) and (7.2–7.3), respectively. The datapath of the MVU unit, shown in Fig. 7.5, consists of the voltage updater units (VUUs), pipeline latency compensation shift registers *PLC Shift Registers*, and ΔV scaling units. The seven-stage pipelined VUUs compute the changes to the neuron voltage values, denoted as ΔV. While accurate representation of 0.04 in the fixed-point numerical format would require a relatively large number of fractional bits, we instead utilize a shift-and-add approximation as $\frac{1}{32}+\frac{1}{128}+\frac{1}{1024}=0.040039\approx 0.04$. The rest of Equation (7.1), in which the input value v is added to the scaled ΔV, is performed by the ΔV scaling units. The inputs to the ΔV scaling units include the change in voltage for each neuron, the original voltage input to the neuron, and the time scale. The original input value v is shifted through the *PLC Shift Registers*. The time scale is implemented using the arithmetic right shift of input values by $-1\times(\log_2 t)$ bit positions, where t denotes the resolution of the time step used in Equation (7.1). For $t = 0.0625$ ms, this corresponds to a 16-bit arithmetic right shift.

The number of iterations used by the VUUs is equal to the shift position used by the ΔV scaling units. The multiplexers at the inputs to the VUUs and the *PLC Shift Registers* can select either the input ports to read voltage, recovery, and network inputs for the VUUs, or to accept the outputs of the VUUs for subsequent iterations. The select line *feedbackEN* asserted by the control unit is responsible for enabling the feedback into the VUUs. For cases in which the number of inputs streaming to the VUUs exceeds the pipeline depth of VUUs, additional output shift registers are added to the output of the ΔV scaling units and to the output of the *PLC Shift Registers*. The number of inputs that will propagate through the VUUs is equal to the number of rows in the voltage, recovery variable, and input memories, which is N/PDeg.

Fig. 7.6 shows the datapath of the designed *RVU Pipeline* module. It consists of the *Recovery Voltage Units* (RVUs), pipeline latency compensation shift registers *PLC Shift Registers*, a bank of comparators, a bank of resetable adders, and output resetting/spiking multiplexers. The four-stage pipelined RVU computes the output recovery variable, as given in Equation (7.2). The outputs *Volt. Outputs* and *Rec. Outputs* of the MVU are passed to the *Recovery Pipeline* unit, shown as *MVU Outputs*. The desired spiking dynamics of neurons, denoted by the Izhikevich parameters $a-d$, are passed through the input port *Neuron Params.*. The *PLC Shift Registers* are

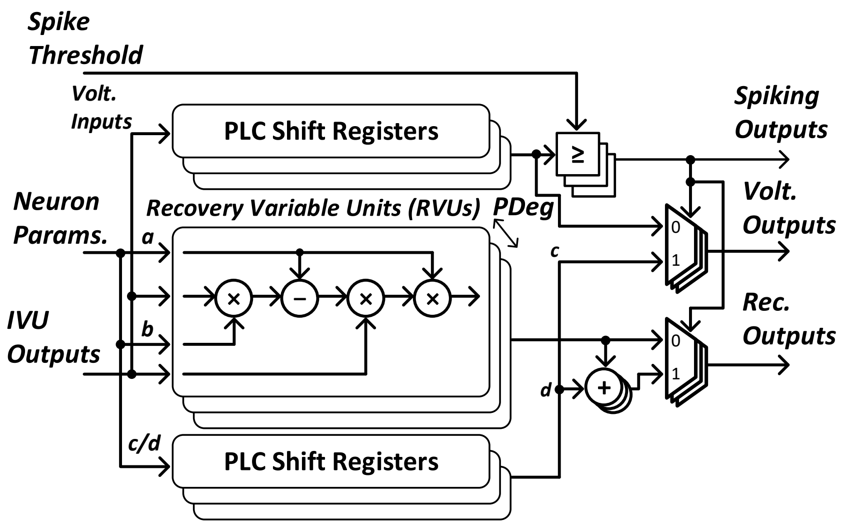


Figure 7.6. The block diagram of the Recovery Variable Updater (RVU).

used to compensate for the latency of the RVU and to propagate the parameters c and d, and voltage v values. If the shifted *Volt. Inputs* values are greater than or equal to the value of the spike threshold in Equation (7.3), a spike is emitted. The output of the comparators are used as the select lines of the multiplexers, which will either pass the original *Volt. Inputs* values to the output if a spike did not occur, or they will pass the shifted reset value c, given the occurrence of a spike. Similarly, the output of the RVUs, the updated recovery variables, will be passed to the output port *Rec. Outputs* if a spike did not occur; otherwise, the updated recovery variables will be added to the shifted parameters d for resetting the recovery variables, as given in Equation (7.3).

The operations of the designed architecture can be briefly explained by the following example. For simplicity, assume three RS neurons, each of which has a synaptic weight of 10 mV to the other two neurons. Also, assume that the current values for the membrane voltage and the recovery variables are $v = [-45, -45, -23]$ and $u = [-13, 12, 10]$, respectively, and $t = 0.5$. Finally, assume that the next external inputs to each neuron are equal to $I = [5, 0, 12]$. First, the neuron input accumulation unit will read input data from the *Input Mem.*. Initially, the accumulated network inputs for each neuron are equal to the values read from the memory, i.e., $I_{\text{acc}} = [5, 0, 12]$. Next, the *Spiking Mem.* is read and its output value is 100, which indicates that the first neuron emitted a spike during the last millisecond of operation. Thus, the synaptic weight value of 10 mV is added to the accumulated network input of neurons ‘2’ and ‘3’ and hence, $I_{\text{acc}} = [5, 10, 22]$. Once the external inputs and synaptic weights are accumulated, the *Neuron State Updater* computes the updated values of $v' = [-25, -52, 98]$ and $u' = [0.15, -0.45, 0.19]$. Since the membrane potential of neuron ‘3’ exceeds the threshold of 35 mV, the membrane potential and the recovery variable for neuron ‘3’ is updated as $v' = [-25, -52, -65]$ and $u' = [0.15, -0.45, 8.19]$. The spiking outputs of the neurons, 001, are then written into the *Spiking Mem.*, which indicates that neuron ‘3’ has fired a spike. This process is repeated for each millisecond of the SNN operation.

7.4 SNN Emulation and Implementation Results

The brain’s neural activity is recorded using either non-invasive or invasive techniques, the latter of which generally provides greater spatial resolutions and decoding accuracy. The raw recorded signals are then amplified and digitized using an analog-to-digital converter (ADC), and filtered to the frequency bands of interest. In order to provide real-time interfacing between biological neurons and a SNN, we must verify that the SNN can operate predefined time constraints.

7.4.1 Real-time SNN Emulation

A typical sampling frequency of 10 kHz [217] requires the SNN processing to be completed within 0.1 ms for real-time operation. Therefore, the number of iterations of Equation (7.1) must be either 8 ($t = 0.125$ ms) or 16 ($t = 0.0625$ ms). Fig. 7.7(a) shows the maximum clock cycle latency for the input accumulation when all 1000 neurons in the network have fired. Fig. 7.7(b) shows the number of spiking neurons for a SNN with 1000 fully-connected neurons. One can see that the maximum spiking rate is only 5.1% of all neurons in the network and hence, it is unlikely that all neurons of the network will fire simultaneously. The average spiking rate is

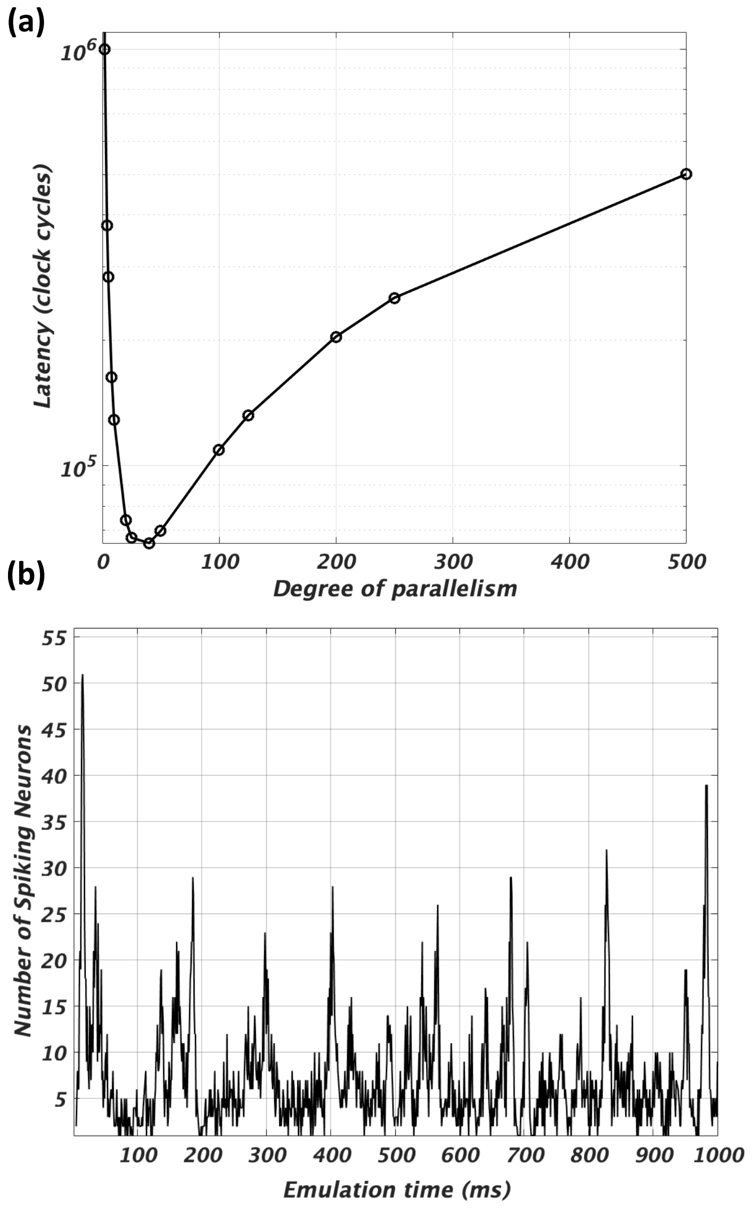


Figure 7.7. (a) The maximum clock latency of the input accumulation module for varying degrees of parallelism, and (b) the number of spiking neurons for a SNN with 1000 fully-connected neurons over one second of emulation.

even significantly smaller, at 0.7413%. The accumulation latency can be given as:

$$((2N/PDeg)^2+20)+SG(N+(PDeg^2-PDeg)), \tag{7.4}$$

where SG denotes the number of SN groups in which at least one SN fired. The maximum number of SN groups N/PDeg gives a maximum latency of $129,020$ clock cycles for a network with $N = 1000$ and $\mathsf{PDeg} = 10$.

Fig. 7.8 shows the latency, based on the number of clock cycles, for updating the neurons' v and u values, with the time resolutions $t = 0.125$ ms and $t = 0.0625$ ms, for a SNN with $N = 1000$ neurons over varying degrees of parallelism. The number of clock cycles required to complete the neuron state updating process can be given as $\left(\frac{N}{PDeg}(V_{Iter}+1)\right)+7$, where $V_{Iter} = 1/t$ denotes the number of iterations required by the membrane voltage updater for the desired time resolution t. Because the designed architecture utilizes shifting operation to implement the required time scaling in Equation (7.1), the time resolution t must be a negative integer power of 2 and hence, the degree of parallelism PDeg must be an even divisor of N. According to Fig. 7.8, FPGAs that can accommodate storage and computational resources for about one-fifth of the number of neurons N in a SNN yield approximately two degrees of magnitude lower accumulation latency than emulating a SNN with a single processing element (e.g., traditional in-order execution with $\mathsf{PDeg} = 1$). This is also the case for the input accumulation process. To quantify the time available for the real-time operation (i.e., completing one iteration of computations within one millisecond), we can determine the number of SN groups to estimate the maximum clock cycle budget of the input accumulation module for each SN group. As shown in Fig. 7.9, for $N = 1000$ and $\mathsf{PDeg} = 10$, the maximum number of SN groups is 44, which corresponds to an input accumulation clock cycle budget of $67,980$. This clock cycle budget can be added to the clock latency of the MVU, which is 1707 for $V_{Iter} = 16$. As a result, the total clock cycle budget for the real-time SNN operation is $69,687$.

7.4.2 Hardware Implementation Results

We have implemented the designed generalized reconfigurable hardware architecture for a SNN with 1000 fully-connected neurons with $\mathsf{PDeg} = 10$ on a Xilinx Artix-7 XC7A200T FPGA. It utilizes 12700 (9.49%) lookup tables (LUTs), 11139 (4.16%) flip-flops (FFs), 14.50

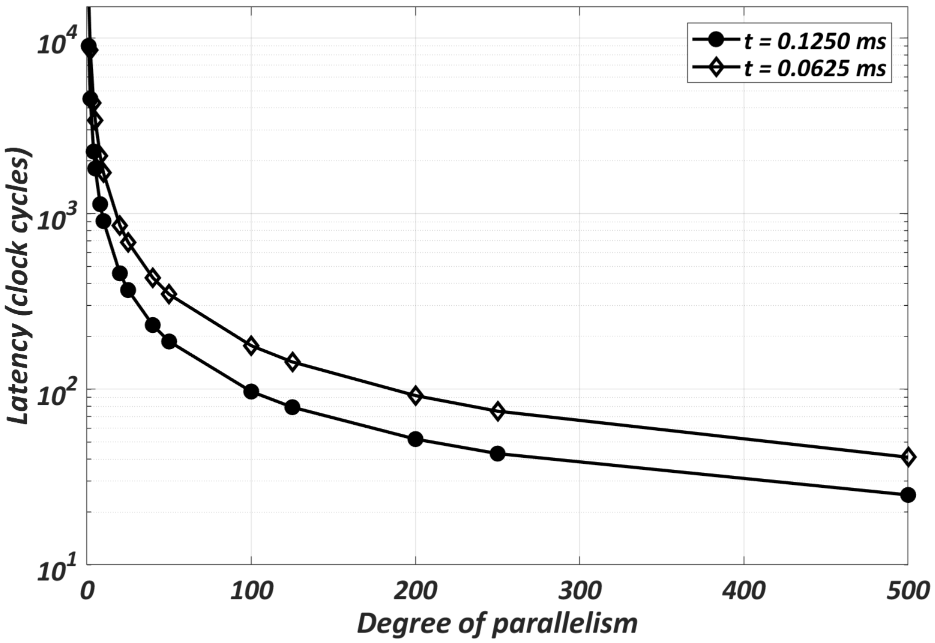


Figure 7.8. The clock cycle budget for computing the neuron state equations with varying degrees of parallelism for a network with N = 1000 neurons over 1000 emulation time steps.

(3.97%) block RAMs (BRAMs), and 110 (14.86%) dedicated multiplication units (DSP48s). This network size is comparable to those of the other single FPGA implementations, for a fair comparison. The iterative computation of the membrane potentials via the MVU with the folded hardware architecture requires a relatively small number of configurable FPGA resources. The clock cycle budget for the real-time operation of the implemented SNN can be determined as 92 MHz $\times$ 1 ms = 92000 cycles, where 92 MHz is the operating frequency of the SNN implemented on the FPGA and one millisecond is required for completing the SNN computations before the next set of neural samples are available based on a 10 kHz sampling rate. The inference speed of the network depends on the degree of parallelism, the time step resolution, and the number of emulated spiking neurons. For example, given an FPGA operating frequency of 92 MHz, degree of parallelism *PDeg* = 10, emulating 1000 spiking neurons with a 0.5 ms time step resolution, each computation time step takes 0.49 ms. Note that the size of the networks that can be implemented is directly limited by the number of computational resources

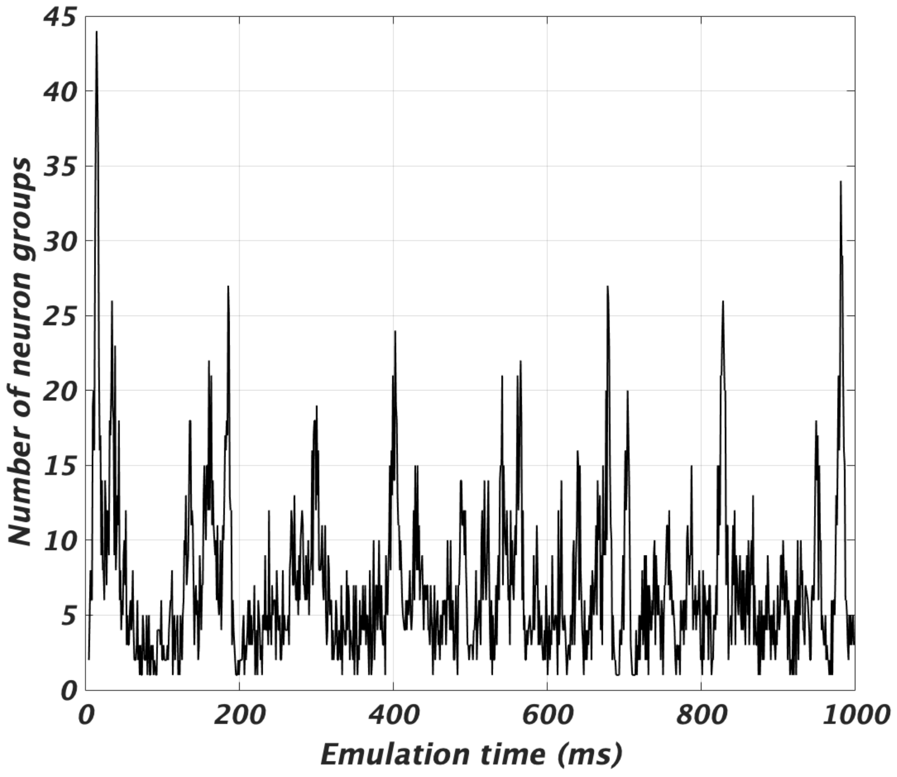


Figure 7.9. The number of SN groups for *N* = 1000 neurons, *PDeg* = 10, over 1000 emulation time steps.

and memory elements available on the target device.

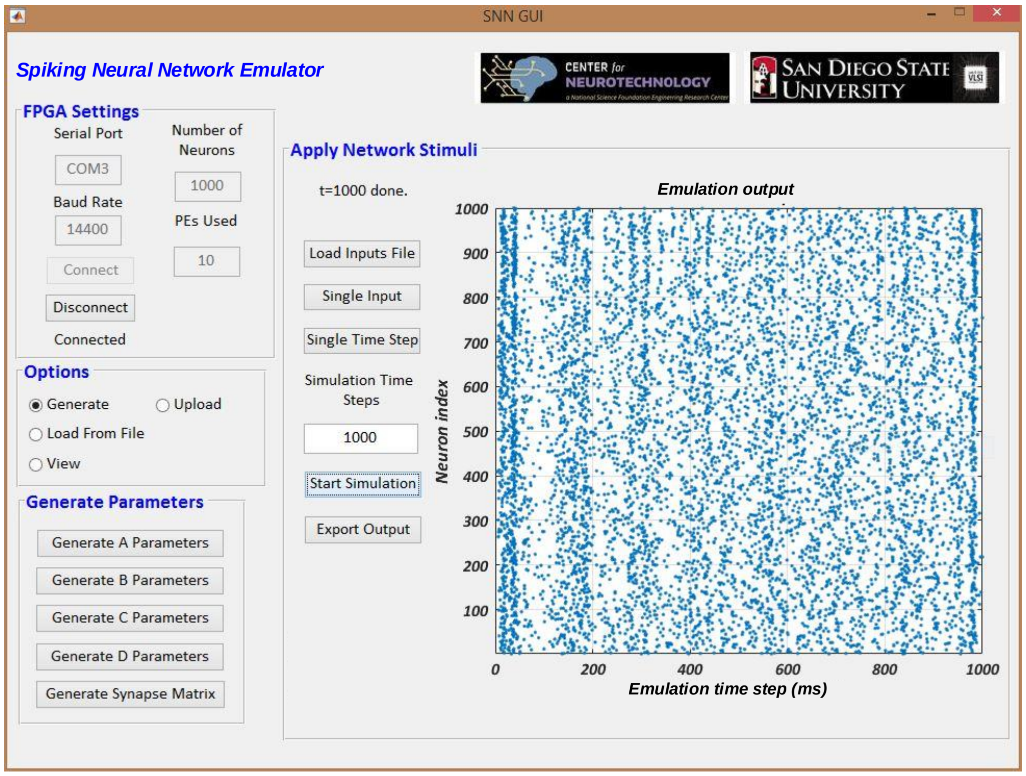
7.4.3 Design Verification

To test the designed and implemented SNN architecture, we have developed a custom MATLAB-based graphical user interface (GUI), shown in Fig. 7.10. The GUI interacts with the FPGA through a serial port. The baud-rate and the parameters of the synthesized and implemented SNN on the FPGA device are specified under the *FPGA Settings* panel. The *Options* panel allows the user to randomly generate network parameters and synaptic weights, load previously generated network parameters and synaptic weights from a file, view the network parameters and synaptic weights currently loaded onto the FPGA, and upload the network parameters and synaptic weights onto the FPGA. The SNN is trained using our custom-developed scripts in Python. The neuron parameters $a-d$ and the synaptic matrix values derived by the training are then loaded onto the SNN hardware through the GUI for functional verification. Once the connection to the FPGA is established, the GUI displays the *Apply Network Stimuli* panel, which allows the user to apply network inputs to the SNN hardware implemented on the FPGA. The scatter plot *Emulation output* then shows the spiking activity of each neuron for each emulation time step. As shown in Fig. 7.10, the emulation output of the implemented SNN utilizing 1000 fully-connected neurons with *PDeg* $= 10$. A neuron has a connection to another neuron in the network through a randomly initialized weight value. Weight values of zero imply that there is no connection between a pre- and post-synaptic pair of neurons. In this specific model, neurons have no self-recurrent connections. Although the SNs are randomly coupled and no synaptic training was employed, the occasional dark vertical lines indicate synchronized firing events across the majority of neurons. This behavior is akin to the mammalian neo-cortex behavior during an awake state and is an indication that the Izhikevich model reproduces the behavior of biological neurons and their spiking dynamics.

In order to evaluate the SNN and Izhikevich model for a practical application, we evaluate our design using the MNIST digit recognition dataset [218]. The dataset consists of 28×28 pixel grayscale handwritten digits with values between 0 and 255. The images are converted into poisson-firing spike trains following the approach in [219]. The firing rate of a pixel is proportional to the pixel intensity, with a maximum firing rate of 350 Hz. Each set of image spike trains are presented for 50 milliseconds, with another 50 millisecond of no input to allow the network to settle before presenting the next image. The weights from input spike trains to spiking neurons are derived through spike-time dependent plasticity (STDP) using a supervised training scheme described in [220]. The weight change rule is given by:

$$\Delta w_{ij}(t) = \mu \cdot \xi_j(t) \sum_{\hat{t}=t-\varepsilon}^{t} s_i(\hat{t}), \tag{7.5}$$

where $w_{ij}(t)$ denotes the weight from input i to neuron j, μ denotes the learning rate, $\xi_j(t)$


SNN GUI

Spiking Neural Network Emulator

FPGA Settings
Serial Port: COM3
Baud Rate: 14400
Number of Neurons: 1000
PEs Used: 10
Connected

Options
[✓ Generate] Upload Load From File View

Apply Network Stimuli
t=1000 done.
Simulation Time Steps: 1000

Figure 7.10. The developed graphical user interface for emulating and testing SNNs on FPGAs.

denotes the STDP-based learning rule for neuron j, ε denotes the STDP window in milliseconds, and $\sum_{\hat{t}=t-\varepsilon}^{t} s_i(\hat{t})$ denotes the number of spikes fired by input i during the time interval $[t-\varepsilon, t]$. The learning rule $\xi_j(t)$ is given by:

$$\xi_j(t) = \begin{cases} +1 & z_j(t) = 1, r_j \neq 1 \in [t-\varepsilon, t], \\ -1 & z_j(t) = 0, r_j = 1 \in [t-\varepsilon, t], \\ 0 & \text{otherwise}, \end{cases}$$

where $z_j(t)$ denotes the target spike train for neuron j and r_j denotes whether neuron j has fired a spike within the STDP window. The above equations can thus be interpreted as local STDP-based rules, whereby neurons that should fire have their input weights strengthened and neurons that should not fire have their weights weakened. If a neuron fires within the STDP time window ε, the neuron is performing as desired and no weight change is required.

While the STDP-based learning rules are suitable for on-chip learning, our proposed custom architecture is not intended to support it. On-chip training needs additional hardware, such as storing a history of spiking activity over a STDP window. Also, for training the synaptic weights from external spiking activity, an additional memory unit would be required to store input spiking history. Combining STDP-based learning rules with stimuli-dependent teaching signal requires an additional memory unit to define which neuron populations should respond to specific stimuli.

In our network, the 100 spiking neurons are divided into ten groups of equal size, where each group is assigned to an input digit class. The target spike trains have a firing rate of β Hz. The spike trains are active during the first 50 milliseconds of image spike train presentation. Table 7.1 gives the neuron model and learning parameters used during our experiments. We use a subset of the MNIST dataset consisting of 60000 images, with 42000 used for training and 18000 used for testing. The training set is presented to the network six times and weights are tuned with the learning rule given in Equation (7.5). After training, the network's prediction is given by the digit group that fires the most spikes.

Table 7.1. The neuron model and learning parameters for Izhikevich-based SNN digit recognition

Neuron parameters		Learning parameters	
Parameter	Value	Parameter	Value
a	0.02	μ	0.0001
b	-0.1	ε	10 ms
c	-55	β	300 Hz
d	6		

Even though the input digits are represented using Poisson-distributed spike trains, the weight values shown in Fig. 7.11 demonstrate that the trained synaptic weights form digit-specific representations of the inputs. For example, it can be seen that the weights for the digit group 0 and 3 resemble handwritten 0s and 3s. The confusion matrix of the network's classification is shown in Fig. 7.12. It is noted that the network is able to accurately classify the handwritten digits with the classification accuracy of 89.8%. Several previously published works have used supervised spike-based methods for training SNNs using the MNIST dataset. A restricted boltzmann machine (RBM) was realized using the leaky I&F neurons in [221] and achieved a classification accuracy of 91.9%. In [222], a spiking convolutional network was implemented using the leaky I&F neurons and achieved a 91.3% classification accuracy. In [223], the Izhikevich neurons were used to implement two layers of visual system neurons (V1 and V2) followed by a classification layer. In total, the network architecture consists of over 71,000 neurons with over 130 million synapses and achieved a classification accuracy of 91.6%. In [224], a two-layer network of stimuli-selective leaky I&F neurons were used to make predictions regarding the particular stimulus presented, and achieved a 96.5% classification accuracy. Unfortunately, the state-of-the-art SNN-based MNIST classifiers offer a lower classification accuracy compared to the non-spiking convolutional neural network models [225]. The accuracy might be improved by employing more layers, increasing the amount of abstract features that can be learned, incorporating conductance-based dynamics into the synaptic weights, and employing additional pre-processing layers that perform edge-detecting [224] and orientation-detecting [222, 223].

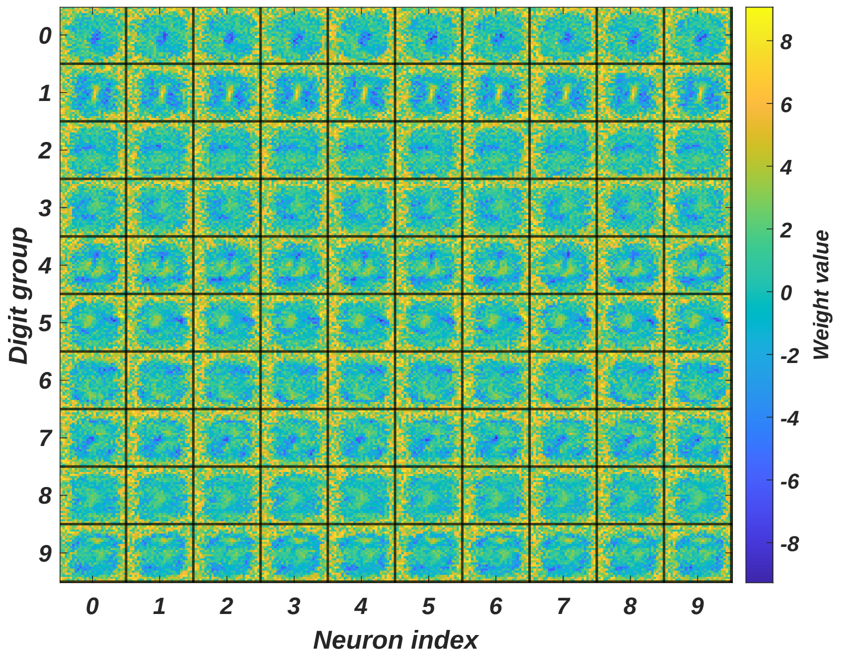


Figure 7.11. Weight matrices of the trained Izhikevich SNN

After obtaining input weights, trained weights can be sent to the network by performing the weighted sums of spike inputs with their respective weight parameter. For the MNIST digit recognition task, for a given time step, the weights from inputs that spike at time t can be accumulated into a single external input I, based on Equation (7.1), thus providing weighted spike train input to the network.

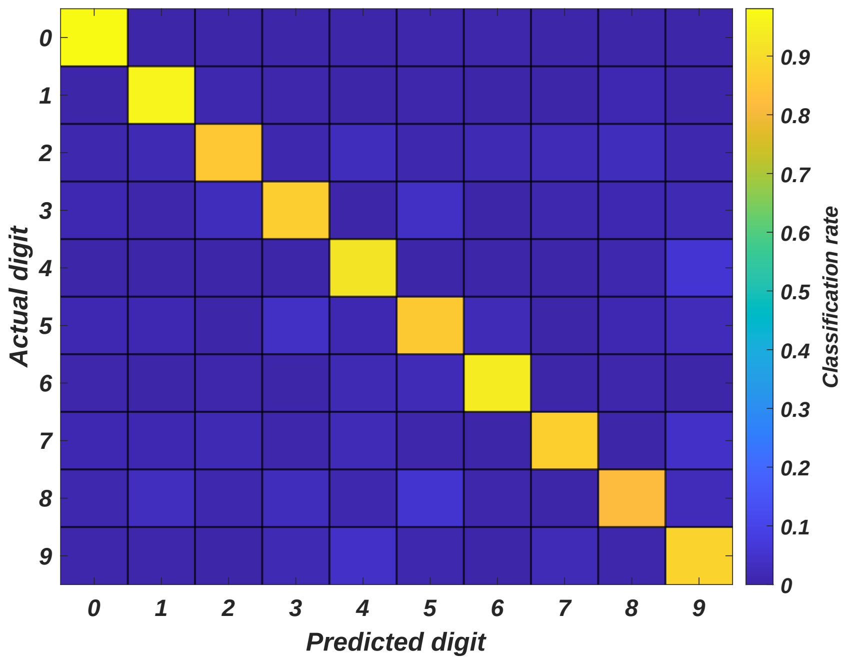


Figure 7.12. Confusion matrix for the SNN-based MNIST digit recognition task

Table 7.2. The characteristics and FPGA implementation results for various SNN realizations

Work	Neuron model	FPGA device	Number of neurons	Synapses per neuron	Time resolution (ms)	Regs. (%)	LUTs (%)	BRAMs (%)	DSP48s (%)	Freq. (MHz)
[201]	IZH	Virtex-5	1024	1024	0.01	32420 (15)	19397 (9)	264 (81.5)	16 (8.3)	133
[202]	IZH	Virtex-4	117	117	1	970 (2)	1598 (3.3)	8 (2.5)	1 (0.19)	84
[203]	IZH	Virtex-6	1440	1440	0.1	48502 (16)	55884 (37)	392 (94)	408 (53)	–
[205]	HH	Kintex-7	500	500	0.00002	2360 (0.57)	5551 (2.7)	–	28 (3.33)	100
[206]	HH	Artix-7	4096	4096	0.0078	25430 (9)	46045 (34)	20 (5.4)	280 (37)	58.8
Ours	IZH	Artix-7	1000	1000	0.0625 – 0.5	11139 (4.1)	12700 (9.49)	14.5 (3.9)	110 (14.8)	92

7.4.4 Comparison

The implementation of a hardware dedicated to emulating SNNs has been of interest in recent years. There are two overarching implementation platforms in use today: (i) programmable devices, including FPGAs and CPUs; and (ii) ASICs. Below, we discuss several state of the art implementations for the two implementation platforms.

Programmable Devices

FPGAs offer extensive flexibility to emulate SNNs with various network topologies and spiking behaviors. Table 7.2 gives the characteristics and FPGA implementation results of various state-of-the-art Izhikevich-based SNN hardware architectures. The SpiNNaker project [207] utilizes general-purpose processors to implement SNNs using a distributed computing approach, where a processing node incorporates 18 ARM968 processors and a combination of processing nodes are used concurrently for simulating SNNs. SNNs are first simulated in software using the PyNN [226] or Nengo [227] frameworks, and then mapped to the SpiNNaker nodes. The distributed computing approach is ideal for simulating large-scale SNNs. While utilizing an array of programmable processors offers a high degree of flexibility, the time resolution scales down to only one ms, which makes it impractical for real-time operation. Moreover, while up to 1000 SNs can be simulated per node, the level of connectivity among neurons is not realistic, with a fan-out of only 100 outputs per neuron. The work in [204] presents NeuroFlow, a scalable distributed computing SNN emulation platform using FPGA-based processors. A NeuroFlow system using six FPGAs can simulate SNNs of up to 600K neurons. The time resolution of one millisecond can be achieved for SNNs of up to 400K neurons. While it is reasonable to compare NeuroFlow and SpiNNaker platforms, as both are targeting large-scale SNNs, our design primarily focuses on a generalized hardware architecture for moderately-sized SNNs with arbitrary interconnect structures and with programmable time resolutions for real-time operation.

Dedicated reconfigurable hardware realization of SNNs has also received interest [201, 202, 203, 204, 205, 206], many of which have focused on accelerating the emulation of large-scale SNNs. The work in [201] presents an FPGA-based implementation of an Izhikevich-based SNN, which supports a fully-connected network of 1024 neurons with over one million synapses. The design uses 16 dedicated synaptic accumulation units, each of which can perform the accumulation of four synapses. The network operations are performed using either single- or double-precision floating-point units and their design operates at 133 MHz with the reported timing resolution of 10 μs. The FPGA implementation of fully-connected SNNs in [202] supports relatively small networks with only 117 neurons, operating at 84 MHz, while achieving a one millisecond time resolution. The work in [203] presents an Izhikevich-based SNN implementation on a Kintex-7 FPGA, capable of simulating over 1000 neurons with a time resolution of 0.1 ms. The work in [205] implements HH-based neurons in a set of tightly integrated cores, supporting up to 500 neurons each, on a Kintex-7 FPGA. Unfortunately, the total memory utilization in terms of number of BRAMs was not reported, which may explain their relatively low register and LUT usage. The work in [206] also presents a core-based approach for implementing HH-based neurons on an Artix-7 FPGA. Rather than supporting a fully-connected network, their work focuses on randomly connected networks, where the connectivity among neurons is generated off-line on a workstation and encoded into an initial seed vector. The seed vector is then used to generate the connectivity matrix on-the-fly instead of storing the full-connectivity on the FPGA. As given in Table 7.2, our SNN hardware architecture is among the most resource efficient designs when implementing similarly sized SNNs, yet being able to realize a range of time resolutions and hence, provides a viable FPGA-based real-time emulation platform. Utilizing a relatively small percentage of on-chip reconfigurable resources along with scalability of the designed hardware architecture makes it suitable for emulating moderately-sized SNNs on a single FPGA.

While the implemented design is more area efficient than the state-of-the-art FPGA-based SNNs, its general-purpose architecture also offers several important design trade-offs. In addition to supporting a "variable" number of neurons (in the order of tens of thousands), it also allows an efficient realization of an "arbitrary" interconnect structure between neurons in the network, which normally requires a relatively large synaptic weight memory. Current FPGA-based SNNs manage the size of the weight memory by only realizing a particular subset of interconnections for a specific application. Another tradeoff is that the designed and implemented SNN architecture supports a wide range of temporal resolutions for added flexibility, while current SNNs designs would support optimized computation of membrane voltages using a fixed time resolution.

ASIC Realizations

ASIC realizations of SNNs include mixed-signal neuromorphic chips [228, 229, 208], which mimic biologically-plausible electrical behavior of neurons [230], and programmable digital SNN implementations [211, 212, 213]. The former designs generally offer lower power consumption and require smaller silicon area [231], whereas the latter designs can leverage the programmability of processors for increased flexibility. To compare our proposed generalized SNN hardware architecture with the state-of-the-art ASIC realizations, our designed SNN architecture is implemented in a standard 32-nm CMOS technology. The CMOS layout of the implemented ASIC is shown in Fig. 7.13, which is estimated to occupy 3.6 mm^2 of silicon area. Post-layout synthesis and simulation results show that the ASIC chip will dissipate 3.6 mW from a 1.16 V supply while operating at 34.7 MHz frequency.

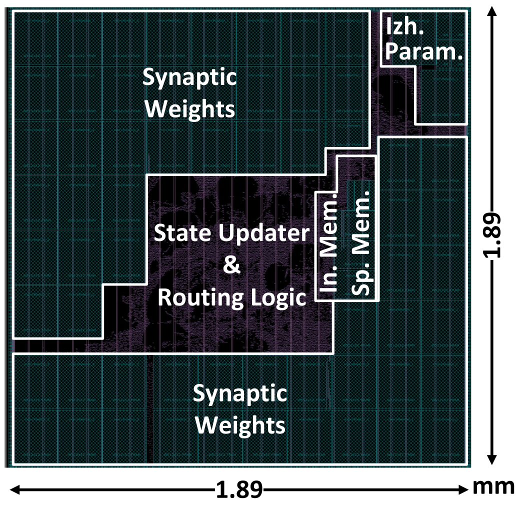


Figure 7.13. Layout of the implemented SNN hardware architecture in a standard 32-nm CMOS process.

Table 7.3 gives the characteristics and ASIC implementation results of various SNN designs. The work in [208] presents a mixed-signal SNN implementation in a standard 180-nm CMOS technology, supporting up to 256 neurons with 16K synapses. The spiking neurons are realized based on the AdExp neuron model [232]. The digital neuron implementation, often referred to as silicon neurons, mimic the behavior of the AdExp neuron with analog circuits.

Table 7.3. The characteristics and ASIC implementation results for various SNN realizations

Work	[208]	[209]	[210]	[211]	[212]	[213]	Ours
Circuit	Mixed-Signal	Mixed-Signal	Digital	Digital	Digital	Digital	Digital
CMOS process (nm)	180	180	45	28	28	14	32
No. spiking dynamics	20[1]	20[1]	3	20	11	6	20
Neurons per core	256	256	256	256	256	1024	256
Synapses per core	16K	128K	64K	64K	64K	1M – 114K	64K
Weight storage	12-bit CAM	Capacitor	1-bit SRAM	3-bit SRAM + 1-bit SRAM	1-bit SRAM	(1-9)-bit SRAM	16-bit SRAM + Izh. Param.
Voltage (V)	1.3 – 1.8	1.8	0.53 – 1.0	0.55 – 1.0	0.7 – 1.05	0.5 – 1.25	1.16
Die area per core (mm^2)	9.6	51.4	4.2	0.086	0.095	0.4	3.6
Power consumption (mW)	0.2 - 2.7	4	0.5	0.4	64	4	3.6
Scaled area per core (mm^2)[2]	0.24	1.28	2.9	0.12	0.13	–	3.6
Scaled power consumption (mW)	0.17–0.85	1.25	0.62–1.88	0.50–1.95	73–312	–	3.6

[1]The AdExp neuron has not yet been proven to exhibit all 20 spiking dynamics.
[2]Scaled to a 32-nm CMOS process utilizing a 1.16V supply voltage following the scheme in [47].

While the AdExp neuron model can exhibit various spiking dynamics, it has not yet been shown to reproduce all of the 20 spiking behaviors modeled by Izhikevich's description [233]. The processing units store the address of the post-synaptic spike in a 10-bit content addressable memory (CAM) and the parameters for the desired dynamics are stored in a 2-bit SRAM. The SNN presented in [209] also implements the AdExp model with silicon neurons in a standard 180-nm CMOS technology. The synaptic weights are stored in on-chip capacitors in the array of synapses. The work in [210] implements a SNN consisting of 256 neurons, based on the I&F neuron model, and 64K synapses in a standard 45-nm CMOS technology. As opposed to the previous approaches in which a neuron's operations are performed using analog circuits, the computations in [210] are performed digitally. The binary synaptic weights are stored in an array of SRAM cells within the cross-bar structure of synapses. The work in [211] presents a digital 256-neuron, 64K-synapse neuromorphic processor. It is implemented in a 28-nm CMOS technology and employs the time-multiplexing technique to emulate the cross-bar structure of synapses. IBM's TrueNorth [212] and Intel's Loihi [213] are also two digital implementations of SNNs. TrueNorth is a multi-core progammable processor implemented in a 28-nm CMOS technology, where each core contains 256 spiking neurons and 64K synapses. Loihi is also a multi-core programmable processor, implemented in a 14-nm CMOS technology, where each core supports 1024 neurons with up to one million synapses.

While commercial SNN processors and distributed computing solutions in [204] and [207] focus on very large-scale SNNs, our architecture and those in [208, 211, 209, 210] are single-chip realizations for a moderate network size. Even though the operating frequency of our design is relatively slow, it is sufficient for real-time emulation of SNNs. For a fair comparison, we have scaled the silicon area and power consumption to a $32-$nm CMOS process utilizing a 1.16-V supply voltage as presented in [47]. We can see that the digital implementations presented in [211, 212, 213] occupy smaller silicon areas than ours, since our design naturally requires more storage elements to define the spiking dynamics on a per-neuron basis. The Izhikevich parameters a and b are stored in 12-bit SRAM units, the c values are stored in 32-bit SRAM units, and the d values are stored in 16-bit SRAM units. While the discussed SNN designs utilize a cross-bar synaptic structure for communication among neurons, the accumulation of synaptic weights in our design is performed in a hybrid sequential-parallel approach with an adjustable degree of parallelism. The synaptic weights are stored in 16-bit SRAM units and for a realistic level of interconnection, the synaptic weight memory may occupy a relatively large silicon area. Note that while our generalized design allows storing the Izhikevich parameters for each individual neuron, to substantially reduce the memory requirement, the Izhikevich parameters a – d can be constrained to specific values for a group or all of the neurons in the employed SNN. Similarly, in [199], seven types of neurons are specified, each with a predefined set of parameter values for $a - d$. Our approach differs in that we aim to provide a robust and flexible platform for emulating diverse SNNs.

While it is shown that our digital SNN architecture consumes acceptable power, recent analog implementations offer a potential alternative candidate. As discussed in Section 7.3, one of the challenges for the efficient realization of spiking neural networks (SNNs) is the process of accumulating synaptic weights. Neuromorphic architectures employ crossbar interconnects to accumulate weighted inputs through current flow. Also, instead of storing weight parameters in SRAM cells, they utilize memristor-based memory, which has a higher density and lower power consumption [234]. They are also suitable passive elements for modeling neuronal synapses with biological learning rules, such as spike-time dependent plasticity [235, 236, 237, 238]. Therefore, in terms of area, power consumption, as well as scalability, the neuromorphic architecture could offer a viable alternative. However, while analog processing provides higher speed than digital processing, unfortunately, noise and inaccuracies accumulate in analog systems. Digital systems can reliably process data and tolerate noise and inaccuracies during neural signal processing. For brain-computer interfaces where the speed of neural signal recording and processing is far less than the speed of digital integrated circuits, low-power realization of spiking neural networks in the digital domain is a feasible alternative candidate.

7.5 Conclusion

This chapter presented an area- and power-efficient scalable reconfigurable hardware architecture for real-time emulation of spiking neural networks (SNNs). We presented the characteristics and implementation results of our designed generalized SNN hardware architecture on both field-programmable gate arrays (FPGAs) and on application-specific integrated circuit (ASIC). Compared to the similarly-sized SNNs implemented on a single FPGA, our design is capable of emulating moderately-sized SNNs in real time while utilizing significantly fewer reconfigurable resources. We also verified the performance of the proposed SNN on the MNIST digit recognition task and showed that it can accurately classify handwritten digits with 89% accuracy. Compared to the state-of-the-art ASIC realizations of SNNs, our design consumes less power utilizing a hybrid parallel-sequential scheme. Moreover, as opposed to the state-of-the-art ASIC realizations, our design's power consumption and real-time processing are not affected by the spiking rate of the emulated SNN. Our design supports various spiking rates based on the number of neurons and allows an adjustable degree of parallelism and time step resolution. The silicon area and low power consumption of the realized generalized SNN hardware architecture make it suitable for single-chip SNN realizations while emulating SNNs of various sizes and interconnect topologies in real-time.

7.6 Publications

The following article was published as an outcome of the research conducted in this chapter: D. Valencia and A. Alimohammad, “A generalized hardware architecture for real-time spiking neural networks,” Neural Computing and Applications, vol. 35, pp. 17821 – 17835, August 2023.

Chapter 8

Power-Efficient In Vivo Brain-Machine Interfaces via Brain-State Estimation

8.1 Introduction

Over the past decade, researchers have studied the activities of individual neurons with respect to their neighboring neurons and their response to various stimuli [239, 240]. Neurons communicate by means of firing electric pulses, called action potentials (APs) or spikes. The electrical activity of neurons can be measured and recorded by multi-electrode arrays (MEAs) in which each intracortical electrode implanted in the motor cortex record cellular electrical activity from a small population of neurons within a few hundred micrometers of the neuron closest to the tip of the electrode [167], with added ambient noise and technical artifacts, such as electrode micro-motion or instrumentation noise. The measured electrical activity inside the gray matter of the brain can be used for muscle control, and also sensory perception, such as seeing and hearing, speech, decision making, and self-control. In a brain-machine interface (BMI) system, spikes are first detected from the background noise by comparing the recorded and filtered voltage waveforms with a threshold, which is commonly estimated as the scaled value of background noise. Neighboring neurons often fire spikes of similar shape and amplitude, however, relative to their distances to an electrode's tip, the shape of spike waveforms may differ among neurons. This fact allows the spiking activity of individual neurons to be separated through the spike sorting process [138]. It has already been verified that robust BMIs can also be implemented without employing spike sorting [141, 241, 242, 39, 11]. In this case, all threshold crossings (TCs) of the recorded and filtered voltage waveforms associated with an electrode (channel) are treated as spikes from one putative neuron. By considering the number of spikes over a given time (spike count) as the feature of interest, the transmission data rate is significantly reduced compared to transmitting the spike waveforms [64]. Three types of signals can be obtained by intracortical recordings: (i) local field potentials (LFPs), which are extracted by low-pass filtering ($<$300 Hz) of the neural activity in the vicinity of an electrode tip; (ii) multi-unit activity (MUA), which is obtained by high-pass filtering of the recorded neural activity and detecting action potentials from the ambient noise; and (iii) single-unit activity (SUA), which are the detected MUA action potentials clustered into different groups associated with putative neurons. Wireless transmission of LFPs requires substantially more data rate, as LFPs are continuous signals whereas SUA and MUA measure the instances of single or an ensemble of spikes and hence, can be represented as discrete events. For brevity, the subsequent references to spikes refers to MUA spikes rather than sorted SUAs. For example, consider a 1000 channel neural recording system operating at 20 kS/s with 12 bits per sample. Transmitting filtered neural signals would require a data rate of 240 Mbps, while transmitting only spike counts, assuming up to one spike per millisecond per recording channel, would reduce the data rate to only one Mbps. The data rate can be even further reduced by spike binning, i.e., transmitting spike counts over a larger time intervals in the order of tens of milliseconds. Assuming a wireless transmission energy of 8.5 pJ/bit [121], the former requires 2 mW of power while the latter requires only 8.5 μW, over 99% reduction.

In synchronous BMIs the user's neural activity is processed within a predefined time frame, while in asynchronous or self-paced BMIs, the user can interact with the BMI at their leisure, which is more convenient in practical applications. When the user is not actively engaged in a BMI task, the power consumption of the in vivo interface can be reduced to extend the operational lifespan of implanted devices. Transitioning the BMI from a "standby" state back to the "active" state requires the BMI to estimate and differentiate between different mental states. In a two state "brain-switch" approach [243, 116, 244], the brain-switch module continuously monitors neural signal features to detect whether the user would like to engage in the underlying BMI activity and hence, only processing neural signals during the "active" state.

Both non-invasive recordings, such as electro-encephalography (EEG) and near-infrared spectroscopy (NIRS), and invasive recordings, such as electro-corticography (ECoG) and intra-cortical MEAs, have been employed for the brain-switch scheme. The non-invasive methods are more relaxed compared to the invasive methods as they do not impose stringent constraints on power consumption. In all the recordings, neural response patterns are used to differentiate between the "standby" and "active" mental states. In some cases, the neural signals are user

Table 8.2. The energy dissipation of three alternative wireless BMI configurations. Configuration I constantly detects and transmits MUAs for all recording channels; Configuration II.a transmits and detects MUAs for a subset of channels for realizing a brain-switch algorithm on an external device; Configuration II.b detects MUAs for a subset of channels and realizes a brain-switch algorithm in vivo. Configurations II.a and II.b enable processing on all channels when the brain-switch algorithm denotes an "active" mental state.

BMI Configuration	**I**	**II.a**	**II.b**
"Active" detection power (mW) / energy (J)	30 / 2592	30 / 108	
"Standby" detection power (mW) / energy (J)	N/A	3 / 248.4	
"Active" / "Standby" transmission energy (J)	0.73/ N/A	0.03 / 0.07	0.03 / 0
Total energy (J)	2622.73	389.50	389.43
Detection energy savings (%)	N/A	86	
Transmission energy savings (%)	N/A	86	96
Total energy savings (%)	N/A	85.14	85.15

Table 8.1. The in vivo BMI states and the in silico signal processing of the designed brain-switch scheme.

BMI states	**User's activities**	**Behavioral example**	**In silico signal processing**
"Standby"	Non-BMI activities	Sleeping	Intention estimation
		Eating	
"Active"	Only BMI activity	Prosthesis control	Neural decoding
	BMI activity + Non-BMI activities	Prosthesis and eating	

modulated, such as the imagined motor movements in the EEG-based studies [245, 246], while in others the response is driven by specific stimuli, such as the event-related desynchronization in ECoG-based studies [243]. For intracortical recordings, changes in both the spike firing rates and the spectral power of the LFPs have been used for mental state estimation [116], however, no practical in vivo realization of the brain-switch scheme has been reported. Moreover, modulation of neural activity unrelated to the BMI application can be ignored to avoid spurious decoding outputs.

Table 8.1 lists the two considered BMI states, the example of the user's activities in each state, and in silico signal processing operation. The "standby" state is related to when the user activity is not related to the underlying BMI application, such as sleeping or for example, if the BMI task is cursor control but the user is eating. The "active" state is related to the user-modulated neural activity for either only the underlying BMI application or also consisting of unrelated activities, such as the user controlling a robotic prosthesis for eating. During the "standby" mode, the in vivo BMI monitors a subset of recording channels and the in silico signal processing executes the intention-estimation algorithm. During the "active" mode, the BMI processes all recording channels using an in silico neural decoding. While the transition from the "standby" to "active" mode is estimated using the designed brain-switch scheme, the transition from "active" to "standby" can be estimated using the outputs of the neural decoder.

For example, consider a 1000-channel implanted wireless BMI used to control assistive devices with three possible configurations. In Configuration I, the device is continuously detecting and transmitting the binned spike counts from all channels, analogous to a synchronous BMI system, which processes all recorded signals within a predefined time frame. Configurations II.a and II.b both realize an asynchronous BMI, in which a relatively small subset of channels are used to realize the designed brain-switch scheme. When the brain-switch algorithm detects BMI activities, the BMI transitions from "standby" to "active" state and all channels are enabled for spike detection and processing. Configuration II.a realizes the brain-switch algorithm in silico on a computer or a portable computational device by processing the received neural signals, however, Configuration II.b realizes the brain-switch algorithm in vivo, which reduces the power consumption of the wireless transmission during the "standby" mode, at the cost of the power consumption and silicon area of the in vivo brain-switch circuitry.

Table 8.2 gives the energy dissipation and savings of the three alternative configurations. Assuming that the power consumption of the analog front end (AFE) for amplifying and digitizing the recorded neural signals are the same for both configurations, the power consumption of the in vivo spike detection and wireless transmission are compared. Employing an analog-to-digital converter with 12 – 16 bits resolution and sampling between 10 kHz – 30 kHz [247, 248, 249, 250], the nominal power consumption for the AFE circuitry ranges between 0.75

μW and 7.3 μW per channel. Assuming that detecting spikes from 1000 channels requires 0.03 mW of power per channel [251], the power consumption of detecting spikes for the implanted BMI interface is approximately 30 mW, for each of the three configurations. If the in vivo BMI interface transmits the spike counts in one millisecond intervals to represent the brain's neural activities, the data rate is reduced to one Mbps with the transmission energy of 8.5 pJ/bit [121]. For configurations II.a and II.b, let's assume that 10% of the channels are sufficient to detect the user's intention and that they are engaged with the BMI for an average of one hour per day. Energy dissipated during "active"/"standby" BMI operation is given as $P \times \Delta_t \times 3.6\ \mathrm{J/mW}$, where P denotes the power consumption (in mW), and Δ_t denotes the amount of time (in hours) over which the power is consumed. Data transmission energy dissipation during "active"/"standby" BMI operation is given as $\eta \times f_b \times \Delta_t \times 3600\ \mathrm{J/W}$, where η denotes the transmission energy of 8.5 pJ/bit [121], and f_b denotes the output data rate for a 1000-channel system (1000 kbps). It can be noted that Configuration II.b offers the highest potential energy savings. Moreover, it can be seen that for the TC-based BMIs, the main source of power consumption is spike detection. Given that the mean detection power of Configuration I is 30 mW, if an SAFT LS14250 battery [252], which has a nominal capacity of 4.32 Watt-hours, is used, the battery life for spike detection is 144 hours. By applying spike detection to a smaller subset of channels during "standby" periods, the mean detection power of Configurations II.a and II.b are reduced to 16.5 mW, a 45% decrease, extending the battery life to 261 hours, an 81% increase. Note that Table 8.2 does not account for the power consumption of the brain-state estimation circuitry in Configuration II.b. Since the power consumption difference between Configurations II.a and II.b is negligible, we propose to realize Configuration II.a for the BMI system and to estimate the user's intention in silico (i.e., in software running on a personal computing device).

This chapter presents the power-efficient realization of the in vivo interface for asynchronous BMIs using a neural network-based brain-switch scheme. The rest of this chapter is organized as follows. Section 8.2 discusses the signal processing and alternative brain-switch algorithms for detecting the user's mental states. The details of the designed asynchronous spike detection module in hardware is discussed in Section 8.3. The estimated silicon area and power consumption of the spike detection application-specific integrated circuit (ASIC) is also presented. As a proof-of-concept, a system-level realization of the designed asynchronous BMI utilizing the designed brain-switch scheme and the in silico neural decoding is presented in Section 8.4. Finally, Section 8.5 makes some concluding remarks.

8.2 Neural Signals and Brain-Switches

To drastically reduce wireless data transmission between the in vivo implanted circuitry and offline software processing, various compression techniques have been used, from relatively simple methods, such as difference encoding [253], to more complex techniques, such as compressed sensing [254, 255]. Wireless data transmission can be further reduced by transmitting neural information only when the user is intending to engage in a BMI-related action. The data rate can further be reduced by only transmitting the neural information for a subset of channels for estimating when the user wants to engage with a BMI task, as shown in Fig. 8.1.

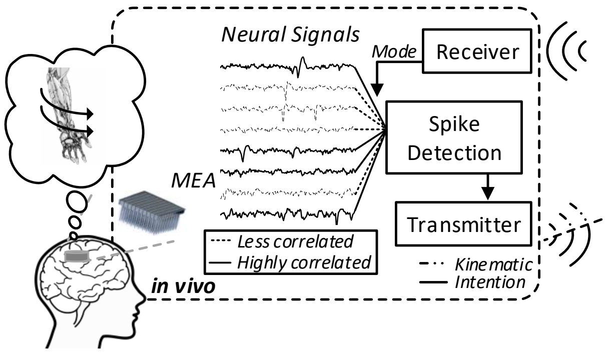


Figure 8.1. The block diagram of the designed intention-aware in vivo BMI.

Implementing a brain-switch effectively involves the detection of a mental state transition from "standby" to "active" modes, which controls the in vivo processing and wireless transmission of neural signals. Three mental state estimation schemes have been previously investigated [244]. The simplest method is a threshold-based approach, in which a specific neural signal or signal feature crossing a threshold is interpreted as a mental state transition. The threshold

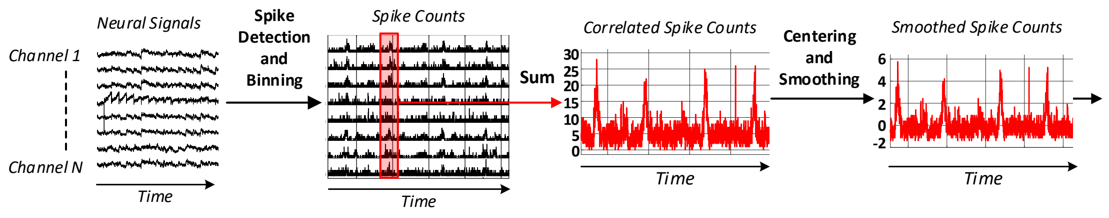


Figure 8.3. Generating an estimate of the correlated channel's spike density. Binned spike counts are added per time step for a subset of correlated channels, followed by centering and smoothing.

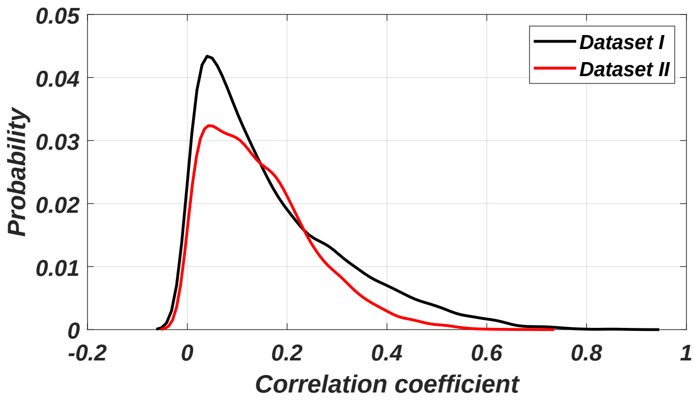


Figure 8.2. The probability distributions of correlation coefficients between recording channels and the state transitions over the trials in Datasets I and II.

value is derived from the training data. The second approach is a classifier-based thresholding technique, in which the neural signals or features are passed to a classification algorithm to produce a continuous output signal, such as the probability of transition, which is then compared to a pre-defined threshold, as in the first scheme. Finally, in a classifier-only-based approach, the output of a classification algorithm is used directly. The output of the classifier may indicate a discrete mental state, such as "idle", "planning", and "movement", which can be used to directly enable or disable further processing based on the predicted mental state. All brain-switch algorithms assume that neural signals exhibit different behavior during mental state transitions. For example, EEG-based brain-switches may detect an increase in the 1 Hz – 4 Hz frequency band power [245, 246], while ECoG-based brain-switches may detect a change in the power of the mu and beta frequency bands for motor execution/imagery [243]. For spike-based signals, the number of SUA spikes during a time window [256] as well as changes in the firing rates [116] have been considered as the features for the brain-switch algorithms. Therefore, we assume that a subset of channels will exhibit enough variations in MUAs that will allow a classification algorithm to relatively reliably detect mental state transitions. We employ two publicly available neural datasets. Datasets I (I140703) and II (L101210) [152], which contain raw recordings from the motor cortex (M1 region) of two Rhesus macaque monkeys employing a 96-channel Utah Array. The recordings were sampled at 30 KHz and then downsampled to 10 KHz. The monkeys performed a cued reach and grasp task to displace an object. During the experiments, the monkeys were presented with a series of cues to indicate that a trial was beginning and to specify one of four combinations of grip and displacement forces to use. After the cues were given, the monkeys released a switch on the table to reach for the object. Upon a successful reach, grasp, and displacement, the monkeys were rewarded. For the active brain-state estimation, the switch release time is considered as the ground-truth intention time. To identify which channels exhibit greater changes during a state transition, a correlation analysis is performed. More specifically, following spike detection, the correlation between the spike counts and a known transition response (i.e., a signal that transitions from zero to one at a pre-defined time) is calculated. Then, the correlation between the spike counts for all channels and the transition from a go cue and the switch release event are calculated. Dataset I and II consist of 153 and 149 trials, respectively. A training subset consisting of the first 70% of the trials are used to analyze the correlation coefficients between the spike counts for each recording channel and the state transition. The mean correlation coefficient over Dataset I and II were 0.15 ($\sigma = 0.13$) and 0.15 ($\sigma = 0.11$), respectively. The probability distributions of correlation coefficients between recording channels and the state transitions over the trials in Datasets I and II are shown in Fig. 8.2. It can be seen that correlation values equal to or greater than 0.30 and 0.25 for Datasets I and II, respectively, constitute the top 15% percent and are considered highly correlated to the "active" brain-state.

After calculating the correlation coefficients for each channel and trial, the mean correlation for the N highest correlated subset of channels is calculated. Since the state transition event is of interest, the response of the correlated channels during this transition time is considered. To represent the activity of the correlated channels, the sum of the correlated channels is computed per time step. The centering via Z-scoring is applied followed by the Gaussian kernel smoothing to estimate the spike density function [257]. The described process is shown in Fig. 8.3, where the correlated spike counts refer to the accumulated spike counts of only the *N* highest correlated channels and the smoothed spike counts represents the estimated spike density function after applying the centering and smoothing.

To reduce the power consumption of the in vivo signal processing, it is preferable to enable as few correlated channels as possible. As shown in Fig. 8.4(a), since the centering and smoothing operations perform a normalization of the signal, the response of a smaller number of correlated channels will be similar to that of a higher number of correlated channels. As shown in Fig. 8.4(a), the behavior of the estimated spike density for the correlated channels begins to deviate from the baseline during the switch release event. Note that the activity of the correlated channels during the BMI and non-BMI operation may impact the decoding performance, however, the brain-switch algorithm is only operating during the non-BMI operations. Fig. 8.4(b) shows the boxplot of the estimated spike density for different number of correlated channels. While the mean variability is similar over different number of channels, it can be seen that the overall variability spread for eight channels is smaller than that of a higher channel counts. Thus, eight highest correlated channels are selected for the brain-switch algorithm. For Dataset I, the mean correlation of the eight highest correlated channels is 0.38 (std. 0.017) for the BMI period, and 0.25 (std. 0.016) during the non-BMI period, a 34% decrease. For Dataset II, the mean correlation is 0.28 (std. 0.02) for the BMI period, and 0.20 (std. 0.005) during the non-BMI period, a 28% decrease.

8.2.1 Brain-switch Algorithms

For estimating the user's intention, three alternative models are considered: the hidden Markov model (HMM) [258], a feed-forward neural network (FNN), and a recurrent neural

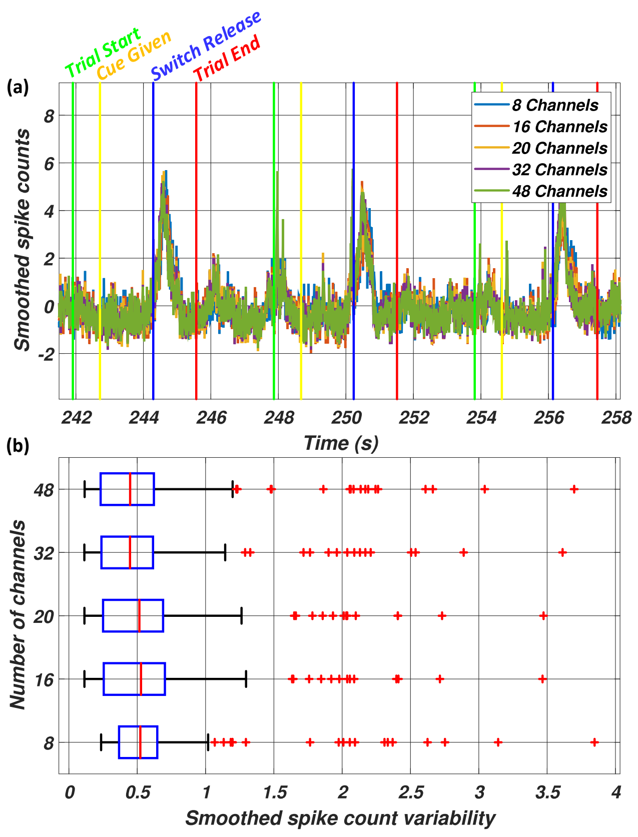


Figure 8.4. (a) A sample from Dataset I of the smoothed spike counts during the trials and (b) the smoothed spike count variability during the non-BMI periods over both datasets.

network (RNN). The HMM is a stochastic model that estimates the probability of being in a particular state based on the input data and the current state. The mental state can be modeled as a two-state HMM, where one is for the "standby" state while the other indicates that the user is in an "active" state. The forward algorithm [258] then calculates the likelihood of being in a Markov state X_i after a sequence of n observations $O_0(t_0 = 0), O_1(t_1 = 1), ..., O_n(t_n = n)$ as $P_i(t_n) = \sum_{j=1}^{M} \left[P_j(t_{n-1})A_{ij}B_{ij}\right]$, where M denotes the total number of Markov states, i and j denote the indices of Markov states, and $P_i(t_n)$ denotes the probability of being in state P_i at time $t = t_n$. We also consider machine learning (ML)-based methods using feed-forward neural networks (FNNs) and recurrent neural networks (RNNs) for brain-state estimation. Both FNNs and RNNs use artificial neurons that each compute an output $y = f(z)$, where z denotes an accumulated weighted input and $f(\cdot)$ denotes a non-linear activation function. The weighted input z is computed by multiplying the inputs with a weight matrix $\mathbf{W_x}$ that is obtained during training. While FNNs process data in a forward direction from the input layer to the output layer, RNNs employ a self-recurrent connection which attempts to learn from temporal features in the data. As a result, RNNs employ additional weight matrices $\mathbf{W_R}$, denoting the self-recurrent weights, in addition to the input weight matrix $\mathbf{W_x}$. Both the FNN and RNNs use the rectified linear unit (ReLU) activation function at the output of the artificial neurons. The RNN also employs the ReLU function in its recurrent connections. The employed FNN and RNN models for mental state estimation have eight artificial neurons in a single hidden layer, with one unit at the output. To quantify the accuracy of the employed brain-state estimation schemes, the F-score and the Pearson CC metric are assessed. The CC is given as:

$$\text{CC} = \frac{\sum_i (t_i - \bar{t})(\hat{t}_i - \bar{\hat{t}})}{\sqrt{\sum_i (t_i - \bar{t})}\sqrt{\sum_i (\hat{t}_i - \bar{\hat{t}})}},$$

where t and $\hat{t}$ denote the actual and estimated intention time, respectively, $\bar{t}$ and $\bar{\hat{t}}$ denote the mean of the actual and estimated intention time, accumulated over i estimations, are assessed. The F-score and the CC metrics can be interpreted as a measure of the accuracy of detecting the intention and as the precision of detection, respectively. Note that in the context of BMIs, an algorithm with a higher F-Score and an acceptable CC is preferable over an algorithm with a relatively low F-Score and a high CC. A highly precise intention estimator, i.e., high CC, will be of no use when the intention is not detected. Previous studies among BCI users have shown that the overall system's reliability is far more important than it's speed [259].

For training the intention detectors, two-second snippets of smoothed spike counts over various animal reaches are extracted from the Datasets I and II. For the intention-positive snippets, the switch release time was aligned at one second. The outputs of the models were a zero, indicating no intention, or one, indicating the BMI-related intention. The FNN has no temporal state and the two-second reach data was divided into sequences of three consecutive, non-overlapping smoothed spike counts. The RNN accepts one input per time step per training sample, while the FNN accepts three inputs per training sample. Both the RNN and FNN models were trained using Python's Tensorflow framework and to avoid overfitting to training data, early stopping was used to monitor the mean squared error metric and stop training when the validation error was no longer reduced. To evaluate the performance of the models after training, a 0.5 threshold was applied to the ReLU outputs to generate the predicted state, e.g., an output greater than or equal to 0.5 was interpreted as an "active" state and as a "standby" state otherwise. Table 8.3 gives the performance of the three developed brain-state estimation schemes over Datasets I and II. It is shown that both of the ML-based models outperform the HMM in detecting the state transitions. The performance of the RNN is consistent over both datasets while the FNN drops in performance has a considerable drop in performance over Dataset I. Thus, for estimating the brain-state, the RNN model is employed.

Table 8.3. The performance of alternative brain-state estimation methods over two datasets.

Dataset	**Method**	**F-Score (std.)**	**CC (std.)**
I	HMM	0.47 (0.08)	0.83 (0.09)
	FNN	0.72 (0.31)	0.93 (0.04)
	RNN	0.83 (0.04)	0.89 (0.05)
II	HMM	0.40 (0.02)	0.83 (0.12)
	FNN	0.92 (0.02)	0.98 (0.02)
	RNN	0.95 (0.03)	0.91 (0.05)

8.3 Hardware Implementation of the Designed Intention-Aware BMI

To detect the spikes of the individual neurons near the tip of an electrode, the contribution of the LFPs is first removed using a band-pass filter. A more computationally-efficient approach to remove the low-frequency components of the signal is employed by subtracting the moving average of the signal, computed over a reasonably short timing window, i.e., $\tilde{x}[n] = x[n] - \bar{x}[n]$, where $x[n]$ denotes the neural signal, k denotes the length of the timing window and

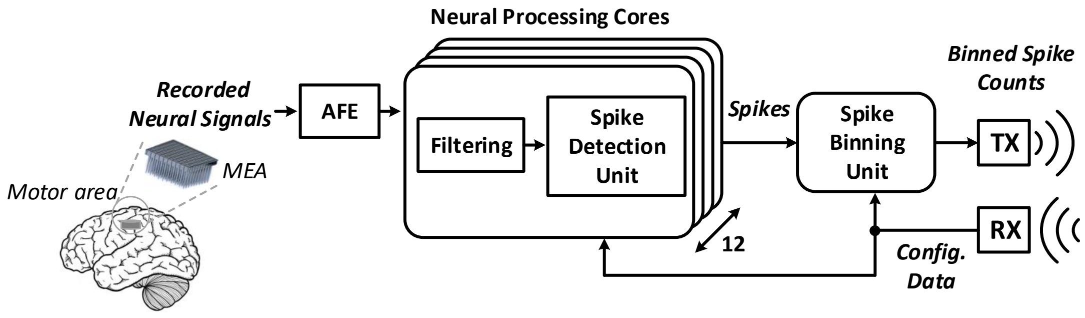


Figure 8.6. The top-level block diagram of the designed intention-aware in vivo BMI.

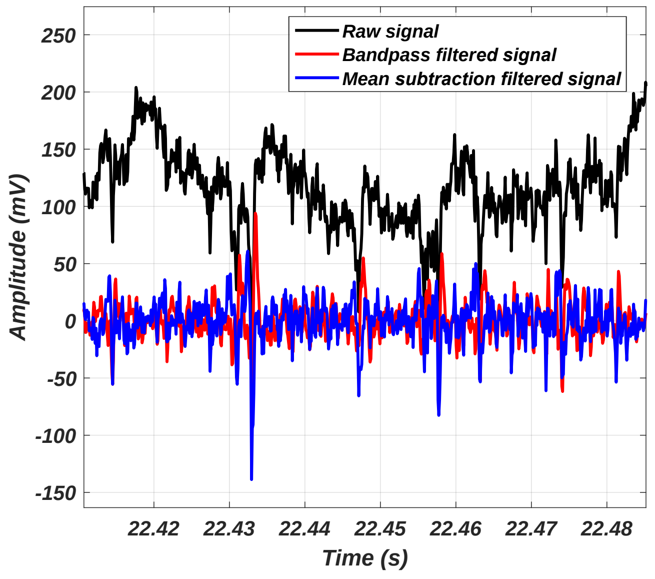


Figure 8.5. The raw neural signal, the bandpass filtered signal, and the mean subtraction filter outputs.

$\bar{x}[n] = \frac{1}{k}\sum_{n}^{n-k} x[n]$ [21]. The mean subtraction filter does not require multiplications, which is a significant saving as in vivo spike detection is a continuous operation. It was found that removing the mean of the neural signal over a relatively small timing window of 800 μs is sufficient for removing the LFPs. With a sampling rate of $f_s = 10$ kHz, this corresponds to computing the average of eight samples, which requires eight additions and one right shift operation per input sample per channel. Fig. 8.5 shows an example of the conventional bandpass filter and the filtered recording using the mean subtraction filter. It can be seen that the employed filter preserves the spikes in the bandpass filtered signal while the low-frequency oscillations are eliminated.

After applying the mean subtraction filter, spikes are first emphasized by applying the absolute value operation to the filtered signal. Then spikes are detected by comparing the emphasized signal with a threshold value, which is computed dynamically as a scaled value of the estimated background noise. The noise is estimated as the mean of the emphasized signal over a one-second time window. The accuracy of the detection unit was measured using the F-score metric as $F = 2T_P/(2T_P + F_N + F_P)$ [260], which accounts for true positives (T_P), false positives (F_P), and false negatives (F_N) spikes over the widely used Wave_Clus datasets [37]. Compared to the energy-based methods, such as the non-linear energy operator (NEO) and the root-mean-square (RMS) noise estimation with an F-Score of 0.94 [113], the employed method shows a comparable F-Score of 0.92 while being less computationally-complex.

The top-level block diagram of the designed asynchronous in vivo BMI is shown in Fig. 8.6. The raw neural signals recorded by the MEA are amplified, digitized by the analog front end (AFE), and is passed to twelve *Neural Processing Cores* (NPCs), where each contains a *Filtering* unit to remove the LFPs and a *Spike Detection Unit* to detect spikes in the form of TCs. Note that the realization of the AFE is beyond the scope of this paper and hence, our subsequent power and energy estimations do not account for the AFE. The spikes are then passed to the *Spike Binning Unit*, which counts the number spikes over periods of 10 milliseconds. Each NPC processes eight channels of interleaved recorded data and the operating frequency of the system is $f = 8 \times f_s = 80$ KHz. The NPCs interleaves data processing of eight recording channels by employing the C-Slow architectural transformation [261]. The eight highest correlated channels are configured through the configuration ports, which will enable or disable processing based on the corresponding NPCs' correlation settings. The correlation settings for each channel are obtained on a workstation prior to the system configuration using labeled data collected during the training process. Each NPC receives an 8-bit configuration word indicating which of its eight interleaved data channels are highly correlated to the user's intention.

To remove LFP components from the recorded neural signals, the moving average of the signal over the previous eight values is subtracted from the current value. To account for

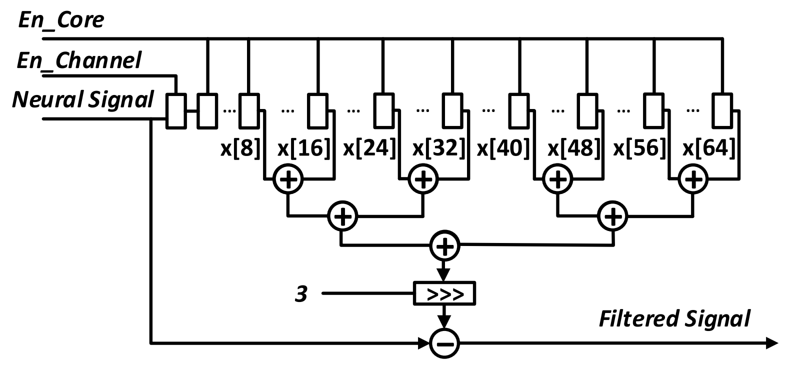


Figure 8.7. The datapath of the LFP-removal moving average filter.

the interleaved data, the outputs of every eighth register of a 64-word shift register are used to compute the mean of the current data channel using an adder tree and an arithmetic right shift, as shown in Fig. 8.7. To reduce the switching activity of the uncorrelated channels, the propagation of new values are disabled by de-asserting the enable signal *EN_Channel*. If an NPC's 8-bit configuration word is equal to zero, the input register enable signal *EN_Core* is de-asserted and no signals will propagate through the delay elements.

The datapath of the *Spike Detection Unit*, as shown in Fig. 8.8, consists of an absolute value unit, a correlation shift register *Corr. Shift Reg*, an accumulation and threshold memory *Acc./Thr. Mem.*, which consists of two memory modules, and a control unit *CTRL*. The *Corr. Shift Reg* is used to enable or disable the accumulation of rectified filtered signals. The *Corr. Shift Reg* is initialized after the training phase to determine the correlation settings for each of the channels processed by the NPCs. The accumulation register memory *ARM* stores the accumulated rectified signal for the eight NPC channels. The threshold register memory *TRM* stores the threshold values, updated approximately every second ($2^{\lfloor \log_2 f_s \rfloor} \times \frac{1}{f_s} = 0.8192$ seconds, where f_s denotes the sampling rate). The *CTRL* unit enables the *TRM* to store the corresponding value from the *ARM* after shifting it 10 bits to the right to compute the scaled threshold value. To detect spikes, the rectified filtered signal is compared to the current channel's threshold value. The comparator output is gated with two signals, the *Corr. Shift Reg* output, which is used to disable transient spikes from uncorrelated channels, and a channel refractory *Chn. Ref* signal, which is used to prevent considering multiple output spikes due to the same threshold crossing event.

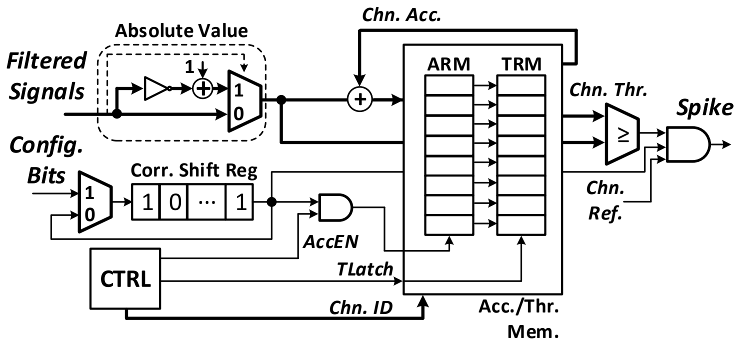


Figure 8.8. The datapath of the spike detection unit.

The spike detection unit can operate in two modes. The first mode of operation is during the system initialization in which a threshold must be derived for each channel processed by the NPC. During this period, a master enable bit is used to allow the processing of uncorrelated channels. This is also the case for when the off-chip intention estimator detects that the user intends to perform a BMI-related activity, where the neural recordings of all channels are required. In the second mode of operation, the master enable bit is de-asserted and only the highest correlated channels are processed.

The spikes generated by the *Spike Detection Unit* are then processed by the *Spike Binning Unit* shown in Fig. 8.9, which accumulates the spikes detected by each of the NPCs in the core count memory *CCM*. The *CCM* stores eight 48-bit words of data, one for each channel of an NPC. Each word is partitioned into twelve 4-bit sections to store the spike counts produced by the 12 NPCs. To compute the accumulated spike counts of the correlated channels, the output *Core Counts* from the *CCM* are added with the masked summing units *MSUs*. The *Bitmask Memory* stores correlated channel information in twelve-bit words, which are used within the *MSUs* to mask the spike counts of uncorrelated channels. The outputs of the MSUs are then accumulated into the correlation sum register *CSR*.

The synthesized ASIC layouts of Configurations I and II.a, implemented in a standard 180-nm CMOS process, is shown in Fig. 8.10. Synthesis was performed using Synopsys

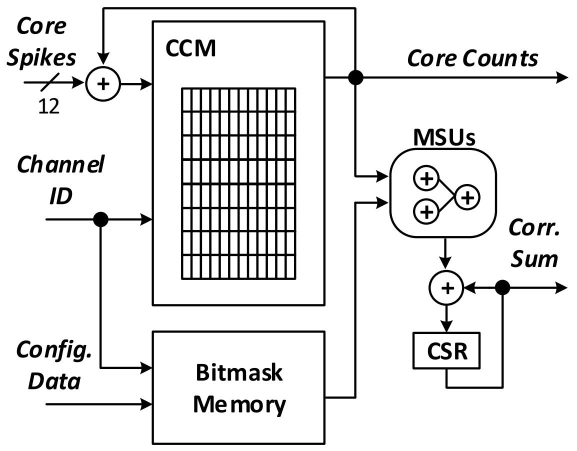


Figure 8.9. The datapath of the spike binning unit.

Design Compiler and the place-and-route was done using Cadence Innovus. The designs were synthesized to support a 96-electrode Utah Array by implementing 12 NPCs, each processing eight channels. To optimize the area and power consumption of the ASIC, we tested the system-level accuracy of the system over various numerical resolutions. Input data, and subsequent internal digital signals, are represented using the fixed-point (WI.WF) number format, where WI and WF denote the number of integer and fraction bits. In our design, data is represented using one integer bit and F fraction bits, where F was between 11 and 2 bits. We found that the system-level accuracy dropped significantly for values of F less than 7. Table 8.4 gives the ASIC characteristics and implementation results for various configurations of (WI.WF). The power consumption of the 96-channel intention-aware circuitry and the power per active channel is approximately 59 μW and 0.63 μW, respectively. The power was estimated by simulating the synthesized and routed design, considering the switching activity of all nodes in the design.

Table 8.5 gives the ASIC characteristics and implementation results for various spike detection systems. For a fair comparison, the implementation results have been scaled to a 180-nm CMOS process with a 1.8 V supply voltage, as described in [47]. Also, we compare the area and power consumption of the spike detection circuitry, when able, for a fair comparison. The design in [45] is a 16-channel BMI with a window-discriminator for spike detection. Window discriminators involve dual threshold values and detect spikes when there is a crossing of both

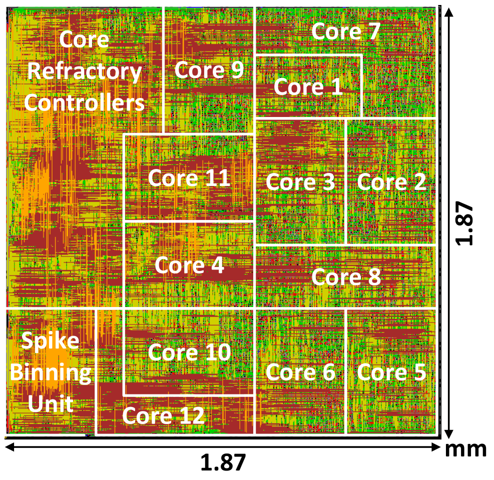


Figure 8.10. The ASIC layout of the designed intention-aware BMI system employing a 96-electrode Utah Array with 12 spike detection cores.

Table 8.5. The ASIC characteristics and implementation results of the state-of-the-art spike detection circuits.

Work	Ours	[45]	[18]	[46]	[262]
Technology (nm)	180	180	180	65	130
Supply voltage (V)	1.8	–	1.8	0.8	1.2
Number of channels	96	16	1	64	16
Normalized area per channel (mm^2)†	0.03	–	0.03	0.05	1.21
Normalized power per channel (μW)†	0.63	4	1.5	4.6	210
Adaptive threshold	Y	N	Y	N	Y
Intention-aware	Y	N	N	N	N
Scaled active / "standby" power consumption (mW)‡ †	0.124 / 0.067 *	0.4	0.15	0.46	21
Scaled energy dissipation (J)$^{\diamond}$ †	0.611	3.04	1.14	3.496	159.6
Relative increased energy dissipation	–	4.97×	1.86×	5.72×	261.21×

† Normalized to a 180-nm CMOS Process with a 1.8 V supply voltage, as described in [47].
‡ Assuming 100 channels and using 10 channels for brain-state estimation.
⋆ Accounting for the power consumption of the brain-state estimation control circuitry.
◇ Assuming 30 minutes of real-time operation over a two hour BMI session.

Table 8.4. The ASIC characteristics and implementation results of the intention-aware spike detection circuitry over various numerical formats

Circuit metric	(WI.WF)			
	(1.11)	(1.9)	(1.7)	(1.5)
Core area (mm^2)	4.49	4.00	3.50	3.01
Intention-aware circuitry power (μW)	85.8	71.9	59.6	49.5
Power per active channel (μW)	0.86	0.66	0.63	0.43
Mean system accuracy (CC)	0.81	0.80	0.80	0.75

the upper and lower threshold values. In [18], an analog implementation of a NEO-based spike detection unit is presented. In [46], the authors present a 64-channel neural signal acquisition system that detects spikes using the NEO-based pre-emphasis and a constant threshold. In [19], a 64-channel neural signal acquisition system is presented. Spikes are detected by an adaptive dual-threshold window comparator. Also, their system supports extracting spike amplitude-related features, such as the time between threshold crossing and negative peak, the time between the negative and positive peak, and the time between the positive peak and the return to baseline. In [262], the authors present a 16-channel exponential component and polynomial component (EC-PC) spike detection system, which involves representing the neural signals in the Hilbert transformed space and predicting occurrences of spikes by estimating the probability that a data point is part of an action potential. The implementation results in Table 8.5 shows that our ASIC design consumes the least power per channel, primarily due to the fact that in an intention-aware BMI system, it is only necessary to detect spikes of a relatively small number of highly correlated channels for the majority of the time. As given in Table 8.5, assuming that 10% of the channels are sufficient for providing adequate brain-state estimation, and assuming one hour of BMI activity in a day, it is clear that our design offers significantly less energy dissipation, 1.8$\times$ less power compared to the state of the art.

While it is shown that the proposed asynchronous system effectively reduces the power consumption of the in vivo interface, system-level power saving schemes may also be applicable for further power reduction. For example, in a BMI system that a user primarily interacts by controlling a cursor on a computer screen, if eye-tracking cameras are used, the BMI system can power down the in vivo recording circuitry should the user not be paying attention to the screen or if the user is asleep. Note that the results given in Table 8.5 assume that the system only estimates the mental states during the "standby" BMI mode and transitions to the "active" state for neural decoding if the BMI-related activity is detected. While the system employs the brain-switch estimation during the "standby" state, which may involve non-BMI activities, as demonstrated in the next section, the false brain-switch estimation has a negligible impact on the overall decoding performance.

8.4 Intention-Aware Decoding

We integrated our designed intention-aware detection unit along with an ML-based neural decoder. During an initial training phase, neural recordings and their associated labeled events are processed to generate training data for the ML-based brain-state estimation and neural decoding algorithms. The labeled training data is then used to determine which channels are highly correlated to the BMI intention and to power-down channels that are not highly-correlated, reducing the power required to operate in vivo spike detection when the user is not actively engaged in the BMI activity. Fig. 8.11 shows the system-level block diagram of the designed wireless BMI system for a motor-cortex neural decoding application. Neural activity is detected and accumulated in 10 ms bins per recording channel. During the user's "active" mental state, multi-unit spikes are detected on each recording channel, and during the "standby" state, only the eight highest correlated channels are processed to detect the user's mental state. The designed system employs two ML-based models in silico. One is the RNN discussed in Section 8.2 and the second is a temporal-convolutional network (TCN) [176] used as a kinematic decoder to map sequences of binned spike counts onto the intended object displacement velocity. When the RNN identifies an "active" mental state, the TCN is enabled for two seconds to decode the user's kinematics and the RNN is disabled and it's internal memory is cleared.

To evaluate the performance of the neural decoding, the datasets described in Section 8.2 are used. During the off-line training phase, the velocity of the displaced object is calculated based on the object's displacement and is smoothed using a moving average filter to reduce instrumentation noise. To generate training data, spikes are counted over 10 ms bins, from the time of switch release to the time that a trial ends. The spike counts are Z-scored and smoothed with a Gaussian kernel before being passed to the decoding model. The training data for Dataset I and II contain 138 and 98 trials, respectively, aligned to the switch release time. Trials that were

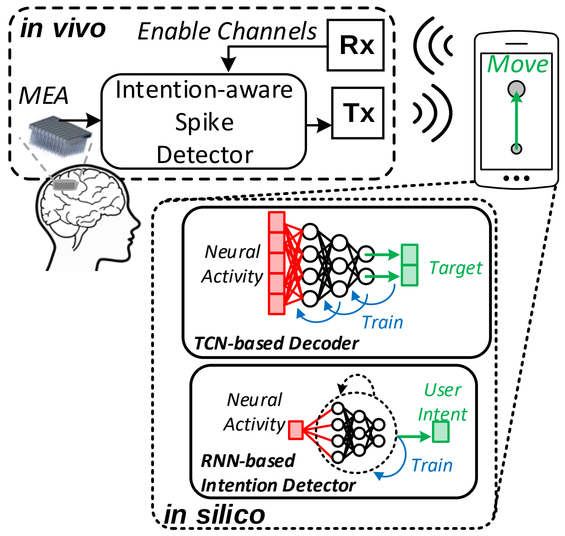


Figure 8.11. The system-level block diagram of our designed wireless BMI system for neural decoding applications.

shorter in time were padded with zeros to match the data lengths during training, however, during BMI operation, both the RNN and TCN models produce an output for intention and kinematic estimations, respectively, per input time step.

Similar to the ML-based intention estimator, the TCN-based decoder was trained using the Tensorflow framework, the RMSprop optimizer, and the root mean square error loss metric. For evaluating regression performance, the CC metric was used. The model was trained for up to 300 epochs and to reduce overfitting, we employed early stopping when the CC no longer increased. The designed TCN-based decoder achieves a mean CC of 0.8 (std. 0.06) over Datasets I and II. Sample outputs of the TCN-based decoder and the estimated intention times from the RNN-based intention estimator over Datasets I and II are shown in Fig. 8.12. For reference, the output of a basic decoder which continuously decodes the neural signals is shown in Fig. 8.12. While the basic decoder will generate spurious outputs among trials, the intention-aware decoder will only generate outputs when the brain-switch algorithm enables the TCN-based decoder. Over the test set shown in Fig. 8.12, the F-score of the RNN-based intention detector was 0.70 and 0.78 over Dataset I and II, respectively. It was found that the mean absolute error of the intention-aware decoder is 0.21 and 0.35 over Datasets I and II, respectively. It was also found that the RNN-based intention detector is more likely to produce a false positive rather than a false negative. However, since the TCN-based decoding model is relatively accurate, the overall performance of the system is not considerably degraded. The primary goal of the proposed design is to enable power savings by decoding only during the "active" mental state, which will inevitably reduce spurious outputs. For Dataset I and II, the error of the velocity decoding is considered as the mean absolute error. It was found that the mean absolute error of the basic velocity decoder is 0.26 and 0.51 over Datasets I and II, respectively. The intention-aware decoder thus has 18.8% and 35% smaller errors than the basic velocity decoder.

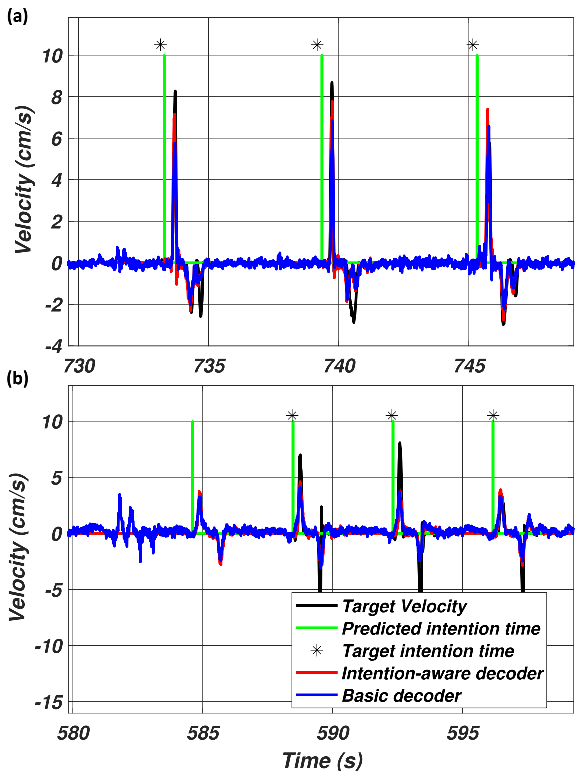


Figure 8.12. The outputs of the basic and intention-aware TCN-based decoders and the estimated intention times using the RNN-based intention estimator over the testing subset of (a) Dataset I and (b) Dataset II.

To evaluate the performance of the BMI system in presence of spiking errors, the spiking error rate (SER) metric [131], which represents errors that can occur in the process of detecting spikes and the wireless transmission of spike counts, was used. To simulate SER, spikes were first detected using the designed detection unit and binned into one millisecond intervals. Due to a neuron's refractory period imposed by the spike detection units, it is assumed that in each one millisecond bin at most one spike can be detected per electrode. To inject errors into the bins, bit-flips were applied to the bins. For example, for an SER of 10^{-2}, there is a 1% chance that a spike or non-spike will be considered as its counterpart. Fig. 8.13 shows the performance degradation of the TCN-based decoder over increasing values of SER. It can be seen that the model offers relatively stable performance for up to about 6% error. The robustness of the model can be further improved by increasing the numerical resolution of signals (i.e., reducing quantization noise during detection), or accounting for spiking errors when the training the model itself, for example, adding error into the spike counts.

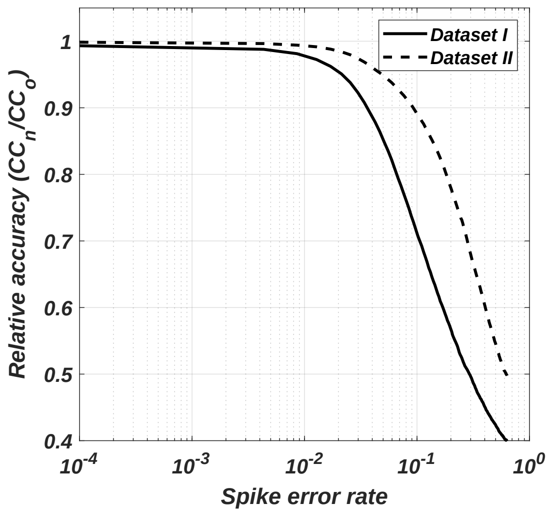


Figure 8.13. The relative accuracy of the TCN-based decoder over increasing values of SER.

8.5 Conclusion

The in vivo spike detection has become increasingly challenging when employing high-density multi-electrode arrays in the order of thousands of recording channels. This chapter demonstrated that an intention-aware brain-machine interface (BMI) system can drastically reduce the power consumption of the in vivo spike detection. It was shown that the machine learning (ML)-based algorithms can be used effectively to estimate the user's intention from a relatively small subset of highly-correlated recording channels, which allows disabling the detection circuitry of the remaining uncorrelated channels. Moreover, since the user is mainly not engaged in the BMI activity throughout the day, the impact of the intention-aware BMI is even more effective. The design and implementation of a 96-channel spike detection unit in a standard 180-nm CMOS process was estimated to occupy 0.03 mm^2 of silicon area and consumes 0.63 μW of power per channel while operating at 80 kHz. The designed in vivo BMI system was used for neural decoding over two neural recordings. It was shown that the ML-based algorithms can reliably detect the user's intention in the presence of up to 6% spiking errors. Additionally, it was shown that compared to the state-of-the-art BMI systems, incorporating intention awareness reduced the total energy consumption by over 1.8$\times$.

8.6 Publications

The following article was published as an outcome of the research conducted in this chapter: D. Valencia, G. Leone, N. Keller, P. P. Mercier, and A. Alimohammad, "Power-efficient in vivo brain-machine interfaces via brain-state estimation," Journal of Neural Engineering, vol. 20, 016032.

Chapter 9

An Efficient Brain-Switch for Asynchronous Brain-Computer Interfaces

9.1 Introduction

Over the past decade, researchers have investigated the activities of individual neurons concerning their neighboring neurons and their response to various stimuli [239, 240]. Neurons communicate by firing electrical pulses, called action potentials (APs) or spikes. The electrical activity of neurons can be measured and recorded by multi-electrode arrays (MEAs). Each intracortical electrode implanted in the motor cortex records extracellular electrical activity from a relatively small population of neurons within a few hundred micrometers of the neuron closest to the tip of the electrode [167]. This measurement is often contaminated with ambient noise and technical artifacts, such as instrumentation noise. The measured electrical activity inside the gray matter of the brain can be used for muscle control, sensory perception (such as seeing and hearing), as well as speech, decision making, and self-control.

The intracortical brain-computer interfaces (iBCIs) predominantly quantify the neural activity of the brain using multi-unit activities (MUAs). Neural spikes embedded in the MUAs are detected and separated from the background noise by comparing the recorded and filtered voltage waveforms with a threshold, commonly considered as the scaled value of the background noise. Neighboring neurons often fire spikes of similar shape and amplitude; however, relative to their distances to an electrode's tip, the shape of spike waveforms may differ among neurons. This fact allows the spiking activity of individual neurons to be separated through the spike sorting process [138]. It has already been demonstrated that robust iBCIs can be implemented without employing computationally-daunting spike sorting [141, 241, 242, 39, 11]. In this case, all threshold crossings (TCs) of the recorded and filtered voltage waveforms associated with an electrode (channel) are treated as spikes from one putative neuron. Instead of transmitting spike waveforms for offline neural decoding and control, transmitting the number of spikes over a given time window as the feature of interest would dramatically reduce the transmission data rate and power consumption [64].

Although spike-based decoding is commonly employed, various studies have shown that LFP signals can alternatively be used for neural decoding [118, 146, 147]. In LFP-based decoding studies, the spectral power is the most commonly employed metric to quantify the neural activity. This is predicated on historical findings indicating a correlation between changes in the neural signal band power and different physiological states, such as oscillations in the gamma band (30 – 90 Hz) during sleep in humans [263, 264]. Similarly, the power in the beta band (15 – 35 Hz) has been linked to movement preparation [265].

Depending on the timing and nature of the interaction between the user's brain activity and the iBCI system, two paradigms exist. In synchronous iBCIs, the user's neural activity is processed within a predefined schedule, such as when the user is instructed to perform the underlying BCI task, such as motor imagery or concentration on a specific stimulus.On the other hand, asynchronous iBCIs allow the user to initiate actions at their own pace without being restricted to specific time intervals or cues. Asynchronous iBCIs aim to offer a more natural and flexible brain-computer interaction.However, to support a self-paced iBCI, the system needs to continuously monitor the user's brain activity. Compared to synchronous BCIs, asynchronous iBCIs are better suited for applications where users require continuous control, such as prosthetic control.

Over the last decade, synchronous iBCIs and the decoding of information, such as movement kinematics (i.e., direction, velocity, etc.), from recorded extracellular signals have received considerable attention. However, the efficient realization of the asynchronous iBCIs and addressing the need to detect when a user desires to use a prosthetic or assistive device have remained relatively understudied. A two-state "active/inactive" "brain-switch" [243, 116, 244, 132] continuously monitors neural signal features to detect whether the user would like to engage in the underlying BCI activity. Consequently, it only processes neural signals during the "active" state. Another important aspect of brain state estimation is that the inclusion of periods of unintended iBCI control could introduce errors to the decoder analysis. For instance, decoding neural activity between periods of active BCI operations may result in unintended, and potentially dangerous, prosthetic movements when the user is not intended to be engaged in the iBCI operation. To eliminate unnecessary and unintended decoding of neural signals during periods of non-BCI activity, we designed an intention estimation system to discriminate between periods of active BCI control and inactive states. We utilized a recurrent neural network and LFPs as the input feature of interest to detect discrete state changes when the user desires to engage in the underlying BCI activity.

The rest of this chapter is organized as follows. Section 9.2 briefly reviews the motivation for employing LFPs over other signal modalities. Section 9.3 discusses the application of LFPs for realizing brain-switches. The relative performance of the designed LFP-based brain-switch is compared against those of the spiking-based models. Section 9.4 presents the hardware architecture and realization of the proposed LFP-based feature extraction unit. Finally, Section 9.5 makes some concluding remarks.

9.2 Feasibility of Employing Local Field Potentials

One invasive method for recording the brain's neural activity is to use penetrating microelectrodes implanted in the related brain area, such as the primary motor cortex. These electrodes provide valuable information for decoding intended muscle movements. Intracortical MEAs, capable of recording hundreds to thousands of channels simultaneously, offer the highest degrees of freedom among neural signal modalities. They have been conventionally used in iBCIs to record and translate neural activities for controlling robotic prostheses and assistive devices. Whether using a single electrode or an MEA, recording action potentials (spikes) from an individual neuron requires placing the electrode tip within 50–100 mm from the neuron. The electrical spike signal best reflects the neural activity of the brain and has a high signal-to-noise ratio (SNR), as well as high temporal and spatial resolutions. Therefore, the firing pattern of spikes is commonly utilized to extract motor-related information effectively.

Depending on the cortical area and layer, however, each electrode may record action potentials of several individual neurons surrounding the electrode tip. If these multiple recorded neurons have distinctive spike shapes, their individual firing rates (i.e., the number of spikes within 30 to 100 ms time bins) can be discriminated by the spike sorting process, yielding single-unit responses, also known as single-unit activity (SUA). If their action potential voltage traces are similar, spike sorting may not be able to differentiate the individual spike waveforms, thus yielding multi-unit activity (MUA), which represent the total firing rates of two or more neurons. MUAs can be obtained by high-pass filtering the raw data at 250 Hz to 5 kHz and counting all waveforms crossing a threshold. An alternative neural signal that has been shown to be well-correlated with MUAs is the spiking band power (SBP), which has a relatively low sampling requirement of 2 kHz [266] and can be considered as a low-frequency version of MUAs.

Fig. 9.1 illustrates the relative spatial resolution of the alternative signal modalities available for intracortical recording, with the center representing the location of the electrode tip. SUAs have a spatial resolution of single units, on the order of 50 μm, while MUAs have a spatial resolution over 100 μm. The SBP is similar to MUAs but can detect the activity of neurons over a larger area of 250 μm. Additionally, the LFP comprises the activity of neurons within 500 μm [142, 29, 143, 266].

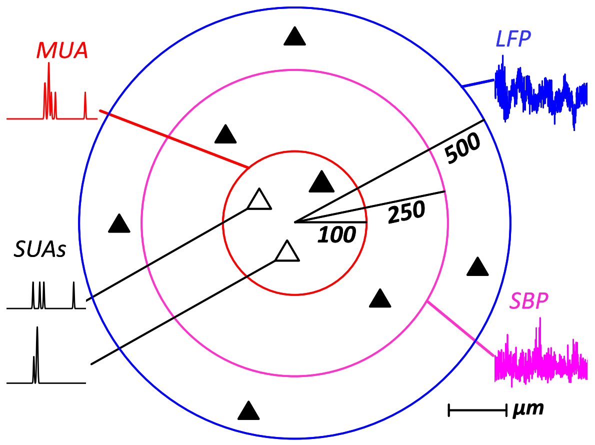


Figure 9.1. The different signal modalities of intracortical recording electrodes.

Maintaining signal stability poses a serious challenge for long-term iBCI recordings. One major drawback of spike-based iBCIs is that over time the electrode tips become increasingly encapsulated by brain tissue scarring, preventing the long-term observation of individual neurons [145]. Fortunately, this encapsulation does not significantly affect lower frequency signals. The LFP is the electrical potential recorded from deep brain tissues, such as the extracellular space in cortical regions. It is typically obtained by filtering the raw continuous data from 0.3 Hz to 500 Hz. Since LFPs mainly reflect the summation of electrical activities surrounding the recording site, they may be less sensitive to small movements or the loss of detectable neurons near the electrode tips [143]. Additionally, the extracellular medium, i.e., portions of cortical tissue between the electrode tip and neurons, acts as a low-pass filter. Consequently, the amplitude of high-frequency action potentials is naturally attenuated with distance. As a result, extracellular action potentials are only reliably detected for electrodes positioned near the recorded neuron. In contrast, low-frequency synaptic potentials attenuate less with distance, allowing them to propagate over larger distances in the extracellular medium. These low-frequency signals may even be recordable as far as the surface of the scalp, which is used in electro-encephalography (EEG) studies.

While early studies on LFPs primarily focused on the temporal analysis of a population of LFP signals, time-domain analyses of LFPs have demonstrated inferior performance for decoding neural information and control. One commonly-employed feature of interest is band-limited power, which can be computed over specific frequency bands, such as delta (1–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), beta (12–30 Hz), gamma (30–80 Hz), or high-frequency (80–500 Hz). It has been demonstrated that the spectral power in the high-frequency (HF) range of 60–200 Hz reflects the summation of spiking signals from the neuronal population near the recording electrode. This range may contain more sub-threshold information that is lost during the spike detection process. Since HF-LFP activity is correlated with neuronal firing rates (multi-unit spike count) [267, 268], it may provide particularly information-rich signals for predicting movement parameters, such as kinematic representations of prosthetic movement variables, even in the absence of clear spiking activity. For instance, a study in [147] reported that various movement intentions, such as the imagined end-point, trajectory, and type of movement, could be reliably

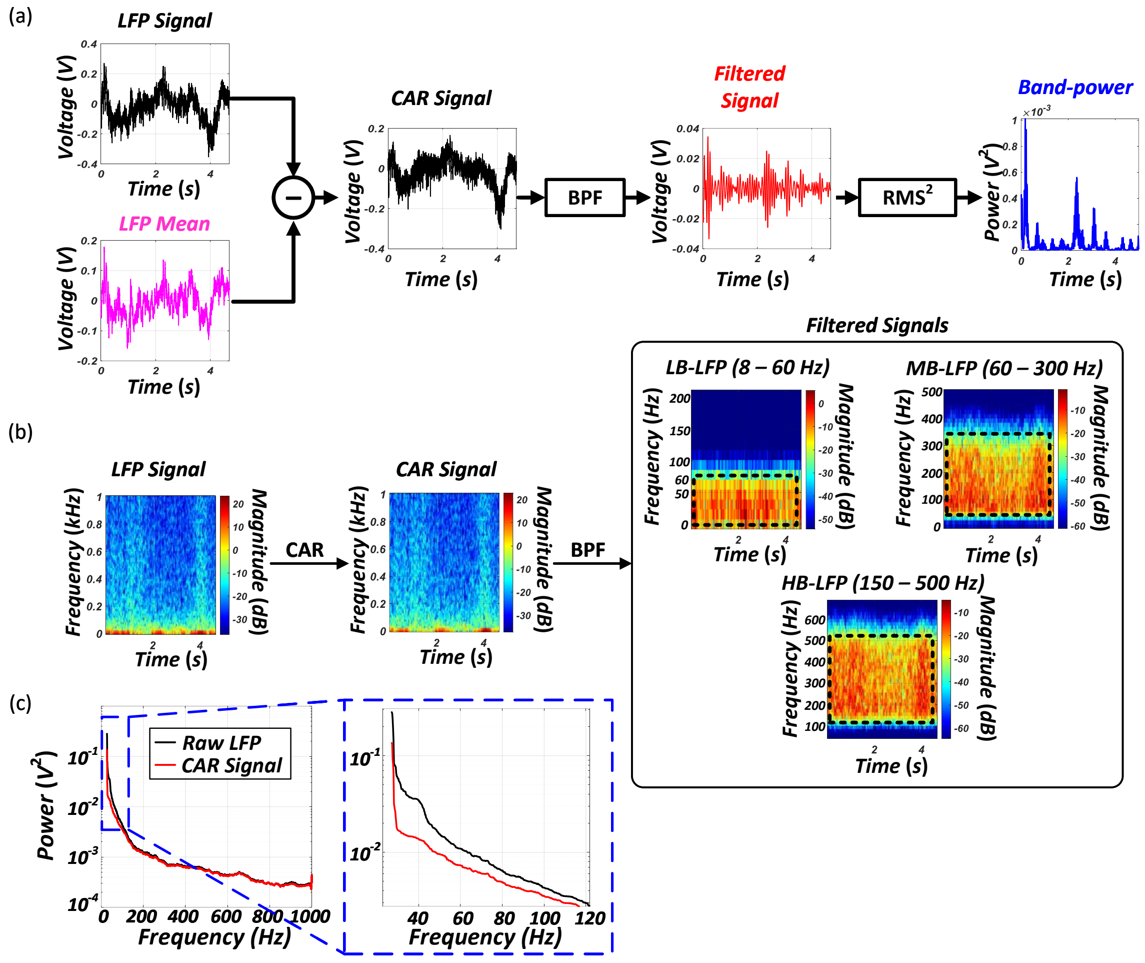


Figure 9.2. (a) The block diagram of the in vivo LFP-based feature extraction pre-processing. (b) The time-frequency power spectrum of the neural signals during different phases of the feature extraction pre-processing. (c) The spectral power of the raw LFP and the CAR signal.

predicted from the LFP signals. Additionally, LFPs are considered a more stable neural signal compared to the single-unit and multi-unit activities, which often suffer from electrode drift, neuron drop-out [144], and scarring from the electrode-tissue interface [145].

Another advantage of LFPs over action potentials is that LFPs can be sampled and processed at significantly lower rates (approximately 1 kHz) compared to spike events being digitized at sampling rates of 10 – 30 kHz for accurate spike detection [131]. A reduced sampling rate has a significant impact on the power consumption of implantable interfaces, potentially increasing the lifespan of implanted devices [131]. Additionally, there is no need for computationally-daunting spike detection and sorting algorithms, which consume 0.6 – 4 μW of power per channel [132, 148]. BCIs based on LFPs may also not require the use of deeply penetrating micro-electrodes, significantly reducing the chance of tissue scarring and potentially increasing the longevity of the recorded neural signals [145]. Therefore, instead of relying on isolating individual action potentials within the 500–5000 Hz band, the low-frequency components ($<$ 500 Hz) of a recorded neural signal can be employed for a stable BCI signal processing [269].

9.3 Design of LFP-based RNN Brain Switches

To reduce the significant power consumption of the in vivo circuitry during non-BCI activities, a neurally-controlled brain-switch can be utilized. This switch enables or disables in vivo neural signal processing and wireless transmission of chosen features for in silico neural decoding. Brain-switch algorithms operate on the assumption that the recorded neural signals exhibit various behaviors during different mental states. For instance, EEG-based brain-switches may detect an increase in the 1 Hz – 4 Hz band power [245, 246], while ECoG-based brain-switches may detect a change in the power of the specific frequency bands for motor execution/imagery [243]. For spike-based neural signal processing, the spike counts during a time interval [256], as well as changes in the firing rates [116], have been considered as underlying features for reported brain-switch algorithms. It has been previously reported that a small subset of recording channels exhibit satisfactory variations in the MUAs, allowing brain-switch algorithms to reliably detect transitions in mental state [132].

Due to the intracortical recording setting, LFPs offer a higher signal-to-noise ratio compared to ECoG and EEG. Moreover, the relaxed requirements of LFP signal acquisition and processing make them a viable modality for brain-switch algorithms, significantly reducing the power consumption compared to spike-based systems. To analyze the feasibility of using LFP features for brain-switch models, we employed two publicly-available neural datasets, datasets

I (I140703) and II (L101210) [152]. These datasets contain raw recordings from the motor cortex (M1 Region) of two Rhesus macaque monkeys using an implanted 96-channel Utah Array. The recordings were initially sampled at 30 KHz and then downsampled to 10 KHz. Further downsampling to 2 kHz was performed to extract low-frequency LFPs. During the experiments, the monkeys performed a cued reach and grasp task to displace an object. The monkeys were presented with a series of cues indicating that the beginning of a trial and specifying one of four combinations of grip and displacement forces to use. Additional details of the animal experiments are described in [152].

9.3.1 LFP Feature Extraction Unit

After signal acquisition and digitization, the common average reference (CAR) is conventionally employed to effectively remove common interference shared across electrodes. The CAR is given as $\hat{\mathbf{X}}[n] = \mathbf{X}[n] - \mu[n]$, where $\mathbf{X}[n]$ denotes the multi-channel neural signal, and $\mu[n]$ denotes the mean over all recording channels at time n. The CAR enhances the SNR and the interpretation of neural activity. Although the CAR is computed over all recording electrodes in most analyses, practical settings typically limit the CAR to a local subset of recording channels, as its impact on the system-level performance is negligible [270]. Following CAR, spectral power is computed for three frequency bands: the low-band LFP (LB-LFP) (8 – 60 Hz), mid-band LFP (MB-LFP) (60 – 300 Hz), and high-band LFP (HB-LFP) (150 – 500 Hz). Spectral power is commonly calculated using Fourier or Morlet wavelet transforms, providing time-frequency decomposition of signals.

A less computationally complex approach is to use a series of bandpass filters with cutoff frequencies for LB-LFP, MB-LFP, and HB-LFP. Once the signals are filtered, the spectral power within each frequency band of interest can be estimated using envelope detection schemes [116] or fast Fourier transform [271]. Applying bandpass filtering allows for attenuating signal components outside the desired frequency range, eliminating the need for Fourier transform to compute the spectral power over all frequencies. Fig. 9.2(a) illustrates the estimation of total power contained within a specified frequency range (band power) using the square of the root mean-square (RMS) of the bandpass filtered LFP signal in the time-domain [272]. Fig. 9.2(b) illustrates the time-frequency plots of the raw LFP signal, the output of the CAR, and the output of the bandpass filter (BPF) over the LB-LFP (8 – 60 Hz), MB-LFP (60 – 300 Hz), and the HB-LFP (150 – 500 Hz). The time-frequency plots in Fig. 9.2(b) demonstrate that the employed BPF effectively isolates the signal within the specified frequency range. Following the implementation of CAR, the multi-channel interference is reduced, leading to a decline in low-frequency power, as depicted in the magnified view presented in Fig. 9.2(c). For Channel 1 of Dataset I, depicted in Fig. 2, the average power of the raw LFP within 0 – 120 Hz is 16 mV^2, while after CAR, the average power in the same frequency band is reduced to 8.2 mV^2.

9.3.2 RNN Model Evaluation

While brain-switch software models may use all recording channels as the input to the algorithm, one opportunity for reducing energy dissipation is to employ a subset of recording channels. This, however, comes at the expense of additional pre-processing and a learning phase. This approach is viable for spike-based iBCIs, where specific neurons or an ensemble of neurons may behave differently during the intention onset. However, in the case of LFPs representing the activity of neuronal populations, it might be possible to utilize either a set of arbitrary recording channels or even a single recording channel as the input for the brain-switch algorithm.

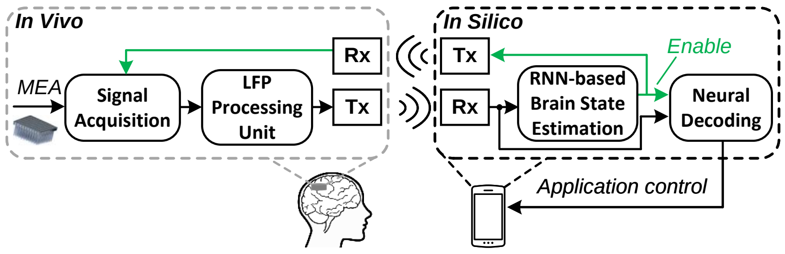


Figure 9.3. The system-level block diagram of the designed brain-switch consisting of the in vivo LFP processing unit and in silico brain-state estimation.

In our previous work [132], we found that machine learning (ML) algorithms, specifically relatively small recurrent neural networks (RNNs) using neural spikes as input performed well for brain-state detection. In this study, we implement an LFP-based brain-switch RNN, as depicted in Fig. 9.3. The brain-switch consists of two units: an in vivo processing system for neural signal processing and LFP feature extraction, and an in silico module that receives wirelessly transmitted features and executes the RNN-based brain-switch model. When the BCI is in the "inactive" state, one recording channel is employed for in silico brain-state estimation. However, when the BCI is in the "active" state, all or a subset of recording channels can be employed for robust neural decoding. Therefore, the output of the brain-switch is utilized to control both the in vivo signal acquisition as well as in silico neural decoding. It is important to note that the neural signal features used for decoding may differ from those used for the brain-state estimation. In vivo neural signals are first enhanced using the CAR. Then an 8-th order bandpass filter, along with a squared RMS module is used to estimate the power within each frequency band. where the number of spikes observed within this time period make up the multi-unit activities [131, 273]. Ranges up to 50 ms have been reported for iBCIs targeting the SBP [266]. The RMS is computed over 10 ms windows as it falls within the range of commonly employed iBCI configurations.

We evaluate the performance of four alternative RNN-based models with different recurrent cell architectures: standard RNN, long short-term memory (LSTM), gated recurrent unit (GRU), and quasi-recurrent neural network (QRNN) [274]. Each RNN cell accepts the previous layer inputs and employs a self-recurrent connection to learn temporal dependencies within the data. The standard RNN cell transmits its hidden state directly. In contrast, the LSTM and GRU cells utilize internal non-linear functions to regulate the amount of data propagated over time. The QRNN employs a convolution operation over time instead of directly propagating its internal state, with its temporal receptive field width dependent on the convolutional kernel width. For performance comparison among the alternative RNN cells, we devised a RNN brain-switch that accepts the LFP bandpower of a single channel to predict the moment of intention. First, the single channel is passed to a fully-connected dense layer with 16 units, where its output connects to an RNN-based layer to learn temporal features within the signals. For a fair comparison among the different cell types, the RNN-based layers have about the same number of parameters. The model with the simple RNN cells employs 62 units, the LSTM employs 34 units, the GRU employs 32 units, and the QRNN employs 25 units. Finally, the outputs of the RNN-based layer passes through a fully-connected dense layer that performs the final regression and predicts the current intention state using a non-linear sigmoid function.

Each of the RNN-based layers has about 4800 parameters and is trained using the mean squared error loss metric and the Adam optimizer. Each is trained using 10-fold cross-validation to provide a fair estimate of its performance. The performance of the designed brain-switch RNN models is measured using the F-score metric given as $F = 2T_P/(2T_P + F_N + F_P)$, where T_P denotes the number of correctly detected intentions, F_N denotes the number of non-detected intentions, and F_P denotes the number of false positive intentions [260]. Figs. 9.4 (a) and (b) show the boxplots of the F-score of the four alternative models for the first recording channel of Dataset I and II, respectively. It can be seen that the GRU-based model provides the most consistent performance over both datasets and training iterations compared to the alternative models. In other words, while the LSTM and QRNN can perform well, whether a given training session will yield an acceptable performance is not deterministic. However, for a well performing training session, the training time is also of importance. Fig. 9.5(a) and (b) show the variation of F-score of the different models over the training period. In general, the GRU-based model achieves a relatively high performance considerably faster than the other models and is also shown to provide stable performance over time compared to the other recurrent neural network models. Therefore, we employ the GRU-based architecture, as shown in Fig. 9.6, to realize the brain-switch.

9.3.3 MEA Channel Selection

To evaluate the performance of the designed brain-switch scheme, the impact of channel and frequency band selections was analyzed. For each recording channel and frequency band, the data was normalized as $\hat{X_{fC}} = (X_{fC} - \mu_{fC})/\sigma_{fC}$, where μ_{fC} and σ_{fC} denote the mean and

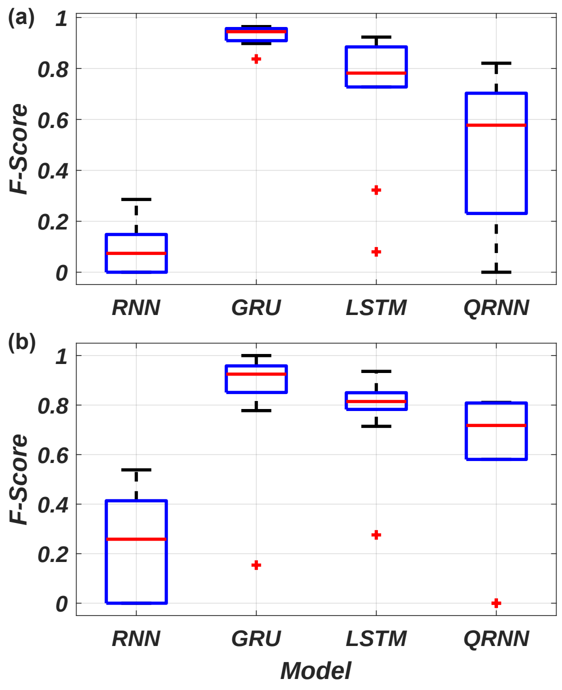


Figure 9.4. The boxplots of the F-score of the four alternative RNN-based models over the first recording channel of (a) Dataset I and (b) Dataset II.

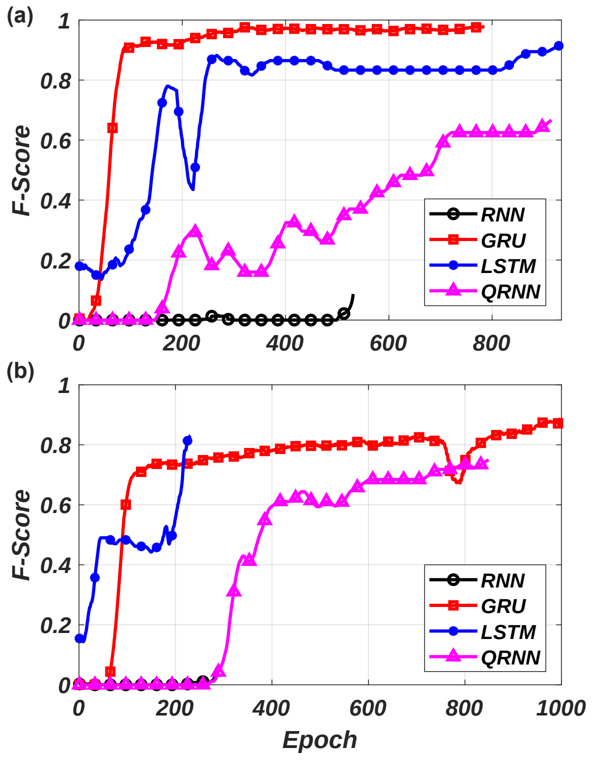


Figure 9.5. The variation of F-score over the training epochs for (a) Dataset I and (b) Dataset II.

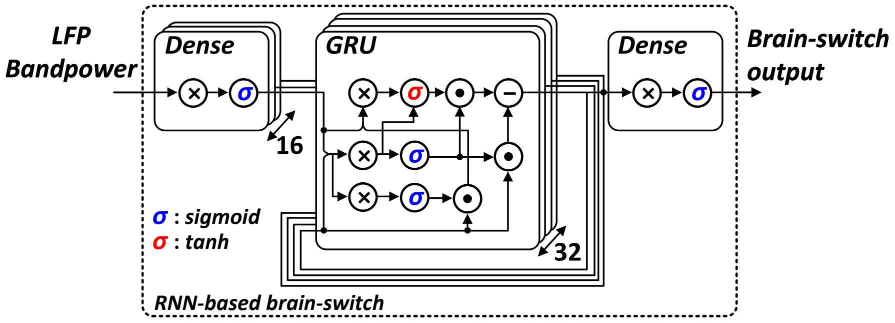


Figure 9.6. The block diagram of the designed GRU-based LFP brain-switch.

standard deviation of signal X of channel C filtered over frequency band f. Deep learning algorithms conventionally rely on a training and testing subset using 70% and 30% of the dataset, respectively. Compared to deep learning datasets that typically have millions of samples for training and testing, a common limitation in the field of computational neuroscience is that training data is relatively sparse. In our work, we have used only the first 30% of the dataset for training, which is consistent with our previously published work [64] to emulate this limitation. It is important to note that using less training data and more testing data implies that our model is not as intensely affected by overfitting. Dataset I and II consist of 153 and 149 trials, respectively. For each trial, the moment the monkey releases a switch is considered as the brain state transition from inactive to active, as used in our previous work [132]. However, to avoid detecting the intention during movement-related potentials when the animal releases the switch, a small randomized jitter time ($\mu = 14 \pm 8$ ms) was subtracted from the switch release time, ensuring that the state transition is detected during the movement planning phase of the trial. The value used for the jitter time has no particular significance other than ensuring that the RNN is trained to predict a brain-state transition prior to a switch released by the subject. The jitter time is modeled as a random variable so that the RNN would appropriately learn to perform the state transition based on the neural signal input patterns rather than performing a brain-state transition at the same time for any given trial. This approach also ensures that the model is only trained on neural signals that occur during the grasp planning stage of the subject's movement.

The designed RNN model was trained on all combinations of recording channels and

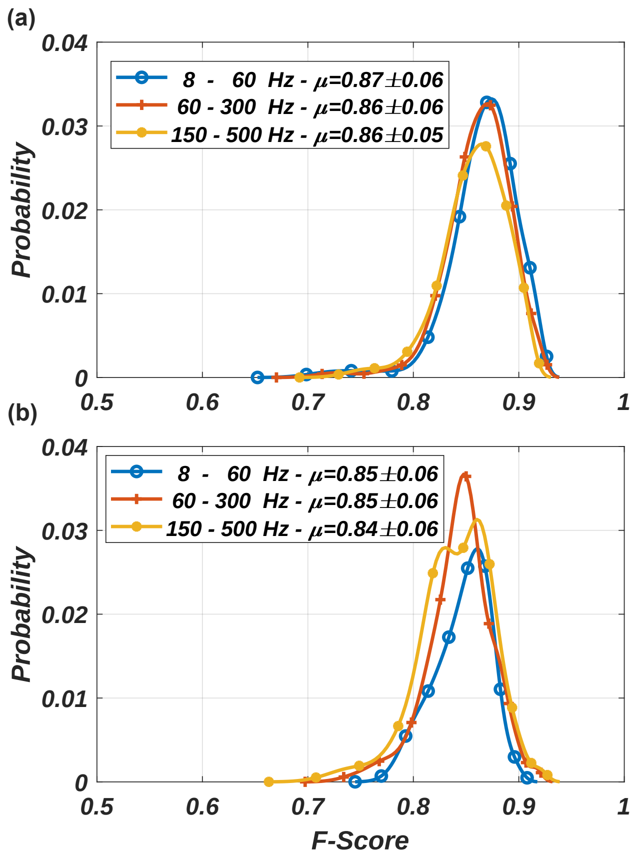


Figure 9.7. The F-score probability distributions for (a) Dataset I and (b) Dataset II over the three frequency bands and all recording channels.

frequency bands, and the F-Score probability distributions for Datasets I and II are shown in Figs. 9.7(a) and (b), respectively. It can be seen that the performance of the designed brain-switch model is agnostic to recording channel and frequency band over both datasets, with a consistent F-Score performance of 0.86 ± 0.06 for Dataset I and approximately 0.80 ± 0.07 for Dataset II. This finding suggests that any particular channel may offer useful information for effective brain-state estimation. However, some channels may have lower performance than others. While beyond the scope of this chapter, one may employ a mean bandpower threshold or other relevant metrics for screening candidate channels. In our previous realization of the brain-switch, which utilized multi-unit activities in the form of spike threshold crossings, we required multiple channels along with a complex channel selection process for sufficiently accurate brain-state estimation [132]. However, we found that a single recording channel of LFPs, which embodies the activity of larger populations of neurons, provides relatively accurate brain-state estimation. This is potentially due to the fact that in the spike-based approach, removing low-frequency components to obtain the action potentials isolates the recorded neural signals to those of only neurons near the electrode tips, which may encode more specific information than that of an ensemble of neurons surrounding a larger region around the electrode.

9.3.4 CAR and Filter Considerations

The results depicted in Figs. 9.7(a) and (b) prompt the question of whether performing the CAR is necessary at all. CAR is commonly employed in LFP studies to remove interference that is common across the recording electrodes. In practice, removing common interference from multi-channel recordings is an effective approach for reducing the impact of redundant features on the decoding algorithm. However, as discussed in Section 9.2, our findings indicate that data from a single channel is adequate for the brain-switch operation. Therefore, the CAR may not be necessary, and the noise may be sufficiently removed by the bandpass filter. Additionally, in the context of efficient hardware realization, removing the CAR would significantly reduce the in vivo signal processing. Computing the CAR would require $2N-1$ additions and one multiplications per clock cycle, which can be expressed as $2N+1$ operations when assuming the complexity of multiplication as twice that of an addition [43]. Considering a clock rate of 2 kHz and a 96-channel Utah electrode array, the CAR would require 386 kOps/second.

Figs. 9.8(a) and (b) show the F-Score probability distributions over Datasets I and II, respectively, and over all combinations of recording channels and frequency bands, without employing the CAR. It can be seen that for Dataset I, there is no significant degradation in F-score compared to those with the CAR for all three frequency bands. For Dataset II, the performance of each band varies, with the LB-LFP (8 – 15 Hz) outperforming the other two bands. Therefore, we employ the band-limited power of the LB-LFP as the input feature for the RNN-based brain-switch algorithm.

Previously published work mainly employ two-sided non-causal filters as they may better

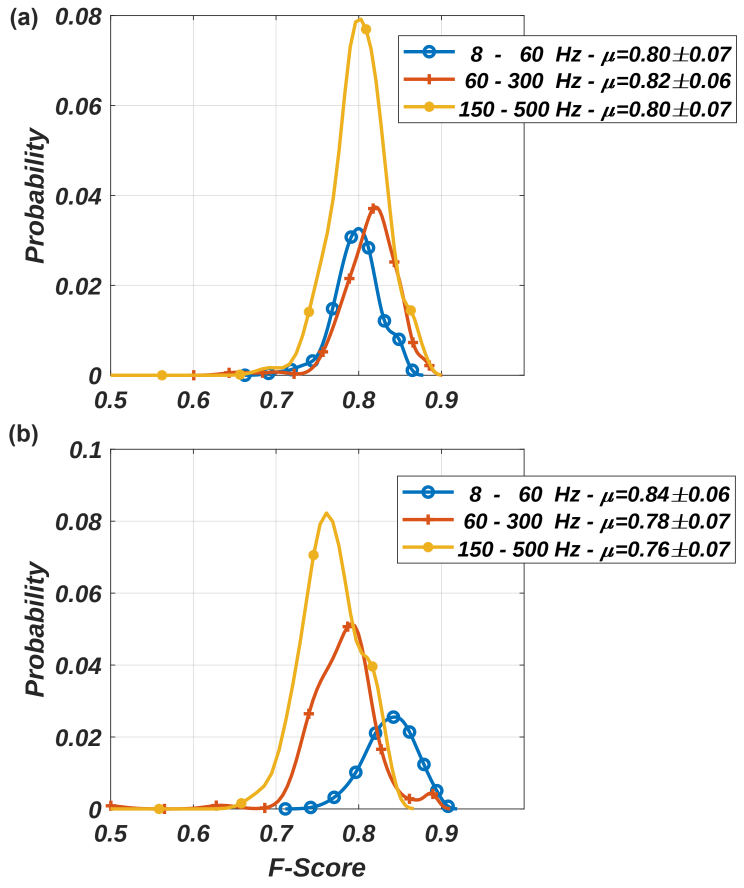


Figure 9.8. The F-score probability distributions for (a) Dataset I and (b) Dataset II over the three frequency bands and all recording channels when removing CAR.

preserve the amplitude and shape of neural signals while not imposing phase distortions [113]. Causal filter realizations, however, are deemed to be preferred for in vivo realizations due to their reduced computational complexity and memory requirements. Fig. 9.9 shows the spectral power density of the first channel of Dataset I over the 8 – 15 Hz frequency band for both non-causal and causal filtering with various filter orders. It is evident that the lower-order filters do not limit the signal to the desired frequency band as accurately as the higher-order filters. While it may appear that the second-order causal filter spectrum shown in Fig. 9.9(d) may not be suitable, as it has a relatively poor frequency selectivity, we investigate its impact on the performance of the designed brain-switch algorithm by training the RNN model using both non-causal and causal filters of orders 2, 4, 6, and 8 and the LB-LFP bandpower for the first channel of Datasets

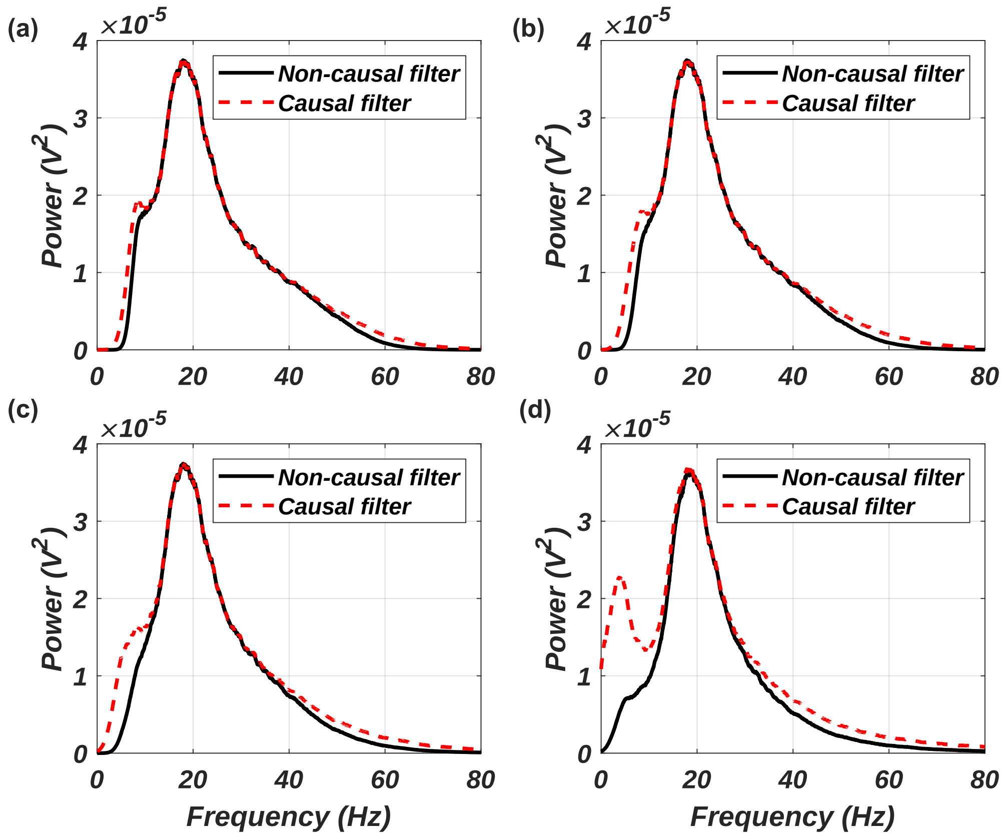


Figure 9.9. The spectral power of the LB-LFP of non-causal and causal filters with filter order (a) 8, (b) 6, (c) 4, and (d) 2 over the first channel of Dataset I.

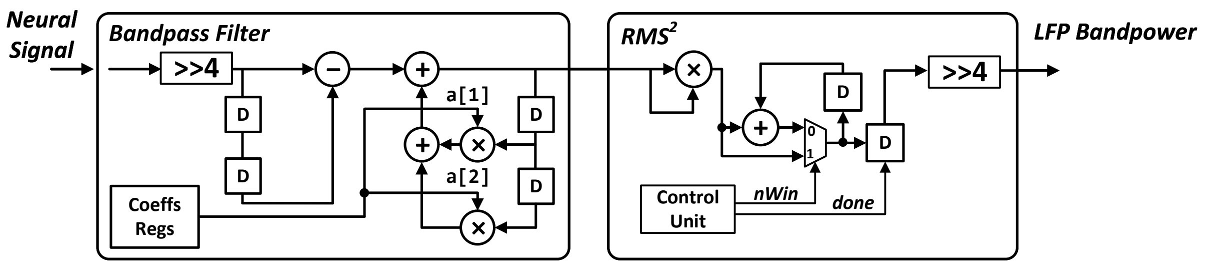


Figure 9.11. The block diagram of the in vivo signal processing for estimating LFP bandpower.

I and II as input. Figs. 9.10(a) and (b) confirm that there is no significant performance variation when employing either non-causal or causal filters. Additionally, it is observed that for Dataset I, the lower-order filters outperform the higher-order filters, employing either non-causal or causal realizations. This suggests that the sharp roll-off characteristic of the higher-order filters is not a strict requirement for the designed RNN-based brain-switch. Therefore, we employ

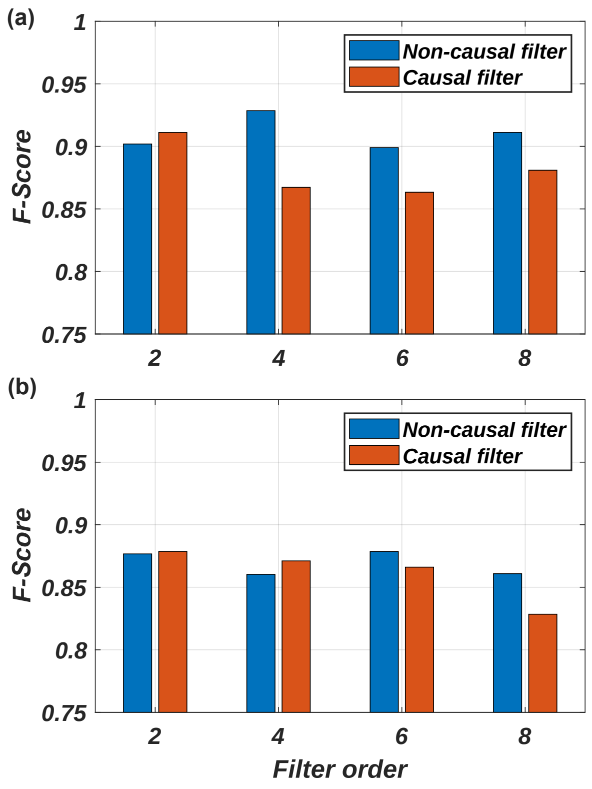


Figure 9.10. The F-score of the LFP-based RNN brain-switch model using various filter orders and filter causalities for the LB-LFP and the first channel of (a) Dataset I and (b) Dataset II.

a second-order causal filter without CAR for in vivo realization. It should be noted that the performance of the proposed GRU-based brain state estimation is dependent on the dataset, as is the case for all machine learning-based algorithms. The performance of the proposed approach, and other similar kinematic decoding tasks, relies on the activation of relevant brain regions. Prior to surgical implantation of microelectrodes, the neural activity of the brain regions are observed using positron emission tomography (PET) or functional magnetic resonance imaging (fMRI). [275, 276].

9.4 Hardware Architecture of the Designed LFP-based Brain Switch

Fig. 9.11 shows the block diagram of the in vivo signal processing for the LFP-based feature extraction, i.e., LFP bandpower estimation. A single channel of the MEA is bandpass filtered at a rate of 2 kS/s. The bandpower is estimated by computing the RMS squared over non-overlapping windows of 8 ms (i.e., 16 samples at 2 kS/s). The bandpass filter is designed using a second-order infinite-impulse response (IIR) Butterworth filter structure [277]. The passband filter coefficients for 8 – 60 Hz were computed using the Filter Design Toolbox in Matlab as $b_0 = 1$, $b_1 = 0$, $b_2 = -1$, $a[0] = 1$, $a[1] = -1.84$, and $a[2] = 0.848$. The filter input scaling factor of 0.075 is approximated as an arithmetic right shift by four bit positions, equivalent to multiplying by 0.0625. To find an optimal tradeoff between area utilization and power consumption, in addition to the direct biquad structure, the time-multiplexed IIR filter structures with the folding factors of 2 and 3, as shown in Figs. 9.12(a) – (b), respectively, were designed and implemented. The direct structure is required to operate at the minimum rate of 2 kHz while the designs with the folding factors of 2 and 3 are required to operate at 4 kHz and 6 kHz, respectively. A control unit, not shown, is used to generate appropriate control signals.

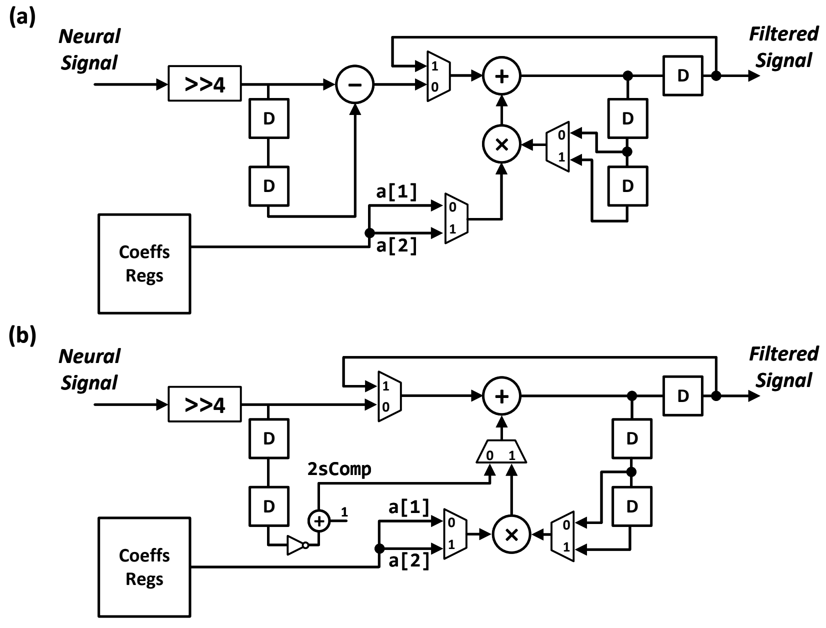


Figure 9.12. The block diagram of the designed band-pass filter with the folding factors of (a) two and (b) three.

The filter output is passed to the *RMS*² unit, as shown in Fig. 9.11. The *control unit* asserts the signal *nWin* every 8 ms, setting the accumulation register to the squared input sample. On the 16-th input from the filter (i.e., after 8 ms), the *control unit* asserts the signal *done* to latch the output of the accumulator into the output register, which is arithmetically right shifted by four bits to compute the mean over the window. The squared RMS is used as the estimate of the bandpower where no square root operation is required.

The computations of the filter and RMS^2 unit are performed in the fixed-point format Q(WI.WF), where WI and WF denote the integer and fractional wordlengths of the signals. Software simulations of the in vivo modules verified that the filter's internal accumulator requires 17 bits in Q(5.12) format and the filter output requires 16 bits in Q(3.13) format. Since the RMS^2 performs a square operation and a right shift for computing the mean, the full-precision computation would require at least 30 fractional bits. To find the optimal number of fractional bits necessary to achieve performance (F-Score) comparable to that of the software simulations shown in Figs. 9.10, various bit-true simulations were performed. Fig. 9.13 shows the F-Scores of the designed RNN-based brain-switch algorithm over different number of fractional bits for representing the RMS bandpower. The analysis over the two datasets showed that a minimum of eight fractional bits offers sufficient accuracy.

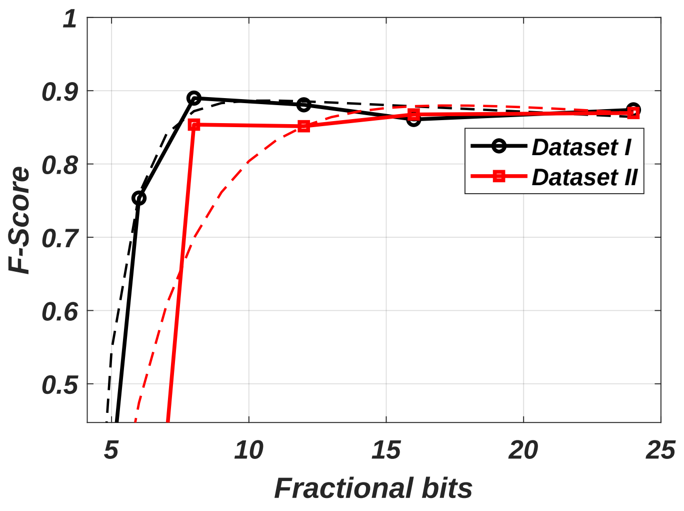


Figure 9.13. The variation of the F-Score of the designed RNN brain-switch over various number of fractional bits used for representing the bandpower, with the dashed lines indicating the trend.

The logic synthesis of the in vivo design shown in Fig. 9.11 was implemented using

Table 9.2. The characteristics and implementation results of various feature extraction ASICs.

Work	Ours (LB-LFP)	Ours (Prog.)	Ours[132]	[18]	[278]	[279]
Signal modality	LFPs		MUAs	MUAs	SBP	SBP
Technology (nm)	180	180	180	180	180	180
Supply voltage (V)	1.8	1.8	1.8	1.8	1.55	0.625
Area per channel (mm^2)	0.09	0.12	0.03	0.03	0.07	–
Power per channel (nW)†	91.8	109.9	630	1500	212	3660

† Normalized to a 180-nm CMOS process with a 1.8 V supply and accounting only for digital backend.

Table 9.1. The area utilization and power consumption of the designed and implemented LFP-based feature extraction unit.

Folding Factor	**Area (mm^2)**	**Power (nW)**
1	0.09	91.8
2	0.10	192.5
3	0.10	160.6
Programmable Frequency Band		
1	0.12	109.9
2	0.12	247.2
3	0.11	279.6

Synopsys Design Compiler, and the place and route was performed using Cadence Innovus in a standard 180-nm CMOS process. The design was also synthesized for both folding factors of the bandpass filter datapath. The power consumption was estimated by employing both datasets, along with the toggling rate information of all signals stored in a Switching Activity Interchange Format (SAIF) file. Table 9.1 gives the area utilization and power consumption of the designed LFP bandpower estimator using the direct structure and using two folding factors. It is shown that the smallest area and the least power consumption are achieved with the direct-form IIR filter structure. While the folding factors of two and three for the bandpass filter datapath have fewer adders and multipliers, the multiplexers require sign and zero extension of inputs with fewer bits to match those of the accumulation registers, which in turn require more area. While the LB-LFP offered the best performance for both employed datasets, it may be beneficial to support MB-LFP and HB-LFP bands as well for practical applications (e.g., neural decoding). The design was therefore modified to support all three sub-bands. The power consumption of the LFP-based feature extraction unit with the programmable frequency band for different folding factors is given in Table 9.1. Similar to that of the ASIC for the LB-LFP, the direct structure (i.e., the folding factor of one) consumes the least power among the three realizations for a marginally larger silicon area.

The ASIC layouts of the synthesized in vivo LFP processing unit, including the IIR filter and the RMS bandpower unit, for (a) the LB-LFP and (b) the programmable frequency band in a standard 180-nm CMOS process are shown in Fig. 9.14. The layout in Fig. 9.14(a) computes the bandpower of the LB-LFP and occupies 0.089 mm^2 of silicon area and consumes 91.87 nW of power from a 1.8 V supply while operating at 2 kHz. The layout in Fig. 9.14(b) computes the bandpower of any one of the LB-LFP, MB-LFP, and HB-LFP bands based on a programmable band-selection input, and occupies 0.117 mm^2 of silicon area and consumes 109.9 nW of power from a 1.8 V supply, while operating at 2 kHz. The power consumption was estimated by simulating the netlist after placement and routing and accounting for the switching activity of all signals.

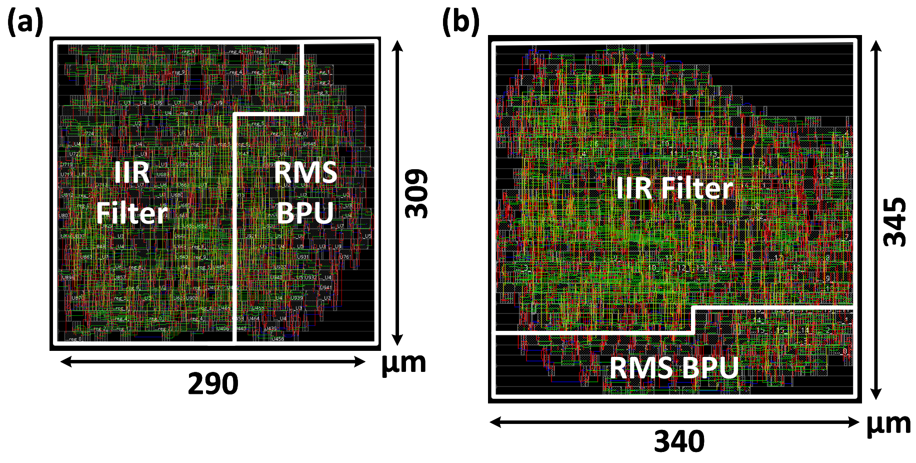


Figure 9.14. The ASIC layouts of the synthesized in vivo LFP-based feature extraction units for (a) the LB-LFP and (b) the programmable frequency band in a standard 180-nm CMOS process.

Table 10.2 gives the characteristics and implementation results of various in vivo feature extraction ASICs. Due to the relatively high temporal resolution of action potentials, analog front-ends (AFEs) for spike-based BCIs must operate at a significantly higher sampling rate and bandwidth than those employing LFPs. First, the power consumption of the low-noise amplifier (LNA) is proportional to the width of the frequency band of interest [131], which for LFPs is about five times smaller than that for spikes. Secondly, conventional AFEs employ successive approximation register (SAR) analog-to-digital converters (ADCs), whose power consumption is relatively linearly proportional to the sampling rate [162, 163], which again are far more efficient for LFPs. Thus, for a fair comparison with the previously published works, the comparison Table II only accounts for the digital backend of all works. In our previous work [132], we designed and implemented an RNN-based brain-switch algorithm employing MUA features. Compared to the designed LFP-based brain switch, the MUA-based design requires a correlation-based learning phase to find the optimal recording channels for detecting the brain's state transitions. Employing the same datasets, the MUA-based brain-switch algorithm achieves a marginally improved F-score of 0.89 compared to the LFP-based brain-switch F-score of 0.84, with a power consumption approximately five times higher. In [18], the authors present an analog implementation of a spike detection unit that performs energy-based thresholding to generate MUAs. In [278], the authors target the low-frequency SBP and achieve a low power consumption of 212 nW per channel. Also targeting the SBP, the design in [279] reports a power consumption of 3.6 μW per recording channel for obtaining the SBP. The proposed LFP-based ASICs, as given in Table 10.2, consume significantly lower power for extracting the LFP features. While it remains to explore how well the SBP can be employed for deteciting the brain-state transitions, due to the LFP's lower frequency filtering and acquisition requirements, the LFP is well positioned as an effective signal modality for brain-state monitoring and behavioral decoding.

One important consideration is the total power consumption of the proposed system, accounting for the signal acquisition analog front-end (AFE) circuitry, digital signal processing, and wireless transmission overhead. Conventional AFE circuitry, such as low-noise amplifiers (LNAs) and analog-to-digital converters (ADC), requires an average of 7.4 μW of power per channel for spike-based systems employing sampling rates on the order of 10 kS/s [160]. As both the bandwidth and sampling rate of the LFPs and SBPs is roughly one-fifth of that for spike-based

BCIs, the signal conditioning AFE for LFPs and SBPs would consume approximately 1.5 μW per recording channel.

The wireless transmission power depends largely on the output bitrate of the in vivo circuitry, as well as the temporal resolution of the signal features. For spike-based systems, a commonly employed range of temporal resolutions is within the range of 1 ms to 10 ms, requiring an output data rate between 0.4 - 1.0 kbps, i.e., one bit per millisecond or $\lceil \log_2 10 \text{ ms} \rceil = 4$ bits per millisecond assuming a one millisecond spike refractory period. The temporal resolution of the proposed LFP feature extraction unit is 10 milliseconds, with 9 bits (Q(1.8)) per estimated bandpower sample and requires 0.9 kbps. Spike-based systems consume between 0.6 and 1.78 μW of power for obtaining MUAs. Given a wireless transmission power of 158 pJ/bit [15], spike-based systems require between 8.06 μW and 9.26 μW per recording channel, accounting for the entire signal acquisition and processing chain, and wireless transmission. The designed LFP-based feature extraction system would require a total of 1.75 μW per recording channel. Based on our previous work implementing on a spike-based brain-switch targeting MUAs, we found that a minimum of 8 recording channels were required for sufficient brain-state transition detection, which would require a total of 64.48 μW of power. In contrast, the designed LFP-based brain-switch requires only a single channel and consumes over 97% less power with only 1.75 μW.

Considering that a single channel of the LFP is sufficient for brain state estimation, an alternative system configuration is to bypass the LFP feature extraction in vivo and transmit the LFP signal directly. Assuming a signal resolution of 16 bits and a sampling rate of 2 kS/s, the required output data rate for a system transmitting the LFP signal directly is 32 kbps. Given the transmission energy of 158 pJ/bit, this would require 5.05 μW of power, over 2.8$\times$ more power than that of the proposed system. For transmitting the LFP signal directly, the increased data rate becomes the primary factor causing increased wireless transmission power requirements. To achieve a comparable power consumption to the proposed design, the LFP signal must be represented with fewer than 6 bits to consume less than 1.8 μW.

9.5 Conclusion

This chapter presented a brain-switch to estiamte when the user involves in the underlying BCI activity based on local field potentials (LFPs) and gated recurrent unit (GRU)-based neural networks. The designed switch is particularly imnportant in the practical realization of asynchronous brain-computer interfaces and offers a comparable or greater effectiveness than that of spike-based systems. It was shown that the performance of the machine learning (ML)-based algorithms for implementing LFP-based brain-switches may not be sensitive to specific recording channels as well as specific frequency bands. Compared to the previously reported brain-switch schemes in which exhaustive learning phases for optimal recording channels are performed prior to neural network training, the brain-switch performance is similar across multiple recording channels, significantly reducing the pre-processing computations. The synthesized LFP-based feature extraction unit in a standard 180-nm CMOS process was estimated to occupy 0.09 mm^2 of silicon area and consumes only 91.8 nW of power while operating at 2 kHz. Compared to the previously published works on in vivo neural signal feature extraction, the proposed LFP-based design consumes the least power and achieves a performance (F-Score) comparable to that of spike-based systems.

9.6 Publications

The following article was published as an outcome of the research conducted in this chapter: D. Valencia, P. P. Mercier, and A. Alimohammad, "An efficient brain-switch for asynchronous brain-computer interfaces," IEEE Transactions on Biomedical Circuits and Systems, May 2024.

Chapter 10

A Hybrid Brain-Computer Interface for Low-power and Efficient Neural Decoding

10.1 Introduction

Brain-computer interfaces (BCIs) are used to record and analyze neural activity. One of the main functions of BCIs is to enable the translation (decoding) of the brain's neural activity into real-world representations, such as controlling the position of a computer cursor on a screen or the movements of a prosthetic limb. Thus, BCIs are key rehabilitation devices for those impaired by spinal cord injury or neurodegenerative diseases. The activity of the brain is made up of the synaptic interactions between neuronal cells. Neurons communicate with each other using action potentials (spikes), which travel along the axon before being passed to another neuron via neuro-transmitters [1]. The neural activity can be recorded non-invasively over the scalp, such as with electro-encephalography (EEG) [280]. Alternatively, invasive recording methods employ implantable micro-electrode arrays (MEAs), typically containing hundreds of electrodes penetrating the brain matter. Early MEAs, such as the Utah Array [31], use fewer than 100 electrodes, while modern high-density recording interfaces employ two to three-thousand recording sites [8, 9, 7]. Although invasive recording methods require surgical procedures for the placement and implantation of the MEAs, the quality of the neural signals is significantly greater than that of EEG, which is preferable for users with severe motor impairment [281, 282].

Previous studies have demonstrated that the response of neurons varies when presented with different stimuli. For example, neurons in the visual cortex show selectivity depending on the orientation of the light stimulus [283]. Similarly, multiple brain regions contain neurons tuned for movement direction, including the motor cortex and dorsal premotor cortex [284]. Researchers initially related the neural activity directly to kinematic variables through manually derived expressions and functions, such as Gaussian and cosine tuning functions [35]. Recent advances in machine learning (ML) have paved the way for a data-driven approach to neural decoding, in which large amounts of neural activity and the corresponding physiological behaviors are used to derive an optimal relationship using neural networks [10].

Fig. 10.1 shows the conventional system configuration for a wireless intracortical BCI.

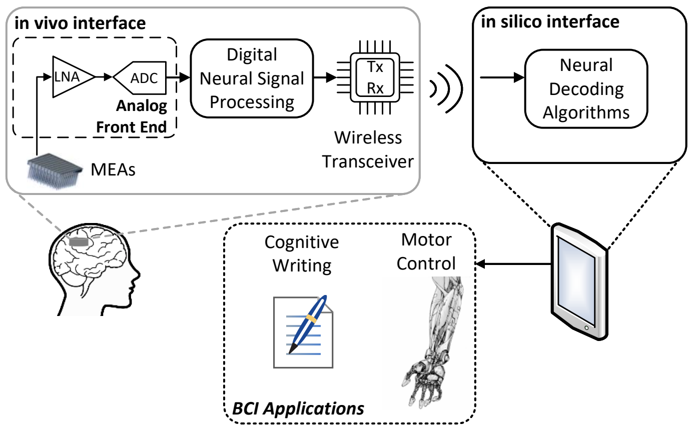


Figure 10.1. The system configuration of a conventional wireless intracortical BCI.

As shown, practical BCI systems are often partitioned into two distinct modules: the in vivo electronic device, which is surgically implanted on the brain to record, process, and transmit neural signal features to an in silico device, which is typically realized on a personal computer or a tablet to execute the decoding algorithm. Prior to digital in vivo signal processing, neural signals are first acquired through the analog front end (AFE), consisting of low-noise amplifiers (LNAs) and analog-to-digital converters (ADCs). The in silico interface also communicates with the underlying BCI application, such as cognitive writing or prosthetic limb control. To limit the amount of data transmitted wirelessly, digital neural signal processing is typically employed in vivo to extract relevant features from the digitized neural signals. The extracted features, as well as the specific decoding algorithms, depend on the BCI application and neural signals that are being recorded.

BCIs can be realized in one of two paradigms. In the synchronous paradigm, the BCI decodes neural activities during predefined time periods. Synchronous BCIs are often employed in laboratory settings, as researchers can readily control when data is collected for analysis and processing, discarding irrelevant neural activity between experiments or trials. Alternatively, the asynchronous paradigm allows the user to initiate BCI operation at their convenience. The asynchronous paradigm, therefore, represents a more practical solution for future BCI technology beyond laboratory settings. While both paradigms perform neural decoding, the asynchronous paradigm also requires a brain-switch (intention estimation unit) to detect when the user wishes to begin the BCI operation. The intention estimation unit must continuously monitor the recorded neural signals for specific variations. Since BCIs operating in asynchronous mode require constant monitoring of the neural activities, they consume more energy to support continuous signal acquisition. Most conventional in vivo processing relies on high-frequency signals (i.e., spikes) to discern the activities of individual neurons. While spike-based signals offer the highest temporal and spatial resolution, they also consume significantly more power compared to low frequency local field potential (LFP) signals. However, spike-based signals have been shown to provide superior performance in neural decoding applications compared to LFPs. To drastically reduce the overall system power consumption, we propose exploiting an intention estimation unit using LFPs to realize an asynchronous BCI. For neural decoding, the system employs spike-based signals to maintain robust performance.

The rest of this chapter is organized as follows. Section 10.2 briefly reviews various neural signal features and compares the effectiveness of employing different features for various operations of BCIs. Section 10.3 presents realizing the asynchronous BCI paradigm using intention estimation. Section 10.4 details the hardware realization of the designed BCI system and compares it to state-of-the-art BCI systems. Section 10.5 discusses the implementation of an FPGA prototype system for verification of the designed hybrid asynchronous BCI. Section 10.6 makes some concluding remarks.

10.2 Neural Signal Features

Invasive intracortical recordings provide a significantly higher signal-to-noise ratio compared to non-invasive methods. Intracortical neural activity can be represented at various scales, as shown in Fig. 10.2. At the largest scale, the neural activity is represented by the LFPs, which provide relatively slow oscillations, reflecting the synaptic activities of neural populations within 500 μm of the recording electrode [142, 29, 143]. While LFPs do not provide the spatial resolution to resolve the activities of individual neurons, they offer a large-scale perspective on the spiking activity of groups of neurons. At the medium scale, the slow oscillations of the LFPs are removed from the recorded neural signals by applying a high-pass filter at 300 Hz. Doing so isolates the signal to the activity of neurons within a 100 μm radius around the recording electrode.

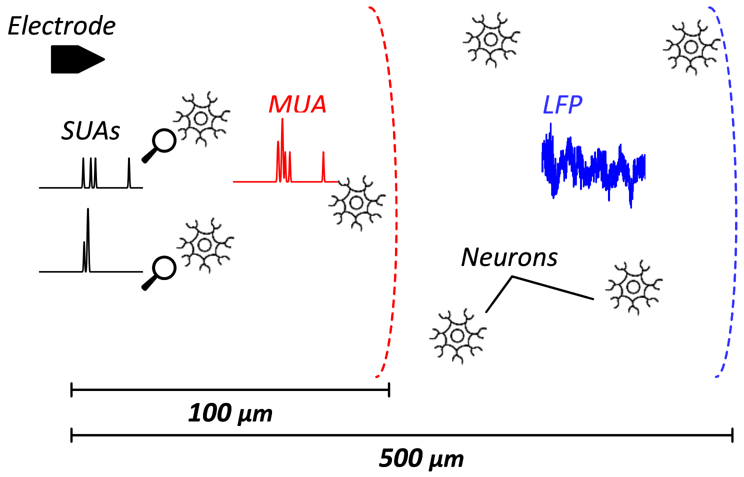


Figure 10.2. The various scales of neural signals using intracortical recording electrodes.

While a single channel of filtered neural signals may be feasible for wireless transmission and subsequent processing, MEAs employ hundreds to thousands of recording sites, making it infeasible to transmit the filtered neural signals directly. For example, a single 96-channel MEA sampling at 10 kS/s with a resolution of 16 bits will generate over 15 Mbps of data. For recording configurations employing multiple MEAs for different brain regions, the amount of data is staggering, as the increased data rate would incur more power consumption for wireless transmission. Given a wireless transmission power of 158 pJ/bit [15], a BCI with 96 channels would require 2.37 mW of power for transmitting the filtered neural signals directly. Wireless brain-implantable devices have a strict power limitation of 7.7 mW to prevent heating of brain tissue beyond 1 °C [251]. The wireless transmission alone would require over 35% of the available power budget. To reduce the total amount of data transmitted, and thus the amount of wireless transmission power, in vivo feature extraction, which involves performing in vivo digital signal processing to select specific signal features, is used. As modern recording devices employ MEAs with channel counts in the thousands, the challenges related to power consumption for recording and transmission are significantly exacerbated [8, 7, 9].

One commonly employed method for reducing the amount of data gathered is to detect action potentials (spikes). After filtering, spikes are emphasized against the background noise via signal pre-emphasis methods, such as the non-linear energy operator [113]. The amplitude of the pre-emphasized signal is compared to a threshold value, set to a scaled estimate of the pre-emphasized signal's standard deviation σ [113]. Crossing this noise threshold indicates that a spike is present at a given recording channel. The number of threshold crossings (TCs) is often accumulated over small time windows on the order of a few milliseconds, a process known as spike binning, resulting in spike counts. Note that because the spike counts denote the presence or lack of spikes, they provide a discretized representation of the neural activity. To transform the spike counts from a discrete signal into continuous multi-unit activity (MUA), the spike counts are convolved with a Gaussian kernel, realized using a finite impulse response (FIR) filter, resulting in an estimate of the spike density function [257]. Conventional spike detection systems impose a biologically plausible one-millisecond refractory period, which denotes the time that must pass before a particular channel can detect another spike. Thus, the number of spikes that can be detected on a given channel over a time period of τ ms is τ spikes. This reduces the output data rate of MUAs significantly compared to the filtered neural signal. For example, considering a 96-channel MEA with 10 ms spike bins (i.e., 4 bits per bin), the required output data rate is 38.4 kbps, a near $400\times$ reduction compared to transmitting the filtered neural signals directly. This would reduce the wireless power consumption to 6.06 μW, less than 1% of the available wireless power budget.

To achieve the lowest scale of granularity, the activities of individual neurons near the tip of the recording electrode are distinguished from one another. While the spike waveforms of different neurons are similar, the distance between the recording electrode and the individual neurons will distort the spike waveform shapes [167], which allows for the activities of different neurons to be distinguished from one another through the spike sorting process [138, 62]. Once sorted, the spike counts for individual neurons can be converted into single-unit activity (SUA) in the same manner as that for generating MUAs.

Conventional LFP processing involves computing the band-limited power within a specific frequency band. The specific frequency bands are chosen based on the underlying study and the relationships between the neural activity and physiology. Historically, the use of these frequency bands is based on correlations between changes in the band-limited power and different physiological states. For example, variations in the beta band (12 – 30 Hz) are correlated with motor activities, such as movement and/or movement preparation [265, 285, 286]. LFP waveform variations may also be used as biomarkers of a particular mental or physiological state, such as intermittent spindles (11 – 14 Hz) in non-rapid eye movement sleep [287], or exhibiting different behaviors among different stages of sleep [263, 264]. The majority of prior work has related the bandpower to a given measurement using linear relationships. Modern machine learning (ML)-based approaches, however, achieve similar or better results without requiring pre-processing of the LFPs [132, 288].

Conventional AFE circuitry requires an average power of 7.4 μW per channel for spike-based signals sampled at 10 kS/s [160, 289, 290]. From the perspective of in vivo processing, employing the LFP from a single channel poses considerable advantages compared to multi-channel spike-based processing. First, the LNA power consumption depends on various factors, such as the acceptable input-referred noise and whether specialized circuitry is required to mitigate flicker noise [291]. Additionally, the LNA power consumption is proportional to the frequency bandwidth of the underlying signal [131]. The ADCs, commonly realized using successive approximation register (SAR) architectures, have a power consumption that is roughly linearly proportional to the sampling rate [162, 163]. Thus, because both the sampling rate and bandwidth are five times smaller for LFPs than those of MUAs, the power consumption of the AFE when employing LFPs is also reduced by a factor of five, about 1.5 μW per channel. This fact motivates the design of a hybrid asynchronous BCI system employing two signals for specific tasks. During the idle state, a single channel of LFP data is processed to reduce the power consumption for both the signal acquisition circuitry as well as for in vivo signal processing. In contrast to MUAs and SUAs, which are continuous representations of discrete temporal events, the features derived from LFPs are continuous valued signals. Note that the reduced sampling rate for LFPs inevitably reduces the overall amount of data that is generated. Additionally, the band-limited power is often computed over small time windows on the order of tens of milliseconds, reducing the output data rates further. For example, a BCI computing the band-limited power over 10 ms windows with 16 bits per sample would require an output data rate of 1.6 kbps per channel, or 153.6 kbps given a 96-channel MEA.

For neural decoding, various studies have shown that useful information can be extracted from the LFP [118, 146, 147]. For example, LFPs recorded from cognitive regions of the brain, such as the posterior parietal cortex, have allowed high-level cognitive decoding compared to low-level continuous motor control [118, 146]. In [147], the authors reported that various movement intentions, including kinematic trajectory, imagined movement endpoints, and movement type, could be decoded reliably from LFPs. However, employing spike-based signals for motor decoding has demonstrated consistently higher performance compared to LFP-based features [144, 35]. Various researchers have studied utilizing both MUAs and LFPs for BCI applications. In [292], the accuracy of velocity decoding was improved by employing both MUAs and LFPs when using a Kalman filter-based decoding algorithm. In [293], similar results were demonstrated, in which LFPs, MUAs, and electro-corticography (ECoG) signals were combined to improve overall decoding performance. While the integration of multiple types of signals has been shown to improve performance, the relative accuracy of the decoded output is not a strict requirement for practical BCI applications. For example, the “thought-to-text” BCI presented in [11] achieved a significant improvement in both communication speed and accuracy by incorporating auto-correct features into the BCI system. Additionally, it has been shown that modern machine learning-based decoding algorithms can tolerate input errors and achieve sufficient decoding performance [131, 132]. Rather than employing both MUAs and LFPs for increased decoding performance, we employ LFP features when the BCI is inactive and the intention estimation unit is monitoring the brain's neural activity while MUAs are employed, and MUAs are employed when the BCI is in active mode for neural decoding.

10.3 Asynchronous BCIs using Intention Estimation

Identifying changes in brain state has been previously studied. For example, studies focusing on EEG-based intention estimation monitored for an increase in band-power within the 1 – 4 Hz frequency band [245, 246]. ECoG-based intention estimation has been performed by monitoring changes in the power of specific frequency bands to distinguish between active and idle states [243]. For brain-switches employing MUAs, increases in firing rates have been monitored to detect changes in the BCI state [116, 132]. Three main approaches by which intention estimation methods are designed [244] are shown in Fig. 10.3. The first scheme uses a threshold-based estimation, such that whenever the neural signal feature crosses a pre-defined threshold, a mental state transition is assumed. The pre-defined threshold value is derived from training data where the mental state transition is known. The second approach uses a classification algorithm to produce a continuous output signal, such as the probability of a mental state transition at the current time. This probability is then compared against a pre-defined probability threshold value, similar to the first method. In the third discrete classifier-based approach, the output of the classification algorithm is used directly.

While most approaches use two states, the output of the classifier can be expanded to distinguish between a variety of discrete mental states, such as "idle", "planning", and "movement". In our previous work [132], we found that classifier-based algorithms employing recurrent neural networks (RNNs) perform relatively accurate intention estimation. An interesting finding was that using a relatively small subset of electrodes with MUAs well-correlated to the onset of user intention provided sufficient performance compared to employing all recording channels. This finding allowed for reduced power consumption during idle periods, as only the

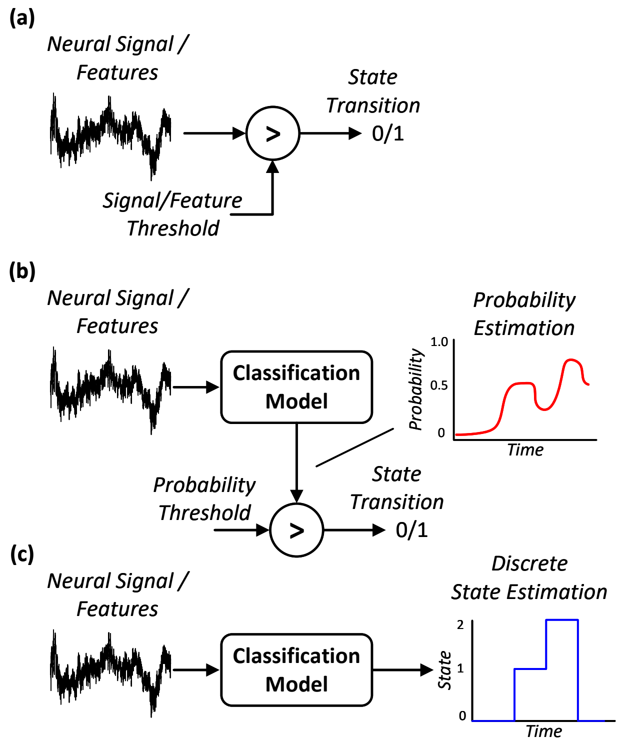


Figure 10.3. The block diagrams of (a) threshold-based, (b) classifier-based, and (c) discrete state classifier-based intention estimation.

eight most correlated channels are enabled for processing. Our recent findings [294] suggest that even a single channel of the LFP contains sufficient information required for accurate intention estimation. Moreover, the performance of the single channel LFP-based brain-switch does not require a complex correlation analysis prior to real-time operation. While both MUAs and LFPs can be used, the significantly lower power consumption required for the acquisition of LFPs makes them a preferable candidate for intention estimation.

As shown in Fig. 10.4, the designed hybrid BCI operates in two distinct modes. During the inactive mode, the BCI transmits LFP features. Off-chip in silico processing is used to perform intention estimation using the LFP features. When an intention is detected, the BCI is switched to the active mode, in which it transmits spike features for in silico neural decoding.

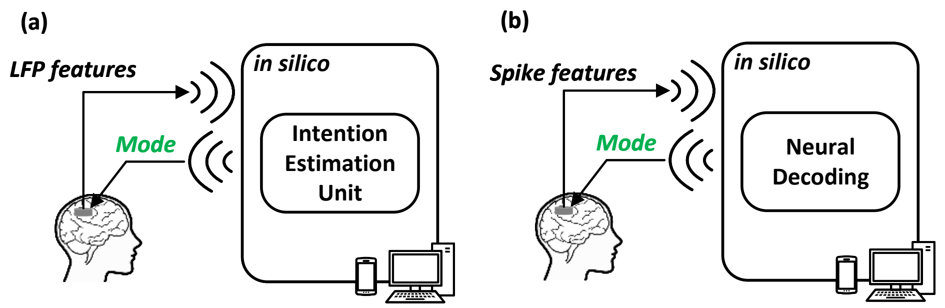


Figure 10.4. The block diagram of the designed hybrid BCI in the (a) inactive mode, and the (b) active mode.

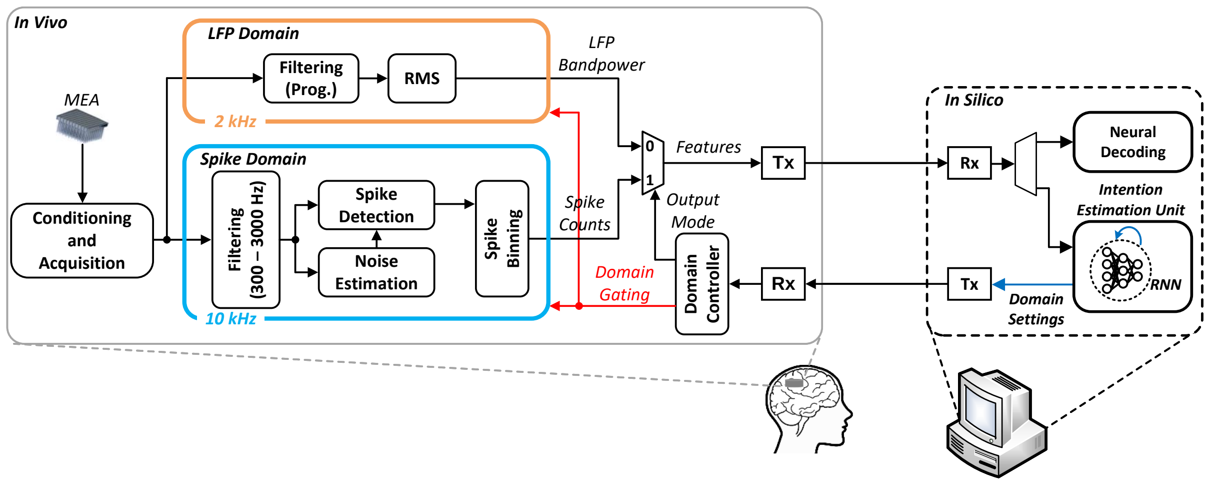


Figure 10.5. The top-level block diagram of the proposed hybrid asynchronous BCI.

10.4 Hardware Realization and Comparison

The top-level block diagram of the proposed hybrid BCI is shown in Fig. 10.5. The in vivo interface consists of an implantable MEA, the AFE, and two different signal processing domains, the LFP domain and the spike domain. The AFE operates in a hybrid mode, dynamically switching between a low-frequency mode to obtain LFPs at a rate of 2 KS/s and a high-frequency mode to obtain signals at a rate of 10 KS/s. Fig. 10.6(a) shows an example configuration of a multi-channel AFE. Multi-channel AFEs commonly employ multiple LNAs, which are time-multiplexed to a single analog-to-digital converter *ADC* [160]. The sampling control logic *SCL* switches the select line *Select* to sample from different LNAs. As shown in Fig. 10.6(b), the *Select* can be toggled at the switching frequency f_c to obtain an interleaved multi-channel data stream with neural signals obtained at a rate of f. To select a particular signal channel M, *Select* can be set to M, and the clock can be gated, as shown in Fig. 10.6(c). Thus, the downsampling and channel selection logic need not be realized in the digital domain.

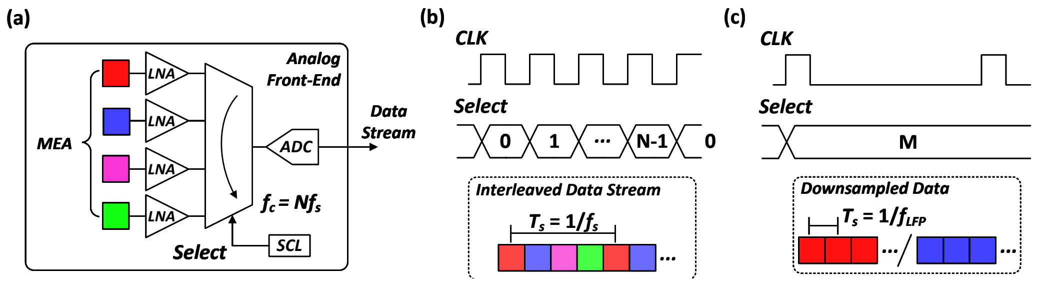


Figure 10.6. The block diagram of (a) the AFE configuration, (b) the interleaved sampling, and (c) the channel-configurable LFP downsampling.

10.4.1 LFP Domain

Following downsampling and channel selection in the AFE, the signal is filtered within a specific frequency band to estimate the power within that band. In our previous work [294], we found that filtering the neural signals between the low-band (LB-LFP: 8 – 60 Hz), mid-band (MB-LFP: 60 – 300 Hz), and high-band (HB-LFP: 150 – 500 Hz) all provided relatively accurate intention estimation. However, it's possible that for certain subjects or datasets, any one of the three bands may outperform the others. Thus, the filtering can be configured dynamically to select any one of the three frequency bands at a time. Finally, the power of the signal within the given frequency band is estimated by computing the root-mean-square (RMS) of the filtered signal. The bandpower is computed within eight millisecond non-overlapping windows to generate outputs at a rate of 125 Hz, a reduction by a factor of 16. Compared to the 3.072 Mbps output bitrate for 16-bit ADCs for 96 channels sampled at 2 KS/s, the output bitrate is reduced to 3.75 kbps for estimated bandpower produced at 125 S/s in 30-bit resolution. The block diagrams of the band-pass filter and RMS estimation units are shown in Figs. 10.7(a) and (b), respectively. Specific details of the IIR band-pass filter and the RMS estimation unit implementations can be found in our earlier work in [294].

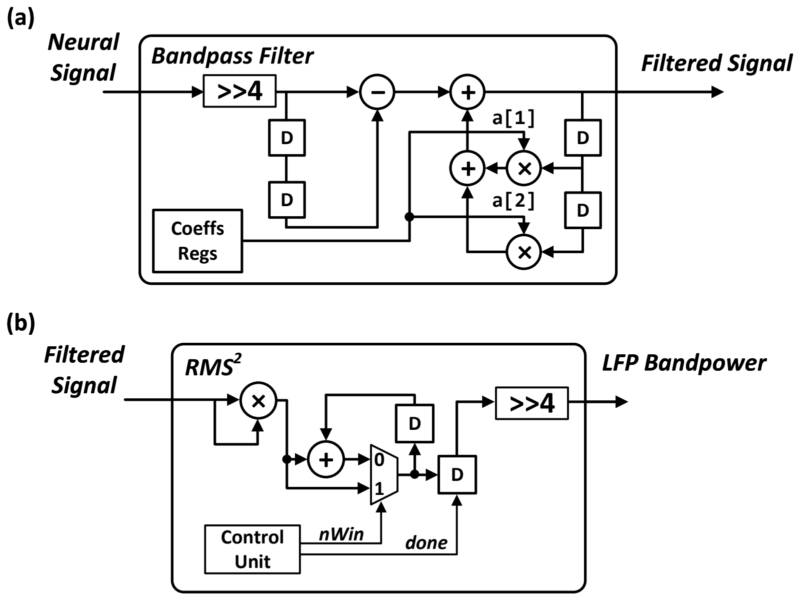


Figure 10.7. The block diagram of (a) the second-order band-pass filter and (b) the RMS estimation unit.

10.4.2 Spike Domain

The interleaved data streams are passed to the spike processing domain which performs filtering, noise estimation, spike detection, and spike binning. Conventional spike processing involves band-pass filtering the neural signal between 300 Hz and 3000 Hz and removing low frequency LFPs from the signal to isolate spiking activities [113]. However, it has been shown that an estimate of the band-pass filtered signal can be obtained by removing low-frequency LFPs with a moving average filter [132]. The block diagram of the utilized approximate LFP-removal filter is shown in Fig. 10.8. It consists of interleaved delay line (IDL) registers, an adder tree, a right shift unit, and a subtraction unit. Due to the interleaving of eight channels of data, every 8-th register in the IDL is passed to the adder tree, which computes the moving average of the given channel over the past 8 samples, i.e., over 800 μs. The moving average filter implements a low-pass filter with a cutoff frequency $f_{lp} = 0.443 f_s/N$, where 0.443 denotes an approximate for the frequency at which the magnitude drops to $1/\sqrt{2}$, f_s denotes the sample rate of the input signal, and N denotes the number of taps of the moving average filter [295]. For $N = 8$ and

$f_s = 10$ kHz, f_{lp} is equal to 558 Hz. Although f_{lp} is relatively high, our previous results do not show significant loss of spiking activities compared to the conventional filtering methods [132]. The estimated LFP is then subtracted from the neural signal to approximate the high-pass filtered signal.

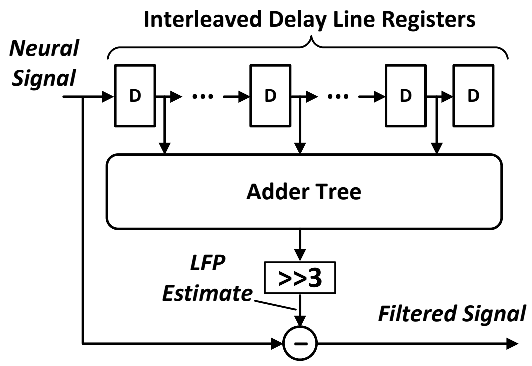


Figure 10.8. The block diagram of the approximate LFP-removal filter.

Following the signal filtering, spike detection is a two-step process involving noise estimation and thresholding. First, the neural signals are pre-emphasized to readily accentuate high-frequency spiking activity from the ambient background noise [138]. Commonly employed pre-emphasis methods compute the absolute value or the energy of the signal via the Teager-Kaiser energy operator [296]. The absolute value has the benefit of a lower computational complexity while the Teager-Kaiser energy operator offers better ambient noise suppression [113]. The RMS of the pre-emphasized neural signal is then computed as an estimate of the ambient noise. A scaled version of the RMS is then used as a threshold for detecting spikes. Multiple studies have found that ML-based decoding algorithms can tolerate errors within the spike-based signals [131, 132]. Therefore, while the Teager-Kaiser energy operator performs better than absolute value-based methods for spike detection, we propose to employ the absolute value for spike detection. Additionally, our previous findings in [132] suggest that it may be feasible to use a relatively simple average of the pre-emphasized signal instead of the RMS. The result is a multiplication-free spike detection circuitry that only requires addition and shift operations. Fig. 10.9 shows the block diagram of the designed spike detection circuitry. The filtered signal consists of the filtered interleaved data samples, and the block diagram shown in Fig. 10.9 processes the signals for multiple channels sequentially. The *Channel Control Unit* reads and updates the correct threshold value stored in the *Threshold Memory*. Similarly, the *Spike Binning Unit* counts the spikes for each of the interleaved channels over the past ten milliseconds to produce spike counts. To prevent the same threshold crossing from being interpreted as multiple spikes, the *Channel Control Unit* imposes a biologically plausible one millisecond refractory period that prevents the *Spike Binning Unit* from incrementing multiple times per spike event. More details about the spike detection implementation are presented in our earlier work in [132].

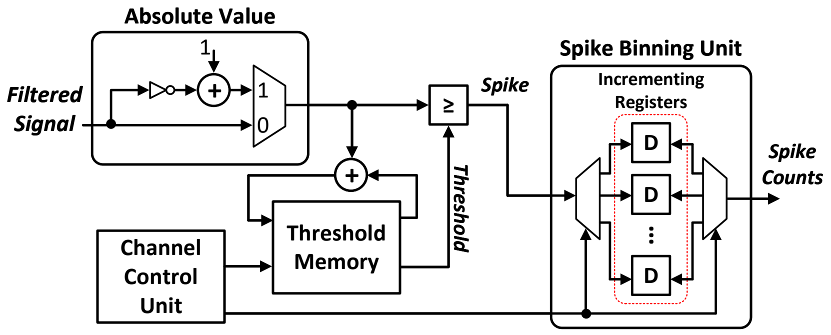


Figure 10.9. The block diagram of the absolute value-based spike detection and binning units.

10.4.3 ASIC Implementation and Comparison

Fig. 10.10 shows the application-specific integrated circuit (ASIC) layout of the designed and implemented hybrid BCI in a standard 180-nm CMOS process. The ASIC is designed to support a 96-channel MEA with multi-channel spike detection using 12 spike detection modules. The ASIC layout of the designed hybrid BCI shown in Fig. 10.10 consists of 12 template memory units (*TM1 – TM12*). The spikes detected by the multi-core spike detection circuitry are passed to the *SBU* to compute the binned spike counts. The LFPs of a single channel are downsampled and processed within the LFP bandpower unit *BPU*.

The operating frequency of the ASIC is 80 kHz with a 1.8V supply voltage. The design has three distinct operating phases, as given in Table 10.1. During the noise estimation phase,

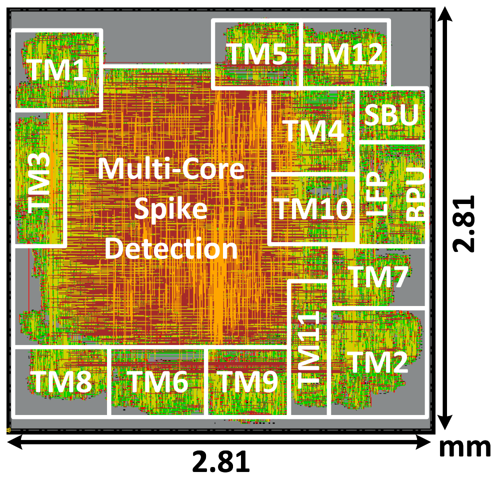


Figure 10.10. The ASIC layout of the designed and implemented hybrid BCI.

which lasts approximately 819 ms (i.e., 2^{13} samples at a rate of 10 KS/s), the spike detection unit estimates initial noise thresholds for all recording channels while simultaneously performing LFP bandpower estimation on a given channel. The next phase is the LFP bandpower estimation, in which the spike detection circuitry is disabled via clock gating and its switching activity is reduced to zero. The only active circuitry during the LFP bandpower estimation phase is the LFP processing domain, which computes the bandpower of a given recording channel. The bandpower samples are transmitted to the in silico interface to detect the user's intention for involving in the BCI application. When the in silico interface detects the user's intention, the spike detection and binning phase is enabled, which will gate the clock of the LFP bandpower estimation module and enable the spike detection module and the binning circuitry modules. The circuits' toggling rate during each phase was estimated by simulating the post-place-and-route netlist with Xilinx Vivado and saving the toggling rate information of each net into a Switching Activity Interchange Format (SAIF) file. Three distinct SAIF files are generated to estimate the power during each of the three different operating phases. The SAIF files are passed to Cadence Innovus, which performs the power estimation. It was observed that the power consumption of the design drops by over 99% during the LFP bandpower estimation phase compared to the initial noise estimation phase. This is due to the significantly lower operating frequency of the

LFP domain relative to that of the spike domain. Additionally, because the LFP domain operates only on a single channel of data, the LFP domain does not require an operating frequency at a multiple of the signal's sampling rate.

Table 10.1. The estimated power consumption of the hybrid BCI for three different operating phases.

Operating Phase	Total Power (μW)	Power/ch. (μW)
Noise estimation	173	1.82
LFP bandpower estimation	0.52	
Spike detection and binning	186	1.93

Table 10.2. The characteristics and implementation results of various neural signal processing ASICs.

Work	Ours	Ours [294]	Ours [132]	[18]	[278]	[279]
Signal modality	MUAs + LFP	LFP	MUAs	MUAs	SBP	SBP
Technology (nm)	180	180	180	180	180	180
Supply voltage (V)	1.8	1.8	1.8	1.8	1.55	0.625
Area per channel (mm^2)	0.08	0.12	0.03	0.03	0.07	–
Power per channel (μW)†	1.93/0.52‡	0.11	0.63	1.5	0.21	3.6
Energy dissipation (mJ)◇	175	39.6	227	540	75.6	1300

† Normalized to a 180-nm CMOS process with a 1.8 V supply and accounting only for digital backend.
‡ Power consumed while the LFP domain is active.
◇ Assuming 15 minutes of active BCI use over a one-hour time period with 100 recording channels.

Table 10.2 gives the characteristics and implementation results of various previously-published neural signal processing ASICs. To approximate energy dissipation, we assume 15 minutes of active BCI operation over a one-hour time period using 100 recording channels. We employed absolute value thresholding and mean-based noise estimation in our earlier work, focusing on spiking signal processing [132]. Similarly, the LFPs were removed by eliminating the low-frequency oscillations with a moving average filter. An analog implementation of an energy-based threshold estimator for spike detection has been reported in [18]. There has been recent interest in the spiking band power (SBP), which is a relatively low-frequency neural signal that has been shown to be well-correlated to multi-unit activities with less signal conditioning and processing requirements [266]. The authors in [278] and [279] target the SBP for neural decoding, achieving relatively low power consumptions of 0.21 μW and 3.6 μW per channel, respectively.

Compared to our previous designs in [294] and [132], which focused specifically on LFPs and MUAs, respectively, it is apparent that targeting a single signal modality allows for lower power consumption per recording channel. This is due to the synthesis tool optimizing the design for a single given operating phase, whereas the multimodal approach requires additional logic and memory to handle both LFP and spike domains. For applications targeting low-frequency signals only, the total energy dissipation is therefore significantly lower than that for systems employing higher frequency MUAs. However, it should be noted that MUAs have been shown to result in a more reliable decoding compared to LFP-based decoding [297, 298, 299]. Recent studies show that SBP may be a viable alternative to MUAs for motor decoding applications [266, 300]. While the spatial specificity of the SBP may be useful for motor decoding, due to its high correlation to spiking activity, it is not yet known whether the SBP can be used for intention estimation. We have previously shown that MUA-based intention estimation is feasible [132]. Employing the SBP may also be feasible, but may similarly require an exhaustive learning phase for estimating optimal recording channels due to its spatial specificity similar to that of MUAs [132].

10.5 FPGA-based BCI System Prototype

For functional verification of the designed hybrid BCI, we implemented a FPGA-based system prototype as shown in Fig. 10.11. The in vivo interface is modeled using a Digilent Zybo Z7 board hosting a Xilinx XC7Z020 Zynq FPGA, while the in silico interface is modeled using a Windows PC workstation. Neural recordings are stored on an SD card that is connected to the Zybo Z7 board. The single core ARM Cortex A9 processor executes a C program to manage the reading of the neural signals via the SD interface controller. The *SD controller* writes data into a *Memory Buffer* within the *Programmable Logic* (PL) of the Zynq SoC, which hosts the hybrid BCI itself. The ARM Processor communicates with the PL via the AXI4 interface to receive the generated neural signal features (LFP bandpower or spikes) from the PL. The *Serial Controller* is used to interface the Zybo Z7 board to the Windows PC Workstation, which is executing a RNN and a Temporal Convolutional Network (TCN). The *Intention Estimation RNN* is used to detect the intention from the LFP bandpower computed on the PL. When the RNN detects an intention, a serial signal is transmitted back to the ARM processor, which will trigger the hybrid BCI to transition to the spiking mode. During the spiking mode, the spike counts are transmitted via the serial interface to the *Velocity Decoding TCN*, which will perform the kinematic decoding and display the inferred kinematic variables on the screen.

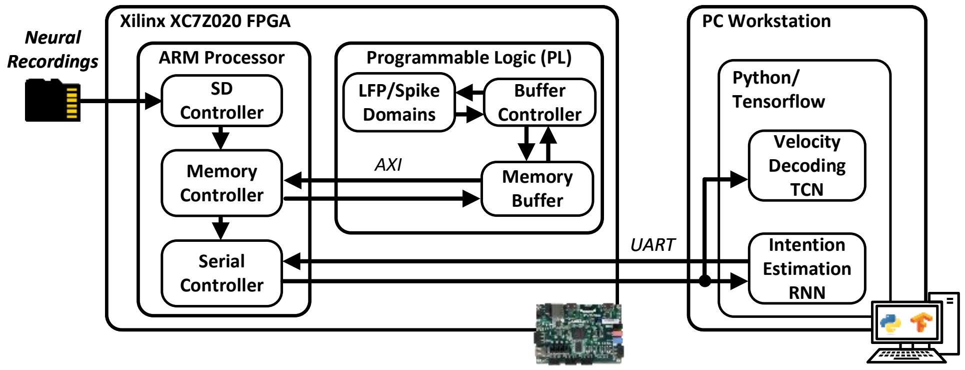


Figure 10.11. The block diagram of the implemented FPGA-based prototype system for functional verification of the designed hybrid asynchronous BCI.

Table 10.3. The resource utilization of the implemented hybrid asynchronous BCI on a Zynq XC7Z020CLG400-1 FPGA.

Resource	Utilization	Available	Utilization %
LUT	5200	19960	37.5
LUTRAM	2946	17400	16.9
FF	25068	106400	23.5
BRAM	72	140	51.4
DSP	7	220	3.1

The resource utilization of the implemented hybrid asynchronous BCI on a Zynq XC7Z020CLG400-1 FPGA is given in Table 10.3. The Zynq-based design was implemented using the Xilinx Vivado block designer and the Xilinx Vitis IDE. The Zynq Processing System (PS) is connected to six dual-port block RAMs (BRAMs) via individual AXI BRAM controllers. The data from each 32-bit BRAM is concatenated and passed as a 192-bit multi-channel input bus to the *LFP/Spike Domains* via a custom BCI wrapper module. To save the appropriate outputs from each respective domain, two additional dual-port BRAMs are employed to store the LFP bandpower and spike count outputs, respectively. Due to the configuration of the spike detection module, which consists of 12 detection cores each time-multiplexed among 8 channels, the spike counts require 36 bits and are stored in a 64-bit BRAM for practical memory indexing purposes. In addition to the instantiated LFP and spike processing domains, the BCI wrapper is an AXI peripheral that contains various control and status registers, allowing the PS to monitor the resulting operation.

Table 10.4. The AXI control and status registers of the developed BCI Wrapper.

Base address offset	Register	Application
+0	`START`	Starts processing batch of data
+8	`BATCH_SIZE`	Defines data batch size
+12	`ENABLE`	Enables domains for processing
+16	`RMS_COUNT`	Returns the number of bandpower values written to LFP BRAM
+24	`LFP_BAND`	Defines the bandpower frequency
+28	`LFP_CHANNEL`	Defines which interleaved channel to employ for bandpower estimation
+30	`LFP_CORE`	Defines which data core to employ for bandpower estimation
+34	`LFP_EN`	Enables the LFP domain
+38	`SPIKE_EN`	Enables the spike domain
+46	`NOISE_STATUS`	Returns the status of the noise estimation phase
+50	`SPIKE_COUNT`	Returns the number of spike count values written to Spike BRAM

A list of various AXI control and status registers of the BCI wrapper is given in Table 10.4. The `START` signal is used to enable processing of a batch of data. The amount of data in a batch is indicated by the `BATCH_SIZE` register. Note that the amount of data refers to the number of rows of memory to read rather than the total number of bytes read from the BRAMs. The `ENABLE` register is used in conjunction with the `LFP_EN` and `SPIKE_EN` to control which of the two domains is currently operating. During the intial noise estimation phase, the spike domain is enabled until the `NOISE_STATUS` register indicates that the noise estimation phase is complete. The LFP domain supports computing the bandpower for the three given frequency bands, and the `LFP_BAND` is used to select one. Since the data passed to the 12 spike detection cores is interleaved, the first row of the buffering memory stores the values $\left[x_1[t], x_9[t], x_{17}[t], \ldots, x_{89}[t]\right]$, where $x_i[t]$ denotes the neural signal of channel i. Therefore, every eight rows of the buffering BRAM contain the values for one single time step at the faster sampling rate of the spike domain. As data is interleaved, the `LFP_CHANNEL` and `LFP_CORE` registers are used to control which of the 96 input channels is processed by the LFP domain. The `LFP_CORE` register value is used to part-select between one of the twelve concatenated 16-bit values to select a particular set of channels, such as 1 – 8 passed to detection core 1, or 89 – 96 passed to detection core 12. To select a particular channel among the set of eight channels, the `LFP_CHANNEL` is used to offset the memory read to emulate the AFE downsampling. For example, setting `LFP_CORE` to 2 and `LFP_CHANNEL` to 1, both zero-indexed, denotes the signals recorded from channel 17 (zero-indexed), as core 3 processes the data from the zero-indexed channels 16 – 23.

For dataflow control, there are three interrupt signals that the PL may issue to the Zynq PS. First, as the spike detector requires an initial estimate for the noise threshold, data packets are processed while the noise interrupt `NOISE_INIT` is zero. When the initial estimate is obtained after 2^{13} samples of data, the `NOISE_INIT` is asserted to one. Following the initial noise estimation phase, the device operates in either the LFP or the Spike mode, depending on configuration settings transmitted from the PC workstation via the UART interface. When set to either mode, the device reads a pre-defined batch of data from the six BRAMs and passes it through the corresponding domain. When in the LFP mode, the bandpower RMS valid interrupt `RMS_VALID` is asserted high after the processing of the current batch is finished. The `RMS_VALID` interrupt is handled in the C code by reading an appropriate number of values from the LFP Bandpower BRAM and transmitting them via the UART interface to the PC workstation. Similarly, the spike count valid interrupt signal `SPIKE_VALID` is asserted high when the current batch is finished processing following spike detection and binning. The C code also reads the appropriate number of values from the spike count BRAM and transmits them through the UART interface to the PC workstation. The `RMS_VALID` and `SPIKE_VALID` interrupts trigger a handler to read from the corresponding BRAMs, using the value stored in the `SPIKE_COUNT` or `RMS_COUNT` registers to read the computed features.

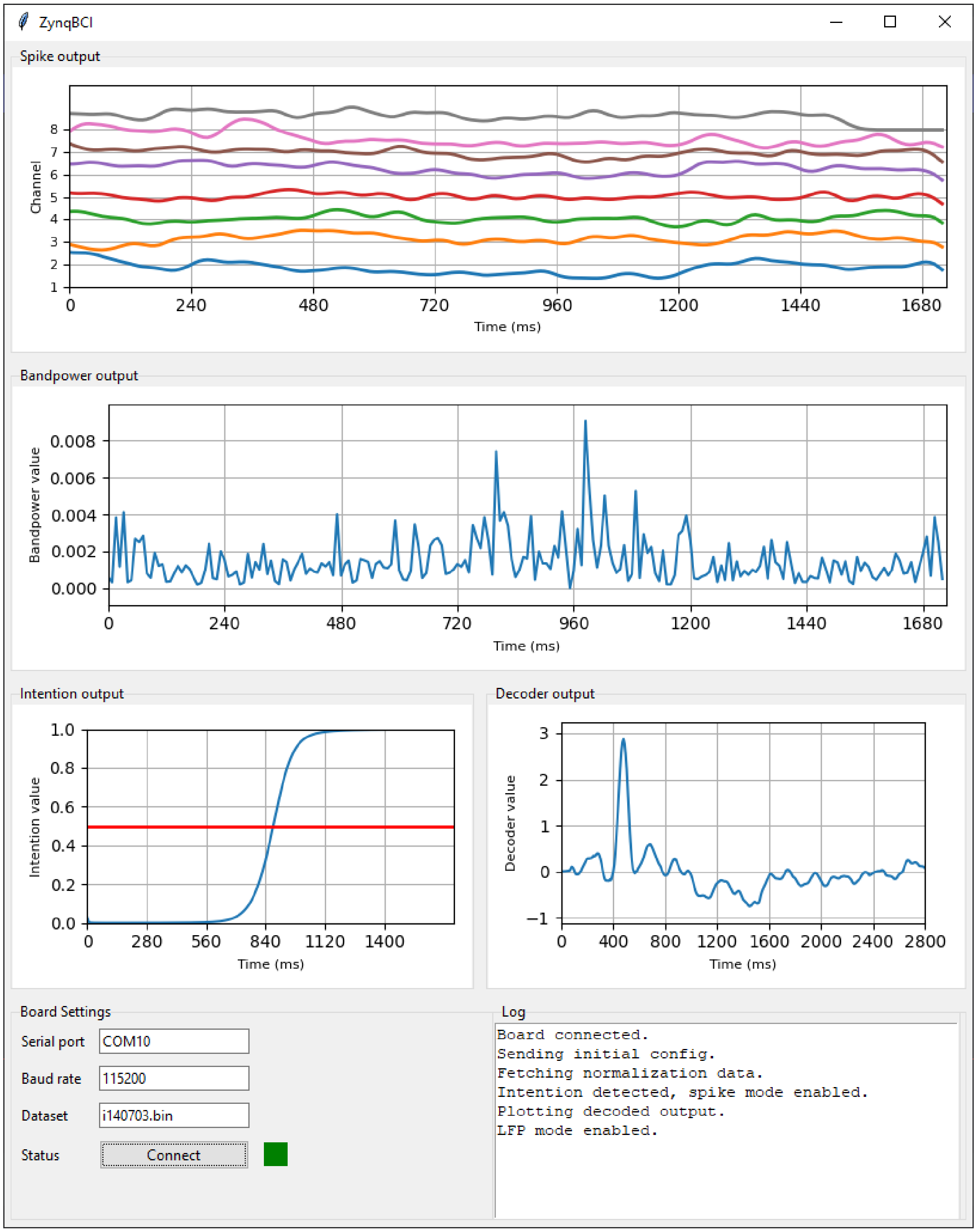
Fig. 10.12 shows the custom-developed graphical user interface (GUI) in Python for functional verification of the designed and implemented hybrid BCI. After the user enters the appropriate serial port, device baud rate, and the underlying dataset, the Connect button is used to establish a connection between the workstation and the Zynq-based hybrid BCI. Then the GUI displays four subplots for visualizing smoothed spike count outputs, bandpower outputs, as well as the outputs from the intention estimation RNN, and the velocity decoder TCN.

```
Algorithm 1: Initialization and Noise Estimation
    Input: Pre-trained ML models, serial connection, batch size, LFP band, LFP core,
           LFP channel
    Result: BCI wrapper configuration, spike detection thresholds
    1. Transmit wrapper configuration via serial interface;
    2. Load pre-trained ML models into the GUI;
    3. Enable Spike detection processing modules; Collect spike counts over three
       seconds;
    4. Smooth spike counts using a Gaussian kernel;
         Window length = 16, $\sigma = 24$ ms;
    5. Enable LFP processing modules; Collect bandpower data over three seconds;
    6. Compute normalization parameters and Z-score spike counts and LFP bandpower;
```

Algorithm 1 describes the initialization process of the GUI and the prototype. The pre-trained ML models for intention estimation and velocity decoding are loaded and initial configuration parameters are transmitted to the FPGA. After initial noise threshold estimation for spike detection, spike count and LFP bandpower data are collected over three seconds. The Z-scoring normalization parameters are then computed and stored for future use. Following this initialization and noise estimation process, Algorithm 2 presents the remaining operations of the prototype system. The default operating phase is the LFP processing, which involves sending a batch of data and passing the received bandpower samples to the intention detection RNN. Once the output of the intention detection RNN exceeds a threshold of 0.5, the spike processing mode is enabled. Then over the next three seconds, batches of data are sent to the FPGA and the received spike counts are smoothed and passed to the decoding TCN. After three seconds, the

Algorithm 2: Intention Estimation and Velocity Decoding

```
Input: Algorithm 1 completed
Result: Movement intention detection, kinematic velocity decoding
1. Enable LFP processing phase;
2. Send a batch of data;
3. Receive bandpower samples through serial port;
4. Normalize bandpower samples using pre-computed normalization parameters;
5. Pass normalized bandpower values to intention detection RNN, produce intention
  output $y_{\text{int}}$;
if $y_{\text{int}} \geq 0.5$ then
    Go to step 6;
else
    Go to step 2;
end
6. Enable spike processing phase;
7. Send a batch of data;
8. Receive spike counts through serial port;
9. Apply Gaussian smoothing and normalize spike counts using pre-computed
  normalization parameters;
10. Pass normalized smoothed spike counts to velocity decoding TCN; produce
  velocity output $y_{\text{vel}}$;
if Time spanned $> 3$ seconds then
    Go to step 1;
else
    Go to step 7;
end
```



Serial port COM10
Baud rate 115200
Dataset i140703.bin

```
Board connected.
Sending initial config.
Fetching normalization data.
Intention detected, spike mode enabled.
Plotting decoded output.
LFP mode enabled.
```

Figure 10.12. The custom-developed graphical user interface for functional verification of the designed and implemented hybrid asynchronous BCI.

system is switched back to the LFP phase and the process is repeated.

10.6 Conclusion

This chapter presented the design and implementation of a hybrid asynchronous brain-computer interface (BCI) system by employing two distinct signal modalities, multi-unit activities and local field potentials (LFPs). The hybrid BCI uses an intention estimation unit to enable asynchronous operation and utilizes low-frequency LFPs to optimize device power consumption. It was shown that the power consumption was reduced by over 73% during user intention estimation. When user intention is detected via an off-chip in silico recurrent neural network, the hybrid BCI is switched to the spike processing phase for accurate neural decoding. The ASIC layout in a standard 180-nm CMOS process was presented and was shown to dissipate less power than alternative single-modality neural signal processor implementations. An FPGA system prototype realized on a Zynq System-on-Chip and a custom-developed graphical user interface for functional verification of the hybrid BCI were also presented.

10.7 Publications

The following article has been submitted for publication as a result of the work in this chapter: D. Valencia, P. P. Mercier, and A. Alimohammad, "A hybrid asynchronous brain-computer interface for low-power and efficient neural decoding," submitted to IEEE Transactions on Biomedical Circuits and Systems.

Chapter 11

Conclusion

This dissertation discussed various approaches to mitigating the challenges currently faced by BCI technology, which hinder progress toward widespread and practical implementation. Firstly, advances in neural recording interfaces have increased the overall amount of data generated, which raises system-level power consumption due to the requirements for signal acquisition and wireless transmission. This dissertation proposed addressing this data overload by reducing the amount of data generated by the recording and processing circuitry. Spike detection was used to represent the neural signal as a discrete representation of neural activity.

Chapter 2 detailed the design of a spike detection circuitry that autonomously adapts to changes in the ambient noise to maintain robust performance. The results demonstrated that our spike detection circuitry extends device lifespan by over 13% compared to the state of the art. Furthermore, the candidate spike detection and noise estimation methods, namely the non-linear energy operator (NEO) and root-mean-square (RMS), displayed higher neural decoding performance than alternative methods. To further reduce the data rate, spike sorting was explored as a method to distinguish the activities of specific neurons near the recording electrode.

Chapters 3 and 4 discussed methods for implementing efficient machine-learning (ML)-based spike sorting using binarized neural networks (BNNs). Chapter 3 presented the first practical implementation of BNNs for neural spike sorting, demonstrating how the designed BNN-based spike sorting circuit achieves high performance while requiring less silicon area and consuming less power compared to the state of the art. Additionally, in comparison to conventional spike sorting approaches, the synthesized BNN-based sorting circuitry can sort the activity of twice as many neurons as the previously published work.

Chapter 4 further expanded on binarized neural networks by studying the effects of partial binarization to mitigate the challenges posed by neural signals with low signal-to-noise ratios (SNRs). The result showed that, compared to previously published work with the most compact area-efficient design, the partially binarized neural network achieves 13% improvement in spike sorting accuracy for low SNR data. Compared to the BNN, the PBNN classifier demonstrated a reduction in power consumption by more than two-fold.

For applications that require continuous neural signals, such as local field potentials (LFPs), Chapter 5 discussed ML-based LFP compression using an autoencoder (AE)-based neural network. The study demonstrated that by leveraging the characteristics of the LFP and the inter-channel correlation of the recording array, various optimizations could significantly reduce the computational complexity and memory requirements of the designed AE model. It was shown that the AE-based compression circuitry achieved a reduction in power consumption of over 97% compared to previously published compression circuits, while maintaining comparable reconstruction performance.

Secondly, no single decoding algorithm is universally suitable for all users, due to both inter-user variations and the non-stationary nature of neural signals. Therefore, a critical step towards practical BCIs is achieving a high degree of neural decoding versatility to adapt and serve various users. This dissertation discussed the design and implementation of two processor architectures for realizing complex ML-based neural decoding algorithms.

Chapter 6 presented the design and implementation of a processor micro-architecture and a custom-developed instruction set for realizing arbitrary ML-based decoding algorithms. The custom instruction set supports sequential processing to enable complex operations, such as dilated convolution and non-linear activation functions. As a proof of concept for implantable neural decoding processors, the power density of the designed processor was reported as 24.4 mW/cm^2, which is within tissue-safe design constraints. While implementing in vivo decoding is feasible, Chapter 6 also discusses the limitations of model complexity concerning real-time neural decoding.

As an alternative, Chapter 7 introduced the design and implementation of a spiking neural network (SNN) processor. This processor employs biologically-plausible models of spiking neurons, which closely emulate the behavior of biological neurons, offering a greater potential for realizing neuron-silicon interaction. Unlike state-of-the-art realizations, the designed SNN processor's power consumption and real-time processing capabilities are independent of the spiking rate of the emulated SNN. The design supports various spiking rates based on the number of neurons and allows for an adjustable temporal resolution, enabling a trade-off between precision and computational complexity. Furthermore, the implemented SNN processor provides a compact single-chip solution that supports various network sizes and synaptic interconnect topologies in real-time.

Conventional BCI realization may compromise autonomy by requiring engagement with the BCI task during pre-defined time periods. Additionally, signal processing and decoding circuitry can be disabled when the user is not engaged, effectively saving power and improving overall decoding performance. To enable BCIs for everyday use, a method is required to achieve autonomous user engagement. By estimating the user's intention, a brain-controlled switch can be effectively realized, allowing users to engage or disengage with the BCI at their discretion. This also enables the implantable device circuitry to operate in an "asynchronous" mode, where specific processing tasks are performed only during active BCI engagement. This dissertation details two methods for implementing an efficient brain-switch for user intention estimation. Chapter 8 discussed the design and implementation of intention-aware spike detection circuitry. It was demonstrated that performing spike detection on a relatively small subset of intention-correlated recording channels can reduce total energy consumption by over 1.8$\times$ compared to previously published spike detection methods. Furthermore, by incorporating intention-awareness, neural decoding errors were reduced by an average of 26%.

Chapter 9 detailed how the LFP can also be used for intention estimation. Instead of transmitting spikes, the designed LFP-based brain switch transmits the bandpower of the LFP from a single recording channel. Compared to the intention-aware spike detection approach, employing the LFP reduced digital power consumption by over 82% while maintaining brain-switch performance comparable to that of the spike-based approach. Considering both signal acquisition and wireless transmission circuitries, the LFP-based brain switch requires 97% less power than the spike-based approach.

Chapter 10 discussed the design and implementation of a hybrid BCI that incorporates the findings from the previous chapters. By utilizing both LFPs and spikes, the device can operate in an asynchronous mode when the user it not engaged with the BCI application and switch to a spike mode to maintain high-performance and robust neural decoding. During the low-power LFP mode, power consumption is reduced by over 73% compared to the spike-mode for robust neural decoding. Additionally, it was shown that the hybrid BCI dissipates 59% less energy compared to previously published work utilizing a single neural signal type.

Chapter 12

Future Work

It is an exciting time for BCI research, a field experiencing rapid growth and development across several domains, including neuroscience, device development, and neural decoding algorithms. The topics discussed in this dissertation provide insights and suggestions for addressing key issues currently facing the field of BCI. However, there remain several areas for future work to tackle these challenges.

Data Overload: A crucial approach to reducing the amount of data, and thereby mitigating issues related to power consumption and wireless transmission, is to reduce the overall volume of recorded data. Lowering the power consumption of the recording front end by prioritizing specific channels based on a high-level task is a promising method for minimizing overall energy dissipation. For example, certain applications may achieve better performance by processing only a targeted subset of neural recording channels. Instead of relying on digital processing to discard data from the remaining channels, the front-end circuitry could be adapted to process only the required set of channels. This concept leads to task-aware neural sampling, where the end-to-end task determines optimized sampling, and ultimately reducing the need for extensive digital signal processing. Further advancements can also be made in the field of neural signal compression, particularly through machine learning (ML)-based methods. This dissertation focused on compressing the LFP; however, there are numerous alternative signals that can be recorded using intracortical penetrating electrodes, each encoding information across different frequency bands. More sophisticated models could potentially reduce information loss during noisy compression. Alternatively, since modern neural decoding algorithms are often ML-based, the compression model could be optimized for specific neural decoding applications. By combining compression and neural decoding, it may be possible to achieve task-aware signal compression. Instead of reconstructing the neural signal, the compressed representation could be used directly for neural decoding. Recent findings suggest that neural activities are constrained to a relatively low-dimensional space, referred to as the neural manifold [301]. This insight indicates that an optimized compression model may achieve higher compression rates by leveraging the inherent robustness of ML-based decoding.

Neural Decoding Versatility: A vital aspect of realizing practical BCIs is the ability to adapt to inter-user variations as well as changes in signal characteristics for a single user over extended periods. Currently, both types of variations are addressed by either retraining a decoding model from scratch or calibrating a pre-existing model to improve its performance. However, frequent training or calibration sessions may become cumbersome for users. To address this, future work should focus on optimizing neural signal adaptation. Recent studies have focused on addressing the changes in neural signal statistics, such as variations in mean firing rate, through a process known as manifold alignment [302, 301, 39]. In this approach, instead of calibrating the decoding model itself, neural signals are adjusted to resemble those recorded during the initial training session, allowing the same decoding model to be reused. Conventional adaptation methods involve complex dimensionality reduction techniques, such as principal components analysis, as part of the signal adaptation processing flow. However, practical BCIs would require this adaptation to be performed autonomously, potentially in vivo, to mitigate performance loss over time. While decoding models can also be adapted, studies have demonstrated that signal adaptation methods can effectively reuse previously trained models [39]. The advent of modern ML techniques could play a pivotal role in developing low-power and efficient signal adaptation methods. One potential avenue for exploration is the use of quantized neural networks, similar to binarized neural networks, to reduce the computational complexity of these models for in vivo implementation. Such networks could be leveraged to develop self-organizing maps or auto-encoders, enabling the creation of unsupervised models capable of maintaining a low-dimensional representation of neural signals over extended periods without requiring labeled training data.

Autonomous User Engagement: Enabling users to dictate when to engage in a BCI task is a crucial mechanism for enhancing user autonomy. It also has practical benefits, such as reducing power consumption and energy dissipation. Research into brain-controlled switches is still in its early stages, as the datasets and user feedback required for such studies are niche and highly specific. Compared to studies focused on the motor-cortex, cognitive-level decoding remains less understood and underexplored. Current brain-switch implementations rely on intention estimation ML models executed in silico on external processing devices, such as laptops. However, this approach necessitates constant wireless signal transmission and continuous power for external processing devices. These requirements may be challenging to maintain for practical BCI applications. To address this, algorithmic optimizations should be pursued to enable implantable autonomous user intention estimation. By reducing wireless transmission to only the initial training or calibration session, it may become unnecessary to transmit the monitored signals during non-BCI use. While algorithmic improvements could make current ML-based intention estimation approaches viable in vivo, an alternative avenue worth exploring involves developing low-complexity signal monitoring methods. Similar to non-invasive neural spelling applications, which monitor for known neurophysiological markers, alternative cognitive neural signal features could be detected. These methods would avoid reliance on computationally-intensive ML-based operations, such as non-linear activation functions and vector-matrix multiplication.

System-level Optimizations: The designs discussed in this dissertation mainly focus on the digital implementations of alternative neural signal processing algorithms. However, there are additional opportunities to reduce power consumption and energy dissipation by employing system-level optimizations. Conventional approaches often make assumptions about the front-end recording circuitry that may not be optimal for specific types of neural signals. For example, optimizing the design of low-noise amplifiers (LNAs) and analog-to-digital converters (ADCs) to adapt to one of the two given signal modalities could significantly benefit system-level power consumption and energy dissipation. The front-end optimizations would further compound the optimizations of the digital circuitry, as sampling rate and signal bit-widths directly impact the power consumption of the digital back-end. Additionally, considering the characteristics of the recording electrodes themselves might yield further system-level optimization. In fact, recent studies incorporate recording array geometry into the spike detection flow to avoid detecting the same spike instance from multiple nearby recording channels [148]. Thus, similar geometry-aware optimizations can be applied to the configuration of the recording front-end circuitry for more efficient channel selection with minimal overhead. These optimizations play a critical role in realizing wireless implantable in vivo neural signal acquisition and digital signal processing circuitry that complies with tissue-safe design constraints.

Bibliography

[1] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition.* Cambridge University Press, 2014.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, "Ion channels and the electrical properties of membranes," in *Molecular Biology of the Cell. 4th edition*. Garland Science, 2002.

[3] P. D. Cheney and E. E. Fetz, "Functional classes of primate corticomotoneuronal cells and their relation to active force," *Journal of Neurophysiology*, vol. 44, no. 4, pp. 773–791, 1980.

[4] E. E. Fetz and P. D. Cheney, "Postspike facilitation of forelimb muscle activity by primate corticomotoneuronal cells," *Journal of Neurophysiology*, vol. 44, no. 4, pp. 751–772, 1980.

[5] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner, "Neuronal population coding of movement direction," *Science*, vol. 233, no. 4771, pp. 1416–1419, 1986.

[6] C. Mercier, K. T. Reilly, C. D. Vargas, A. Aballea, and A. Sirigu, "Mapping phantom movement representations in the motor cortex of amputees," *Brain*, vol. 129, no. 8, pp. 2202–2210, 2006.

[7] K. Sahasrabuddhe, A. A. Khan, A. P. Singh, T. M. Stern, Y. Ng, A. Tadić, P. Orel, C. LaReau, D. Pouzzner, K. Nishimura *et al.*, "The argo: A high channel count recording system for neural recording in vivo," *Journal of Neural Engineering*, vol. 18, no. 1, p. 015002, 2021.

[8] E. Musk *et al.*, "An integrated brain-machine interface platform with thousands of channels," *Journal of medical Internet research*, vol. 21, no. 10, p. e16194, 2019.

[9] N. A. Steinmetz, C. Aydin, A. Lebedeva, M. Okun, M. Pachitariu, M. Bauza, M. Beau, J. Bhagat, C. Böhm, M. Broux *et al.*, "Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings," *Science*, vol. 372, no. 6539, p. eabf4588, 2021.

[10] J. I. Glaser, A. S. Benjamin, R. H. Chowdhury, M. G. Perich, L. E. Miller, and K. P. Kording, "Machine learning for neural decoding," *Eneuro*, vol. 7, no. 4, 2020.

[11] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy, "High-performance brain-to-text communication via handwriting," *Nature*, vol. 593, no. 7858, pp. 249–254, 2021.

[12] S. Olsen, J. Zhang, K.-F. Liang, M. Lam, U. Riaz, and J. C. Kao, "An artificial intelligence that increases simulated brain–computer interface performance," *Journal of Neural Engineering*, vol. 18, no. 4, p. 046053, 2021.

[13] A. Défossez, C. Caucheteux, J. Rapin, O. Kabeli, and J.-R. King, "Decoding speech perception from non-invasive brain recordings," *Nature Machine Intelligence*, vol. 5, no. 10, pp. 1097–1107, 2023.

[14] S. L. Metzger, K. T. Littlejohn, A. B. Silva, D. A. Moses, M. P. Seaton, R. Wang, M. E. Dougherty, J. R. Liu, P. Wu, M. A. Berger *et al.*, "A high-performance neuroprosthesis for speech decoding and avatar control," *Nature*, vol. 620, no. 7976, pp. 1037–1046, 2023.

[15] J. Rosenthal and M. S. Reynolds, "A 158 pj/bit 1.0 mbps bluetooth low energy (BLE) compatible backscatter communication system for wireless sensing," in *Conference on Wireless Sensors and Sensor Networks (WiSNet)*. IEEE, 2019, pp. 1–3.

[16] S. M. Won, L. Cai, P. Gutruf, and J. A. Rogers, "Wireless and battery-free technologies for neuroengineering," *Nature Biomedical Engineering*, pp. 1–19, 2021.

[17] P. D. Wolf and W. Reichert, "Thermal considerations for the design of an implanted cortical brain–machine interface (bmi)," *Indwelling Neural Implants: Strategies for Contending with the In Vivo Environment*, pp. 33–38, 2008.

[18] E. Koutsos, S. E. Paraskevopoulou, and T. G. Constandinou, "A 1.5 μw NEO-based spike detector with adaptive-threshold for calibration-free multichannel neural interfaces," in *IEEE International Symposium on Circuits and Systems*, 2013, pp. 1922–1925.

[19] M. Delgado-Restituto, A. Rodriguez-Perez, A. Darie, C. Soto-Sánchez, E. Fernández-Jover, and A. Rodriguez-Vazquez, "System-level design of a 64-channel low power neural spike recording sensor," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 420–433, 2017.

[20] S. Gibson, "Neural Spike Sorting in Hardware: From Theory to Practice," Ph.D. dissertation, University of California, Los Angeles, 2012.

[21] Z. Zhang and T. Constandinou, "Adaptive spike detection and hardware optimization towards autonomous, high-channel-count BMIs." *Journal of Neuroscience Methods*, pp. 109 103–109 103, 2021.

[22] M. Zamani, J. Sokolić, D. Jiang, F. Renna, M. R. Rodrigues, and A. Demosthenous, "Accurate, very low computational complexity spike sorting using unsupervised matched subspace learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 2, pp. 221–231, 2020.

[23] Y. Yang, S. Boling, and A. Mason, "A hardware-efficient scalable spike sorting neural signal processor module for implantable high-channel-count brain machine interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 4, pp. 743–754, 2017.

[24] A. Do *et al.*, "An area-efficient 128-channel spike sorting processor for real-time neural recording with 0.175 μW/channel in 65-nm CMOS," *IEEE Transactions on VLSI Systems*, vol. 27, no. 1, pp. 126–137, 2019.

[25] L. Guo, A. Weiße, S. M. A. Zeinolabedin, F. M. Schüffny, M. Stolba, Q. Ma, Z. Wang, S. Scholze, A. Dixius, M. Berthel *et al.*, "68-channel neural signal processing system-on-chip with integrated feature extraction, compression, and hardware accelerators for neuroprosthetics in 22 nm fdsoi," *Frontiers in Neuroscience*, vol. 18, p. 1432750, 2024.

[26] M. Pachitariu, S. Sridhar, J. Pennington, and C. Stringer, "Spike sorting with kilosort4," *Nature Methods*, pp. 1–8, 2024.

[27] E. Maynard, N. Hatsopoulos, C. Ojakangas, B. Acuna, J. Sanes, R. Normann, and J. Donoghue, "Neuronal interactions improve cortical population coding of movement direction," *Journal of Neuroscience*, vol. 19, no. 18, pp. 8083–8093, 1999.

[28] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried, "Invariant visual representation by single neurons in the human brain," *Nature*, vol. 435, no. 7045, pp. 1102–1107, 2005.

[29] G. Buzsáki, C. A. Anastassiou, and C. Koch, "The origin of extracellular fields and currents – EEG, ECoG, LFP and spikes," *Nature Reviews Neuroscience*, vol. 13, no. 6, pp. 407–420, 2012.

[30] I. E. Ohiorhenuan, F. Mechler, K. P. Purpura, A. M. Schmid, Q. Hu, and J. D. Victor, "Sparse coding and high-order correlations in fine-scale cortical networks," *Nature*, vol. 466, no. 7306, pp. 617–621, 2010.

[31] E. M. Maynard, C. T. Nordhausen, and R. A. Normann, "The utah intracortical electrode array: a recording structure for potential brain-computer interfaces," *Electroencephalography and Clinical Neurophysiology*, vol. 102, no. 3, pp. 228–239, 1997.

[32] G. Buzsáki, "Large-scale recording of neuronal ensembles," *Nature Neuroscience*, vol. 7, no. 5, pp. 446–451, 2004.

[33] R. Mukamel and I. Fried, "Human intracranial recordings and cognitive neuroscience," *Annual Review of Psychology*, vol. 63, pp. 511–537, 2012.

[34] T. Boraud, E. Bezard, B. Bioulac, and C. E. Gross, "From single extracellular unit recording in experimental and human parkinsonism to the development of a functional concept of the role played by the basal ganglia in motor control," *Progress in Neurobiology*, vol. 66, no. 4, pp. 265–283, 2002.

[35] W.-k. Tam, T. Wu, Q. Zhao, E. Keefer, and Z. Yang, “Human motor decoding from neural signals: a review,” *BMC Biomedical Engineering*, vol. 1, no. 1, pp. 1–22, 2019.

[36] R. Q. Quiroga, “What is the real shape of extracellular spikes?” *Journal of Neuroscience Methods*, vol. 177, no. 1, pp. 194–198, 2009.

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

[38] J. F. Kaiser, “On a simple algorithm to calculate the ‘energy’ of a signal,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1990, pp. 381 – 384.

[39] E. R. Oby, S. Perel, P. T. Sadtler, D. A. Ruff, J. L. Mischel, D. F. Montez, M. R. Cohen, A. P. Batista, and S. M. Chase, “Extracellular voltage threshold settings can be tuned for optimal encoding of movement and stimulus parameters,” *Journal of Neural Engineering*, vol. 13, no. 3, p. 036009, 2016.

[40] K. Guillory and R. Normann, “A 100-channel system for real time detection and storage of extracellular spike waveforms,” *Journal of neuroscience methods*, vol. 91, no. 1-2, pp. 21–29, 1999.

[41] E. Biffi, D. Ghezzi, A. Pedrocchi, and G. Ferrigno, “Development and validation of a spike detection and classification algorithm aimed at implementation on hardware devices,” *Computational Intelligence and Neuroscience*, 2010.

[42] K. S. Al-Kharabsheh, I. M. AlTurani, A. M. I. AlTurani, and N. I. Zanoon, “Review on sorting algorithms a comparative study,” *International Journal of Computer Science and Security (IJCSS)*, vol. 7, no. 3, pp. 120–126, 2013.

[43] H. Mora-Mora, J. Mora-Pascual, J. M. García-Chamizo, and A. Jimeno-Morenilla, “Real-time arithmetic unit,” *Real-Time Systems*, vol. 34, no. 1, pp. 53–79, 2006.

[44] I. Koren, *Computer arithmetic algorithms*. AK Peters/CRC Press, 2018.

[45] X. Liu, M. Zhang, A. G. Richardson, T. H. Lucas, and J. 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*, vol. 11, no. 4, pp. 729–742, 2016.

[46] W. Biederman, D. J. Yeager, N. Narevsky, J. Leverett, R. Neely, J. M. Carmena, E. Alon, and J. 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*, vol. 50, no. 4, pp. 1038–1047, 2015.

[47] A. Stillmaker, Z. Xiao, and B. Baas, "Toward mode accurate scaling estimates of cmos circuits from 180nm to 22nm," *VLSI Computation Lab, ECE Department, University of California, Davis, Tech. Rep. ECE-VCL-2011-4*, vol. 4, p. m8, 2011.

[48] M. S. Kim, A. A. Del Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient mitchell's approximate log multipliers for convolutional neural networks," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 660–675, 2018.

[49] Michael Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *NetworkL Comput. Neural Syst*, pp. 53–78, 1998.

[50] E. M. Trautmann, S. D. Stavisky, S. Lahiri, K. C. Ames, M. T. Kaufman, D. J. O'Shea, S. Vyas, X. Sun, S. I. Ryu, S. Ganguli *et al.*, "Accurate estimation of neural population dynamics without spike sorting," *Neuron*, vol. 103, no. 2, pp. 292–308, 2019.

[51] C. A. Chestek, V. Gilja, P. Nuyujukian, J. D. Foster, J. M. Fan, M. T. Kaufman, M. M. Churchland, Z. Rivera-Alvidrez, J. P. Cunningham, S. 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*, vol. 8, no. 4, p. 045005, 2011.

[52] M. Taketani and M. Baudry, *Advances in network electrophysiology*. Springer, 2010.

[53] F. Hamilton, T. Berry, and T. Sauer, "Tracking intracellular dynamics through extracellular measurements," *PloS one*, vol. 13, no. 10, p. e0205031, 2018.

[54] M. A. Long and A. K. Lee, "Intracellular recording in behaving animals," *Current Opinion in Neurobiology*, vol. 22, no. 1, pp. 34–44, 2012.

[55] M. Pachitariu, N. A. Steinmetz, S. N. Kadir, M. Carandini, and K. D. Harris, "Fast and accurate spike sorting of high-channel count probes with kilosort," *Advances in Neural Information Processing Systems*, vol. 29, pp. 4448–4456, 2016.

[56] W. W. Teka, K. C. Hamade, W. H. Barnett, T. Kim, S. N. Markin, I. A. Rybak, and Y. I. Molkov, "From the motor cortex to the movement and back again," *PloS one*, vol. 12, no. 6, p. e0179288, 2017.

[57] Y. Salimpour, H. Soltanian-Zadeh, S. Salehi, N. Emadi, and M. Abouzari, "Neuronal spike train analysis in likelihood space," *PloS one*, vol. 6, no. 6, p. e21256, 2011.

[58] J. E. O'Doherty, M. M. B. Cardoso, J. G. Makin, and P. N. Sabes, "Nonhuman Primate Reaching with Multichannel Sensorimotor Cortex Electrophysiology," May 2017. [Online]. Available: https://doi.org/10.5281/zenodo.583331

[59] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-Theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[60] K. Mizuseki, A. Sirota, E. Pastalkova, and G. Buzsáki, "Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop," *Neuron*, vol. 64, no. 2, pp. 267–280, 2009.

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

[62] D. Valencia and A. Alimohammad, "An efficient hardware architecture for template matching-based spike sorting," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 3, pp. 481–492, 2019.

[63] D. Valencia and A. Alimohammad, "A real-time spike sorting system using parallel osort clustering," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1700–1713, 2019.

[64] D. Valencia, J. Thies, and A. Alimohammad, "Frameworks for efficient brain-computer interfacing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1714–1722, 2019.

[65] V. Karkare, S. Gibson, and D. Markovic, "A 130-μw, 64-channel neural spike-sorting DSP chip," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 5, pp. 1214–1222, 2011.

[66] V. Karkare, S. Gibson, and D. Markovic, "A 75-μw, 16-channel neural spike-sorting processor with unsupervised clustering," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 9, pp. 2230–2238, 2013.

[67] Y. Liu, J. Sheng, and M. C. Herbordt, "A hardware design for in-brain neural spike sorting," in *IEEE High Performance Extreme Computing Conference*, 2016, pp. 1–6.

[68] M. Zamani, D. Jiang, and A. Demosthenous, "An adaptive neural spike processor with embedded active learning for improved unsupervised sorting accuracy," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 3, pp. 665–676, 2018.

[69] H. Xu *et al.*, "Unsupervised and real-time spike sorting chip for neural signal processing in hippocampal prosthesis," *Journal of Neuroscience Methods*, vol. 311, pp. 111–121, 2019.

[70] H. Ando, K. Takizawa, T. Yoshida, K. Matsushita, M. Hirata, and T. Suzuki, "Wireless multichannel neural recording with a 128-mbps uwb transmitter for an implantable brain-machine interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 6, pp. 1068–1078, 2016.

[71] S. Gibson, J. W. Judy, and D. Marković, "An FPGA-based platform for accelerated offline spike sorting," *Journal of Neuroscience Methods*, vol. 215, no. 1, pp. 1–11, 2013.

[72] J. Dragas, D. Jäckel, A. Hierlemann, and F. Franke, "Complexity optimization and high-throughput low-latency hardware implementation of a multi-electrode spike-sorting algorithm," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 2, pp. 149–158, 2015.

[73] L. Schäffer, Z. Nagy, Z. Kineses, and R. Fiáth, "FPGA-based neural probe positioning to improve spike sorting with OSort algorithm," in *IEEE International Symposium on Circuits and Systems*, 2017, pp. 1–4.

[74] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.

[75] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, and C. Poon, "Real-time FPGA-based multichannel spike sorting using hebbian eigenfilters," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 4, pp. 502 – 515, 2011.

[76] Z. Jiang, J. P. Cerqueira, S. Kim, Q. Wang, and M. Seok, "1.74-μw/ch, 95.3%-accurate spike-sorting hardware based on bayesian decision," in *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits).* IEEE, 2016, pp. 1–2.

[77] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," in *Advances in Neural Information Processing Systems*, 2017, pp. 5109–5118.

[78] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[79] R. R. Harrison *et al.*, "A low-power integrated circuit for a wireless 100-electrode neural recording system," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 123–133, 2007.

[80] I. Obeid and P. D. Wolf, "Evaluation of spike-detection algorithms for a brain-machine interface application," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 905–911, 2004.

[81] K. H. Kim and S. J. Kim, "A wavelet-based method for action potential detection from extracellular neural signal recording with low signal-to-noise ratio," *IEEE Transactions on Biomedical Engineering*, vol. 50, no. 8, pp. 999 – 1011, 2003.

[82] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics.* Berkeley, Calif.: University of California Press, 1967, pp. 281–297.

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

[84] P. O. Olukanmi, F. Nelwamondo, and T. Marwala, "k-means-lite: Real time clustering for large datasets," in *International Conference on Soft Computing & Machine Intelligence*. IEEE, 2018, pp. 54–59.

[85] D. Valencia, S. F. Fard, and A. Alimohammad, "An artificial neural network processor with a custom instruction set architecture for embedded applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020.

[86] S. Wu *et al.*, "Training and inference with integers in deep neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HJGXzmspb

[87] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[88] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, 2019.

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

[90] L. Geiger and P. Team, "Larq: An open-source library for training binarized neural networks," *Journal of Open Source Software*, vol. 5, no. 45, p. 1746, Jan. 2020. [Online]. Available: https://doi.org/10.21105/joss.01746

[91] J. Martinez, C. Pedreira, M. J. Ison, and R. Q. Quiroga, "Realistic simulation of extracellular recordings," *Journal of neuroscience methods*, vol. 184, no. 2, pp. 285–293, 2009.

[92] M. K. Awais and J. M. Andrew, "On-chip feature extraction for spike sorting in high density implantable neural recording systems," in *Biomedical Circuits and Systems Conference*. IEEE, 2010, pp. 13–16.

[93] S. Gibson, J. W. Judy, and D. Markovic, "Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 5, pp. 469–478, 2010.

[94] C. Pedreira, J. Martinez, M. J. Ison, and R. Q. Quiroga, "How many neurons can we see with current spike sorting algorithms?" *Journal of Neuroscience Methods*, vol. 211, no. 1, pp. 58–65, 2012.

[95] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

[96] S. Gibson, "Neural spike sorting in hardware: From theory to practice," Ph.D. dissertation, University of California Los Angeles, 2012.

[97] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, "Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, 2012.

[98] S. Kim, P. Tathireddy, R. A. Normann, and F. Solzbacher, "Thermal impact of an active 3-D microelectrode array implanted in the brain," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 4, pp. 493–501, 2007.

[99] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks," in *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS).* IEEE, 2017, pp. 173–177.

[100] E. F. Chang and G. K. Anumanchipalli, "Toward a speech neuroprosthesis," *Journal of the American Medical Association*, vol. 323, no. 5, pp. 413–414, 2020.

[101] M. Kocaturk, H. O. Gulcur, and R. Canbeyli, "Toward building hybrid biological/in silico neural networks for motor neuroprosthetic control," *Frontiers in Neurorobotics*, vol. 9, p. 8, 2015.

[102] D. Valencia and A. Alimohammad, "Neural spike sorting using binarized neural networks." *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2020.

[103] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[104] U. A. Korat and A. Alimohammad, "A reconfigurable hardware architecture for principal component analysis," *Circuits, Systems, and Signal Processing*, vol. 38, no. 5, pp. 2097–2113, 2019.

[105] M. V. Lopes, E. Aguiar, E. Santana, E. Santana, and A. K. Barros, "ICA feature extraction for spike sorting of single-channel records," in *IEEE Biosignals and Biorobotics Conference*, 2013, pp. 1–5.

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

[107] P. Li, M. Liu, X. Zhang, and H. Chen, "Efficient online feature extraction algorithm for spike sorting in a multichannel FPGA-based neural recording system," in *IEEE Biomedical Circuits and Systems Conference*, 2014, pp. 1–4.

[108] Z. Nadasdy, R. Q. Quiroga, Y. Ben-Shaul, B. Pesaran, D. A. Wagenaar, and R. A. Andersen, "Comparison of unsupervised algorithms for on-line and off-line spike sorting," in *Proceedings of the Annual Meeting of the Society for Neuroscience*, 2002.

[109] S. E. Paraskevopoulou, D. Y. Barsakcioglu, M. R. Saberi, A. Eftekhar, and T. G. Constandinou, "Feature extraction using first and second derivative extrema for real-time and hardware-efficient spike sorting," *Journal of Neuroscience Methods*, vol. 215, no. 1, pp. 29–37, 2013.

[110] M. Zamani and A. Demosthenous, "Feature extraction using extrema sampling of discrete derivatives for spike sorting in implantable upper-limb neural prostheses," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 716–726, 2014.

[111] H. W. Lilliefors, “On the Kolmogorov-Smirnov test for normality with mean and variance unknown,” *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[112] W. Tang, G. Hua, and L. Wang, “How to train a compact binary neural network with high accuracy?” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[113] D. Valencia, P. P. Mercier, and A. Alimohammad, “In vivo neural spike detection with adaptive noise estimation,” *Journal of Neural Engineering*, vol. 19, no. 4, p. 046018, 2022.

[114] G. H. Mulliken, S. Musallam, and R. A. Andersen, “Decoding trajectories from posterior parietal cortex ensembles,” *Journal of Neuroscience*, vol. 28, no. 48, pp. 12 913–12 926, 2008.

[115] J. E. Downey, J. M. Weiss, K. Muelling, A. Venkatraman, J.-S. Valois, M. Hebert, J. A. Bagnell, A. B. Schwartz, and J. L. Collinger, “Blending of brain-machine interface and vision-guided autonomous robotics improves neuroprosthetic arm performance during grasping,” *Journal of Neuroengineering and Rehabilitation*, vol. 13, no. 1, pp. 1–12, 2016.

[116] J. J. Williams, R. N. Tien, Y. Inoue, and A. B. Schwartz, “Idle state classification using spiking activity and local field potentials in a brain computer interface,” in *IEEE International Conference of the Engineering in Medicine and Biology Society*, 2016, pp. 1572–1575.

[117] H. Scherberger, M. R. Jarvis, and R. A. Andersen, “Cortical local field potential encodes movement intentions in the posterior parietal cortex,” *Neuron*, vol. 46, no. 2, pp. 347–354, 2005.

[118] M. Filippini, A. Morris, R. Breveglieri, K. Hadjidimitrakis, and P. Fattori, “Decoding of standard and non-standard visuomotor associations from parietal cortex,” *Journal of Neural Engineering*, vol. 17, no. 4, p. 046027, 2020.

[119] G. K. Anumanchipalli, J. Chartier, and E. F. Chang, “Speech synthesis from neural decoding of spoken sentences,” *Nature*, vol. 568, no. 7753, pp. 493–498, 2019.

[120] R. R. Harrison, “The design of integrated circuits to observe brain activity,” *Proceedings of the IEEE*, vol. 96, no. 7, pp. 1203–1216, 2008.

[121] H. Miranda and T. H. Meng, “A programmable pulse UWB transmitter with 34% energy efficiency for multichannel neuro-recording systems,” in *IEEE Custom Integrated Circuits Conference*, 2010, pp. 1–4.

[122] A. Hennessy and A. Alimohammad, “Design and implementation of a digital secure code-shifted reference UWB transmitter and receiver,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 7, pp. 1927–1936, 2017.

[123] A. Hennessy and A. Alimohammad, "Compact digital implementation of a non-coherent IR-UWB transmitter and receiver," *IET Communications*, vol. 12, no. 20, pp. 2616–2622, 2018.

[124] M. Song, Y. Huang, H. J. Visser, J. Romme, and Y.-H. Liu, "An energy-efficient and high-data-rate IR-UWB transmitter for intracortical neural sensing interfaces," *IEEE journal of solid-state circuits*, vol. 57, no. 12, pp. 3656–3668, 2022.

[125] F. Jiang, M. Lin, X. Chen, L. Zhang, and X. Sheng, "A novel low power high speed gbps uwb transmitter design using fractional DTC and high spectrum efficiency intra-chip PPM," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2023.

[126] Y. Zhao, L. Tang, R. Rennaker, C. Hutchens, and T. S. Ibrahim, "Studies in RF power communication, SAR, and temperature elevation in wireless implantable neural interfaces," *PloS One*, vol. 8, no. 11, p. e77759, 2013.

[127] S.-Y. Park, J. Cho, K. Lee, and E. Yoon, "Dynamic power reduction in scalable neural recording interface using spatiotemporal correlation and temporal sparsity of neural signals," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 1102–1114, 2018.

[128] Y. Khazaei, A. A. Shahkooh, and A. M. Sodagar, "Spatial redundancy reduction in multi-channel implantable neural recording microsystems," in *IEEE Conference of the Engineering in Medicine & Biology Society*, 2020, pp. 898–901.

[129] A. Cuevas-López, E. Pérez-Montoyo, V. J. López-Madrona, S. Canals, and D. Moratal, "Low-power lossless data compression for wireless brain electrophysiology," *Sensors*, vol. 22, no. 10, p. 3676, 2022.

[130] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[131] N. Even-Chen, D. G. Muratore, S. D. Stavisky, L. R. Hochberg, J. M. Henderson, B. Murmann, and K. V. Shenoy, "Power-saving design opportunities for wireless intracortical brain–computer interfaces," *Nature Biomedical Engineering*, vol. 4, no. 10, pp. 984–996, 2020.

[132] D. Valencia, G. Leone, N. Keller, P. P. Mercier, and A. Alimohammad, "Power-efficient in vivo brain-machine interfaces via brain-state estimation," *Journal of Neural Engineering*, vol. 20, no. 1, p. 016032, 2023.

[133] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[134] S. Schmale, B. Knoop, J. Hoeffmann, D. Peters-Drolshagen, and S. Paul, "Joint compression of neural action potentials and local field potentials," in *IEEE Asilomar Conference on Signals, Systems and Computers*, 2013, pp. 1823–1827.

[135] M. Shoaran, M. H. Kamal, C. Pollo, P. Vandergheynst, and A. Schmid, "Compact low-power cortical recording architecture for compressive multichannel data acquisition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 6, pp. 857–870, 2014.

[136] B. Sun and W. Zhao, "Compressed sensing of extracellular neurophysiology signals: A review," *Frontiers in Neuroscience*, vol. 15, p. 682063, 2021.

[137] D. A. Henze, Z. Borhegyi, J. Csicsvari, A. Mamiya, K. D. Harris, and G. Buzsaki, "Intracellular features predicted by extracellular recordings in the hippocampus in vivo," *Journal of Neurophysiology*, vol. 84, no. 1, pp. 390–400, 2000.

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

[139] D. Valencia and A. Alimohammad, "Partially binarized neural networks for efficient spike sorting," *Biomedical Engineering Letters*, vol. 13, no. 1, pp. 73–83, 2023.

[140] J. Thies and A. Alimohammad, "Compact and low-power neural spike compression using undercomplete autoencoders," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 8, pp. 1529–1538, 2019.

[141] S. Todorova, P. Sadtler, A. Batista, S. Chase, and V. Ventura, "To sort or not to sort: the impact of spike-sorting on neural decoding performance," *Journal of Neural Engineering*, vol. 11, no. 5, p. 056005, 2014.

[142] S. Katzner, I. Nauhaus, A. Benucci, V. Bonin, D. L. Ringach, and M. Carandini, "Local origin of field potentials in visual cortex," *Neuron*, vol. 61, no. 1, pp. 35–41, 2009.

[143] S. Saha, K. A. Mamun, K. Ahmed, R. Mostafa, G. R. Naik, S. Darvishi, A. H. Khandoker, and M. Baumert, "Progress in brain computer interface: Challenges and opportunities," *Frontiers in Systems Neuroscience*, vol. 15, p. 578875, 2021.

[144] J. E. Downey, N. Schwed, S. M. Chase, A. B. Schwartz, and J. L. Collinger, "Intracortical recording stability in human brain–computer interface users," *Journal of Neural Engineering*, vol. 15, no. 4, p. 046016, 2018.

[145] J. W. Salatino, K. A. Ludwig, T. D. Kozai, and E. K. Purcell, "Glial responses to implanted electrodes in the brain," *Nature Biomedical Engineering*, vol. 1, no. 11, pp. 862–877, 2017.

[146] S. Musallam, B. Corneil, B. Greger, H. Scherberger, and R. A. Andersen, "Cognitive control signals for neural prosthetics," *Science*, vol. 305, no. 5681, pp. 258–262, 2004.

[147] T. Aflalo, S. Kellis, C. Klaes, B. Lee, Y. Shi, K. Pejsa, K. Shanfield, S. Hayes-Jackson, M. Aisen, C. Heck *et al.*, "Decoding motor imagery from the posterior parietal cortex of a tetraplegic human," *Science*, vol. 348, no. 6237, pp. 906–910, 2015.

[148] Y. Chen, B. Tacca, Y. Chen, D. Biswas, G. Gielen, F. Catthoor, M. Verhelst, and C. M. Lopez, "A 384-channel online-spike-sorting IC using unsupervised geo-osort clustering and achieving 0.0013 mm^2/ch and 1.78μw/ch," in *IEEE International Solid-State Circuits Conference*. IEEE, 2023, pp. 486–488.

[149] J. E. O'Doherty, M. M. B. Cardoso, J. G. Makin, and P. N. Sabes, "Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology," May 2017. [Online]. Available: https://doi.org/10.5281/zenodo.788569

[150] E.-R. Ardelean, A. Coporîie, A.-M. Ichim, M. Dînșoreanu, and R. C. Mureșan, "A study of autoencoders as a feature extraction technique for spike sorting," *Plos One*, vol. 18, no. 3, p. e0282810, 2023.

[151] J. Eom, I. Y. Park, S. Kim, H. Jang, S. Park, Y. Huh, and D. Hwang, "Deep-learned spike representations and sorting via an ensemble of auto-encoders," *Neural Networks*, vol. 134, pp. 131–142, 2021.

[152] T. Brochier, L. Zehl, Y. Hao, M. Duret, J. Sprenger, M. Denker, S. Grün, and A. Riehle, "Massively parallel recordings in macaque motor cortex during an instructed delayed reach-to-grasp task," *Scientific Data*, vol. 5, no. 1, pp. 1–23, 2018.

[153] W. H. Greub, *Linear Algebra*. Springer Science & Business Media, 2012, vol. 23.

[154] N. Ahmadi, M. L. Cavuto, P. Feng, L. B. Leene, M. Maslik, F. Mazza, O. Savolainen, K. M. Szostak, C.-S. Bouganis, J. Ekanayake *et al.*, "Towards a distributed, chronically-implantable neural interface," in *International IEEE/EMBS Conference on Neural Engineering*. IEEE, 2019, pp. 719–724.

[155] M. M. Ghanbari, D. K. Piech, K. Shen, S. F. Alamouti, C. Yalcin, B. C. Johnson, J. M. Carmena, M. M. Maharbiz, and R. Muller, "A sub-mm^3 ultrasonic free-floating implant for multi-mote neural recording," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 11, pp. 3017–3030, 2019.

[156] J. Lee, V. Leung, A.-H. Lee, J. Huang, P. Asbeck, P. P. Mercier, S. Shellhammer, L. Larson, F. Laiwalla, and A. Nurmikko, "Neural recording and stimulation using wireless networks of microimplants," *Nature Electronics*, vol. 4, no. 8, pp. 604–614, 2021.

[157] F. Hashemi Noshahr, M. Nabavi, and M. Sawan, "Multi-channel neural recording implants: A review," *Sensors*, vol. 20, no. 3, p. 904, 2020.

[158] N. Li, M. Osborn, G. Wang, and M. Sawan, "A digital multichannel neural signal processing system using compressed sensing," *Digital Signal Processing*, vol. 55, pp. 64–77, 2016.

[159] X. Liu, M. Zhang, T. Xiong, A. G. Richardson, T. H. Lucas, P. S. Chin, R. Etienne-Cummings, T. D. Tran, and J. Van der Spiegel, "A fully integrated wireless compressed sensing neural signal acquisition system for chronic recording and brain machine interface," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 4, pp. 874–883, 2016.

[160] J. Li, X. Liu, W. Mao, T. Chen, and H. Yu, "Advances in neural recording and stimulation integrated circuits," *Frontiers in Neuroscience*, vol. 15, p. 663204, 2021.

[161] S. Mondal, C.-L. Hsu, R. Jafari, and D. Hall, "A dynamically reconfigurable ECG analog front-end with a 2.5$\times$ data-dependent power reduction," in *IEEE Custom Integrated Circuits Conference*. IEEE, 2017, pp. 1–4.

[162] M. Yip and A. P. Chandrakasan, "A resolution-reconfigurable 5-to-10-bit 0.4-to-1 v power scalable SAR ADC for sensor applications," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 6, pp. 1453–1464, 2013.

[163] H. Wang, X. Wang, A. Barfidokht, J. Park, J. Wang, and P. P. Mercier, "A battery-powered wireless ion sensing system consuming 5.5 nw of average power," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2043–2053, 2018.

[164] M. Jang, W.-H. Yu, C. Lee, M. Hays, P. Wang, N. Vitale, P. Tandon, P. Yan, P.-I. Mak, Y. Chae *et al.*, "A 1024-channel 268 nw/pixel 36x36 μm 2/ch data-compressive neural recording IC for high-bandwidth brain-computer interfaces," in *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2023, pp. 1–2.

[165] A. L. Owens, T. J. Denison, H. Versnel, M. Rebbert, M. Peckerar, and S. A. Shamma, "Multi-electrode array for measuring evoked potentials from surface of ferret primary auditory cortex," *Journal of neuroscience methods*, vol. 58, no. 1-2, pp. 209–220, 1995.

[166] D. A. Borton, M. Yin, J. Aceros, and A. Nurmikko, "An implantable wireless neural interface for recording cortical circuit dynamics in moving primates," *Journal of Neural Engineering*, vol. 10, no. 2, 2013.

[167] C. Gold, D. A. Henze, C. Koch, and G. Buzsaki, "On the origin of the extracellular action potential waveform: a modeling study," *Journal of Neurophysiology*, vol. 95, no. 5, pp. 3113–3128, 2006.

[168] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical control of a prosthetic arm for self-feeding," *Nature*, vol. 453, no. 7198, pp. 1098–1101, 2008.

[169] S. Panzeri and S. R. Schultz, "A unified approach to the study of temporal, correlational, and rate coding," *Neural Computation*, vol. 13, no. 6, pp. 1311–1349, 2001.

[170] E. E. Fetz, "Temporal coding in neural populations?" *Science*, vol. 278, no. 5345, pp. 1901–1902, 1997.

[171] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[172] S. P. Strong, R. Koberle, R. R. D. R. Van Steveninck, and W. Bialek, "Entropy and information in neural spike trains," *Physical Review Letters*, vol. 80, no. 1, p. 197, 1998.

[173] M. K. Chung, “Gaussian kernel smoothing,” *arXiv preprint arXiv:2007.09539*, 2020.

[174] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14, 1967, pp. 281–297.

[175] J. L. Elman, “Finding structure in time,” *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[176] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, “Temporal convolutional networks: A unified approach to action segmentation,” in *European Conference on Computer Vision*. Springer, 2016, pp. 47–54.

[177] J. C. Kao, S. D. Stavisky, D. Sussillo, P. Nuyujukian, and K. V. Shenoy, “Information systems opportunities in brain–machine interface decoders,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 666–682, 2014.

[178] G. W. Fraser, S. M. Chase, A. Whitford, and A. B. Schwartz, “Control of a brain–computer interface without spike sorting,” *Journal of Neural Engineering*, vol. 6, no. 5, p. 055004, 2009.

[179] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” *arXiv preprint arXiv:1611.03530*, 2016.

[180] D. Klabjan and X. Zhu, “Neural network retraining for model serving,” *arXiv preprint arXiv:2004.14203*, 2020.

[181] D. Maltoni and V. Lomonaco, “Continuous learning in single-incremental-task scenarios,” *Neural Networks*, vol. 116, pp. 56–73, 2019.

[182] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[183] J. L. Collinger, B. Wodlinger, J. E. Downey, W. Wang, E. C. Tyler-Kabara, D. J. Weber, A. J. McMorland, M. Velliste, M. L. Boninger, and A. B. Schwartz, “High-performance neuroprosthetic control by an individual with tetraplegia,” *The Lancet*, vol. 381, no. 9866, pp. 557–564, 2013.

[184] NeuraLink. (2021) Monkey MindPong. [Online]. Available: https://neuralink.com/blog/

[185] National Spinal Cord Injury Statistical Center, “Facts and figures at a glance,” *University of Alabama at Birmingham*, 2016.

[186] X.-X. Yin, L. Sun, Y. Fu, R. Lu, and Y. Zhang, “U-net-based medical image segmentation,” *Journal of Healthcare Engineering*, vol. 2022, 2022.

[187] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[188] A. Raganato and J. Tiedemann, "An analysis of encoder representations in transformer-based machine translation," in *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. The Association for Computational Linguistics, 2018.

[189] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[190] P. Bonifazi *et al.*, "In vitro large-scale experimental and theoretical studies for the realization of bi-directional brain-prostheses," *Frontiers in Neural Circuits*, vol. 7, p. 40, 2013.

[191] A. Nawrot, T. Pistohl, S. Schrader, U. Hehl, V. Rodriguez, and A. Aertsen, "Embedding living neurons into simulated neural networks," in *IEEE Conference on Neural Engineering*, 2003, pp. 229–232.

[192] A. Zbrzeski *et al.*, "Bio-inspired controller on an FPGA applied to closed-loop diaphragmatic stimulation," *Frontiers in Neuroscience*, vol. 10, p. 275, 2016.

[193] Y. Mosbacher, F. Khoyratee, M. Goldin, S. Kanner, Y. Malakai, M. Silva, F. Grassia, Y. B. Simon, J. Cortes, A. Barzilai *et al.*, "Toward neuroprosthetic real-time communication from in silico to biological neuronal network via patterned optogenetic stimulation," *Scientific Reports*, vol. 10, no. 1, pp. 1–16, 2020.

[194] S. Buccelli, Y. Bornat, I. Colombi, M. Ambroise, L. Martines, V. Pasquale, M. Bisio, J. Tessadori, P. Nowak, F. Grassia *et al.*, "A neuromorphic prosthesis to restore communication in neuronal networks," *IScience*, vol. 19, pp. 402–414, 2019.

[195] T. Berger *et al.*, "A cortical neural prosthesis for restoring and enhancing memory," *Journal of Neural Engineering*, vol. 8, no. 4, 2011.

[196] T. Berger *et al.*, "A hippocampal cognitive prosthesis: multi-input, multi-output nonlinear modeling and vlsi implementation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 2, pp. 198–211, 2012.

[197] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron," *Brain Research Bulletin*, vol. 50, no. 5, pp. 303–304, 1999.

[198] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[199] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[200] E. M. Izhikevich, “Which model to use for cortical spiking neurons?” *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[201] W. Luk, D. Thomas, “FPGA accelerated simulation of biologically plausible spiking neural networks,” in *IEEE Symposium on Field Programmable Custom Computing Machines*, 2009, pp. 45–52.

[202] M. Ambroise, T. Levi, Y. Bornat, and S. Saighi, “Biorealistic spiking neural network on FPGA,” in *IEEE Information Sciences and Systems*, 2013, pp. 1–6.

[203] D. Pani *et al.*, “An FPGA platform for real-time simulation of spiking neuronal networks,” *Frontiers in Neuroscience*, vol. 11, no. 90, pp. 1–13, 2017.

[204] K. Cheung, S. R. Schultz, and W. Luk, “NeuroFlow: a general purpose spiking neural network simulation platform using customizable processors,” *Frontiers in Neuroscience*, vol. 9, no. 516, pp. 1–15, 2016.

[205] F. Khoyratee, F. Grassia, S. Saïghi, and T. Levi, “Optimized real-time biomimetic neural network on FPGA for bio-hybridization,” *Frontiers in Neuroscience*, vol. 13, p. 377, 2019.

[206] K. Akbarzadeh-Sherbaf, B. Abdoli, S. Safari, and A.-H. Vahabie, “A scalable FPGA architecture for randomly connected networks of hodgkin-huxley neurons,” *Frontiers in Neuroscience*, vol. 12, p. 698, 2018.

[207] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, “The spinnaker project,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[208] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, “A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs),” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 106–122, 2017.

[209] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, “A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses,” *Frontiers in Neuroscience*, vol. 9, p. 141, 2015.

[210] J. S. Seo *et al.*, “A 45 nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons,” in *IEEE Custom Integrated Circuits Conference*, 2011, pp. 1–4.

[211] C. Frenkel *et al.*, “A 0.086-mm^2 12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 1, pp. 145–158, 2019.

[212] F. Akopyan *et al.*, “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[213] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[214] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He *et al.*, "Towards artificial general intelligence with hybrid tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[215] R.B. Wells, "Cortical neurons and circuits: A tutorial introduction [online]. 2005 [cit. 2008-02-05]," *URL: http://www.mrc.uidaho.edu/˜rwells/techdocs/CorticalNeuronsandCircuits.pdf.*

[216] K. K. Parhi, C.-Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined dsp architectures," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 1, pp. 29–43, 1992.

[217] D. Tsai, D. Sawyer, A. Bradd, R. Yuste, and K. L. Shepard, "A very large-scale microelectrode array for cellular-resolution electrophysiology," *Nature Communications*, vol. 8, no. 1, p. 1802, 2017.

[218] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[219] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.

[220] A. Tavanaei and A. Maida, "Bp-stdp: Approximating backpropagation using spike timing dependent plasticity," *Neurocomputing*, vol. 330, pp. 39–47, 2019.

[221] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in Neuroscience*, vol. 7, p. 272, 2014.

[222] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, "Feedforward categorization on aer motion events using cortex-like features in a spiking neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1963–1978, 2014.

[223] M. Beyeler, N. D. Dutt, and J. L. Krichmar, "Categorization and decision-making in a neurobiologically plausible spiking network using a stdp-like learning rule," *Neural Networks*, vol. 48, pp. 109–124, 2013.

[224] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Computation*, vol. 19, no. 11, pp. 2881–2912, 2007.

[225] S. An, M. Lee, S. Park, H. Yang, and J. So, "An ensemble of simple convolutional neural network models for mnist digit recognition," *arXiv preprint arXiv:2008.10400*, 2020.

[226] A. P. Davison, D. Brüderle, J. M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "Pynn: a common interface for neuronal network simulators," *Frontiers in Neuroinformatics*, vol. 2, p. 11, 2009.

[227] T. C. Stewart, B. Tripp, and C. Eliasmith, "Python scripting in the nengo simulator," *Frontiers in Neuroinformatics*, vol. 3, p. 7, 2009.

[228] C. Hofstötter *et al.*, "The cerebellum chip: an analog VLSI implementation of a cerebellar model of classical conditioning," in *Advances in Neural Information Processing Systems*, 2005, pp. 577–584.

[229] J. Park, S. Ha, T. Yu, E. Neftci, and G. Cauwenberghs, "A 65k-neuron 73-mevents/s 22-pj/event asynchronous micro-pipelined integrate-and-fire array transceiver," in *IEEE BioCAS Proceedings*, 2014, pp. 675–678.

[230] C. Mead, "Analog VLSI and neutral systems," *NASA STI/Recon Technical Report A*, vol. 90, 1989.

[231] A. Joubert, B. Belhadj, O. Temam, and R. Héliot, "Hardware spiking neurons design: Analog or digital?" in *IEEE International Joint Conference on Neural Networks*, 2012, pp. 1–5.

[232] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of Neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.

[233] R. Naud, N. Marcille, C. Clopath, and W. Gerstner, "Firing patterns in the adaptive exponential integrate-and-fire model," *Biological Cybernetics*, vol. 99, no. 4-5, p. 335, 2008.

[234] S. Pal, V. Gupta, W. H. Ki, and A. Islam, "Design and development of memristor-based rram," *IET Circuits, Devices & Systems*, vol. 13, no. 4, pp. 548–557, 2019.

[235] M. Chu, B. Kim, S. Park, H. Hwang, M. Jeon, B. H. Lee, and B.-G. Lee, "Neuromorphic hardware system for visual pattern recognition with memristor array and cmos neuron," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2410–2419, 2014.

[236] A. Shukla and U. Ganguly, "An on-chip trainable and the clock-less spiking neural network with 1r memristive synapses," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 4, pp. 884–893, 2018.

[237] B. Chen, H. Yang, F. Zhuge, Y. Li, T.-C. Chang, Y.-H. He, W. Yang, N. Xu, and X.-S. Miao, "Optimal tuning of memristor conductance variation in spiking neural networks for online unsupervised learning," *IEEE Transactions on Electron Devices*, vol. 66, no. 6, pp. 2844–2849, 2019.

[238] N. Zheng and P. Mazumder, “Learning in memristor crossbar-based spiking neural networks through modulation of weight-dependent spike-timing-dependent plasticity,” *IEEE Transactions on Nanotechnology*, vol. 17, no. 3, pp. 520–532, 2018.

[239] W. Wu, M. J. Black, Y. Gao, E. Bienenstock, M. Serruya, A. Shaikhouni, and J. P. Donoghue, “Neural decoding of cursor motion using a kalman filter,” *Advances in neural information processing systems*, pp. 133–140, 2003.

[240] M. Serruya, N. Hatsopoulos, M. Fellows, L. Paninski, and J. Donoghue, “Robustness of neuroprosthetic decoding algorithms,” *Biological Cybernetics*, vol. 88, no. 3, pp. 219–228, 2003.

[241] E. Stark and M. Abeles, “Predicting movement from multiunit activity,” *Journal of Neuroscience*, vol. 27, no. 31, pp. 8387–8394, 2007.

[242] J. D. Wander and R. P. Rao, “Brain–computer interfaces: a powerful tool for scientific inquiry,” *Current Opinion in Neurobiology*, vol. 25, pp. 70–75, 2014.

[243] J. J. Williams, A. G. Rouse, S. Thongpang, J. C. Williams, and D. W. Moran, “Differentiating closed-loop cortical intention from rest: building an asynchronous electrocorticographic bci,” *Journal of Neural Engineering*, vol. 10, no. 4, p. 046001, 2013.

[244] C.-H. Han, K.-R. Müller, and H.-J. Hwang, “Brain-switches for asynchronous brain–computer interfaces: A systematic review,” *Electronics*, vol. 9, no. 3, p. 422, 2020.

[245] Z. Bozorgzadeh, G. E. Birch, and S. G. Mason, “The lf-asd brain computer interface: on-line identification of imagined finger flexions in the spontaneous eeg of able-bodied subjects,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4. IEEE, 2000, pp. 2385–2388.

[246] J. F. Borisoff, S. G. Mason, and G. E. Birch, “Brain interface research for asynchronous control applications,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 160–164, 2006.

[247] A. Samiei and H. Hashemi, “A bidirectional neural interface soc with adaptive iir stimulation artifact cancelers,” *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2142–2157, 2021.

[248] L. Shen, N. Lu, and N. Sun, “A 1-v 0.25-μw inverter stacking amplifier with 1.07 noise efficiency factor,” *IEEE Journal of Solid-State Circuits*, vol. 53, no. 3, pp. 896–905, 2018.

[249] H. Chandrakumar and D. Marković, “A 15.2-enob 5-khz bw 4.5-μw chopped ct $\delta\sigma$-adc for artifact-tolerant neural recording front ends,” *IEEE Journal of Solid-State Circuits*, vol. 53, no. 12, pp. 3470–3483, 2018.

[250] S. Yoshimoto, T. Araki, T. Uemura, T. Nezu, T. Sekitani, T. Suzuki, F. Yoshida, and M. Hirata, “Implantable wireless 64-channel system with flexible ecog electrode and optogenetics probe,” in *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2016, pp. 476–479.

[251] Z. Zhang, O. W. Savolainen, and T. G. Constandinou, "Algorithm and hardware considerations for real-time neural signal on-implant processing," *Journal of Neural Engineering*, vol. 19, no. 1, p. 016029, 2022.

[252] J. D. Simeral *et al.*, "Home use of a percutaneous wireless intracortical brain-computer interface by individuals with tetraplegia," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 7, pp. 2313–2325, 2021.

[253] R. Sedgewick and K. Wayne, "Algorithms," 2011.

[254] A. Wang, Z. Jin, C. Song, and W. Xu, "Adaptive compressed sensing architecture in wireless brain-computer interface," in *Proceedings of the Design Automation Conference*, 2015, pp. 1–6.

[255] R. R. Shrivastwa, V. Pudi, C. Duo, R. So, A. Chattopadhyay, and G. Cuntai, "A brain–computer interface framework based on compressive sensing and deep learning," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 90–96, 2020.

[256] N. Achtman, A. Afshar, G. Santhanam, M. Y. Byron, S. I. Ryu, and K. V. Shenoy, "Free-paced high-performance brain–computer interfaces," *Journal of Neural Engineering*, vol. 4, no. 3, p. 336, 2007.

[257] A. Szűcs, "Applications of the spike density function in analysis of neuronal firing patterns," *Journal of Neuroscience Methods*, vol. 81, no. 1-2, pp. 159–167, 1998.

[258] L. Rabiner and B. Juang, "An introduction to hidden markov models," *IEEE Acoustics, Speech, and Signal Processing Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[259] A. Kübler, E. M. Holz, A. Riccio, C. Zickler, T. Kaufmann, S. C. Kleih, P. Staiger-Sälzer, L. Desideri, E.-J. Hoogerwerf, and D. Mattia, "The user-centered design as novel perspective for evaluating the usability of BCI-controlled applications," *PloS One*, vol. 9, no. 12, p. e112392, 2014.

[260] C. J. Van Rijsbergen, "Information retrieval. 2nd. newton, ma," 1979.

[261] C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming," in *Caltech Conference on Very Large Scale Integration*. Springer, 1983, pp. 87–116.

[262] T. Wu, J. Xu, Y. Lian, A. Khalili, A. Rastegarnia, C. Guan, and Z. Yang, "A 16-channel nonparametric spike detection asic based on EC-PC decomposition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 1, pp. 3–17, 2015.

[263] E. Christensen, A. Abosch, J. A. Thompson, and J. Zylberberg, "Inferring sleep stage from local field potentials recorded in the subthalamic nucleus of parkinson's patients," *Journal of Sleep Research*, vol. 28, no. 4, p. e12806, 2019.

[264] A. R. Adamantidis, C. Gutierrez Herrera, and T. C. Gent, "Oscillating circuitries in the sleeping brain," *Nature Reviews Neuroscience*, vol. 20, no. 12, pp. 746–762, 2019.

[265] B. E. Kilavik, M. Zaepffel, A. Brovelli, W. A. MacKay, and A. Riehle, "The ups and downs of beta oscillations in sensorimotor cortex," *Experimental neurology*, vol. 245, pp. 15–26, 2013.

[266] S. R. Nason, A. K. Vaskov, M. S. Willsey, E. J. Welle, H. An, P. P. Vu, A. J. Bullard, C. S. Nu, J. C. Kao, K. V. Shenoy *et al.*, "A low-power band of neuronal spiking activity dominated by local single units improves the performance of brain–machine interfaces," *Nature Biomedical Engineering*, vol. 4, no. 10, pp. 973–983, 2020.

[267] D. A. Heldman, W. Wang, S. S. Chan, and D. W. Moran, "Local field potential spectral tuning in motor cortex during reaching," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 180–183, 2006.

[268] J. A. Perge, S. Zhang, W. Q. Malik, M. L. Homer, S. Cash, G. Friehs, E. N. Eskandar, J. P. Donoghue, and L. R. Hochberg, "Reliability of directional information in unsorted spikes and local field potentials recorded in human motor cortex," *Journal of Neural Engineering*, vol. 11, no. 4, p. 046007, 2014.

[269] T. Milekovic, A. A. Sarma, D. Bacher, J. D. Simeral, J. Saab, C. Pandarinath, B. L. Sorice, C. Blabe, E. M. Oakley, K. R. Tringale *et al.*, "Stable long-term bci-enabled communication in als and locked-in syndrome using lfp signals," *Journal of Neurophysiology*, vol. 120, no. 7, pp. 343–360, 2018.

[270] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Impact of referencing scheme on decoding performance of lfp-based brain-machine interface," *Journal of Neural Engineering*, vol. 18, no. 1, p. 016028, 2021.

[271] F. J. Harris, *Multirate signal processing for communication systems*. CRC Press, 2022.

[272] J. Tamarkin, "A new proof of parseval's identity for trigonometric functions," *Annals of Mathematics*, pp. 541–547, 1926.

[273] M. M. Shanechi, A. L. Orsborn, H. G. Moorman, S. Gowda, S. Dangi, and J. M. Carmena, "Rapid control and feedback rates enhance neuroprosthetic control," *Nature communications*, vol. 8, no. 1, p. 13825, 2017.

[274] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," *arXiv preprint arXiv:1611.01576*, 2016.

[275] H. Van Mier, L. Tempel, J. Perlmutter, M. Raichle, and S. Petersen, "Changes in brain activity during motor learning measured with pet: effects of hand of performance and practice," *Journal of Neurophysiology*, vol. 80, no. 4, pp. 2177–2199, 1998.

[276] D. M. Mehler, A. N. Williams, F. Krause, M. Lührs, R. G. Wise, D. L. Turner, D. E. Linden, and J. R. Whittaker, "The bold response in primary motor cortex and supplementary motor area during kinesthetic motor imagery based graded fmri neurofeedback," *Neuroimage*, vol. 184, pp. 36–44, 2019.

[277] V. Shinde, G. Jai Kumar, D. Valencia, and A. Alimohammad, "High-throughput and compact reconfigurable architectures for recursive filters," *IET Communications*, vol. 12, no. 13, pp. 1616–1623, 2018.

[278] G. Atzeni, J. Lim, J. Liao, A. Novello, J. Lee, E. Moon, M. Barrow, J. Letner, J. Costello, S. R. Nason *et al.*, "A 260$\times$ 274 μm 2 572 nw neural recording micromote using near-infrared power transfer and an rf data uplink," in *Symposium on VLSI Technology and Circuits*. IEEE, 2022, pp. 64–65.

[279] H. An, S. R. Nason-Tomaszewski, J. Lim, K. Kwon, M. S. Willsey, P. G. Patil, H.-S. Kim, D. Sylvester, C. A. Chestek, and D. Blaauw, "A power-efficient brain-machine interface system with a sub-mw feature extraction and decoding asic demonstrated in nonhuman primates," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 16, no. 3, pp. 395–408, 2022.

[280] J.-W. Hong, C. Yoon, K. Jo, J. H. Won, and S. Park, "Recent advances in recording and modulation technologies for next-generation neural interfaces," *Iscience*, vol. 24, no. 12, 2021.

[281] D. Katoozian, H. Hosseini-Nejad, M.-R. Abolghasemi Dehaqani, A. Shoeibi, and J. Manuel Gorriz, "A hardware efficient intra-cortical neural decoding approach based on spike train temporal information," *Integrated Computer-Aided Engineering*, vol. 29, no. 4, pp. 431–445, 2022.

[282] J. Lin, D. Lai, Z. Wan, L. Feng, J. Zhu, J. Zhang, Y. Wang, and K. Xu, "Representation and decoding of bilateral arm motor imagery using unilateral cerebral lfp signals," *Frontiers in Human Neuroscience*, vol. 17, p. 1168017, 2023.

[283] N. V. Swindale, "Orientation tuning curves: empirical description and estimation of parameters," *Biological Cybernetics*, vol. 78, pp. 45–56, 1998.

[284] S. Fabbri, A. Caramazza, and A. Lingnau, "Tuning curves for movement direction in the human visuomotor system," *Journal of Neuroscience*, vol. 30, no. 40, pp. 13 488–13 498, 2010.

[285] S. N. Baker, "Oscillatory interactions between sensorimotor cortex and the periphery," *Current opinion in neurobiology*, vol. 17, no. 6, pp. 649–655, 2007.

[286] E. E. Fetz, "Volitional control of cortical oscillations and synchrony," *Neuron*, vol. 77, no. 2, pp. 216–218, 2013.

[287] H.-V. Ngo, J. Fell, and B. Staresina, "Sleep spindles mediate hippocampal-neocortical coupling during long-duration ripples," *elife*, vol. 9, p. e57011, 2020.

[288] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, “Decoding hand kinematics from local field potentials using long short-term memory (lstm) network,” in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*. IEEE, 2019, pp. 415–419.

[289] N. S. K. Fathy, J. Huang, and P. P. Mercier, “A digitally assisted multiplexed neural recording system with dynamic electrode offset cancellation via an lms interference-canceling filter,” *IEEE Journal of Solid-State Circuits*, vol. 57, no. 3, pp. 953–964, 2021.

[290] N. S. K. Fathy, R. Vatsyayan, A. M. Bourhis, S. A. Dayeh, and P. P. Mercier, “A 0.00179 mm 2/ch chopper-stabilized tdma neural recording system with dynamic eov cancellation and predictive mixed-signal impedance boosting,” *IEEE Transactions on Biomedical Circuits and Systems*, 2024.

[291] S. Mondal, C.-L. Hsu, R. Jafari, and D. A. Hall, “A dynamically reconfigurable ecg analog front-end with a 2.5$\times$ data-dependent power reduction,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 15, no. 5, pp. 1066–1078, 2021.

[292] S. D. Stavisky, J. C. Kao, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy, “Hybrid decoding of both spikes and low-frequency local field potentials for brain-machine interfaces,” in *International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2014, pp. 3041–3044.

[293] H.-L. Hsieh and M. M. Shanechi, “Multiscale brain-machine interface decoders,” in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2016, pp. 6361–6364.

[294] D. Valencia, P. P. Mercier, and A. Alimohammad, “An efficient brain-switch for asynchronous brain-computer interfaces,” *IEEE Transactions on Biomedical Circuits and Systems*, 2024.

[295] D. G. Manolakis and V. K. Ingle, *Applied Digital Signal Processing: Theory and Practice*. Cambridge University Press, 2011.

[296] J. F. Kaiser, “On a simple algorithm to calculate the’energy’of a signal,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1990, pp. 381–384.

[297] A. K. Bansal, W. Truccolo, C. E. Vargas-Irwin, and J. P. Donoghue, “Decoding 3d reach and grasp from hybrid signals in motor and premotor cortices: spikes, multiunit activity, and local field potentials,” *Journal of Neurophysiology*, vol. 107, no. 5, pp. 1337–1355, 2012.

[298] R. D. Flint, C. Ethier, E. R. Oby, L. E. Miller, and M. W. Slutzky, “Local field potentials allow accurate decoding of muscle activity,” *Journal of Neurophysiology*, vol. 108, no. 1, pp. 18–24, 2012.

[299] J. Zhuang, W. Truccolo, C. Vargas-Irwin, and J. P. Donoghue, “Decoding 3-d reach and grasp kinematics from high-frequency local field potentials in primate primary motor cortex,” *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 7, pp. 1774–1784, 2010.

[300] S. R. Nason, M. J. Mender, A. K. Vaskov, M. S. Willsey, N. G. Kumar, T. A. Kung, P. G. Patil, and C. A. Chestek, “Real-time linear prediction of simultaneous and independent movements of two finger groups using an intracortical brain-machine interface,” *Neuron*, vol. 109, no. 19, pp. 3164–3177, 2021.

[301] J. A. Gallego, M. G. Perich, R. H. Chowdhury, S. A. Solla, and L. E. Miller, “Long-term stability of cortical population dynamics underlying consistent behavior,” *Nature Neuroscience*, vol. 23, no. 2, pp. 260–270, 2020.

[302] J. A. Gallego, M. G. Perich, L. E. Miller, and S. A. Solla, “Neural manifolds for the control of movement,” *Neuron*, vol. 94, no. 5, pp. 978–984, 2017.