A 130-µW, 64-Channel Spike-Sorting DSP Chip

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical Engineering

by

Vaibhav Pradeep Karkare

2009
© Copyright by
Vaibhav Pradeep Karkare
2009
The thesis of Vaibhav Pradeep Karkare is approved.

Rajeev Jain

Jack W. Judy

Dejan Marković, Committee Chair

University of California, Los Angeles

2009
To my parents, Maya and Pradeep Karkare...
# Table of Contents

1 Introduction ................................................. 1  
  1.1 Spike-sorting process ........................................ 3  
  1.2 Motivation for on-chip spike sorting .......................... 5  
  1.3 Design constraints and previous work ........................ 6  
  1.4 Organization of thesis ....................................... 7  

2 Spike-Sorting Algorithm Evaluation ............................ 9  
  2.1 Evaluation methods ............................................ 9  
  2.2 Algorithm evaluation results .................................. 12  

3 Spike-Sorting DSP Architecture ................................ 16  
  3.1 Functional description ........................................ 16  
  3.2 Architecture optimization .................................... 18  
  3.3 Circuit-level implementation .................................. 22  

4 Design and Verification Flow .................................. 27  
  4.1 Matlab/Simulink-based chip design flow ........................ 27  
  4.2 ASIC implementation ......................................... 29  
  4.3 Matlab-controlled ASIC verification flow ........................ 32  

5 Results and Conclusions ...................................... 35  
  5.1 Chip measurement results ..................................... 35  
  5.2 Conclusions ................................................... 39  

iv
A Efficiency Comparison of Analog versus Digital Spike-Detection 40

B Spike-Sorting Methods ........................................ 44

References ............................................................. 49
## List of Figures

1.1 The Utah array: A 100-channel neural recording electrode array. ........................................ 2
1.2 Spike-sorting process. ........................................................................................................... 3
1.3 Design process used for the spike-sorting DSP ASIC. .......................................................... 7

2.1 Methodology used for performance evaluation of spike-sorting algorithms. ............................. 11
2.2 Median probability of detection versus computational complexity for spike-detection algorithms. .................................................................................................................. 13
2.3 Median classification accuracy versus computational complexity for FE algorithms. .............. 13
2.4 Median classification accuracy for DD and DWT FE algorithms when used with different alignment methods. ........................................................................ 14

3.1 16-channel DSP core architecture. ......................................................................................... 17
3.2 Energy-delay tradeoff for a single-channel spike-sorting DSP core. ........................................ 19
3.3 Data-stream interleaving. ....................................................................................................... 20
3.4 Energy-delay tradeoff for spike-sorting DSP core for various degrees of interleaving. ............. 21
3.5 Energy-per-sample for 64 channels (left axis) and area savings (right axis) versus number of channels interleaved. ................................................................. 22
3.6 Power gating: NMOS and PMOS headers. .............................................................................. 23
3.7 \( I_{ON}/I_{OFF} \) versus \( V_{DD} \) for a fixed \( V_{DS} \) of 10 mV. .................................................. 24
3.8 Leakage current versus delay increase for NMOS and PMOS headers. ................................. 25
## List of Tables

5.1 Chip Summary ................................................. 37
5.2 Comparison with Previous Work .............................. 38
I would like to thank Prof. Dejan Marković for the constant guidance and support throughout this work. Completing this thesis would have been impossible without his help. I am also thankful to Sarah Gibson for her work on the evaluation of spike-sorting algorithms. She also helped me make almost every presentation and figure that was needed to present this work. My sincere acknowledgments are due to Chia-Hsiang Yang and Chengcheng Wang for their help during the tapeout. My roommates BibhuDutta Sahoo, Abhishek Ghosh, and Nitesh Singhal were also a tremendous help during the tapeout. They were willing to cut down on their sleep to help me fix DRC errors! I am also thankful to my group members Rashmi Nanda and Victoria Wang for several discussions during the course of this work. My sincere thanks to Prof. Jack Judy for his suggestions and encouragement during the project. My acknowledgments are due to Dr. Rick Staba at the Reed Neuroscience Research Center for allowing access to human neural recordings for testing our spike-sorting DSP chip. I would also like to thank Preeti Mulage for her comments on the thesis. I would like to express my deep gratitude towards my parents, family, and friends for their constant love, encouragement, and support.

This material is based upon work supported by the National Science Foundation under Grant No. 0847088. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Abstract of the Thesis

A 130-$\mu$W, 64-Channel Spike-Sorting DSP Chip

by

Vaibhav Pradeep Karkare
Master of Science in Electrical Engineering
University of California, Los Angeles, 2009
Professor Dejan Marković, Chair

Spike sorting is an important processing step in various neuroscientific and clinical studies. An on-chip spike-sorting DSP must provide data-rate reduction while maintaining a power density less than $800$ $\mu$W/mm$^2$. Most existing designs either provide only spike detection for multi-channel processing, or they provide detection and feature extraction only for a single channel. We demonstrate a chip for detection, alignment, and feature extraction simultaneously for 64 channels. Spike-sorting algorithms identified from a complexity-performance analysis are implemented on an ASIC using a Matlab/Simulink-based architecture design framework. The chip has a modular architecture which allows it to be configured to process 16, 32, 48, or 64 channels. Inactive cores are power-gated to reduce power consumption when the chip operates for less than 64 channels. The chip is implemented in a 90-nm commercial CMOS process and has a power dissipation of 130 $\mu$W (power density of 30 $\mu$W/mm$^2$) when processing all 64 channels. A data-rate reduction of 91.25% (11.71 Mbps to 1.02 Mbps) is achieved.
CHAPTER 1

Introduction

Neurons in the brain communicate with each other using electrical signals known as “action potentials” or “spikes”. In the recent past, a significant progress has been made in our understanding of the brain, owed largely to the analysis of electrical signals recorded from the brain. In modern neuro-scientific studies and clinical procedures, neural signals are recorded from the brain using implanted electrode arrays. Figure 1.1 shows an example of a 100-channel neural recording electrode array developed at the University of Utah. The array consists of 100 electrodes located at a pitch of 400 µm.

Each electrode in such an electrode array records signals from multiple neurons. While analyzing the collective signal of a group of neurons is interesting for some studies, many neuroscientific analyses require knowledge of single-neuron activities. For instance, correlations between an applied stimulus and the activity of an individual neuron or the correlation between the activity of different individual neurons can only be observed from single-unit neural recordings. Further, single-unit neural recordings are shown to significantly enhance the performance of brain-machine interfaces [1]. Therefore, it is important to classify the multi-unit action potentials recorded by a single electrode based on their source neurons. This process of classifying the recorded action potentials according to the neurons from which they originate is called spike sorting. Such a classification of action potentials based on the source neurons is possible due to the fact that the shape
of the action potential generated by a given neuron is different from the rest of the neurons.

The process of spike sorting is challenging due to several reasons. Even though the action potential of a neuron can be distinguished from other neurons, the action potential of a given neuron can vary with time. Electrode movement with respect to brain degrades the signal received from a given neuron over time. Further, the non-stationary background noise (which is a combination of biological and thermal noise) makes it difficult to classify action potentials. Spike sorting, therefore, requires special algorithms which can perform classification in the low SNR situation found at the recording electrode. These spike-sorting algorithms can also be considered as lossy data compression techniques in which the objective is to retain the signal of interest, namely action potentials, while the rest of the signal is discarded. Since the spikes are sparse in the recorded neural data, spike sorting provides us with a significant reduction in the output data rate. Several references like [2] give a good overview of the most popular spike-sorting algorithms. The following section provides a brief discussion about the spike-sorting process.
1.1 Spike-sorting process

Spike sorting can be divided into three major steps: (1) detection and alignment, (2) feature extraction (FE), and (3) clustering. Figure 1.2 illustrates these spike-sorting steps. Recorded neural data contains action potentials and local field potentials (commonly referred to as LFPs). LFPs have a higher amplitude (of around 2 mV to 6 mV) compared to that of action potentials, which have amplitudes ranging from 10 µV to 1 mV. LFPs occupy a frequency band of 0.5 to 250 Hz, while action potentials typically occupy a range of 300 Hz to 3 kHz. Raw neural data recorded from the brain is amplified, digitized, and high-pass filtered to remove the LFP\(^1\). These operations are grouped under the “analog front end” module in Fig. 1.2. Spike sorting, which is the focus of the work presented in this thesis, is performed on this filtered and digitized neural data.

The first spike-sorting step is spike detection. As the name indicates, the objective of this step is to search for action potentials in the raw neural data. This is typically done by placing a threshold on the raw data or on a signal derived from the raw data. Several detection methods are used in the neuroscientific

\(^1\)Local field potentials (LFPs) are of interest to clinicians for several studies like diagnosis of epileptic patients. However, the focus of this thesis is spike sorting. Hence we assume LFP to be filtered immediately after the signal is recorded.
community which differ from each other in the signal transformation applied and the thresholding method used. Some detection methods perform detection in the analog domain as opposed to the digital domain. An efficiency comparison of analog and digital threshold detection methods is presented in Appendix A. Spike detection provides us with approximately 80% reduction in the data-rate. If we assume a sampling rate of 24 kHz and a relatively high neuronal firing rate of 100 Hz, detection would provide us with an output data rate that is $5 \times$ lower than the data rate for raw signals. Recorded spikes are aligned to a common reference like the maximum spike amplitude or maximum spike derivative in order to avoid mis-aligned spikes from the same neuron being classified into different groups.

There are two major problems with direct transmission of spike waveforms extracted after detection. First, the background noise makes it difficult to classify the time-domain waveforms. Second, we need to reduce the data rate below the rate required for transmission of spike waveforms in order to support a higher number of channels. In order to better separate the spike from each other, the spikes are transformed to a feature space. This transformation process is called feature extraction and is the second major spike-sorting step. Feature extraction also reduces the data that needs to be output per spike, thereby reducing the output data rate and the computational complexity of the processing that follows. The discrete wavelet transform [3], principal component analysis [4], the integral transform [5], and discrete derivatives [6] are some examples of the transformations used for feature extraction.

The extracted features of the recorded action potentials are finally passed through clustering, where the input spike features are grouped into clusters. Most clustering methods require storage of the entire data to be classified, in order to map each spike to its class. K-means and fuzzy c-means are examples of such
clustering methods. These methods also require a user input for the number of clusters present in the data. Since the precise number of neurons is not known, the user often has to guess the number of clusters present in the data, which makes the entire clustering process susceptible to human errors. In order to avoid these limitations, unsupervised clustering algorithms like super-paramagnetic clustering [7] and online clustering [8] can be used.

1.2 Motivation for on-chip spike sorting

In a traditional neural recording system, the data recorded from the brain is usually sent unamplified outside the body using wires. Amplification, filtering and digitization is performed externally. Spike sorting on the filtered data is performed offline in software [2]. This setup for neural recording has several disadvantages. It does not allow real-time processing of data, which is a pre-requisite for brain-machine interfaces. Also, it can only support a limited number of channels. The wires used to transfer data outside the body hinder free movement of the subject and also increase the risk of infection. Wireless transmission of data is highly desirable, since it avoids most of the above mentioned problems. However, power constraints allow raw data to be transmitted wirelessly only for a limited number of channels. Performing spike sorting on a chip implanted inside the skull is a promising option to enable the processing and wireless transmission of information for a large number of channels. Spike sorting on an implantable chip provides fast processing, thereby enabling the real-time, many-channel, closed-loop control necessary for brain-machine interfaces. The data reduction provided by spike sorting enables wireless transmission of data for a large number of channels.
1.3 Design constraints and previous work

An implantable spike-sorting DSP chip is subject to aggressive power density constraints. The power density of 800 $\mu$W/mm$^2$ has been shown to damage brain cells [9]. We would therefore, like to have a power density much lower than this limit. It has also been shown that in order to avoid tissue damage, the circuit should not increase the temperature of the neurons by more than 1 degree centigrade [10]. To support wireless transmission for a large number of channels while meeting the power density constraint, the DSP must provide significant data-rate reduction. Implantable ICs should also have a small area, to allow for the possibility to integrate the DSP, along with analog frontend and RF circuits, with the base of an implanted electrode array, which is 16 mm$^2$ for the Utah electrode array.$^2$

Spike-sorting DSPs that satisfy the constraints for an implantable device have been published earlier in literature. However, most of these DSPs either provide only detection for multiple channels [11], or they provide detection and feature extraction but only support a single channel at a time [12], [13]. Also, these spike-sorting DSPs do not include alignment, which is known to have a significant impact on the spike classification results. In this work, we demonstrate a spike-sorting DSP chip that can perform detection, alignment, and feature extraction simultaneously for 64 channels. The power density of the chip is 30 $\mu$W/mm$^2$, which is $27 \times$ lower than the power density limit.

$^2$The base plate has been widened in previous work from this conventional size to integrate circuits at the base of the array
1.4 Organization of thesis

A technology-driven algorithm-architecture selection process, illustrated in Fig. 1.3, is used in the design of the spike-sorting DSP. A complexity-performance analysis of several spike-sorting algorithms is performed to select the hardware-friendly algorithms. Energy-efficient choices are made in the algorithm and circuit design process, which lead to a low-power ASIC implementation for the multi-channel spike-sorting DSP. The remainder of this thesis describes each of these steps in the design process, along with the ASIC implementation details.

Chapter 2 presents the analysis for selection of spike-sorting algorithms. The
spike-sorting DSP architecture and the techniques used for architecture and circuit implementation are presented in Chapter 3. Chapter 4 describes the Matlab/Simulink-based design flow, ASIC implementation and the Matlab-controlled ASIC test flow. Finally, Chapter 5 presents the chip results and conclusions of the thesis.
CHAPTER 2

Spike-Sorting Algorithm Evaluation

Several algorithms have been published for each of the spike-sorting steps referred to in Fig. 1.2. However, there is no agreement in literature about which algorithms are best suited for hardware implementation. Complexity-accuracy tradeoffs of several spike-sorting algorithms \(^1\) are analyzed in [14]. We use the algorithms selected from this analysis for our ASIC implementation. This chapter provides a summary of the algorithm evaluation presented in [14].

2.1 Evaluation methods

Probability of detection (\(P_D\)), probability of false alarm (\(P_{FA}\)), and classification accuracy (CA) were used to evaluate the algorithm accuracy. The accuracy numbers for each algorithm are evaluated using simulated neural data across a wide range of SNRs. We choose to use simulated neural data since data with known spike times and classes are required to evaluate the algorithm performance. We used a data-driven neural-signal simulator introduced in [15] to generate test data. The simulator contains a library of 594 average spike shapes taken from a collection of real physiological recordings. Along with the test data (“raw data”), the simulator also generates a file containing the true spike time and the true spike class for each actual spike in the data file, which can be used to calculate accuracy.

\(^1\)A review of these algorithms is presented in Appendix B
Figure 2.1: Methodology used for performance evaluation of spike-sorting algorithms.

(Fig. 2.1). We used this simulator to generate 96 data sets at 17 different noise levels, for a total of 1632 data sets. First, 24 sets of spikes were chosen from the library randomly. From each of these 24 sets, 4 data sets were generated: 2 where the amplitudes of all spikes were equal (normalized to one) and 2 where the amplitudes were unequal. Each of these groups included one data set where the firing rate of each neuron was equal (40 Hz) and one where the firing rates varied (from 5 to 40 Hz). Finally, we generated 17 versions of each of the 96 data sets at different noise levels, where the standard deviation of the noise ranged from $\sigma = 0$ to 0.4. This range allowed us to explore very high to very low SNRs.
All data sets were 60 s in length and generated at a sampling rate of 24 kHz. The SNR values for each data set were calculated using

\[ SNR_{dB} = 10 \log_{10} \frac{P_{signal}}{P_{noise}}. \]  

(2.1)

\( P_{signal} \) of each data set was calculated by generating the 60-s signal with no noise added and calculating the power as the sum-of-squares. For \( P_{noise} \), a 60-s signal was generated at each standard deviation of noise \((\sigma = 0, 0.025, 0.05, \ldots, 0.4)\) where the firing rates of all neurons were set to 0 Hz. \( P_{noise} \) was then calculated as the sum-of-squares.

All detection, alignment and FE algorithms were tested in Matlab on the test data described above. In order to evaluate the quality of each detection method, the false-alarm and detection rates were calculated. The false-alarm rate is defined as

\[ P_{FA} = \frac{\text{number of false alarms}}{\text{number of true negatives}}, \]  

(2.2)

and the detection rate is defined as

\[ P_D = 1 - P_M, \quad P_M = \frac{\text{number of misses}}{\text{number of true positives}}. \]  

(2.3)

We defined true positives as the samples within known spikes and true negatives as all other samples. False alarms are all the samples within a detected spike that are not part of a true spike, and misses are all samples of true spikes that are not part of detected spikes.

As depicted in Fig. 2.1, the accuracy of each FE method was calculated as follows. For each signal over the range of SNRs, true spikes were “detected” using the file of true spike times. This assured that the accuracy calculations for the FE methods were not affected by the possible inaccuracies of spike detection methods. Features from the detected spikes were then extracted using the FE methods. The results were clustered using the Matlab implementation of fuzzy...
c-means, the most accurate out of all clustering methods tried. The accuracy was then calculated by comparing the computed cluster assignment of each spike to the actual identities of each spike. The computational complexity of each of the algorithms was estimated using its respective equations. For detection algorithms, complexity is estimated as the sum of the operations per sample required for pre-emphasis and the operations per sample required for threshold calculations, assuming 10s of training for threshold calculations per 1 hr of data. For FE algorithms, complexity is estimated per spike, assuming 72 samples (3 ms) per spike. We define the complexity as $N_{\text{additions}} + 10N_{\text{multiplications}}$. The analysis for determining the performance of alignment methods is very similar to the analysis described above for analysis of feature extraction methods.

## 2.2 Algorithm evaluation results

This section summarizes the results of the complexity-accuracy tradeoffs for spike sorting algorithms. A brief description of the algorithms referred to in this section can be found in Appendix B. It is found that the probability of detection ($P_D$)
Figure 2.3: Median classification accuracy versus computational complexity for FE algorithms.

for NEO is higher than Stationary wavelet tranform product (SWTP) across the entire SNR range. The probability of detection for absolute value detection is comparable to NEO for high SNRs but it is much lower than NEO for low SNRs. The probability of false alarm ($P_{FA}$) for NEO is better than SWTP and comparable to that of absolute value threshold. Figure 2.2 shows $P_D$ (averaged over all data sets and SNRs) versus computational complexity for different detection methods. Methods that lie in the upper left corner (high accuracy, low complexity) are optimal for our application. Because the NEO method lies at the knee point of this curve, we chose it as the optimal spike detection method for our application. Figure 2.3 shows the classification accuracy versus normalized complexity (averaged over all data sets and SNRs) for several feature extraction methods: Integral transform (IT), Discrete derivatives (DD), Discrete wavelet transform (DWT), and Principal component analysis (PCA). We choose the knee point of the curve corresponding to the DD algorithm as our chosen FE method. Figure 2.4 shows the classification accuracy for different alignment methods evaluated for the DD and DWT FE methods. The plot shows the median accuracy for
Figure 2.4: Median classification accuracy for DD and DWT FE algorithms when used with different alignment methods.

The alignment methods and the shaded area around the plot shows the first and the third quartiles. As seen from this curve, alignment to the maximum derivative gives us the best classification accuracy among all the alignment methods. The complexity of different alignment methods are very similar. Therefore, we decided to use the maximum derivative alignment for our spike-sorting DSP.
CHAPTER 3

Spike-Sorting DSP Architecture

There are several ways in which the algorithms identified from the analysis discussed in Chapter 2 can be mapped into hardware. While dealing with power-limited systems, it is important to make energy-efficient architecture choices to minimize the power consumption of the DSP. This chapter provides a description of a single-channel DSP core followed by the analysis for architecture selection for the multi-channel DSP. Circuit-level techniques used for implementing the spike-sorting DSP ASIC are also described.

3.1 Functional description

Based upon the analysis presented in Chapter 2, we selected the non-linear energy operator (NEO) for detection, maximum derivative for alignment, and discrete derivatives for feature extraction. Figure 3.1 shows the block diagram for a single-channel spike-sorting DSP core. The NEO block calculates \( \psi(n) \), defined by:

\[
\psi(n) = x^2(n) - x(n + 1) \cdot x(n - 1)
\]

for each input sample \( x(n) \). During the training period, the threshold calculation module calculates the threshold as:

\[
Thr = C \frac{1}{N} \sum_{n=1}^{N} \psi(n).
\]
We chose the number of samples in the training period $N$ to be a power of 2 in order to reduce the division by $N$ to a simple shift operation. $C$ in Eq. 3.2 is a constant empirically chosen to be 8 in our implementation [14]. The detector signals a spike-detection event when $\psi(n)$ exceeds the calculated threshold. When a spike is detected, 72 samples (24 samples of preamble and 48 samples post-threshold-crossing) of raw data are saved in the register-bank memory as the recorded action potential. The maximum derivative block calculates the offset for a 48-sample subsection of the spike such that the 11$^{th}$ sample of the subsection is the point of maximum derivative of the recorded spike.

$$\text{Offset} = \arg \max_n (s(n) - s(n - 1)).$$  \hfill (3.3)

The offset is defined by the following equation

$$\text{Offset} = \arg \max_n (s(n) - s(n - 1)).$$  \hfill (3.4)

This offset is used to read in the aligned spike. Discrete derivative features of the aligned spike defined by

$$dd_\delta(n) = s(n) - s(n - \delta),$$  \hfill (3.5)
where $s(n)$ is the $n^{th}$ sample of the spike and $\delta$ is equal to 1, 3, and 7, are calculated. Uniform sampling of the coefficients is performed to select seven coefficients for each level of derivatives extracted. Since only coefficients that would be retained after sampling are evaluated, discrete derivative calculation and coefficient selection is not strictly a two-step process. In addition to the processing described above, control logic is needed for proper scheduling of all the design modules.

Eight-bit input data samples enter the single-channel DSP at a rate of 192 kbps. The aligned spikes identified from the raw data have a rate of 38.4 kbps. Finally, the extracted spike features have a rate of 16.8 kbps. We thus see that a 91.25% data rate reduction is obtained from the input raw data to the output spike features. The detection and alignment modules occupy 33% and 47% of the area of a single-channel spike-sorting DSP core. Feature extraction and the control logic constitute the remaining 20%. The spike-sorting DSP core presented above is a register-dominated design with 50% of the 0.1 mm$^2$ area of the single-channel core being occupied by registers. It is interesting to note that the low operating frequencies and register-dominated nature make the design of the spike-sorting DSP different from the designs in conventional DSP ASICs, which tend to be performance constrained and logic-intensive systems.

### 3.2 Architecture optimization

Now that we are familiar with a single-channel spike-sorting DSP and its features, let us analyze the design from an energy-delay perspective. The DSP has a critical path delay of 466 FO4 inverters at the nominal supply voltage. This implies that the design is $2 \times 10^3$-times faster than the application requirement at the minimum delay point (MDP) shown in Fig. 3.2. The energy-delay curve shown
Figure 3.2: Energy-delay tradeoff for a single-channel spike-sorting DSP core.

in the red line in Fig. 3.2 is based on the assumption that the design is clocked at the maximum frequency at each supply voltage. However, in the case where the application delay is fixed, there is no reward for early computation as the circuit keeps leaking for the remainder of the clock cycle. In this application, the switching activity at the input is estimated to be 0.05. The switching activity is low due to two major reasons. First, the sampling rate for the design is around 4 times higher than the Nyquist rate, which means a high correlation among the data samples. Second, the alignment and feature extraction modules which occupy about 70% area of the design, switch only when a spike is detected. This implies a factor-of-5 reduction in switching activity of these modules even for a high spike firing rate of 100 Hz. The low operating frequency combined with the low sampling rate hints at a leakage-power-dominated design. Infact, if the design is operated at the nominal supply voltage of 1.2 V at the application delay, we would end up at a point which is orders of magnitude higher in energy than
the MEP. At this point, the DSP is heavily leakage dominated. In order to bring
the design to a lower energy, it is desirable to scale down the supply voltage to
trade off the excess delay in favor of energy reduction. However, if the voltage for
the single-channel core is scaled to the lowest point that meets the application
data rate requirement, the design is placed at a point beyond the MEP on the
E-D curve (marked as “Required Delay” on Fig. 3.2). We therefore, decided to
use data-stream interleaving to bring the operating point between the MDP and
the MEP of the design and to lower the total energy consumption.

The concept of data-stream interleaving is illustrated in Fig. 3.3 [16]. The
direct mapped architecture in Fig. 3.3(a), which uses N processing elements
(PEs), is replaced with a single processing element that works at N times the
frequency in Fig. 3.3(b). Data-stream interleaving saves the area and leakage
power of logic since the logic is shared between multiple channels. Also, the logic
depth of the data path is reduced with interleaving. As a result, the MEP of
the design is pushed to higher delays and lower voltages. The total energy con-
sumed reduces, as seen from the energy-delay curves, for two and eight-channel
Interleaving also increases the switching power of registers, since the same number of registers now switch at a higher frequency. The register leakage power, register area, and logic leakage power stay almost constant. Beyond 16-channel interleaving, the increase in register switching dominates over the reduction in energy due to savings in logic leakage.

Figure 3.5 shows a plot for energy per channel versus number of channels interleaved. It is seen that due to the opposite trends of reduction in logic leakage versus increase in register switching power, the total energy per sample has a minimum with respect to the number of channels interleaved at a point where the increase due to register switching balances the decrease in energy due to logic leakage. The energy per channel has a minimum at 4-to 8-channel interleaving and increases significantly beyond 16-channel interleaving.

Interleaving also reduces the area of the design as the logic is shared between
multiple channels. However, the area of the registers is unaffected by interleaving. As the number of channels interleaved increases, the ratio of logic to register area goes down and the savings in area saturate. From Fig. 3.5 it can be seen that the area savings of the DSP remain almost constant beyond 16-channel interleaving. In order to have the maximum area savings with a low energy implementation, we decided to interleave 16 channels per core, which gave us an area savings of 47%. The area savings are lower than conventional designs due to the register dominated nature of the DSP.

Thus from the above analysis, we chose to implement the 64-channel DSP with a modular architecture consisting of four cores, each of which supports a 16-channel interleaved data stream.

3.3 Circuit-level implementation

After deciding the architecture to be used for the spike-sorting DSP, we used the techniques of power gating, logic restructuring, and wordlength optimization for
the circuit-level implementation. We used power gating with sleep devices for both $V_{DD}$ and GND rails to reduce the leakage power consumption. In conventional designs, a PMOS header is needed at the $V_{DD}$ line. This is because the source of an NMOS transistor can only rise up to a $V_{TN}$ below its gate voltage. This limits the core voltage to $V_{DD} - V_{TN}$ in a design which operates at the nominal supply voltage. However in our design, we use a reduced supply voltage at the core, while the voltage on the sleep signal can be pushed up to 1.2 V. We can therefore, use a NMOS header for power gating the $V_{DD}$ rail. The two options for power gating the $V_{DD}$ rail are illustrated in Fig. 3.6. The NMOS sleep transistor when ON has a $V_{GS}$ greater than the core $V_{DD}$, while maintaining negative $V_{GS}$ in the sleep mode. The high overdrive in the active mode and the higher mobility of the NMOS device imply that we need a much narrower NMOS device than the corresponding PMOS to supply a given active-mode current to the core. The reduced width of the NMOS device leads to reduced leakage in the sleep mode. Figure 3.7 shows the ratio of the maximum on-state and off-state currents that

Figure 3.6: Power gating: NMOS and PMOS headers.
can be provided by the NMOS and PMOS header devices for different supply voltages, while maintaining a fixed $V_{DS}$ in the active and sleep modes. We find that for voltages lower than 0.7 V the NMOS header has a higher $I_{ON}/I_{OFF}$ than the PMOS and hence is a more energy-efficient choice. Figure 3.8 shows the leakage current versus delay increase for a test circuit when gated with NMOS and PMOS sleep devices. It was seen that the NMOS sleep device leads to around 10-times lower leakage than the PMOS for the same delay increase. We therefore used NMOS sleep devices for both $V_{DD}$ and GND rails. Power gating allowed us to achieve a 70% reduction in the power of inactive cores and a 30% reduction in the power of active cores. In this chip, one of the four cores was not gated for debug purposes.

We also used logic restructuring and wordlength optimization to further reduce the power and area of the DSP. The logic was designed so as to avoid redundant signal switching. As an illustration, consider the circuit shown in Fig. 3.9 for accumulation of $\psi(n)$ for threshold calculation in the training phase. The output of the accumulation node is gated such that the division for averaging only
happens once at the end of the training period, thereby avoiding the redundant switching as $\psi(n)$ is being accumulated. Gating all inputs to the alignment and feature-extraction modules such that they switch only when a spike is detected is another example for logic restructuring to reduce signal switching. This gave us a 5× reduction in the signal switching power for detection and alignment modules. Opportunities for hardware sharing were also exploited while implementing the logic. For instance, the circuit for calculation of $\psi(n)$ was shared between the training and detection phases. Wordlengths for the signal nodes in the circuit were decided using an automated wordlength optimization tool [17]. Iterative constraints were specified on the MSE at the signal of interest until detection.
or classification errors occur. For instance, it was determined that a wordlength of 31 bits is needed at the accumulation node for calculation of $Thr$, to avoid detection errors. Wordlength optimization gave us a 15% area reduction compared to a design with fixed MSE of $5 \times 10^{-6}$ (the input MSE). The savings due to wordlength optimization are limited since we have a register dominated system. Since the inputs and the outputs of a register are constrained to have the same wordlength, there is limited scope for tuning the signal wordlengths for this design when compared to conventional DSP ASICs.
CHAPTER 4

Design and Verification Flow

In this chapter we describe the chip design and verification flow used for the spike-sorting DSP chip. We used a Matlab/Simulink-based graphical design environment for architecture design. The ASIC was implemented in IBM 90-nm, standard V_T, bulk-CMOS process. Chip verification was performed using a Matlab-controlled, FPGA-driven test interface.

4.1 Matlab/Simulink-based chip design flow

A Matlab/Simulink-based graphical flow is used to design the DSP core. Figure 4.1 shows the design process. The algorithm is implemented as a Simulink model using the Synplify DSP block set. The Synplify DSP block set contains basic building blocks like adders, multipliers, RAMs, registers, etc. The Simulink implementation of the algorithm, built using the Synplify DSP block set, provides a bit-true, cycle-accurate representation of the algorithm. As an illustration, Fig. 4.2 shows the Simulink model for calculation of the NEO operator \( \psi(n) \). Once the spike-sorting algorithms are implemented in Simulink, data from the neural simulator [15] is used to test the functionality of the Simulink model. The HDL code for the design is auto-generated from the Simulink model along with a testbench for verification of the HDL code. The HDL is simulated in Modelsim for functional verification following which it is synthesized using Synopsys.
Design Compiler. Since we intend to use supply voltage scaling on the design, the delay specified in the synthesis constraints accounts for the delay increase at the reduced voltage. The script first synthesizes the design at a 20% increased frequency. We then use incremental compilation to back off from the delay slack in the following synthesis iteration. This allows the tool to re-size the design to a lower energy point. Following synthesis, the delay annotated netlist is simulated in Modelsim to check for its functional verification. Formal verification of the netlist with the HDL code was also performed using the Formality tool from Synopsys. Switching activities from the test vectors are annotated in the synthesis flow for better power estimation. The power, area, and delay estimates from the synthesis reports can be obtained from the Simulink model using an automated flow. This implies that various architectural choices can be evaluated by only changing the Simulink model. Place and Route (PNR) was performed using Encounter tool from Cadence. In order to power gate the multi-core design, PNR for each of the cores was performed separately. The power lines were
routed at the block level using the PNR tool. These cores were then hierarchically instantiated into the top level module and the PNR tool was used for signal routing between the cores. Custom macros were included in the PNR flow my including their LEF (Libarary Exchange Format) file as in input to the PNR tool. Power routing was done manually at the top level. The gds file generated from Encounter was imported into Cadence Virtuoso for the manual layout edits \(^1\). Design rule checking (DRC) and Layout-vs-Schematic (LVS) checks were performed in the Calibre tool from Mentor Graphics. The algorithm description only needs to be entered once in the entire design flow. Thus, the design flow avoids design re-entry which is common to the traditional design flow.

### 4.2 ASIC implementation

The 64-channel spike-sorting DSP was implemented in IBM 90-nm, standard \( V_T \), bulk-CMOS process with 8 metal layers. The design was implemented using a modular architecture consisting of four cores that process 16 channels each. The modularity in architecture allows us to extend the design easily to support a higher number of channels. Figure 4.3 shows the die photograph of the chip.

\(^1\)We needed to fix antenna DRC rules by “metal jogging” on the long interconnects.
64-channel interleaved input data enters the chip at a clock of 1.6 MHz and is split into four streams of 400 kHz each by a serial-parallel (S/P) converter. The slower clock is derived on-chip from the faster clock. At the output, data from the four cores is combined by the parallel-serial (P/S) converter into a single 64-channel interleaved stream. Since we intend to operate the core at a reduced supply voltage, a level converter with cross coupled PMOS devices was designed. The level converter shown in Fig. 4.4 provides a 1.2 V output swing at the I/O pads. Figure 4.5 shows the dual-well layout of the level converter and Fig. 4.6 shows a plot of the delay of the level converter versus the $V_{DDL}$ (core voltage). The $V_{DDH}$ (I/O pad voltage) for Fig. 4.6 is set to 1.2 V. The level converter can provide at speeds of up to 500 MHz when converting 0.3 V signals to 1.2 V. The chip consists of 650 k gates and has a die area of 7.07 mm$^2$ and a core area of 4 mm$^2$. It can support three modes of operations to output: (1) raw data, (2) aligned spikes or (3) spike features. The threshold for detection is calculated on-chip as opposed to many previous spike-sorting DSPs [11][12][13]. The training
Figure 4.4: Schematic of level converter.

Figure 4.5: Dual-well layout of level-converter circuit.

Figure 4.6: Delay vs. $V_{DDL}$ for level converter. $V_{DDH} = 1.2$ V.
period for threshold calculation can be initiated independently for each channel. This implies that a subset of noisy channels can be trained more often than the rest, without affecting the detection for the other channels.

4.3 Matlab-controlled ASIC verification flow

ASIC verification was performed with an FPGA-aided test environment using the Interconnect Break Out Board (IBOB) FPGA board [18]. IBOBs are fitted with Xilinx Virtex-II Pro 2VP50 FPGAs. The 2VP50 has 53136 logic cells, 232 18-kbit BlockRAMs, 232 18×18 multipliers, and 2 PowerPC processors. The IBOB interfaces with the ASIC board through the GPIO interface. Figure 4.7 shows a photograph of the chip test setup. The FPGA is programmed using the BEE platform studio based designed flow which uses the Matlab/Simulink-based environment [19]. BEE XPS is used to generate the bit file from the Simulink model, which is then programmed into the FPGA on IBOB using iMPACT tool.
from Xilinx ISE. For illustration, Fig. 4.8 shows the Simulink model used to program the FPGA for ASIC verification. The test vectors used for ASIC verification are the same as the ones used to test the algorithms during the design phase. The IBOB feeds the test patterns to the ASIC board through the GPIO interface. It also reads the outputs from the ASIC using the same interface and performs a cycle-accurate comparison of the outputs with the expected results.

The ASIC board is a 6-layer board and contains voltage regulators required for the different voltage domains present on the chip. It also includes a current sense circuit to allow us to measure the low current levels that the chip is expected to draw. Figure 4.9 shows a schematic of the current sense circuit used for power measurements. We need to measure currents from 0.4 V and 2.5 V supply voltages. Since the current drawn from both these supplies are small, the input common-mode level for the amplifier is very close to the power supply/ground if a single 3.3-V supply is used. The offset of the amplifier is around 5 mV while
Figure 4.9: Schematic for on-board current-sensing circuit.

The current being drawn from the chip is around 100 µA. To be able to reliably measure the voltage drop across a the sensitive 100 Ω current sense resistor, we need a gain of about 60 dB for our measurements. This high gain cannot be obtained if the output common-mode level is close to either power rail. We therefore, choose to operate the Op-amps at a ± 5 V supply which allows the OPAMPs to give a high gain at both 0.4 and 2.4 V common mode level inputs.
CHAPTER 5

Results and Conclusions

5.1 Chip measurement results

The chip was tested to be functional in all the three modes of operation which output raw data, aligned spikes and spike features. Figure 5.1 shows a sample output for simulated neural data with an SNR of $-2.2$ dB. Figure 5.1(a) shows a snapshot of raw data. Figure 5.1(b) shows the aligned spikes and Fig. 5.1(c) shows the uniformly extracted feature extraction coefficients. The section of test data. It is known a priori that the raw data consists of two neurons which as seen in Fig. 5.1(b) are difficult to distinguish in the time domain. However, these neurons can be separated in the feature-extracted domain as seen from Fig. 5.1(c). The clustering accuracy of the data was found to be 72% when time domain data was used for classification. However, when extracted features from Fig. 5.1(c) are used, the clustering accuracy improves to 92%. Figure 5.2 shows the sample output for human data. The data was recorded using one of the nine 40 $\mu$m diameter electrodes from the hippocampal formation of a human epilepsy patient at UCLA. Figure 5.1(a) shows a snapshot of raw data. Figure 5.1(b) shows the aligned spikes and Fig. 5.1(c) shows the spike features. The clustering output of this data using spike features was found to be consistent with manual observations from neuroscientists.

The chip was designed for 0.3 V operation, but could be verified on silicon.
Figure 5.1: Sample output for simulated neural data with SNR=$-2.2$dB.

Figure 5.2: Sample output for human neural data.
upto core voltages down to 0.55 V. Beyond this voltage incorrect outputs were observed even for the raw data mode. The power of the chip is measured to be 130 μW when operated in feature-extraction mode for 64 channels (2 μW/channel). When a single core is operated with the remaining cores in sleep mode, the power consumed by the chip is 52 μW (3 μW/channel). The power for single-core operation is not 4 times lower than the power for four-core operation because the sleep devices do not completely eliminate power consumption in the inactive cores. Leakage and clocking are the dominant contributors of power in the register-dominated spike-sorting DSP. Signal switching power only contributes around 5 to 10% of the total DSP power. Table 5.1 summarizes the performance metrics for the chip. A data rate reduction of 91.25% is obtained as raw data at the rate of 11.71 Mbps is converted into aligned features at the rate of 1.02 Mbps. The accuracy of the chip for simulated neural data illustrated earlier (with SNR

<table>
<thead>
<tr>
<th>Technology</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core V_DD</td>
<td>0.55 V</td>
</tr>
<tr>
<td>I/O Voltage</td>
<td>1.2 V / 2.5 V</td>
</tr>
<tr>
<td>Gate Count</td>
<td>650 k</td>
</tr>
<tr>
<td>Clock Domains</td>
<td>0.4 MHz, 1.6 MHz</td>
</tr>
<tr>
<td>Power</td>
<td>2 μW/channel</td>
</tr>
<tr>
<td>Data Reduction</td>
<td>91.25% (11.71 to 1.02 Mbps)</td>
</tr>
<tr>
<td>No. of channels</td>
<td>16, 32, 48 or 64</td>
</tr>
<tr>
<td>SNR</td>
<td>-2.2 dB, Median</td>
</tr>
<tr>
<td>P_D</td>
<td>86 %, 87 %</td>
</tr>
<tr>
<td>P_FA</td>
<td>1 %, 5 %</td>
</tr>
<tr>
<td>Class. Accuracy</td>
<td>92 %, 77 %</td>
</tr>
</tbody>
</table>
Table 5.2: Comparison with Previous Work

<table>
<thead>
<tr>
<th>Reference</th>
<th>[11]</th>
<th>[12]</th>
<th>[13]</th>
<th>[20]*</th>
<th>This work [21]</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Channels</td>
<td>96</td>
<td>32</td>
<td>1</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Power (µW/channel)</td>
<td>104</td>
<td>75</td>
<td>100</td>
<td>14.6</td>
<td>2.03</td>
</tr>
<tr>
<td>Area (mm²/channel)</td>
<td>-</td>
<td>0.11</td>
<td>1.58</td>
<td>0.01</td>
<td>0.06</td>
</tr>
<tr>
<td>Power Density (µW/mm²)</td>
<td>-</td>
<td>680</td>
<td>60</td>
<td>1460</td>
<td>30</td>
</tr>
<tr>
<td>Process (nm)</td>
<td>FPGA</td>
<td>500</td>
<td>350</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>Core Voltage (V)</td>
<td>-</td>
<td>3</td>
<td>3.3</td>
<td>1.08</td>
<td>0.55</td>
</tr>
<tr>
<td>Detection</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Alignment</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Feature Extraction</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

* Numbers are synthesis estimates, not silicon results.

0.22dB is given by a $P_D$ of 86%, $P_{FA}$ of 1% and classification accuracy of 92%. The corresponding median numbers for the entire SNR range of $-15$ dB to $20$ dB are 87%, 5% and 77% respectively. The power density of the chip is 30 µW/mm², which is 27× lower than the 800 µW/mm² limit.

Table 5.2 shows a comparison of the spike-sorting DSP with previous work for various parameters. Our design achieves 7× lower power than the previous DSPs [20]. The power density is 2× lower than the minimum reported for previous DSPs [13]. It should be noted that the DSPs mentioned in Table 5.2 differ in functionality implemented. For instance, none of the previous DSPs include alignment. The complexity of the algorithms implemented is also different. For instance, [11] uses absolute value threshold which is much simpler than NEO. However, we found that absolute value threshold [14] has poor performance for low SNR situations. Also, results for reference [20] are not from silicon imple-
mentation and seem to violate the power density constraint. Therefore, it might not be feasible to use the design in [20] as an implantable spike-sorting DSP chip.

5.2 Conclusions

Most existing spike-sorting DSP designs provide for only detection for multiple channels or provide for detection, alignment, and feature extraction but only process a single channel at a time. In this thesis, we demonstrated a chip that can perform detection, alignment, and feature extraction simultaneously for 64 channels. The 64-channel spike-sorting DSP chip implemented in a commercial 90-nm CMOS process has a power density of 30 $\mu$W/mm$^2$ which is $27 \times$ lower than the power density limit and $2 \times$ lower than previous work [13]. The chip has a die area of 7.07 mm$^2$ and a core area of 4 mm$^2$. The area required is 0.06 mm$^2$/channel, which implies that the DSP would occupy 37.5% of the base area of the Utah electrode for a 100-channel design. The DSP achieves a data-rate reduction of 91.25%. The output data rate of 1.02 Mbps for 64 channels is equivalent to that of raw data for 5 channels.

The low power density and area is achieved due to careful selection of algorithms along with energy efficient architecture and circuit design. The demonstration of a low power spike-sorting DSP motivates us to explore opportunities for increasing the digital processing in order to reduce the total system power. Our future work includes incorporation of on-the-fly clustering and inclusion of digitally-assisted analog front-end and RF circuits. In addition, the standard cell registers and register-based memories used on this chip could be replaced with custom, energy-efficient registers and memories in future implementations to further lower the area and power of the DSP.
APPENDIX A

Efficiency Comparison of Analog versus Digital Spike-Detection

In [22] we compared the efficiency of analog and digital spike detection. This appendix gives a summary of the analysis presented in [22]. Some neural recording systems perform spike detection in the analog domain while others choose to perform spike detection in the digital domain. The motivation for this analysis is that analog spike detection could be more power-efficient since the ADC would only need to run when there are spikes, whereas in digital detection the ADC must constantly be running, since detection occurs only after sampling. However, performing computations in the digital domain has the advantage that digital-design techniques, such as voltage scaling and interleaving, can be employed that are not easily applied to the analog domain.

We chose to analyze the efficiency of NEO and absolute value spike detection for two output modes: Pulse Output mode and Spike Output mode. In the Pulse Output mode, a pulse is output whenever the input data exceeds the threshold. In the Spike Output mode the detected spike is captured with a preamble and output. The power ($P_{\text{ADC}}$) and area ($A_{\text{ADC}}$) of the ADC are modeled by Equations A.1 and A.2. To compare the power of ADCs of different speeds ($F_S$), resolutions ($B$), and technology, the Figure-of-Merit (FoM) shown in Eq. A.1 is used, where $A_0$ and $B_0$ are the area and resolution, respectively, of
the baseline ADC.

\[ P_{\text{ADC}} = \text{FoM} \cdot F_s \cdot 2^B \]  
\hspace{1cm} (A.1)

\[ A_{\text{ADC}} = A_0 \times 2^{2(B-B_0)} = 0.03 \times 2^{2(B-8)} \]  
\hspace{1cm} (A.2)

Power increases linearly with speed, and exponentially with resolution (and often faster than \(2^B\)). Area is primarily determined by the resolution. For an 8-bit, 24 kSa/s, 100 fF/conv-step ADC, the power consumption is expected to be 614 nW. Silicon area ranges from 0.021 to 0.24 mm\(^2\). We use 0.03 mm\(^2\) (\(A_0\)) and 8 bits (\(B_0\)) as our baseline ADC. To scale to other resolutions (\(B\)), we use Eq. A.2. While the power dissipation of the high-efficiency ADCs previously mentioned is dominated by dynamic power, we allocate 10\% of the reported power as standby power (for biasing and references). Including readout circuitry for buffered samples, the effective power is given by

\[ P_{\text{ADC,eff}} = (0.9 \cdot \text{FoM} \cdot 2^B + P_{\text{buf}}/F_s) \cdot r_D + 0.1 \cdot \text{FoM} \cdot 2^B \cdot F_s \]  
\hspace{1cm} (A.3)

where \(F_s\) is the sampling rate (24 kSa/s) and \(r_D\) is the detection rate in samples/s. We also assume that interleaving the ADC with up to 64 channels has little impact on the overall performance (which is true for modern-day CMOS implementations). Analog circuits to perform NEO and absolute-value spike detection are described in [22]. The major components of analog detection are the circuits for implementation of absolute value/NEO and the analog memory. The power consumption of the detection circuits is analyzed across a wide range of SNRs and firing rates.
In order to obtain power and area estimates for the digital implementations of the spike-detection algorithms, both the absolute-value threshold and the NEO detection methods (Spike Output and Pulse Output modes as explained earlier) were implemented in the Matlab/Simulink-based design environment. Each of the above algorithms was implemented with 2, 4, 8, 16 and 32-channel data-stream interleaving to determine a power-area efficient implementation. The RTL was auto-generated from the Simulink model using the Synplify DSP blockset. Power and area estimates were then obtained from the synthesis reports for these designs when synthesized with DC compiler from Synposys. Simulated neural data was input to RTL simulations to obtain switching activity estimates for the design. These estimates were then annotated into the synthesis flow to obtain power estimates for the digital spike-detection module. We find that 8-channel interleaving is the most energy-efficient implementation for the detection algorithms considered. Also, we observe that area and power for Spike Output mode are significantly higher than those for Pulse Output mode. This is due to the additional logic and memory required to provide the detected spikes (with preamble) as the output. It should be noted that we use a register-based memory for our design to guarantee functional operation at low supply voltages. However, use of a custom low-voltage memory would reduce the power difference between Pulse Output and Spike Output mode implementations. We find that variation in SNR and firing rate does not cause significant variations in the power consumed by the DSP. This result is expected, due to two major reasons: (a) SNR and firing rate do not affect the ADC power for digital detection (b) SNR and firing rate only affect the switching power of a portion of the DSP, which does not cause a significant change in the total power of the DSP. Hence, we expect the results of the above analysis to be valid for a wide range SNR and firing rates.
From the analysis we find that for each algorithm and output mode, a crossover point exists between analog and digital, indicating that for lower resolutions digital spike detection is more power/area-efficient, while for higher resolutions analog spike detection is more power-efficient. We find that the power for analog detection in the pulse mode is constant across bit resolution. In the case of the Spike Output mode, the power for analog detection does increase with bit resolution, since the ADC power increases. However, the increase in analog detection power due to the ADC is less than that for digital detection. This is because the ADC is only active for a limited time in case of analog detection. As for the area trade off, there is a crossover between analog and digital implementations in the case of Pulse Output mode. At higher bit resolution, the area for analog implementation is less since the area of the ADC dominates. However for the Spike Output mode, the area of digital detection is less than that of the analog detection. We conclude that digital spike detection is better for resolutions of up to 8 or 9 bits, depending on the algorithm used. Until this point, the ADC power is small. Hence the saving in ADC power achieved by analog detection does not outweigh the lower power cost of the DSP implementation. However, since the power and area of the ADC scale exponentially with bit resolution, the ADC starts to dominate at higher resolution. The analog detection, therefore, is more power efficient in this domain.
APPENDIX B

Spike-Sorting Methods

Spike sorting can be divided into three main steps: (1) **Spike detection and Alignment** is the process of separating spikes from raw data and aligning to a common reference. (2) **Feature extraction** is the process of transforming spikes into a feature space that better separates them from each other. Feature extraction also reduces the dimensionality of the data. (3) **Clustering** is the process where clusters are found in the extracted features, which allow us to map the neurons to their source action potentials. In this appendix, we present an introduction of spike detection and feature-extraction algorithms which were used for analysis in [14].

**Spike Detection:** Spike-detection algorithms involve two main steps, pre-emphasis of the spike and application of a threshold [23]. This section describes three methods of pre-emphasis: absolute value, nonlinear energy operator, and stationary wavelet transform product. The expressions for automatically determining the threshold for each method are also stated. Spike detection using all of the methods was accomplished as follows: When a sample in the pre-emphasized signal crosses the threshold, a 3-ms window is applied to the signal and the result is saved as a spike. This window-length was chosen because a spike-duration is unlikely to be longer than this and because it assures that we do not capture more than one spike from the same neuron in the window.

**Absolute Value** A simple, commonly used detection method is to apply a
threshold to the voltage of the waveform. This threshold can be applied to either
the raw (filtered) waveform or to the absolute value of the waveform. Applying
a threshold to the absolute value of the signal is more intuitive, since spikes can
either be positive- or negative-going. The absolute value threshold is confirmed
to be better than a simple threshold in [24]. As in [15], the threshold $Thr$ was
automatically set to

$$Thr = 4\sigma_N, \quad \sigma_N = \text{median}\left\{\frac{|x(n)|}{0.6745}\right\},$$

(B.1)

where $x(n)$ is a sample of the waveform at time $n$ and $\sigma_N$ is an estimate of the
standard deviation of the noise.

Nonlinear Energy Operator The nonlinear energy operator (NEO), also
called the Teager energy operator (TEO) has been proposed for use in spike
detection ([23,24]). In discrete time, the NEO $\psi$ is defined as

$$\psi[x(n)] = x^2(n) - x(n + 1) \cdot x(n - 1).$$

(B.2)

The NEO is large only when the signal is both high in power (i.e., $x^2(n)$ is large)
and high in frequency (i.e., $x(n)$ is large while $x(n + 1)$ and $x(n - 1)$ are small).
Since a spike by definition is characterized by localized high frequencies and an
increase in instantaneous energy [23], this method has an obvious advantage over
methods that look only at an increase in signal energy or amplitude without
regarding the frequency. Similar to the method in [23], the threshold $Thr$ was
automatically set to a scaled version of the mean of the NEO:

$$Thr = C \frac{1}{N} \sum_{n=1}^{N} \psi[x(n)],$$

(B.3)

where $N$ is the number of samples in the signal. The scale was empirically chosen
to be $C = 8$ and then used as a constant.
Stationary Wavelet Transform Product (SWTP) The discrete wavelet transform, originally presented in [25], is ideally suited for the detection of signals in noise (e.g., edge detection, speech detection). Here we use the method presented in [26]. First, the stationary wavelet transform (SWT) is calculated at 5 consecutive dyadic scales \( W(2^j, n), j = 1, \ldots, 5 \). Then the scale \( 2^{j_{\text{max}}} \) with the largest sum of absolute values is found:

\[
j_{\text{max}} = \arg \max_{j \in \{1, \ldots, 5\}} \left( \sum_{n=1}^{N} |W(2^j, n)| \right).
\]  

(B.4)

From here, we calculate the point-wise product \( P(n) \) between the SWT at this scale and the SWTs at the two previous scales:

\[
P(n) = \prod_{j=j_{\text{max}}-2}^{j_{\text{max}}} |W(2^j, n)|.
\]  

(B.5)

This product is then smoothed by convolving it with a Bartlett window \( w(n) \) in order to eliminate spurious peaks, and a threshold is applied. Again, the threshold \( Thr \) was automatically set to a scaled version of the mean of this result:

\[
Thr = C \frac{1}{N} \sum_{n=1}^{N} w(n) * P(n),
\]  

(B.6)

where \( N \) is the number of samples in the signal. The scale was empirically chosen to be \( C = 2 \) and then used as a constant.

**Feature Extraction:** Feature extraction (FE) emphasizes the difference between waveforms and reduces the dimensionality of the data. In this section, we describe the principal component analysis, the discrete wavelet transform, discrete derivatives, and the integral transform algorithms for feature extraction.

**Principal Component Analysis (PCA)** PCA finds an orthogonal basis (i.e., the “principal components” or PCs) for the data such that it captures the directions in which the data has the largest variation. We express each spike as a
series of PC coefficients $c_i$:

$$c_i = \sum_{n=1}^{N} PC_i(n) \cdot s(n), \quad (B.7)$$

where $s$ is a spike, $N$ is the number of samples in a spike/PC, and $PC_i$ is the $i$th PC. The PCs are found by performing eigen-value decomposition of the covariance matrix of the data. [27] provides a detailed description of the method. PCA results in the same number of coefficients as samples in the original spikes ($N$). However, as most of the energy is captured in the first few components, we kept only the 3 largest PC scores for our analysis.

**Discrete Wavelet Transform (DWT)** The DWT has been proposed for FE by [15]. The DWT promises to be a good method for FE since it is a multi-resolution technique that provides good time resolution at high frequencies and good frequency resolution at low frequencies. The DWT is also appealing because it can be implemented using a series of filter banks, keeping the complexity relatively low. We used the Haar wavelet because it provided one of the highest levels of accuracy out of all the wavelets tested while being the simplest to implement. The DWT yields about the same number of expansion coefficients as samples in the original spike. We then perform dimensionality reduction using the Lilliefors test for normality, taking the 10 coefficients whose distribution over all spikes differ most from the normal distribution.\(^1\)

**Discrete Derivatives (DD)** A method similar to the DWT but simpler was presented in [6]. This method called discrete derivatives, requires us to compute the discrete derivatives by computing the slope at each sample point, over a number of different time scales:

$$dd_\delta(n) = s(n) - s(n - \delta), \quad (B.8)$$

\(^1\)The number of coefficients has not yet been fully optimized.
where $s$ is a spike. We chose to use $\delta = 1, 3,$ and $7$. This yielded about $3 \times$ more “expansion coefficients” as samples in the original spike, so we again reduced the dimensionality to 10 using the Lilliefors test.$^1$

**Integral Transform (IT)** In the IT method [5], spikes are classified based on the areas under the positive and negative phases, $I_A$ and $I_B$, respectively:

$$I_A = \frac{1}{N_A} \sum_{n=n_A}^{n_A+N_A} s(n), \quad I_B = \frac{1}{N_B} \sum_{n=n_B}^{n_B+N_B} s(n),$$

where $s$ is the spike, $N_A$ is the number of samples in the positive phase and $N_B$ the number of samples in the negative phase of the spike. $N_A$ and $N_B$ are both determined by offline training. This method is appealing because of the simple hardware implementation presented. Since only two features are extracted from each spike ($I_A$ and $I_B$), the resulting dimensionality of this method is 2.
References


stem with implantable data compression circuitry,” *IEEE Journal of Solid-

“A 128-Channel 6mW Wireless Neural Recording IC with On-the-Fly Spike
Sorting and UWB Transmitter,” in *International Solid-State Circuits Con-

[14] S. Gibson, J. Judy, and D. Markovic, “Comparison of Spike-Sorting Algo-

and sorting with wavelets and superparamagnetic clustering,” *Neural Com-

[16] D. M. Markovic, “A power/area optimal approach to vlsi signal processing,”
Ph.D. dissertation, EECS Department, University of California, Berkeley,
TechRpts/2006/EECS-2006-65.html

Department, University of California, Berkeley, May 2004.

berkeley.edu/wiki/index.php/IBOB

urable computing system,” *IEEE Design & Test of Computers*, vol. 22, no. 2,


[21] V. Karkare, S. Gibson, and D. Markovic, “A 130 µW, 64-Channel, Spike-
Sorting DSP Chip,” in *Asian Solid-State Circuits Conference*, November
2009.

[22] S. Gibson, R. Chandler, V. Karkare, J. Judy, and D. Markovic, “An Ef-


