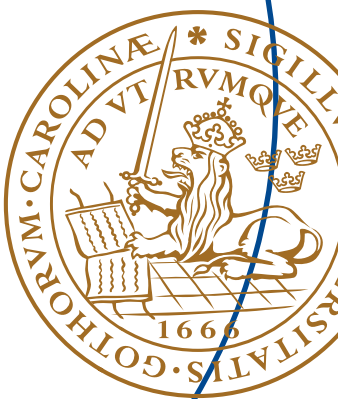
Master's Thesis

A Low Energy Data Compression Sub-V_T ASIC for Brain Implants

Longyang Lin
Shuai Cheng



Department of Electrical and Information Technology,
Faculty of Engineering, LTH, Lund University, January 2014

LUND UNIVERSITY

DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY

MASTER OF SCIENCE THESIS

A Low Energy Data Compression Sub-V_T ASIC for Brain Implants

Author:
Longyang Lin
Shuai Cheng

Supervisor:
Palmi Thor Thorbergsson
Oskar Andersson
Joachim Rodrigues

Lund January 6, 2014

The Department of Electrical and Information Technology
Lund University
Box 118, S-221 00 LUND
SWEDEN

This thesis is set in Computer Modern 10pt,
with the LaTeX Documentation System

Abstract

In this report, a digital implementation of a data compressor for wireless brain machine interface with an attractive area and low energy cost is presented. This design consists of pre-processing filters, spike detectors, a spike compressor and the Serial Peripheral Interface (SPI) IO protocol, targeting the 65 nm CMOS technology. Area and energy dissipation have been dramatically reduced by resources sharing, architectural optimizations and using a standard-cell based latch memory. Moreover, the compression ratio for each channel is adjustable based on the channel quality and the requirement of spike reconstruction accuracy. The loss of the reconstruction accuracy of the fixed-point digital implementation is less than 0.1% compares to the full precision Matlab model. Additionally aggressive voltage scaling (down to the sub-V_T region), clock gating and multiple clock domains have been performed resulting in a total die area of 900×500 μm^2 for 16 channels. Energy dissipation in the sub-V_T region is estimated using a high-level sub-V_T energy model. The estimated value is 1.03 pJ/clock cycle, which is $30\times$ improvement compared to the standard super-V_T implementation without clock gating.

Acknowledgements

We would like to thank our supervisor, Palmi Thor Thorbergsson who has given great help in the understanding of algorithms and BMIs. We would like to thank our supervisor, Oskar Andersson who always patiently provided the detailed technical supports regarding the digital implementation. Our special thanks would go to Joachim Rodrigues for his inspiring supervision and giving the invaluable tapeout chance. Finally, our appreciation would extend to our families who always stand behind us, and our SoC classmates for all the bright ideas and generous help.

Longyang Lin, Shuai Cheng
Lund, January 2014

Contents

List of Tables

List of Figures

List of Acronyms

ABS ABSolute value

ASIC Application Specific Integrated Circuit

BMI Brain Machine Interface

CMOS Complementary Metal–Oxide–Semiconductor

CNS Central Nervous System

DRC Design Rule Check

EMV Energy Minimum Voltage

FIR digital filter with a Finite Impulse Response

FIFO First-In-First-Out

FSM Finite State Machine

HDL Hardware Description Language

HVT High Voltage Threshold

IIR digital filter with a Infinite Impulse Response

LFP Local Field Potential

LLSVT Leakage Standard-Threshold

MISO Master In, Slave Out data

MOSI Master Out, Slave In data

NEO Nonlinear Energy Operator

P&R Place-and-Route

RTL Register Transfer Level

SCLK Serial Data Clock

SCLs Standard-Cell Libraries

SCM Standard-Cell based Memories

SDF Standard Delay Format

SPI Serial Peripheral Interface

SRAM Static Random Access Memory

SNR Signal-to-Noise Ratios

SVD Singular Value Decomposition

SVT Standard Voltage Threshold

SWTP Statinary Wavelet Transform Product Detection

Tcl Tool command language

VHDL Very High Speed Integrated Circuit Hardware Description Language

CHAPTER 1

Introduction

1.1 Neurophysiological Background

The central nervous system (CNS) is the information processing centre for the nervous system. Through probing the CNS for its neuronal activities while simultaneously observing behaviour, the underlying connection between brain signal and behaviour can be illustrated with specific simulation and modelling. Such knowledge enlightens clinical potentials in neurological dysfunction and allows the treatment of the symptoms of neurological disease. Moreover, the physical functions that have been lost due to neural injury can be retrieved by building up a feedback route to the nervous system with means of electrical simulation.

Neurons are the basic electrically excitable cells of the CNS and they are able to communicate with neurons and other cell types through specialized junctions called synapses, at which electrical or electrochemical signals can be transmitted from one cell to another. A typical neuron has four distinct parts or regions (see Fig. 1.1). The first part is the cell body (or soma). This is the neuron's center for metabolic functions and protein synthesis. The second and third parts are process structures that extend away from the cell body. Generally speaking, the function of a process is to be a conduit

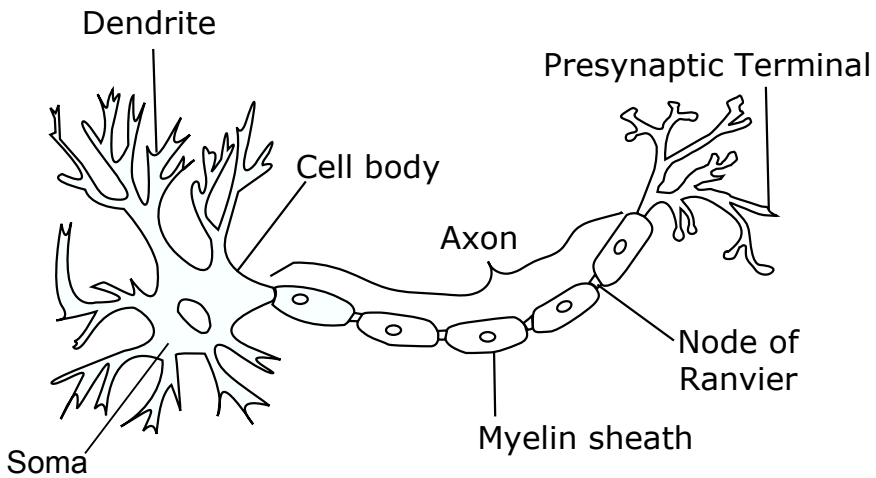


Fig. 1.1: A basic architecture of the building box of a typical neuron. (partially adopted from [1] and [2])

through, which signals flow to or away from the cell body. Input signals from other neurons are typically received through its dendrites. The output signal to other neurons flows along its axon. A neuron may have many thousands of dendrites, but it will have only one axon. The fourth distinct part of a neuron lies at the end of the axon, the presynaptic terminals. These are the structures that contain neurotransmitters. Neurotransmitters are the chemical medium through which signals flow from one neuron to the next at chemical synapses. In summary, the function of a neuron is to receive an input signal from other neurons, to process that signal, and then to send an output to other neurons [2].

The study of neural activity can be performed by extracellular neural recording [2]. It provides a method of measuring the change in electric potential associated with the activation of a single neuron using a microelectrode system. When a neuron generates an action potential, the signal propagates along the neuron as a current that flows in and out of the cell through excitable membrane regions in the soma and axon. The action potential is a short-lasting electrical signal, which rapidly rises and falls and is similar to the electrical signals in electronic devices. A microelectrode is implanted into the brain, where it can record the rate of change in voltage with respect to time. The extracellular signal consists of several components. The first one is the *spiking* component that reflects the action potentials elicited in

neurons that are close to the recording electrode. The other components of the extracellular signal are physiological noise, representing spiking activity from distant neurons, and low-frequency local field potentials (LFPs) that represent synaptic input to neurons close to the recording electrode. Most of LFPs reside in the lower part of the frequency spectrum compared to the spiking component. Hence, the most straightforward way to remove LFPs from the recorded signal is high-pass filtering [2].

1.2 Motivation

Recently, Brain Machine Interfaces (BMIs) have become an important research tools in neuroscience by providing a bidirectional communication between the CNS and the outside world. A typical BMI is shown in Fig. 1.2. It acquires signals from the CNS, processes them and forwards the output to a research application or to an actuator. The function of the actuator is generating feedback that is modulated by the acquired neural activity. The feedback has various forms like neural stimulation, prosthesis control, muscular stimulation or the operation of a personal computer [2]. The connection between the parts of a BMI can be realized by wire or wireless [3] [4]. The traditional form of connection between the measurement system is implemented by wire. When performing acute experiments on anesthetized subjects, the mobility of the subject and risks for surgical or post-surgical complications are usually insignificant. Hence, the wired connection is not a problem in these case. However, in experiments on awake and freely moving subjects or in clinical applications where the acquisition device is chronically implanted, the wires and the bulk of the equipment, as well as the potential risks for post-surgical complications will become significant. Hence, we are interested in exploring wireless BMIs to avoid complication inferred while using wires [2].

Despite the advantages of wireless BMIs, the paramount problems in wireless BMIs involve a restricted wireless link capacity and energy provision. From the point of view of clinical application, the measurement device to be implanted has to sustain through a long life span , where reliability and patient safety are of major significance. This eliminates the need for replacing the energy source, which involves invasive surgery. Moreover, the increasing number of recording channels results in the rising demand of energy provision and wireless link capacity [5]. In order to overcome the

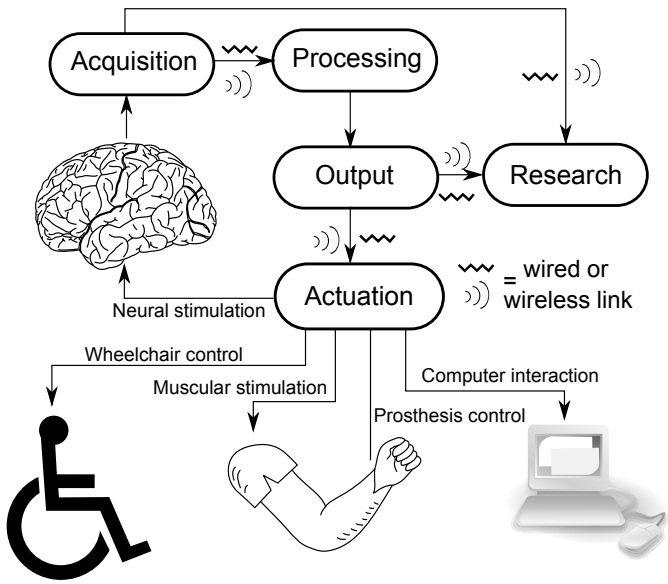


Fig. 1.2: A typical brain-machine interface (BMI).

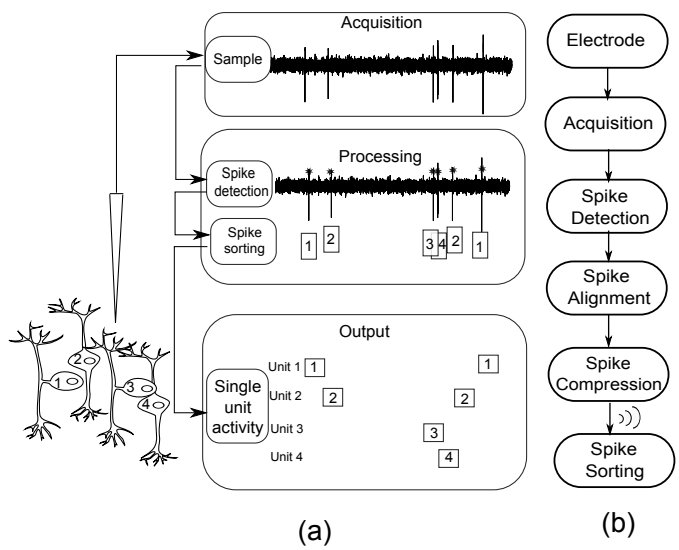


Fig. 1.3: Basic process of neural signal in BMI: (a) An illustration of basic process of wireless BMI. (b) A basic process blocks of wireless BMI.

bottleneck of channel capacity and maximize battery life, data reduction is the most effective and practically feasible method as compared to increasing the channel capacity since it will lead to a reduction of power and area consumption as well [2]. Thus, a spike compression step is introduced in Fig. 1.3 (b) to perform data reduction.

The data reduction is performed in two steps. The first step minimizes the acquired data from target neurons by minimizing sampling rate and resolution of data. In [6], the authors concluded that a sampling rate of 16-31 kHz and resolution of 9 bits are sufficient for the best performance in spike detection and spike sorting with realistic recording Signal-to-Noise Ratio (SNR). The second step removes any redundancies for the data to be transmitted through a compression algorithm. In previous research [2], five different compression algorithms are compared in terms of computational complexity and performance. The conclusion shows the most feasible approach is using a fixed generic compression basis. These fixed bases are derived from spike data by singular value decomposition (SVD), which ensuring the majority of the data is described by a minimal number of compression coefficients. The compression coefficients are ordered by significance by using SVD and the coefficient selection simply involves selecting the first coefficients. It should be noted that these bases are not derived from the detected spike waveforms each time, but from a large pre-recorded assembly of spike waveforms that cover a wide range of shapes. Thus, the significant information of the detected spike waveforms lies in the lower end of the coefficient spectrum and the coefficient selection becomes straight-forward.

In wireless BMIs, a measurement system that contains the electrode, the acquisition part and some of the processing part is implanted into the subject along with a transceiver. The implanted part of the BMI sends the acquired data to an external device that is used in processing and analyzing the incoming measurement data, as well as controlling the measurement device. In Fig. 1.3 (a), a illustration of a basic process of wireless BMI is given. An implanting microelectrode into the CNS is a common way of acquiring signals in BMI applications. The extracellular recordings can be used in studying the single unit activity of individual neurons. The main processing steps include spike detection and spike sorting. In Fig. 1.3 (b), some basic process blocks of neural signal in wireless BMI are illustrated. The acquisition unit is used in performing pre-processing of acquired signal, which includes the amplifier, an A/D converter, and filters. The next step is to perform spike detection and record their timing when the spike gets de-

tected. The final step is to use classification algorithms to sort the extracted spike waveforms and assigning them to correct neurons. A spike alignment step is often included to enhance the spike sorting accuracy. In wireless BMI, one more step called spike compression is needed before sending the signal to external devices.

Various different system architectures were compared in previous work [2], involving different combinations of spike detection, spike alignment, spike compression, spike reconstruction and spike sorting. The authors also concluded that the most feasible architecture of a wireless BMI consists of absolute value threshold detection, maximum value spike alignment and compression with a fixed generic basis that is derived from a large assembly of empirically found spike waveforms. The basic structure of this system is shown in Fig. 1.3 (b). The subject of this thesis project is to implement this architecture in hardware with the requirement of hardware efficiency and energy efficiency. Several features such as resource sharing, architecture optimization and the use of the standard-cell based latch memories, have been applied resulting in a dramatic area and energy reduction. Furthermore, low power techniques such as aggressive voltage scaling (downto sub-V_T region), clock gating and multiple clock domain have been performed aiming at an ultra low energy dissipation. The rest of this report is organized as follows. Chapter 2 introduces the algorithm that has been selected for implementation in this project. Chapter 3 presents the detailed hardware implementation of each unit and the ideas of hardware sharing. The discussion of speed requirement in the proposed design is also introduced in Chapter 3. Chapter 4 illustrates the low energy approaches consisting of multi-clock domain, aggressive supply voltage reduction and the use of standard-cell based latch memories. Chapter 5 summarizes the results and analyzes the performance. Chapter 6 gives the conclusion and the ideas of future investigation.

CHAPTER 2

Algorithm

The algorithm implemented in this project is a fixed generic bases compression, where the basis was obtained by performing singular value decomposition on a set of empirically found mean spike waveforms and is independent on the test recordings [2]. The desired building block for this algorithm is shown in Fig. 2.1. In the following sections, the functionality of each block is detailed.

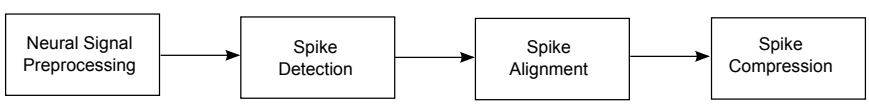


Fig. 2.1: An illustration of the building blocks of the fixed generic bases algorithm.

2.1 Neural Signal Preprocessing

Typically, the peak to peak amplitude of the recorded signal from the electrode is less than 1 mV, requiring amplification before digitization by the A/D converter. The amplifier can be included inside the A/D converter as

most of the commercial A/D converter support amplification. In this paper [6], the authors systematically analyzed the influence of the sampling rate and resolution on the performance of spike detection and spike sorting in a BMI system. Moveover, the authors concluded that a sampling rate of 16-31 kHz with 9 bits resolution was sufficient for maximizing performance at a realistic recording SNR. However, the resolution needs to be increased if dynamic range of the A/D converter does not exactly match with the amplitude range of the spike signal. Besides, both a highpass filter and a lowpass filter are needed since both the high-frequency spiking components and low-frequency LFPs are essential for analysis of neurophysiological data. Figure 2.2 illustrates a neural recording signal as well as the highpass filtered signal and lowpass filtered signal, which are time zooming in for occurrence of one spike. Typically the cut-off frequency is 300 Hz for both the highpass filter and the lowpass filter. Note that the comparison between different studies should be considered carefully if the filtering is performed in a different way since it influences the shape of spikes [7]. In this project, all the filters were implemented by the digital filters, which can provide much better frequency response.

2.2 Spike Detection

Spike detection aims at separating the spike from the recorded signal. The detection is usually based on a threshold, where the values above a given threshold are considered as the location of spike. Meanwhile, the time-stamps that indicate the occurrence of spike in the neurons also need to be recorded. In this paper [6], the performance of several spike detection algorithms were compared in terms of sampling rate, noise level and number of target neurons. The spike detection algorithms that were implemented and compared in this paper [6] were *ABSolute value* (ABS) [8], *Nonlinear Energy Operator* (NEO) [9], *Stationary Wavelet Transform Product Detection* (SWTP) [10] and *Matched Filter Detection* (MF) [11]. The discussion was carried out mainly in terms of sampling rate and true positive detection rate. Among them, ABS is the simplest method of spike detection that provides a good combination of performance and computational complexity [6], [12], [11]. Moreover, ABS is also attractive in hardware implementation due to its simplicity. Different approaches to generate the threshold for ABS detection are described and compared in the section 2.3.

2.3 Noise Estimation

In the ABS detection, a threshold is defined as a multiple of an estimate of the background noise level. The formula is shown as

$$T = s \cdot \sigma_N \tag{2.1}$$

where σ_N is the estimated background noise and s is the scale factor. The scale factor is used in generating a reasonable threshold value by scaling the noise level. This scale factor generally varies from case to case and in this design, the value is equal to 6. Three methods to estimate the noise level were introduced in following sections: exponential averaging filter, mean filter and median filter.

2.3.1 Exponential Smoothing Filter

The exponential smoothing filter is a filter that can be applied to a set of time series data to produce smoothed data. The simplest form of exponential smoothing filter is given by the formula 2.2

$$\begin{aligned} S_1 &= X_0 \\ S_n &= \alpha X_{n-1} + (1-\alpha) S_{n-1}, n > 1 \end{aligned} \tag{2.2}$$

where the smoothed statistic S_n is a simple weighed average of the previous observation X_{n-1} and the previous smoothed statistic S_{n-1}. In the proposed design, S_n refers to the background noise level. The factor α here refers to the smoothing factor ($0 < \alpha < 1$), for which larger values of α closer to 1 actually reduce the effect of smoothing, and give greater weight to recent changes in the data. In the case with $\alpha = 1$ the output of data series is identical to the original data series delayed one time unit. While, the values of α closer to zero have a greater smoothing effect and are less sensitive to recent changes. Simple exponential smoothing is easily applied, and unlike some other smoothing methods, this technique does not require any minimum number of observations to be made before it begins to produce results. However, in practice, a "reasonable average" value will not be achieved until several samples have been averaged together. In other words, an appropriate estimation of background noise level will not be achieved until a given amount of samples calculated together, for example, a constant signal will take approximately $3/\alpha$ stages to reach 95% of the actual value (See more detailed discussion in section 2.3.3).

Table 2.1: Frequently used sorting algorithms

Algorithm	Memory	Time	
		Average	Worst
Bubble	$O(1)$	$O(n^2)$	$O(n^2)$
Selection	$O(1)$	$O(n^2)$	$O(n^2)$
Insertion	$O(1)$	$O(n^2)$	$O(n^2)$
Shell	$O(1)$	$O(n(\log_2 n)^2)$	$O(n(\log_2 n)^2)$
Merge	$O(n)$	$O(n \log_2 n)$	$O(n \log_2 n)$
Quick	$O(\log_2 n)$	$O(n \log_2 n)$	$O(n^2)$
Median-finding Algorithm	$O(\log_2 n)$	$O(n)$	$O(n^2)$

2.3.2 Median and Mean Filters

In the median or mean filter, a window slides across a set of data, then the median or mean value of the samples inside the window is chosen to be the output in these type of filters. The median filter is considered as a more robust estimation than the mean filter, since it is less sensitive to extreme values. In this design, the noise measurement time for each channel was assumed as 82 ms. Thus, 2048 neural signals are needed for performing the noise measurement according to 25 kHz sample rate. That means the window size of the median filter is 2048. The estimation of background noise is defined as [8]

$$\sigma_N = median(\frac{abs(v)}{0.6745}) \tag{2.3}$$

where v is the sampled signal and σ_N is the background noise level.

The sorting algorithm is the critical computational block of the median filter. Some widely used sort algorithms are compared in this section in terms of time performance and memory usage. All of them are shown in table 2.1 and a brief description is given below. Among them, the median-finding sorting algorithm is the most feasible choice in terms of performance and memory usage.

- **Bubble Sort**: This algorithm is very straightforward and easy to implement. The algorithm starts on the first item on the left of the data array, comparing adjacent items and keeps moving the larger one to the right. After this, the same process will be applied to sort the

remaining $N-1$ items. This method normally performs $O(n^2)$ in both average and worst case. Thus, this algorithm is very inefficient, and rarely used in practice.

- **Selection Sort**: The process of this algorithm is performed by selecting the smallest element in specific order, then putting them in proper position in the final list. This process is repeated once for every element of the list. Thus, selection sort is an in-place sorting algorithm, which has $O(n^2)$ complexity for both worst and average case.

- **Insertion Sort**: The main logic of insertion sort is implemented by sorting one element in the array at a time, which means the memory requirement is $O(1)$. Assume the algorithm starts processing from the beginning of a set of data. When a value is smaller than the previously viewed value, move the new value into its correct position and shift all of the previous values forward. The average case and worst case are $O(n^2)$. However, in some cases, like a partially ordered list, this algorithm will correctly reorder the list faster than selection sort [13].

- **Shell Sort**: The basic idea of shell sorting is based on the insertion sort algorithm and is also known as the diminishing increment sort. It starts by comparing the elements far apart, then the elements less far apart, and finally comparing adjacent elements. The number of sorting operations in each is limited due to the pre-sorted of the sequence obtained in the preceding steps.

- **Merge Sort**: This sorting algorithm is a "divide and conquer" algorithm, which works in two steps. Firstly, dividing the unsorted list into n sublists, until each containing only one element. The elements in these sublists are considered as sorted. Then repeat merging sublists to produced new sublists until there is only one sublist remaining. This will be the final sorted list.

- **Quick Sort**: The basic idea of the quick sort algorithm is a "divide and conquer" procedure. The algorithm selects a partitioning element that is called "pivot" and partitions the data into two parts around this "pivot". Assume an array of n numbers $x_1, ..., x_n$. The quick sort is done by first rearranging it to two subarrays $x_1, ..., x_{j-1}$ and $x_{j+1}, ..., x_n$. Elements in the sub arrays are greater and less than x_j

respectively. Then the same procedure is recursively applied to the subarrays [13].

- **Median-finding Algorithm**: This algorithm is similar to the quick sort algorithm but the recursion is done only on one of the subarrays and not on both. Assume a i^{th} order statistic is the median number of the array. First of all, the algorithm will randomly select a pivot and partitions the data in two parts around this pivot. Thus, the elements are greater and less than the pivot respectively in these two parts. If the pivot is the i_{th} element of the data then the process is done, otherwise apply recursion to the sub array that contains the i_{th} element. The worst case for this algorithm is $O(n^2)$ and only occurs when each partition creates two subarrays of size 1 and $n-1$ respectively. This rarely occurs as the pivot is selected randomly. The average case is $O(n)$ [13].

2.3.3 Comparison between Exponential Filter and Median Filter

As mentioned in section 2.3.1 and section 2.3.2, the estimation of background noise level can be realized by a median filter or a exponential filter. These two algorithms will be compared in terms of memory usage, effect on accuracy of spike sorting, and complexity of hardware implementation. First of all, the memory usage of median filter is extremely larger than exponential filter, it requires 2kB (2048*8 bits) memory to store the data for calculation of median value. Whereas the memory usage of the exponential filter is only two register according to 2.2. Obviously, the memory usage of the median filter can be reduced from the point of view of the whole system if the memory is shared with the spike compression unit. It is feasible to do this sharing because the noise estimation is not required to execute all the time. In other words, the update of background noise level is periodic or can be controlled by the user through the I/O interface. Thus, the 2kB memory can be used in storing the data for spike compression unit when the noise estimation is not executing. It should be noted that the noise estimation unit is shared by several channels. The disadvantage of the memory sharing is that it might decrease the accuracy of the spike detection and affect the accuracy of spike sorting eventually. For example, when the noise estimation is running, all the memory is used in storing the data for calculation of median value until it finishes. In the meantime, new neural signals keep coming and get discarded

since there is no space for storing these new coming signals. Thus, the accuracy of spike detection might decrease if these abandoned neural signals contain spikes. It is obvious that the accuracy of spike sorting can be kept if using extra registers to store the new coming neural, but it is not wise since the memory usage will increase and it is hard to tell how large extra registers are needed since it depends on the sorting algorithm and spike firing rate. For a exponential filter, the memory usage just involves two registers which is much smaller than the median filter and it will generate the estimation value of noise level continuously. In other words, it will keep track of noise level and maintain the accuracy of spike sorting since the memory used in exponential filter is independent with spike compression. The complexity of hardware implementation of exponential filter is also lower than the median filter which involves implementation of sorting algorithm.

In summary, the exponential filter has lower complexity of hardware implementation, no effect on accuracy of spike sorting, and smaller memory usage. Thus, it is the most feasible alternative in the proposed design.

2.4 Spike Alignment

The spike alignment is necessary to achieve maximum accuracy in spike sorting. For a given neuron, the detected spikes are typically not sampled at the same time instances within the noise-free spike waveform. This is attributed to noise and asynchronicity between the sampling of the signal and the firing of action potentials [2]. Thus, the extracted spike waveforms are misaligned to each other since the spike detection threshold is not crossed at the same relative time between spikes. This phenomenon is referred to as *spike_detection_jitter* and it will introduce an apparent deviation in shape between the spike waveforms from the same neuron.

Spike alignment eliminates the influence of *spike_detection_jitter* by identifying the location of a reference point (a maximum value in the proposed architecture), then shifting the detected waveform to ensure the reference point occur at the same absolute point for all spike waveform from same neuron [2]. In the proposed design, the reference point is 16. That means the maximum absolute value will be shifted by 16.

2.5 Spike Compression and Reconstruction

The basic idea of spike compression is first transforming the detected spike waveform onto a full set of basis waveforms, then reducing the dimensionality, which means only transmitting the significant coefficients and discarding the rest. The amplitude of the majority of transform coefficients are small whereas the minority of coefficients has high amplitude. Thus, the detected spike waveform can be roughly described by the small subset of compression basis waveforms. The selection of a compression basis is crucial for compression since it is beneficial both in terms of data reduction and computational complexity. In this paper [14], the conclusion shows that the spike library of paper [15] can be described by the first six principal components, indicating that spike waveforms could be compressed with a fixed generic compression basis derived from a large set of experimentally obtained spike waveforms and always using the first six or even more compression coefficients. The number of fixed compression coefficients for transmission will vary depending on signal quality. The transformation of spike waveform is defined as

$$W_c = B_c^T S \tag{2.4}$$

Where the $M \times N$ matrix S contains the M sample long spike waveforms in its column, the $M \times N$ matrix B_c contains the M sample long basis waveforms of the compression basis in its columns and the $M \times N$ matrix W_c contains the full set of transformation coefficients. The compression only involves selecting K of the total set of M transformation coefficients and discarding the rest, thus, the compression was referred as the $K \times N$ dimensionality reduction matrix B_d. The eight basis waveform from the fixed compression bases was shown in Fig. 2.3.

The transmitted coefficients are thus described by

$$W_d = B_d W_c = B_d B_c^T S \tag{2.5}$$

Since the compression and dimensionality reduction bases were known to the reconstruction process, the reconstructed spike waveform is obtained as

$$S = B_c B_d^T W_d \tag{2.6}$$

2.6 Test Data

The synthetic test signals used in this project were generated by a novel approach described in this paper [16]. The recorded signals were five minutes long and include 16 independent channels with various SNR. In the proposed design, the spike compression unit is shared by 16 channels in order to minimize the hardware cost. After the spike waveform is detected and aligned in a particular channel, the compression unit will be occupied by this channel. Thus, the critical case in the proposed design occurs when all 16 channels detect the spike at the same time, which means all channels need to do compression at the same time. In order to test this critical situation, the recorded signals from 16 channels were identical. The test data was generated by the same method as described in this paper [2]. We assumed a space of hollow cylinder center at z-axis, which has inner and outer boundaries of 120 μm and 250 μm respectively. The floor and ceiling of the hollow cylinder were $\pm$ 250 μm respectively. One particular electrode site was set at the position of (0,0,0) μm for acquiring signals from the neuron. The noise neurons were placed at random positions within the space of the hollow cylinder. Four target neurons with the neural model derived in this paper [16] were placed inside the hollow space of the cylinder. The coordinates of four target neurons were (10,20,-2) μm, (-2,18,20) μm, (-20,-5,-10) μm and (16,-13,15) μm (see Fig. 2.4 and Fig. 2.5). The choice for these coordinates was to acquire high SNR recording (with mean SNR of 25 dB). For the medium and low SNR recording, the target neuron should be moved from the electrode site, which resulted in decreasing their spike amplitudes and thus decreasing the SNR. The positions of medium and low SNR recordings were chosen empirically, which was obtained by simply multiplied the coordinates given above by the factors of 1.5 and 2 respectively. Assume the inter-spike intervals of all neurons were described by gamma distribution [17]. A random mean firing rate for each noise neuron was chosen from a uniform distribution between 1 and 50 spikes/second. For each target neuron, the random mean firing rate was between 1 and 10 spikes/second. The SNR for a given recording and a given electrode site was defined as

$$SNR = 20 \log_{10} (\frac{S_{pp}}{\sigma_N}) \tag{2.7}$$

where S_{pp} is the peak to peak amplitude of the mean spike waveform of the target neuron detected by the electrode site and σ_N is the standard

deviation of background noise (described in section 2.3.2).

2.7 Performance Measurement

The performance of this system was estimated in terms of spike sorting accuracy and spike reconstruction accuracy [2]. Spike reconstruction accuracy C_{mean} was calculated by using the cross-correlation function between the reconstructed spike and the mean spike for the target neuron. The maximum value of the result refers as spike reconstruction accuracy. The spike sorting accuracy was estimated in terms of overall sorting accuracy (P_{ID}), which is described as

$$P_{ID} = \frac{N_{ID}}{M} \tag{2.8}$$

where N_{ID} refers to the overall number of correctly classified spikes and M is the total number of spikes [16].

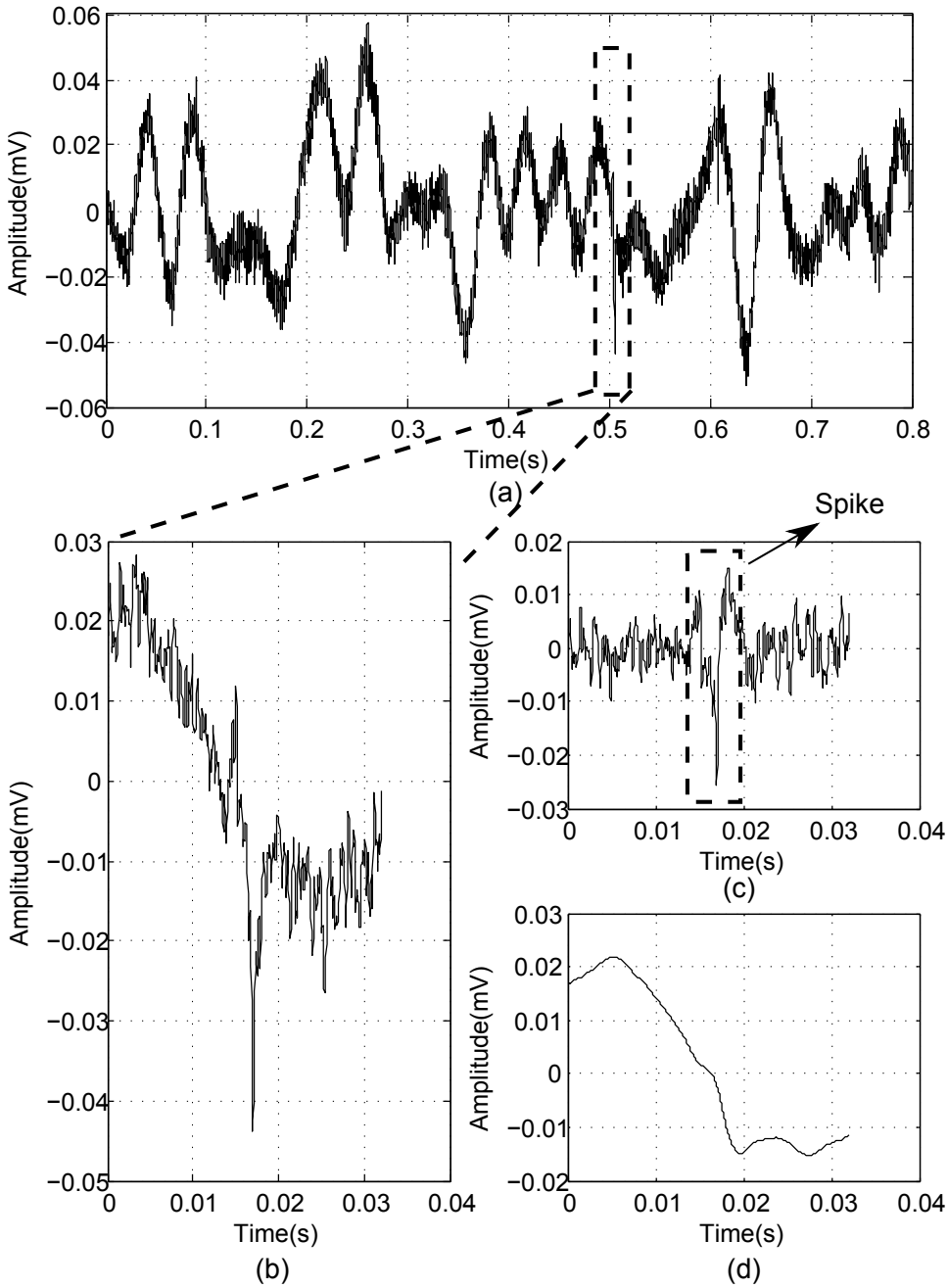


Fig. 2.2: An illustration of highpass and lowpass filtering for neural recording signal. (a) Neural recording signal. (b) Partially zoomed Neural recording signal. (c) Highpass filtered signal (the peak in the middle is the occurrence of one spike). (d) Lowpass filtered signal.

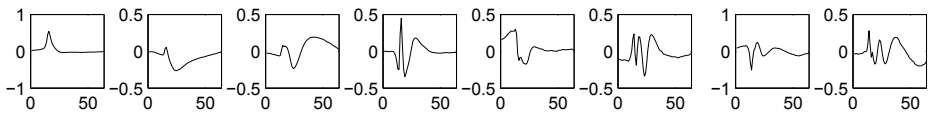


Fig. 2.3: First 8 basis waveform from the fixed compression bases.

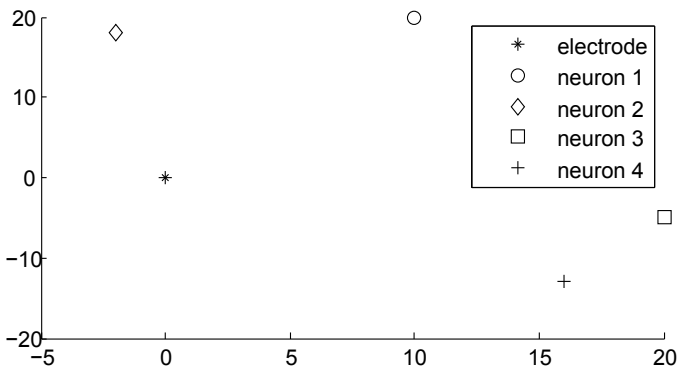


Fig. 2.4: Coordinates of electrode site and neurons in x-y view.

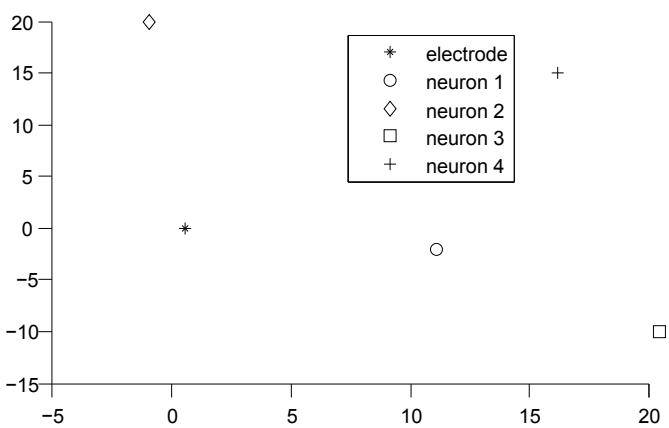


Fig. 2.5: Coordinates of electrode site and neurons in x-z view.

CHAPTER 3

Implementation

This section gives the specific implementation of each building block. The process flow of system with a single channel is shown in Fig. 3.1. The LFP signal, spike time information and compression coefficients are outputs as mentioned in chapter 2. In order to minimize the number of IO pins and increase the communication capability, a customized serial peripheral interface (SPI) was used as IO protocol. Besides, SRAM was needed for storing the spike waveform and compression coefficients. When considering

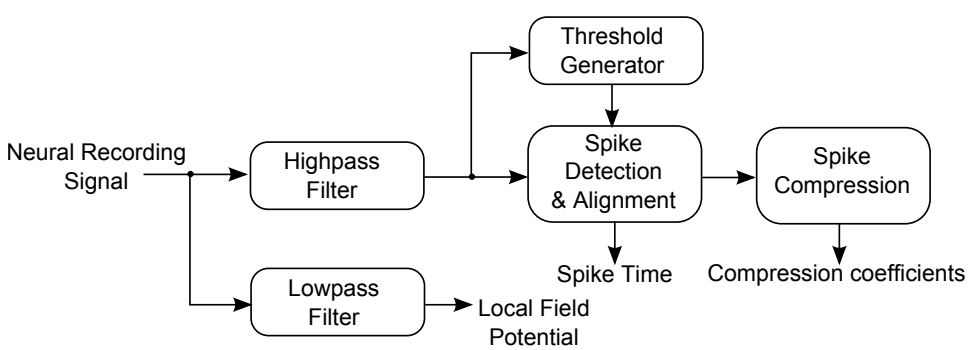


Fig. 3.1: Process flow of a single channel system.

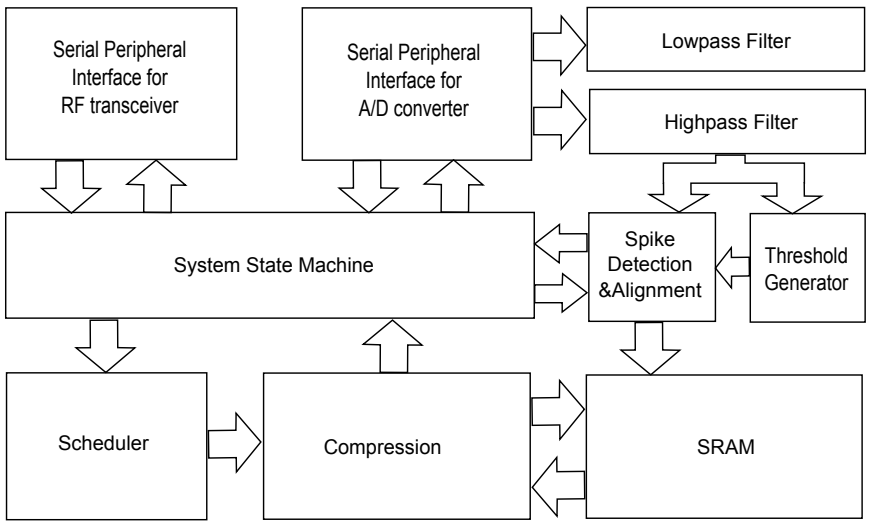


Fig. 3.2: An overview of the system architecture.

the multiple channel system and hardware resources sharing, a scheduler must be implemented to arrange the resources occupation. Moreover, a system state machine was designed to control the state of the overall system. Figure 3.2 illustrates a detailed system architecture.

3.1 System State Machine

The function of the system state machine is to decode and respond the commands sent by users through the SPI. The detailed description of each SPI command is presented in Appendix 1. The brief state diagram of SPI decoder is shown in Fig. 3.3. Furthermore, an ASM chart of this state machine with more detailed information is illustrated in Fig. 3.4. The system state machine consists of eight states:

- **IDLE**: The state machine listens to the signal "$SPI\_command\_valid$", which will be set high if received a new command. Once a new command has been received, the state machine will move to a different state based on the content of the command.

- **WRITE** α β: This state writes new values of α and β to the corresponding parameter registers of the particular channel. This operation

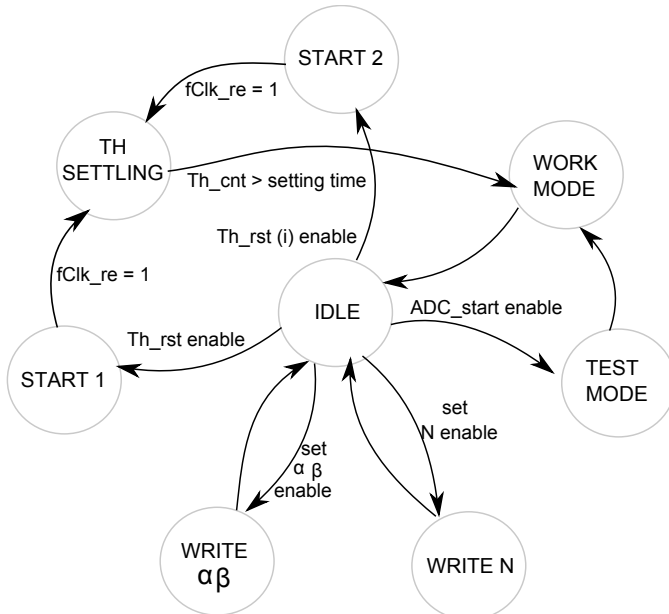


Fig. 3.3: State diagram of SPI command decoder.

will be finished within one system clock cycle. Hence, the state machine will move back to "IDLE" state in the next clock cycle.

- **WRITE N**: This state writes new value of N (Number of compression coefficients per detected spike) to the corresponding parameter register of the particular channel. This operation will be finished within one system clock cycle. Hence, the state machine will go back to "IDLE" state in the next system clock cycle.

- **START 1**: The system will move to this state when the signal "Th_reg_rst" is asserted. The value of threshold registers of all channels is set to zero. When signal "$fClk\_re$" is asserted (indicating the rising edge of the clock used by the filters), the state machine will move to "TH SETTLING" state.

- **START 2**: The system will jump to this state when The specific bit of bus signal "Th_reg_rst" is asserted and the threshold register of the particular channel is set to zero. When signal "$fClk\_re$" is asserted, the state machine will move to "TH SETTLING" state.

- **TH SETTLING**: This state enables a counter for counting the time of the exponential filters. Once the time counter exceed the settling time, the state machine will go to "WORK MODE" state.
- **WORK MODE**: The spike detectors are enabled during this state. Besides, the system will send the result to the external device once a spike has been detected and compressed. However, the state machine will go back to "IDLE" state if the signal "Sm_intr" is asserted (an interrupt command has been received).
- **TEST MODE**: The signal "$Test\_mode\_en$" will be asserted to enable the test mode. In the next system clock cycle, the state machine will jump to "WORK MODE" state for fast testing and debugging the functionality without the time counter of exponential filters.

3.2 Filters

In signal processing, a filter is a device or process that removes some unwanted components or features from a signal. The digital filters are either Infinite Impulse Response (IIR) filters or Finite Impulse Response (FIR) filters. In the proposed design, all the filters were implemented by IIR filter. The advantage of IIR filter over FIR filter is that IIR filter usually requires fewer coefficients to achieve similar filtering operation. Fewer coefficients means fewer multipliers, registers, and adders. Thus, IIR filters are the primary choice since it works faster, and requires less memory space. However, the IIR filter is prone to overflow and instability due to its recursion feature. Hence, the appropriate coefficients should be chosen to design a stable IIR filter. Overflow avoidance can be achieved by data truncation during the process of calculation as well. The detail of implementation of filters is described below.

3.2.1 Highpass and Lowpass Filters

In the proposed design, highpass filters are applied to the sampled neural signal in order to remove the low frequency LFP from the higher frequency spike component. The frequency of component with spike is typically larger than 300 Hz. The lowpass filters with 300 Hz cut-off frequency are used in isolating the LFP information since the frequency of LFP information is

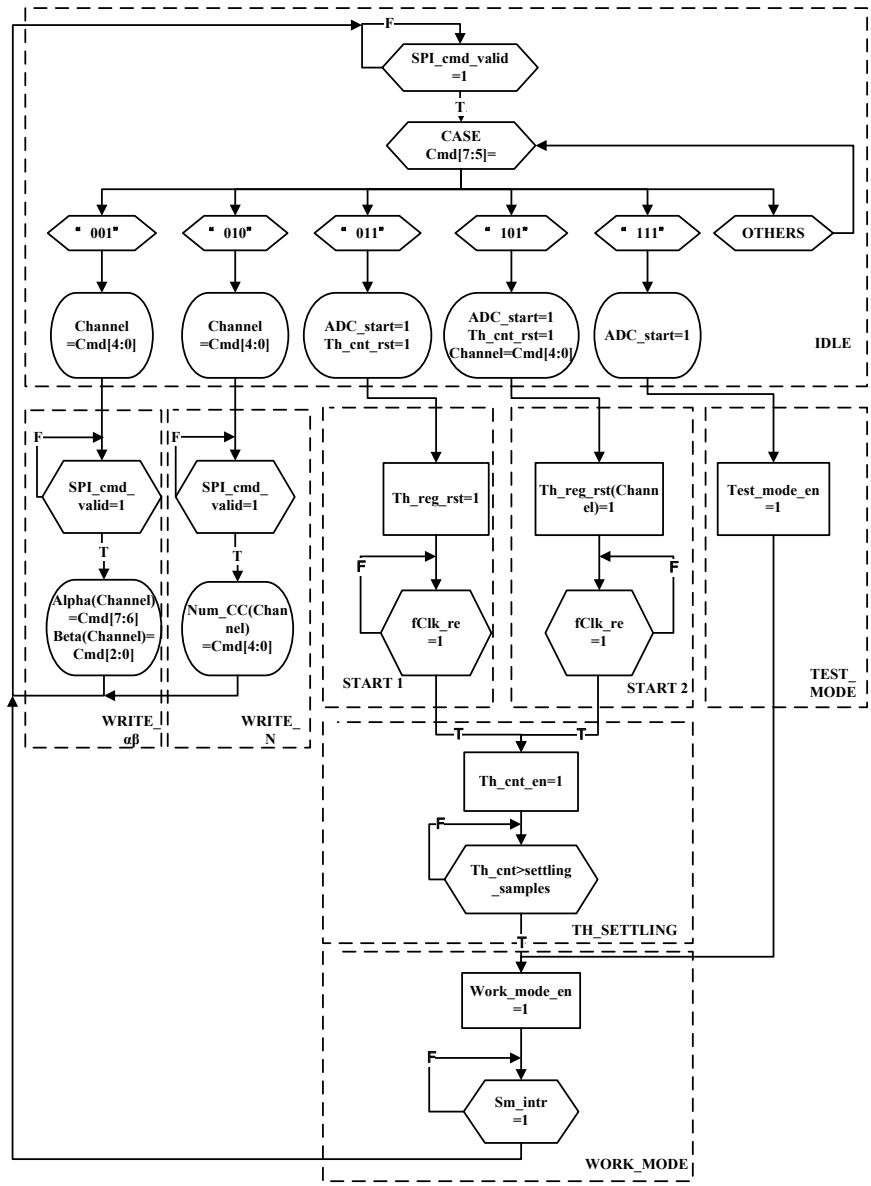


Fig. 3.4: ASM chart of system state machine.

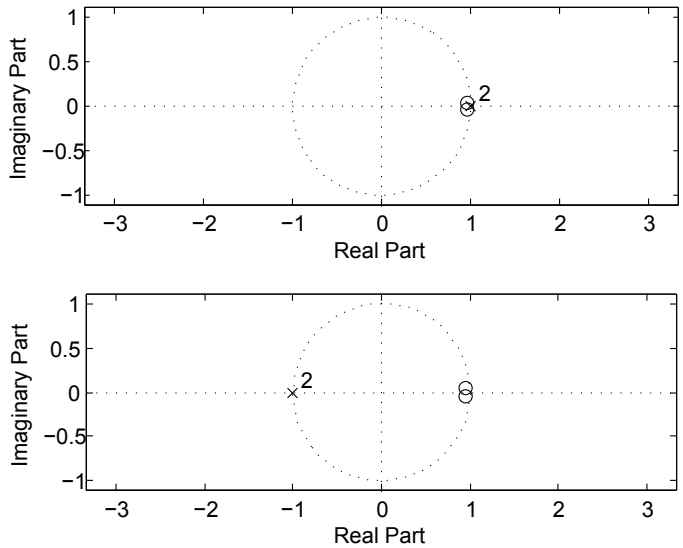


Fig. 3.5: Zero-Pole plot of highpass and lowpass filter.

located in the lower part of the frequency spectrum. All IIR filters in the proposed design are second order and simply implemented by Butterworth filter with direct form I structure. The general equation of both highpass and lowpass filter is described by

$$a_1 y(n) = b_1 x(n) + b_2 x(n-1) + b_3 x(n-2) - a_2 y(n-1) - a_3 y(n-2) \quad (3.1)$$

Where the numerator coefficients b and denominator coefficients a were shown in Table 3.1.

An IIR filter is stable if the root of the the denominator of transfer function has an absolute value that is less than one. In other words, when the poles of filter in the z complex plane have an absolute value that is less then one, i.e lie within the unit circle. The zero-pole plot in z complex plane of highpass and lowpass filters is shown in Fig. 3.5, which indicates that these filters are stable since the poles are inside the unit circle.

In hardware implementation, wordlength of data is especially important from the point of view of area and power consumption. A larger wordlength will lead to higher accuracy, larger area and power consumption. In order to choose a appropriate wordlength, some simulations were performed to analyse the relation between accuracy and wordlength. The hardware structure of both highpass and lowpass filters and the detail of wordlength changes

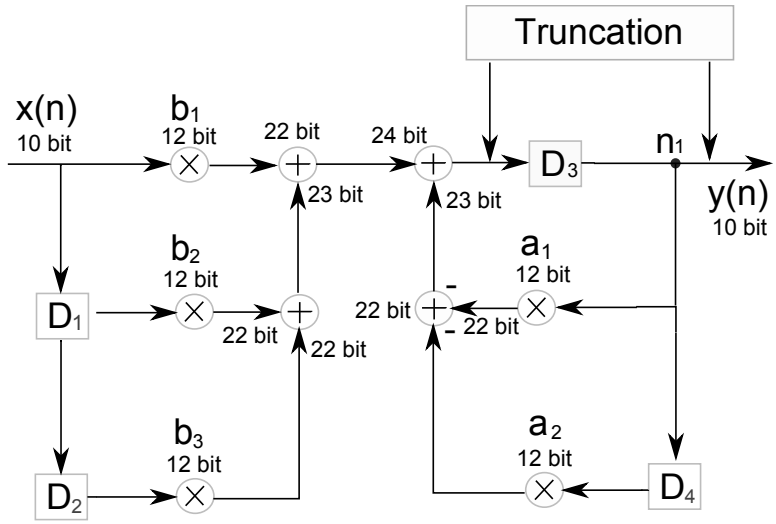


Fig. 3.6: Hardware structure of highpass and lowpass filter.

during the process is introduced in Fig. 3.6.

Table 3.1: Coefficients of highpass and lowpass filter.

	Numerator			Denominator		
	b_1	b_2	b_3	a_1	a_2	a_3
Highpass	0.9648	-1.9297	0.9648	1	-1.9287	0.9316
Lowpass	3.4618e-4	6.9189e-4	3.4618e-4	1	-1.9463	0.9482

Figure 3.6 illustrates that the wordlength will increase in every cycle of calculation. The hardware cost of register will also increase if the data is saved in full precision. Hence, the data is truncated before saving in the register D_3 and after the node n_1. The wordlength of input data x_n and output data y_n were assumed to be identical. A simulation was performed to find out the appropriate wordlength of data in register D_3. The mean value and standard deviation value of quantization errors were chosen as the measurements for wordlength simulation. Here the normalized quantization error is described as

$$\varepsilon = \frac{abs(\widetilde{S} - S)}{\max(abs(S))} \times 100\% \tag{3.2}$$

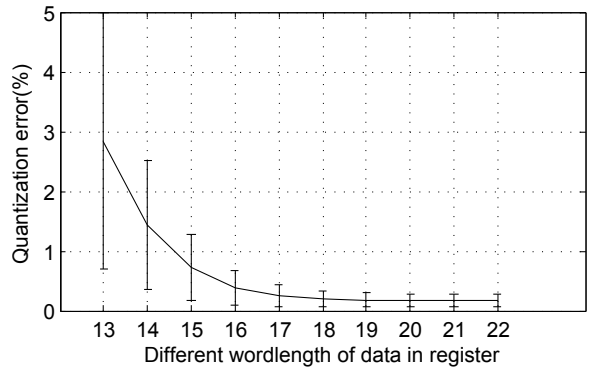


Fig. 3.7: Quantization error of different wordlength of highpass filter.

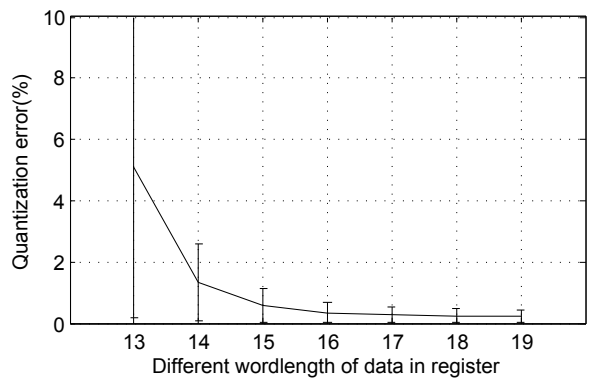


Fig. 3.8: Quantization error of different wordlength in lowpass filter.

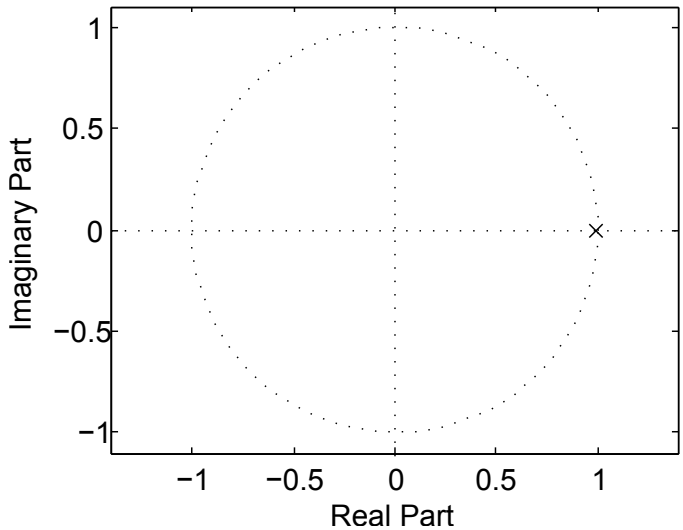


Fig. 3.9: Zero-pole plot of exponential filter.

where the $\widetilde{S}$ is quantized signal, which is processed by filter in hardware and S is the signal processed by filter in full precision. The simulation result showing the relation between different wordlength on register and quantization error is given by Fig. 3.7 and Fig. 3.8.

As shown in Fig. 3.7 and Fig. 3.8, a wordlength of 17 or larger gives a negligible quantization error but do not gain more in precision. Whereas the wordlength of 15 or smaller has a large quantization error. Thus, 16 bits wordlength was chosen for both case since it gives less than 1% quantization error with a relatively small area consumption.

3.2.2 Exponential Smoothing Filters

In this design, exponential smoothing filter is implemented by a very simple IIR filter to be applied to estimate the background noise level. The definition of exponential filter is described in section 2.3.1. The smoothing factors in 2.2 were chosen empirically to provide reasonable background noise level, which consist of four values in the proposed design: 0.1, 0.01, 0.001, 0.0001. The hardware structure is given by Fig. 3.10. The zero-pole plot in z complex plane is given by Fig. 3.9. The poles in Fig. 3.9 are inside the unit circle which means this filter is stable.

As Fig. 3.10 shows, the feedback loop of IIR will increase the hardware

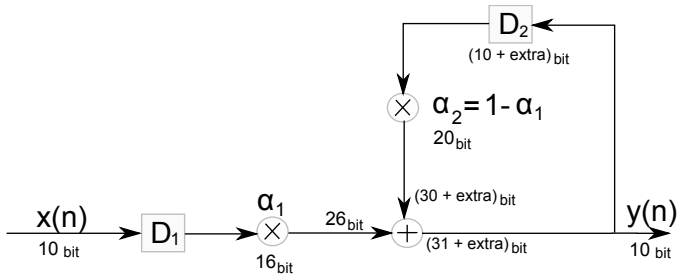


Fig. 3.10: Hardware structure of exponential filter.

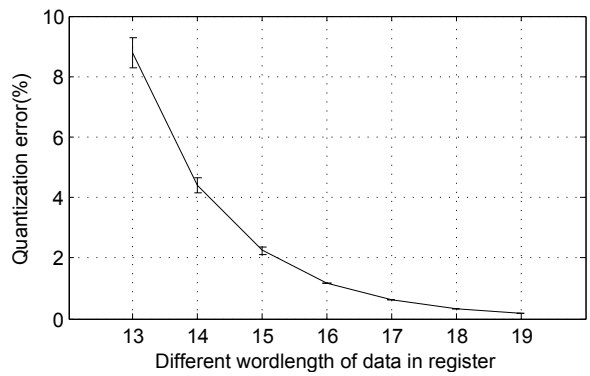


Fig. 3.11: Quantization error of different wordlength in exponential filter.

cost of register as the wordlength increases. Hence, the truncation need to be done before the result stores back on the register D_2, which leads to the quantization error. Thus, a simulation was performed to find out the appropriate wordlength. The result showing in Fig. 3.11 illustrates that the wordlength of 17 or even larger will not gain more in accuracy. However, using small wordlengthes like 15 bits or even smaller will lead to a larger quantization error. Hence, 16 bits wordlength was chosen which gives a quantization error around 1% and relatively small area.

3.3 Spike Detection and Alignment

As mentioned in chapter 2.2 and chapter 2.4, the next steps after neural signal pre-processing by filters are spike detection and spike alignment. In the proposed design, these process steps include:

- Calculating the address of SRAM for storing the neural signal samples;
- Making a decision by comparing the absolute value between the pre-processed signal and the threshold;
- Recording the time once a spike is detected (note: this is the relative time from the previous detection);
- Finding out the maximum absolute value of the next ten incoming samples;
- Deciding which part of SRAM will be assigned to spike compression unit.

Figure 3.12 shows the flow chart which can be easily implemented in a FSM (finite state machine). As shown in Fig. 3.12, the spike detection and alignment block will assert a "valid" signal to indicate that the data samples stored on the memory are ready for partial data compression (see section 3.4 for details of data compression and memory sharing). Once the "valid" signal is asserted, the channel ID will be recorded in the compression scheduler. It should be noticed that there is an individual spike detection and alignment unit for each channel. Furthermore, there are some methods of sharing resources between different channels are introduced in section 3.6. As shown in Fig. 3.12, 10 samples are needed to be stored on memory and compared to perform spike alignment. Considering the critical case that the first sample of these 10 samples is the maximum absolute value of this detected spike (the maximum absolute value should be the 16th sample of the entire spike according to spike alignment algorithm mentioned in Section 2.4), after the spike alignment, 26 valid samples of the detected spike are known and needed to be stored in order to process the spike compression. Hence, the memory cost from the critical case is 26 samples/channel.

3.4 Spike Compression

The transformation function for the spike compression described in 2.5 (Chapter 2) can be expressed as follows:

$$W_d = B_d W_c = B_d B_c^T S = \begin{bmatrix} Bc_{0,0} & \cdots & Bc_{0,63} \\ \vdots & \ddots & \vdots \\ Bc_{N-1,1} & \cdots & Bc_{N-1,63} \end{bmatrix} \cdot \begin{bmatrix} S_0 \\ \vdots \\ S_{63} \end{bmatrix} = \begin{bmatrix} W_{d0} \\ \vdots \\ W_{dN-1} \end{bmatrix}$$

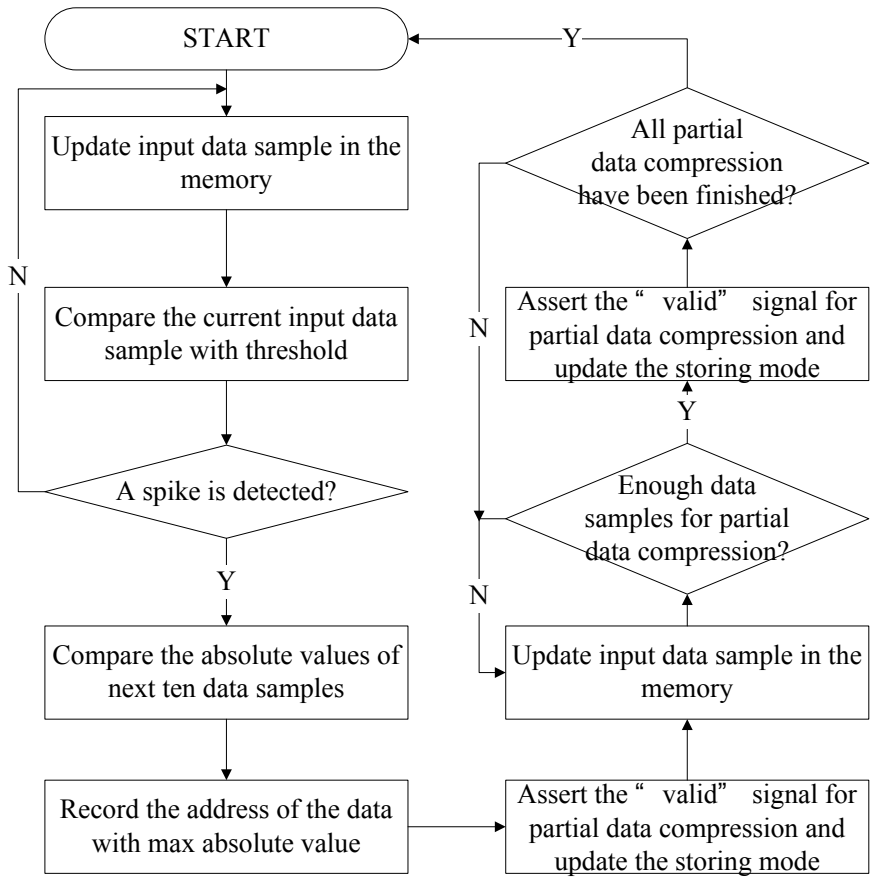


Fig. 3.12: Flow chart of spike detection and alignment.

where the dimensionality reduction matrix B_d is only aimed at selecting N of the total set of transformation coefficients. Here a spike includes 64 data samples (S_0 to S_{63}) corresponding to 2.56 ms recording time (sampling rate is 25 kHz). Since the parameter N (1 to 16) is configurable by SPI commands, the maximum number for storing B_c should be considered as 16 when implementing the ROM. In general, it requires a memory that can store all the 64 data samples (8 bits per sample) of a spike for calculating such a matrix multiplication. Figure 3.13 shows a general implementation of the matrix multiplier. It should be noted that the number of multipliers and adders can be increased to speed up this process but the area will also increase and the position of the data samples on the memory need to be modified. When extending the design to M channels, it requires M times memory usage and multipliers since the compression unit is simply duplicated for each channel. Assuming the least usage of multipliers per channel (i.e. 1 multiplier/channel) and 32 channels in total (it is a typical case M=32), the utilization of RAM, ROM and multipliers are 16 kbits (512 bits/channel), 256 kbits and 32 respectively. In order to reduce the total memory usage, another more efficient architecture based on the following equation is proposed:

$$W_d = \begin{bmatrix} Bc_{0,0} & \cdots & Bc_{0,63} \\ \vdots & \ddots & \vdots \\ Bc_{N-1,1} & \cdots & Bc_{N-1,63} \end{bmatrix} \cdot \begin{bmatrix} S_0 \\ \vdots \\ S_{63} \end{bmatrix} =$$

$$\begin{bmatrix} Bc_{0,0} & \cdots & Bc_{0,15} \\ \vdots & \ddots & \vdots \\ Bc_{N-1,0} & \cdots & Bc_{N-1,15} \end{bmatrix} \cdot \begin{bmatrix} S_0 \\ \vdots \\ S_{15} \end{bmatrix} + \begin{bmatrix} Bc_{0,16} & \cdots & Bc_{0,31} \\ \vdots & \ddots & \vdots \\ Bc_{N-1,16} & \cdots & Bc_{N-1,31} \end{bmatrix} \cdot \begin{bmatrix} S_{16} \\ \vdots \\ S_{31} \end{bmatrix} +$$

$$\begin{bmatrix} Bc_{0,32} & \cdots & Bc_{0,47} \\ \vdots & \ddots & \vdots \\ Bc_{N-1,32} & \cdots & Bc_{N-1,47} \end{bmatrix} \cdot \begin{bmatrix} S_{32} \\ \vdots \\ S_{47} \end{bmatrix} + \begin{bmatrix} Bc_{0,48} & \cdots & Bc_{0,63} \\ \vdots & \ddots & \vdots \\ Bc_{N-1,48} & \cdots & Bc_{N-1,63} \end{bmatrix} \cdot \begin{bmatrix} S_{48} \\ \vdots \\ S_{63} \end{bmatrix}$$

which splits the entire matrix multiplication into 4 smaller parts with each contains 16 data samples. This architecture consists of a FSM, 16 multipliers, 16 adders, 32 SRAMs and 1 ROM as shown in Fig. 3.14. The FSM is responsible for controlling address and data in the RAM and ROM so that the multipliers and adders can always load the correct data at the correct time. Besides, in order to decrease the critical path, some registers are

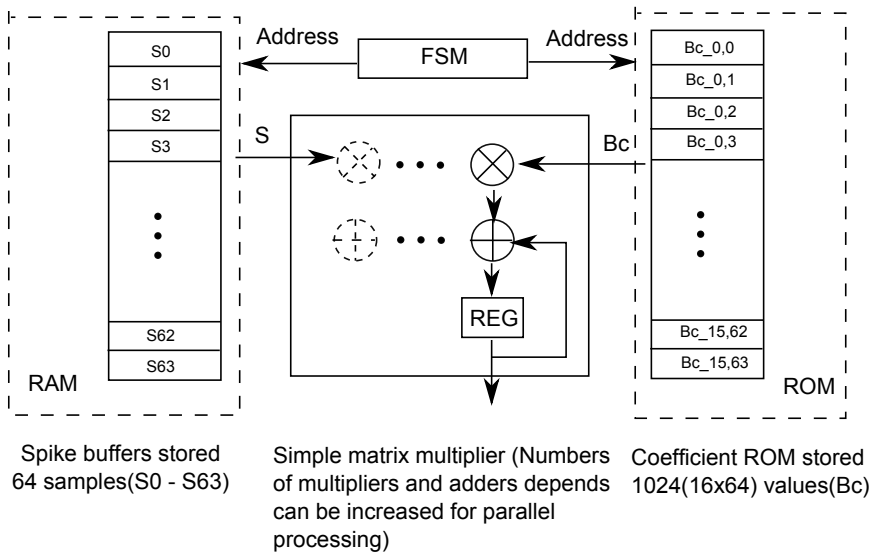


Fig. 3.13: An illustration of the general matrix multiplier.

used in copying the data in the RAM that is being used for the current computation. Moreover, it is important to design a convenient data map for the ROM which can simplify the address calculation. It is obvious that the more splits of the matrix multiplication result in the less memory usage for storing the data. However, based on the algorithm of spike detection and alignment (see Fig. 3.12), at least 16 data samples being stored in the memory once the spike detection and alignment is finished. Hence, 16 data samples of each partial matrix multiplication is the minimum number that can be chosen for splitting. A comparison of the resource usage between the general matrix multiplier and the proposed matrix multiplier for 32 channels is shown in table 3.2 where the calculation of the minimum required frequency is derived in section 3.7. Instead of having 32 RAMs with each storing the data samples for the same channel, the proposed matrix multiplier spreads the data of one channel into 32 RAMs so that it can read all the data of a specific channel in one clock cycle to speed up the computation (see section 3.4.1). Hence only one proposed matrix multiplier is needed, which is shared for all the 32 channels and it still provides higher throughput and less energy consumption (less min frequency) than the general one, which is parallel processing. Moreover, the total area cost of the memory will be further reduced when increasing the amount of channels.

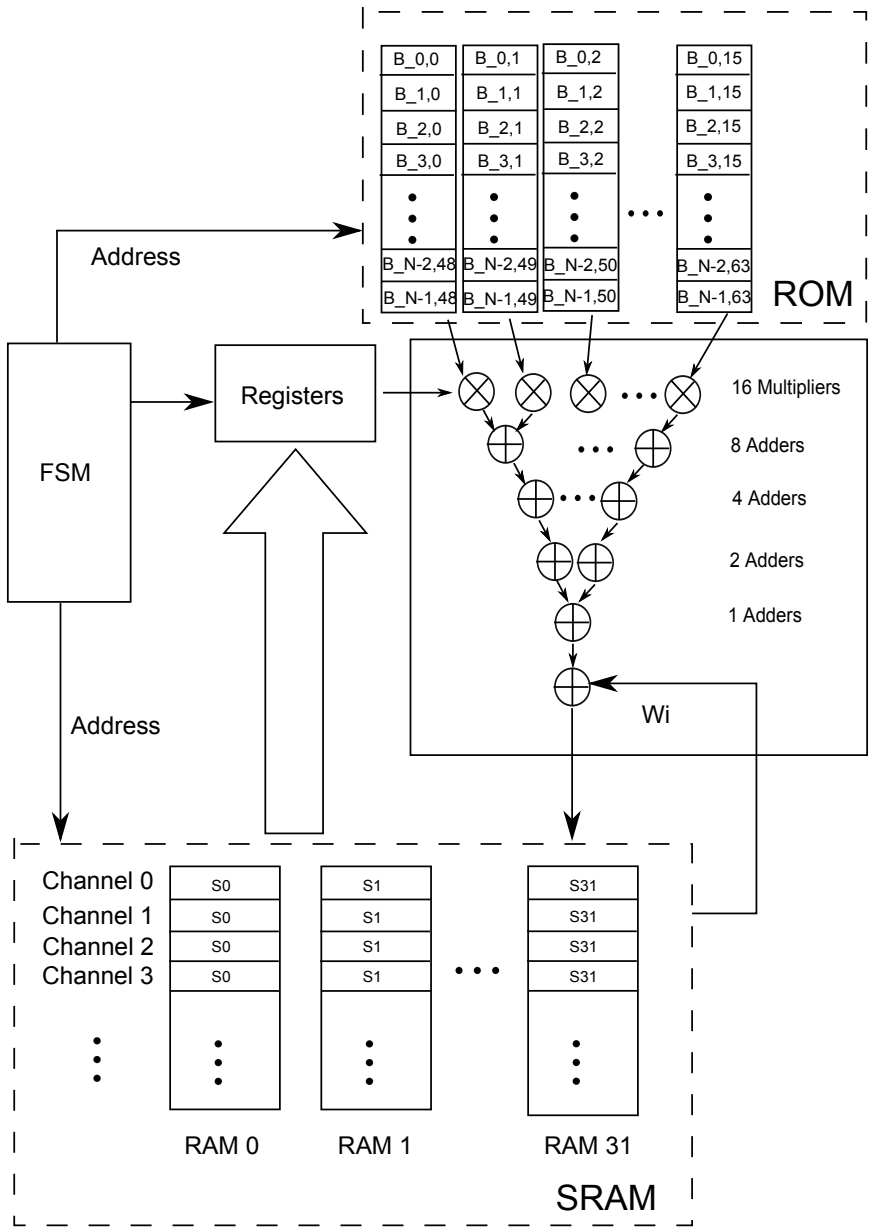


Fig. 3.14: An illustration of the proposed matrix multiplier.

Table 3.2: Comparison of the resource usage of matrix multiplier (32 channels).

	General	Customized	Reduction
RAM(bits/channel)	512	256	50%
ROM(bits/channel)	8k	256	97%
Multiplier/channel	1	0.5	50%
Adder/channel	1	0.5	50%
Min frequency(MHz)	25.6	7.6	72%

3.4.1 Memory Sharing

There are 32 RAMs which stores n data sample (8bits) from n channels respectively (n depends on the total number of channels). The A/D converter (supporting multi-channel) is assumed to sample the data sequentially (for example, $S0$ from channel 0, $S0$ from channel 1, ..., $S0$ from channel n-1, $S1$ from channel 0, ...). The memory works as a circular buffer for each channel. There are two modes of the address calculation for the new coming data samples: mode 1 is that the data will be circularly updated within all the 32 slots; mode 2 is that the data will be circularly updated within only 16 slots. Figure 3.15 shows an example of how the data is stored in the memory with the description of each sub-figure as follows:

(a) Now considering channel 0, the system will make a decision that a spike is detected once $S16$ is sampled (cause the absolute value of $S16$ is larger than the threshold). According to logic flow of spike detection and alignment described in Fig. 3.12, the system will keep updating the next ten data samples ($S17 - S26$) to find out the maximum absolute value before moving to spike compression. Assuming that $S22$ is the maximum absolute value (this value can vary from $S16$ to $S26$ in this case). Based on the alignment algorithm, the maximum absolute value must be the 16th sample of the entire 64 samples before performing spike compression.

(b) In order to have a clear look for the spike alignment, the identify numbers of data sample have been reordered. In other words, $S7 - S22$ are reordered to $S0 - S15$, and $S23 - S26$ are changed to $S16 - S29$. Now the maximum absolute value is named $S15$. $S0 - S15$ are the first 16 samples of the spike that will be compressed.

(c) After the compression of the first part ($S0 - S15$), the temporary results $W_0 - W_{N-1}$ will be saved in the same slot of $S0 - S15$. It should be noted that N is the number of compression results per spike (see command WRITE(C,N-1) in Appendix 1), which varies from 1 to 16 (here the critical case N = 16 is considered). In order to process the next partial compression of $S16 - S31$, $S20 - S31$ are still needed to be sampled ($S16 - S19$ are already existing during the previous spike alignment operation). Hence, the matrix multiplier unit is idle during this sampling time (the sampling rate is much slower than the system clock). This provides an opportunity that the matrix multiplier can be shared to several channels once it fulfils the timing requirement (timing analysis is detailed in section 3.7).

(d,e) The updated temporary results $W_1 - W_N$ of the other three sections ($S17 - S31$, $S32 - S47$, $S48 - S63$) will be saved in the same place where the updated data samples will be saved in the other 16 slots in order to avoid covering the temporary results. This means that the address calculation of memory needs to be modified. After the compressions of all the 4 parts finished, the address calculation of memory will be back to mode 1, which means that the compression results have to be sent out in time so that they won't be covered by the new input data (see section 3.7).

3.5 Serial Peripheral Interface

The proposed design communicates with A/D converter and RF transceiver through a standard serial peripheral interface (SPI). As shown in Fig. 3.16, a standard SPI consists of four signals: an active-low slave select (SS); a serial data clock (SCLK) with a base value of zero; a "Master Out, Slave In" data line (MOSI) to send the data to the slave device; and a "Master In, Slave Out" data line (MISO) to receive the commands from the slave device. There are two SPI masters in this design: one is responsible for receiving data from A/D converter (see section 3.5.1) while another one is used in sending results and receiving user's commands through RF transceiver (see section 3.5.2). This means that the A/D converter and the RF transceiver always being the SPI slaves. A general timing diagram of a SPI master with SPI options of CPOL=0 and CPHA=0 (information about SPI configuration is introduced in this paper [18]) is illustrated in Fig. 3.17. The slave device should sample the MOSI signal on the rising edge of "SCLK" and update the MISO signal on the falling edge of "SCLK".

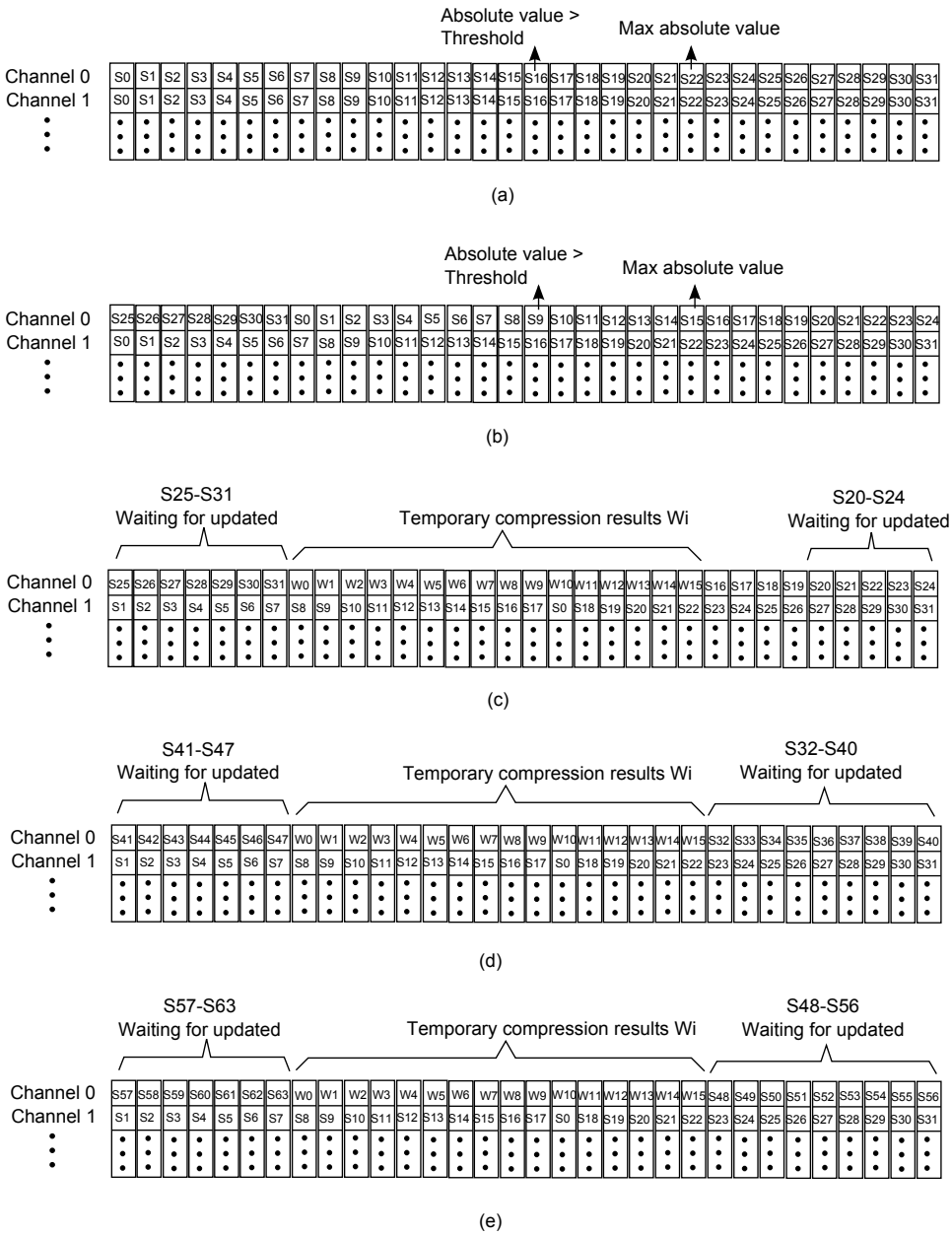


Fig. 3.15: An example of data storage in memory.

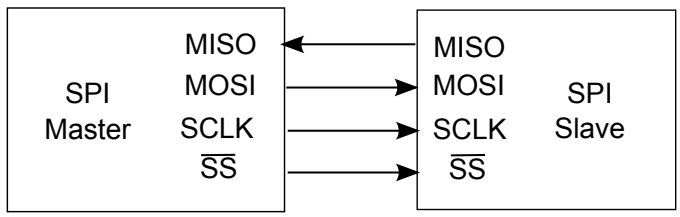


Fig. 3.16: An illustration of connection between a single SPI master and a single SPI slave.

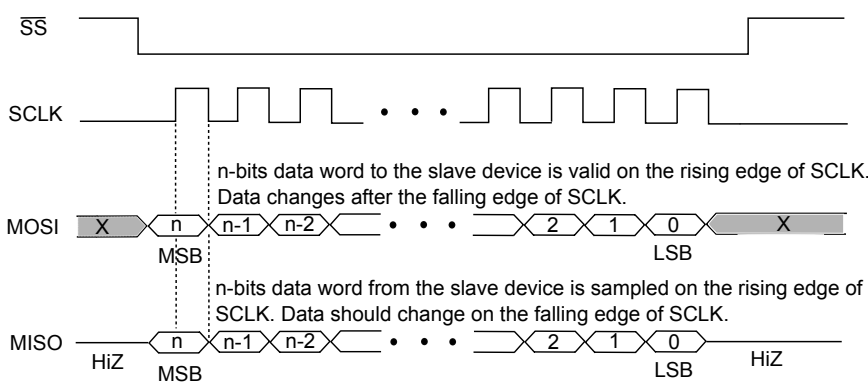


Fig. 3.17: Timing diagram of a SPI master.

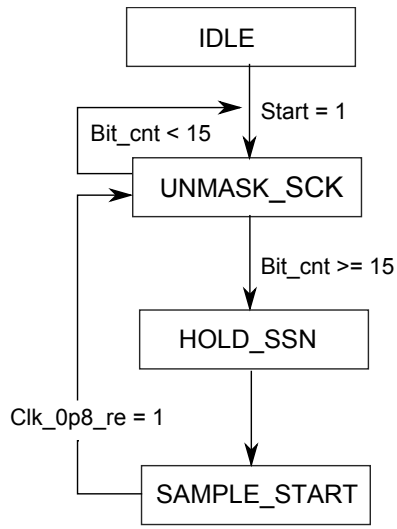


Fig. 3.18: SPI master control state machine for A/D converter.

3.5.1 SPI master for A/D converter

This SPI master is responsible for controlling the SPI slave on the A/D converter and receiving the sample data from the A/D converter, which has 10 bits wordlength. Here the SPI master is just a simplified version where the MOSI signal has been removed since we assumed that the A/D converter only works in a certain mode. This design requires a customized A/D converter otherwise the architecture will be more complicated in order to configure a commercial A/D converter.

The SPI master generates slave select enable signal (SS), serial clock signal (SCLK) and controls the receiver register. If the "Start" signal is asserted, the state machine will start to receive the sample data from A/D converter and the "SCLK" signal continues to transition. Due to the CPHA = 0, the slave select signal will negate after the required hold time and then re-assert between consecutive byte transfers [18]). The SPI control state machine is shown in Fig. 3.18. The SPI control state machine remains in the IDLE state until the "Start" signal is asserted. Then the state machine moves to the UNMASK_SCK state to assert the "SS" signal and generates clock mask signals that control when the clock is output to the SPI bus. In this state, the sample data acquired from ADC stored on the receiver register. After that, the state machine will jump to the HOLD_SSN state.

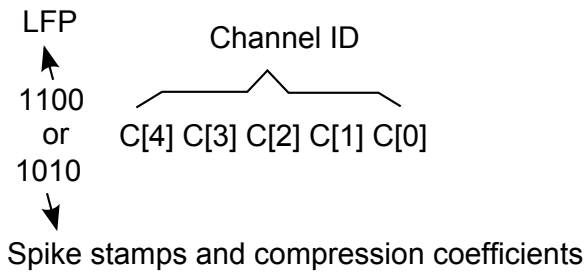


Fig. 3.19: An illustration of output data header.

This state machine will ensure that "SS" will be asserted 1 "SCLK" period before the first "SCLK" edge, meeting the "SS" setup time requirement of most SPI slave devices. The clock mask signal is asserted to disable the "SCLK" signal in this state machine. After that, the state machine will move to SAMPLE_START, which will be waiting for the signal "clk_0p8_re" to be asserted. After "clk_0p8_re" is asserted, it will go back to UNMASK_SCK state to receive new samples.

3.5.2 SPI Master for RF Transceiver

This SPI master is used in controlling the SPI slave on the RF transceiver through "SS" and "SCLK" signals, and transmitting three types of results : LFP (8 bits), spike timestamps (16 bits), and the compression coefficients (8 bits). An 8-bit header was used in specifying these data types and the channel ID is illustrated in Fig. 3.19. The LFP signal is sent by down sampling ratio of 16, which can be turned off by user command as well. However the spike timestamps and the corresponding compression coefficients will only be sent whenever the compression of a detected spike is finished.

The SPI control state machine is shown in Fig. 3.20. Once the assertion of the "Start" signal is finished, which indicates that either LFP signals or compression coefficients are ready, the state machine will move to ASSERT_SSN to assert "SS" signal. The UNMASK_SCK1 and the UNMASK_SCK2 states will reset the bit/byte counters, and transmit the corresponding header, which is generated according to the channel ID and the data content. The XMIT_DATA1 and XMIT_DATA2 states are responsible for transmitting the LFP signals or the spike timestamps with the compression coefficients. Moreover, the HOLD_SSN state is similar to the one in the SPI master for A/D converter.

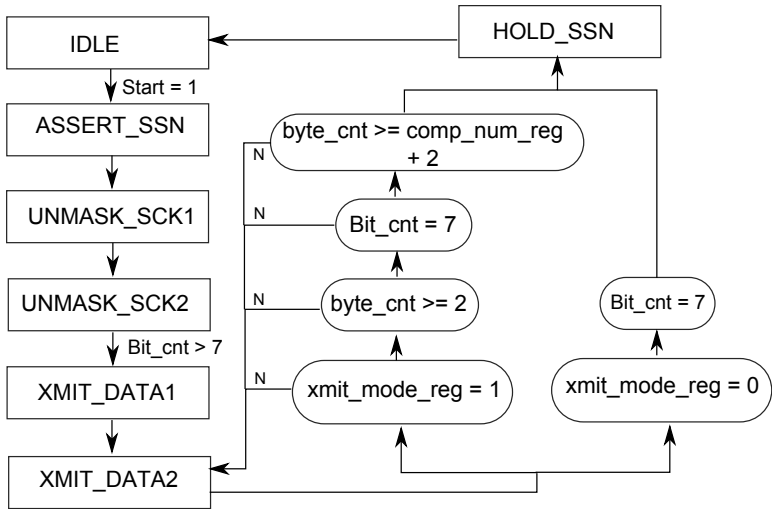


Fig. 3.20: SPI master control state machine for RF Transceiver.

3.6 Scheduler and Resources Sharing for Multi-channels

As mentioned in section 3.4, the area can be decreased by sharing the hardware resources between multiple channels as much as possible. The duplicated blocks (Filters, Spike detection and alignment) can be split into two parts: sequential logic and combinational logic. The sequential logic includes the independent register of channel and the combinational logic contains the state machine logic and the multiplier. (see Fig. 3.21). It is very difficult and complicated to share the sequential logic between multiple channels due to the channel independence. However, the combinational logic is still possible to be shared as shown in Fig. 3.22. An extra multiplexer from 1 to N (N depends on the total number of channels) is needed to select the channel. Moreover, the clock frequency will be N times faster in order to achieve the same speed as the non-sharing system. The speed is not a problem even though it is 32 (assuming 32 channels) times of the sampling rate, as the sampling rate is 25 kHz for each channel, which is rather slow for 65 nm CMOS technology. Actually when considering that this is a sub-V_T system, parallelism is rather ineffective in reducing energy [19], which means that it is still more energy efficient to share the combinational logic for several channels (see Chapter 5 for more detailed information).

It is easy to generate the channel selecting signal sequentially for the

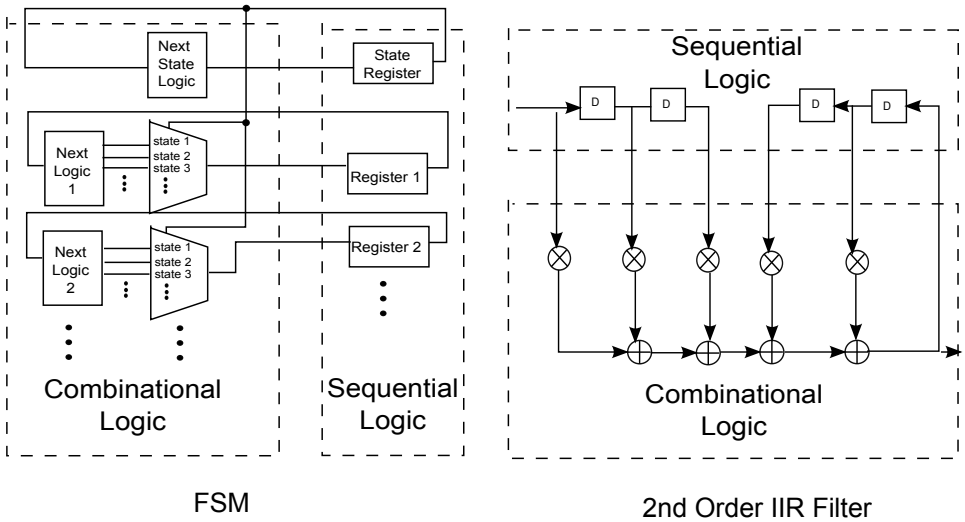


Fig. 3.21: An example of separation of sequential logic and combinational logic for FSM and IIR filter.

multiplexer of the filters and spike detectors since the A/D converter is also assumed to sample the data sequentially for each channel (i.e. inputs are sequentially sampled from channel 0 to channel N-1). However, the spike occurrence of each channel is independent and unpredictable, which requires a scheduler for the compression unit to arrange the tasks. This scheduler is implemented by a FIFO, which records the channel ID that is ready for compression. In addition, one more scheduler is required to arrange the channel ID which has the desired output and is ready to be sent out. The depth of these two schedulers is based on the maximum possible number of channels need to be processed at the same time, which is analysed in Section 3.7.

3.7 Speed Requirement Analysis

This section analyses the timing requirement of different parts in a M channel system in order to achieve a 25 kHz (40 us) sampling rate. The timing requirement derived here is restricted by the design specification (sample rate), which is different from the one from circuit nature (critical path). By calculating the minimum required clock frequency of different parts, the

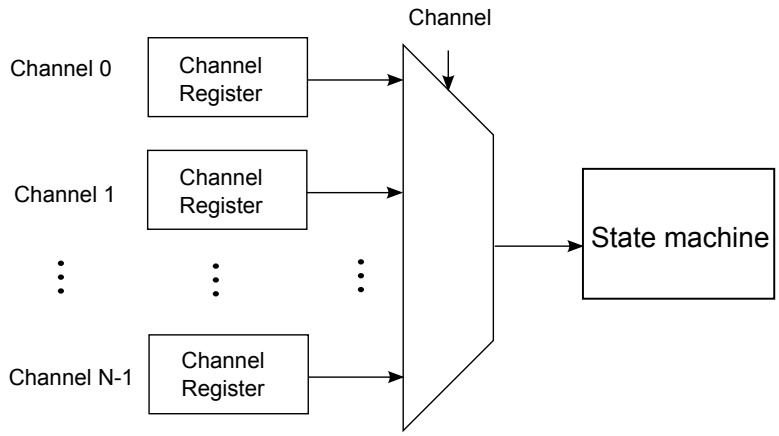


Fig. 3.22: An illustration of state machine sharing for multi-channel system.

speed constrain for the overall system can be determined.

3.7.1 Speed Requirement for Spike Compressor

As the spike compressor is the main computing part of the entire design and its speed may also influence the timing requirement of other parts, this is the start point to analyse the speed requirement. Here are some useful parameters for this timing analysis: the time interval between the data sampling from different channel is 40/M us (this is also the fastest speed to schedule the compression task); it takes 19 (given by N+3) clock cycles for the compression unit to finish a partial matrix multiplication for one channel when the maximum number N = 16 is chosen (N is the number of compression results per spike). The critical case of the compressor scheduler is that all the M channels are ready for scheduling sequentially with a time interval $40/M$. Once all the M channels have been scheduled during this time instance, there will be no more scheduling for any channel in the next several time instances unless the new 16 data samples have been filled up. Hence, the critical case is the compressor unit must finish all the tasks that are scheduled by the scheduler within 40 us so that the new coming data do not overlap the old data (here the old data means that it is still in the waiting list of compression). A formula which can be derived from this

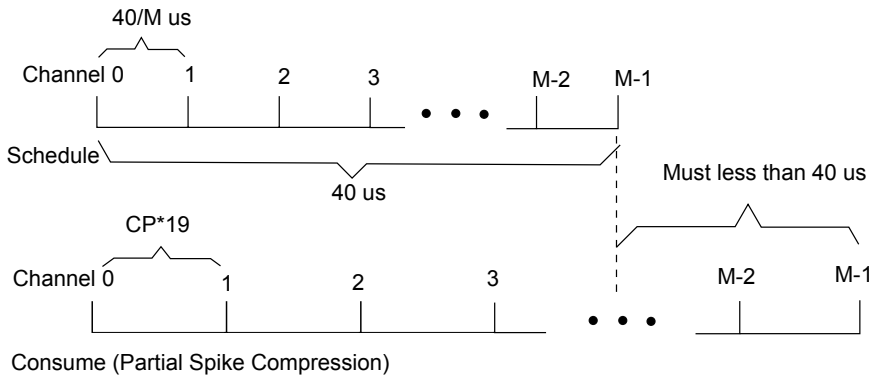


Fig. 3.23: An illustration of scheduling and consuming.

situation is shown as follows:

$$K \times Tp \times 19 < 40us \tag{3.3}$$

where Tp is the clock period of the compressor and K is the maximum number of channels need to be scheduled (i.e., the depth of the compressor scheduler), which is given by:

$$K = M - 40/(Tp \times 19). \tag{3.4}$$

Hence, the required clock frequency (MHz) of the compressor is:

$$f = 1/Tp > 19 \times M/80 \tag{3.5}$$

where the minimum required clock frequency of M=16 channels is 3.8 MHz and the depth of the compressor scheduler K is 8. The same result can be derived from another point of view:

$$M \times (Consuming\ interval - Schedule\ interval) < 40\ us \tag{3.6}$$

where the consuming interval is $Tp \times 19$ and schedule interval is $40/M$. Figure 3.23 illustrates the relationship between scheduling and consuming (here the response time consumes for the first scheduling is neglected).

3.7.2 Speed Requirement for the SPI Master Communicated with RF Transceiver

Similar method can also be applied to calculate the speed requirement of the SPI master for the RF transceiver as well as the depth of the output

scheduler. In this case, the timing constrain is 16*40 us in order to have a finite size for the output scheduler (16*40 us is the time interval between two spikes at the same channel, which is caused by the spike detection and alignment algorithm, but in reality this depends on the neuron firing rate, which will be much larger than this value). In addition, the critical case costs 152 clock cycles to finish one spike transmission of one channel, resulting in a consuming interval $Tp' * 152$, where Tp' indicates the clock period of this SPI master. Here the schedule interval depends on the speed of compressor, which is $Tp*19$. The Tp indicates the clock period of the compressor. Hence a similar equation like 3.6 can be proposed:

$$M \times (Tp' \times 152 - Tp \times 19) < 16 \times 40 \; us \tag{3.7}$$

Assuming that the same clock is used in this SPI master and the compressor (i.e., $Tp' = Tp$), the minimum required clock frequency is 3.3 MHz for M=16. Since this value is less than the minimum required clock frequency of the compressor (the assumption of $Tp' = Tp$ could not be established), $Tp = 1/3.8$ for M=16 are inserted in 3.7 again, resulting that the minimum required clock frequency in SPI master for RF transceiver is $f = 1/Tp' =$ 6.8 MHz for M=16. Similarly the depth of output scheduler is equal to $M - 40/(Tp' \times 152)$, which is 15 for M=16.

3.7.3 Speed Requirement for the SPI Master Communicated with A/D Converter

The sampling rate of the A/D converter is determined by the request sending rate of the SPI master. If the value of sampling rate/channel is fixed, then request sending speed also needs to be fixed. For example, assuming that a A/D converter samples the data form M channels sequentially with a constant request sending rate R, therefore the data sampling rate of each channel will be R/M. In other words, the time consume of SPI master to receive the data of one channel is 1/R. Considering that it takes 11 clock cycles to receive one data sample using SPI (see Section 3.5.1 for detail), the operating frequency for the SPI master must be faster than 11*R, which can be written as:

$$f > 11 \times R = 11 \times M \times Sampling\ rate \tag{3.8}$$

where the sampling rate is typically 25 kHz as mentioned before, therefore the minimum required clock frequency of SPI master for A/D converter is 4.4 MHz assume M=16.

3.7.4 Speed Requirement for Other Modules

If there is no shared resource for the multi-channel system, the timing requirement for the filters and spike detectors of each channel is just equal to the sampling rate. However, as the combinational blocks are shared, the minimum required frequency for the filters and spike detectors is 25*M kHz, which is 400 kHz for M=16. For the system state machine, there is no individual speed requirement but it is determined by the maximum clock frequency if there are multiple clocks in the system.

CHAPTER 4

Low Energy Approaches

Power consumption is becoming an increasingly important aspect of CMOS devices that are energy autonomous and extremely small sized. As one of such applications, wireless BMIs are limited in terms of energy provision. Hence, some methods have been applied for reducing the power consumption, which are described in the following sections.

4.1 Energy Model

This section presents the energy model that is used for simulating the energy dissipation in the sub-V_T region. In general, there are two types of power consumption, dynamic and static. Dynamic power is consumed during logic transitions on nets consisting of two components, switching power and internal power (short-circuit). Static power is due to the leakage current that flows whenever the gates are supplied by an energy source. The total energy dissipation of the static digital CMOS can be modelled as

$$E_{total} = E_{switching} + E_{internal} + E_{leakage} \tag{4.1}$$

where the internal energy dissipation $E_{internal}$ is neglected since it is only a small portion of the overall energy [20]. Based on this fundamental, a

Table 4.1: Minimum required clock frequency (MHz) of different parts.

	M=32	M=16
SPI master 1	8.8	4.4
SPI master 2	6.6	3.3
Data compressor	7.6	3.8
All filters	0.8	0.4
System state machine	8.8	4.4

SPICE-accurate model was derived in this paper [21] as following:

$$E_{total} = C_{inv} V_{DD}^2 \left[\mu_e k_{cap} + k_{crit} k_{leak} e^{-V_{DD}/(nU_t)} \right] \tag{4.2}$$

where C_{inv} is the switched capacitance of an inverter; V_{DD} is the supply voltage applied to the implementation; μ_e is the average switching activity per N samples operations; k_{cap} is the scaling factor that is calculated by the total capacitance normalized by a single inverter capacitance; k_{crit} is the critical path delay per sample in terms of an inverter delay; k_{leak} refers the average leakage scaling factor normalized by the average leakage current of a single inverter; n is the slope factor depends on process, and U_t is the thermal voltage, known as 26 mV at 300 K. All these parameters can be obtained during synthesis and high level simulations. Besides, it should be noted that this model is based on an important assumption: the design is running at the maximum clock frequency. Hence, for the speed constrained design, which means that the design is working under the achievable speed at the energy-minimum operating point, then the total energy dissipation is modified as

$$E_{total} = \mu_e k_{cap} C_{inv} V_{DD}^2 + k_{leak} I_0 V_{DD} T_{clk} \tag{4.3}$$

where I_0 represents the average leakage current of a single inverter [21] and T_{clk} is the clock period.

4.2 Multiple Clock Domains and Power Domains

As calculated in Chapter 3.7, different parts of the design have different speed requirements (see table 4.1). Thanks to the relation between power

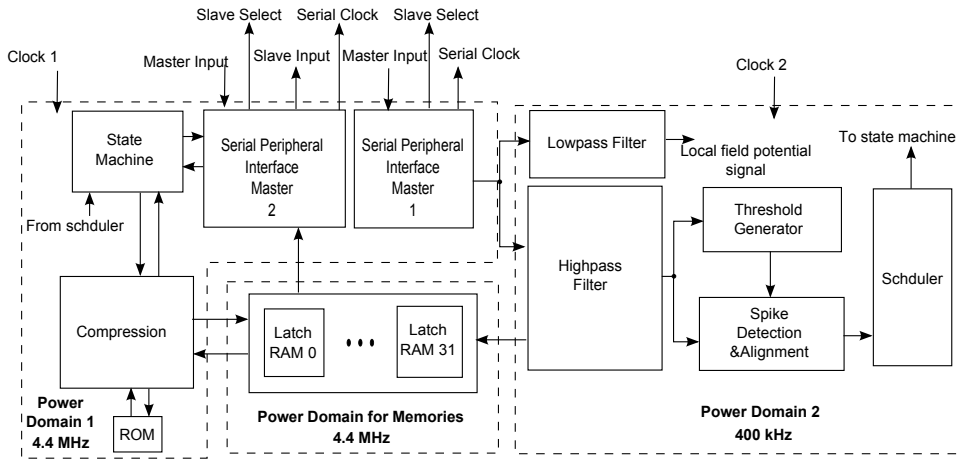


Fig. 4.1: The block diagram of different power domains in the design.

and supply voltage, a lower supply voltage will result in a much lower power consumption but also a slower speed. To achieve maximum speed and lower power consumption at the same time, each part of the design should operate at the lowest supply voltage that can achieve the minimum required speed. However, this means that there would be 4 clock domains in this design, which increases the area overhead and the complexity of communication between clock domains and power domains. Actually, the speed requirements for SPI master 1, SPI master 2, data compressor and system state machine are quite close to each other. It gives the possibility of combining them into one clock domain, where the maximum value of these speed requirement is used (i.e., 4.4 MHz for M=16). Hence, only two clocks are used in this design, which are 4.4 MHz and 400 kHz (the final tapeout is aimed at 16 channels). It should be noted that one of these two clocks is derived from another, which means that clock 2 (400 kHz) derived from clock 1 (4.4 MHz) divided by 11. Using a derived clock ensure that the design is synchronous and resulting in a much easier signal communication between clock domains without the help of a synchronizer [22]. In addition, although the speed of memories is identical with the data compressor, there exists an extra power domain of memories in order to perform a separate power measurement. In summary, there are 2 clock domains and 3 power domains in this design as shown in Fig. 4.1.

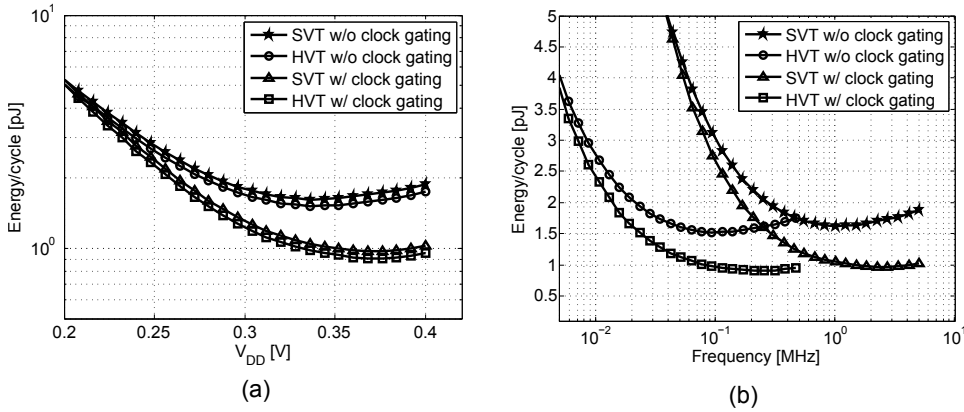


Fig. 4.2: Energy curves versus: (a) Supply voltage; (b) Maximum operational frequency.

4.3 Supply Voltage Reduction

Voltage scaling is a very effective way to reduce energy consumption. By applying aggressive voltage scaling, where the system will operate in the sub-threshold region, the power and energy consumption can be reduced by up to 2 orders of magnitude [19]. The energy dissipation shown in Fig. 4.2 is calculated by using the model in Section 4.1. The energy minimum voltage (EMV) is 0.33 V for both high-threshold (HVT) and standard-threshold (SVT) standard cell libraries. To fulfil the speed requirement, the design will operate at around 0.4 V (SVT cells) instead of 0.33 V, resulting in a slightly higher energy dissipation. However, when compared to the design operating at 1.2 V, it achieves a reduction of 94% in energy consumption. It should be noted that this energy simulation was performed without the memories, and the supply voltage of both power domain 1 and power domain 2 are identical. That is because the optimum supply voltage (i.e, the supply voltage that can fulfil the speed requirement with the minimum energy dissipation) of power domain 1 is very close to the one of power domain 2, which means that they can combine into one power domain. However, in order to measure the energy dissipation of these two power domains separately, two power domains remain in the final tapeout.

4.4 Standard-cell Based Latch Memories

After applied the aggressive supply voltage reduction to the system to achieve ultra low power design, the conventional SRAM using 6-transistor (6T) bit-cells are inoperable in the sub-V_T domain. An attractive alternative from this paper [23] is the use of standard-cell based memories (SCMs) that provides high reliability, low area cost for small storage requirements (a few kb), and straightforward synthesis which reduces the design effort. As the total size of memory used in this design is 4 kb (16 channels), it will perfectly match the optimum case of SCMs, giving a relatively low area cost. The measurement value of EMV for this latch memories is 0.4 V with a energy dissipation of 29 fJ/bit-access, but in our case the supply voltage should be slightly higher (nearly 0.5 V) due to the speed constrain, which will end up with a energy dissipation of 34 fJ/bit-access.

CHAPTER 5

Results and Discussions

This chapter presents the analysis and discussion about the results of hardware implementation organized from section 5.1 to section 5.4. Section 5.1 analyses the fixed point performance including the wordlength simulation result of the spike compressor and the comparison with the floating point Matlab model; The influence of different channel compression ratio on area and processing time is discussed in Section 5.2; The synthesis results with the corresponding strategies are presented in Section 5.3; The layout after place-and-route is shown and discussed in Section 5.4.

5.1 Fixed Point Performance

In hardware implementation, wordlength is one of the key points with large influence on the performance and area. It is obvious that there is a trade-off between the performance and area: increasing the wordlength usually improve the performance as well as increasing the area. In the proposed design, the high performance refers to the high accuracy of results or the low quantization error. In general, we would like to achieve a reasonable accuracy (reasonable quantization error) with the smallest area. Hence some simulations were performed in order to explore the relation between the

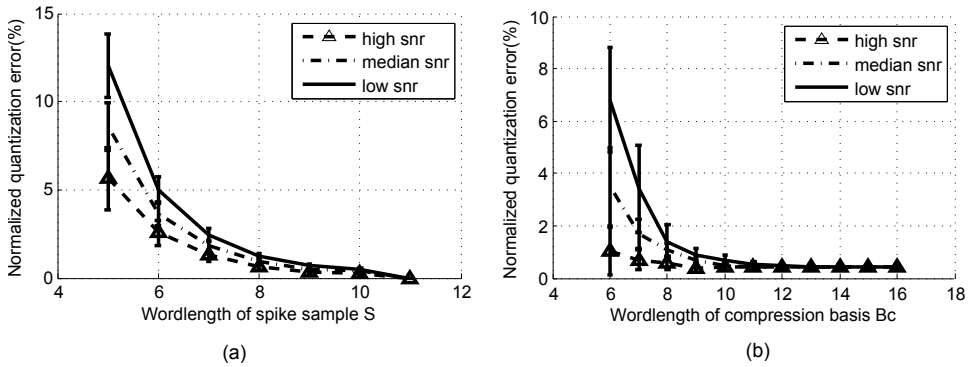


Fig. 5.1: Wordlength simulation of: (a) Spike sample S; (b) Compression basis B_c.

wordlength and the accuracy (quantization error). Figure 5.1 shows the wordlength simulation results of the spike waveform and the compression basis. Here the normalized quantization error of a given spike i is calculated by:

$$\varepsilon_{\mathbf{i}} = \frac{abs(\widetilde{\mathbf{Si}} - \mathbf{Si})}{\max(abs(\mathbf{Si}))} \times 100\% \tag{5.1}$$

where $\widetilde{\mathbf{Si}}$ is the quantized waveform of spike i and $\mathbf{Si}$ is the waveform of spike i with full precision. ε_i is an array with 64 elements (the same numbers as spike waveform) where only the maximum value will be selected as a representative. As mentioned in Chapter 2, the test data of each SNR scenario consists of thousands of spike. Hence the plot in Fig. 5.1 is an average value of thousands of spikes with the error bar standing for the standard deviation of these values.

As shown in Fig. 5.1, using too small wordlength such as 6 bits or an even smaller one will lead to a very large quantization error. However, the wordlength close to 10 bits does not affect the precision significantly. Hence 8 bits wordlength was chosen for both spike waveform and compression basis since the normalized quantization error is less than 2% even in low SNR scenario. Once the wordlengths of spike waveform S and compression basis B_c were determined, the final simulation of the wordlength of compression results W_d was performed. The result is shown in Fig. 5.2. For the same reason, 8 bits wordlength for compression waveform W_d as a normalized quantization error is around 5%. Moreover, another important reason for

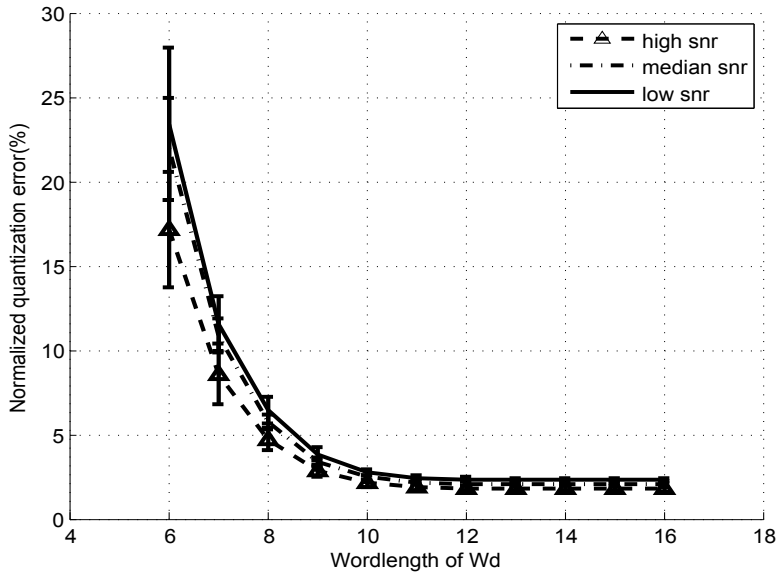


Fig. 5.2: Wordlength simulation for W_d.

this choice is that the memory can be simply shared between spike waveform S and compression waveform W_d if the wordlength of W_d is the same as the one of S. Besides, 8 bits are equal to 1 byte which is convenient for the further process in RF transmitter.

The performance of the fixed point hardware implementation was estimated in terms of spike sorting accuracy and spike reconstruction accuracy. The calculation of overall spike sorting accuracy P_{ID} was described in Chapter 2. Here the spike reconstruction accuracy is the maximum value of the cross-correlation result between the reconstructed spike and the mean spike of the target neuron (see Chapter 2). Figure 5.3 and Figure 5.4 show the comparison between the fixed point hardware implementation and the floating point Matlab model for spike reconstruction accuracy and spike sorting accuracy respectively. These two figures indicate that there is very small loss for the hardware implementation with 8 bits wordlength of the spike waveform, compression basis and compression waveform. Furthermore, both the spike sorting accuracy and reconstruction accuracy decrease as the SNR decreases. That is because lower SNR lead to a larger quantization error which can be seen in Fig. 5.2. Meanwhile, a higher number of compression basis dimension results in a higher precision but also a lower compression ratio.

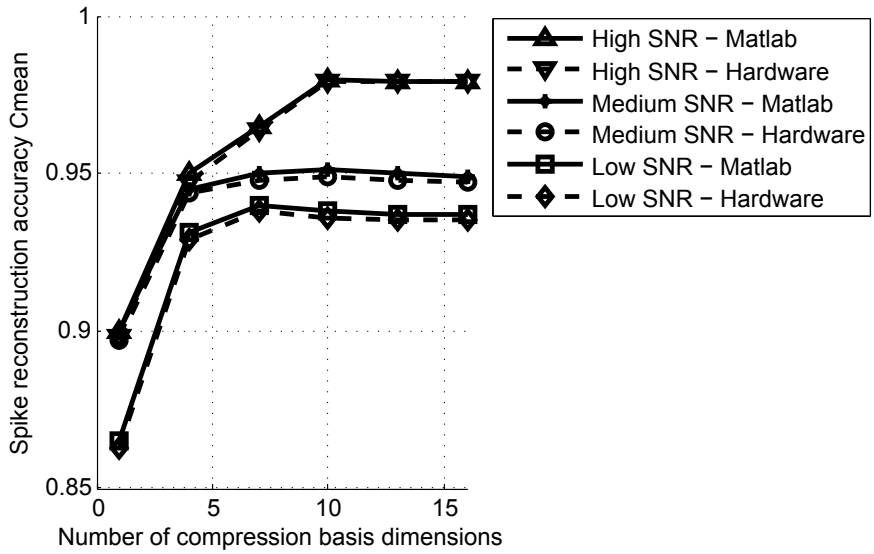


Fig. 5.3: Spike reconstruction accuracy versus number of compression basis dimension (inverse to compression ratio).

For example, in low SNR scenario, using 2 as the number of compression basis dimension gives a compression ratio of 32, a spike reconstruction accuracy of 0.88 and a spike sorting accuracy of 0.57; where using 8 gives a compression ratio of 8, a spike reconstruction accuracy of 0.94 and a spike sorting accuracy of 0.63.

5.2 Compression Ratio

The relation between spike compression ratio and compression basis dimension is

$$CR_{spike} = \frac{64}{Compression\ basis\ dimension} \tag{5.2}$$

where CR_{spike} is the spike compression ratio, it varies from 4 to 64 since the compression basis dimension is from 1 to 16. In general, larger compression basis dimension (smaller compression ratio) requires higher memory usage. However, in this design, the compression ratio does not influence the overall memory usage. The reason is that the memory used in spike alignment component is shared with spike compression component, and the spike sample storage of spike alignment determines the maximum memory usage of the

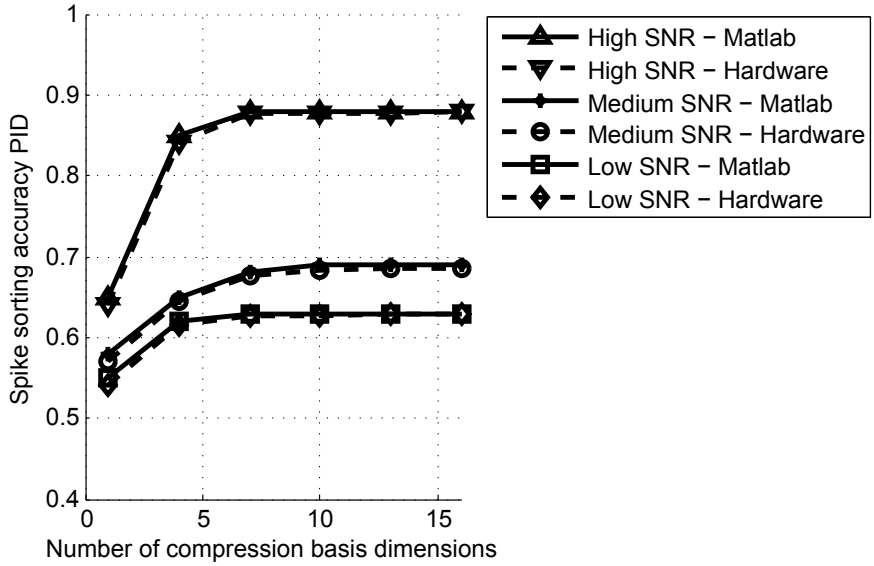


Fig. 5.4: Spike sorting accuracy versus number of compression basis dimension (inverse to compression ratio).

design. It also means that the memory used in spike compression component is less than the memory used in spike alignment component. Thus, in terms of overall memory usage, increasing the compression ratio does not influence the total area significantly. When this compression ratio is compared with other works, it should be noted that the proposed design transmits spike data only when there is a detection of a spike. Hence, a more realistic definition of compression ratio for each channel is given by

$$CR_{channel} = \frac{NB}{NA} = \frac{SR * W}{FR * (Compression\ basis\ dimension + ST) + LFP} \tag{5.3}$$

where NB and NA are the amount of data before and after compression respectively. SR is sample rate, FR is the spike firing rate and W is the wordlength of sample data. The output data after compression contains compression coefficients of spike, some overhead bits for recording spike time (ST) , and the LFP data (LFP). As mentioned in the previous sections, the sampling rate of this design is 25 kHz, the wordlength of sample data is 16, overhead bits of recording spike time are 24, the wordlength of compression coefficient is 8, and the LFP data is 12504 bits/sec. For example, assuming

Table 5.1: Channel compression ratio for different spike firing rate and spike compression ratio.

Spike firing rate FR	CR_{spike}	$CR_{channel}$ with LFP	$CR_{channel}$ without LFP
10 spikes per second	4	28	263
10 spikes per second	16	30	714
10 spikes per second	64	31	1250
100 spikes per second	4	14	26
100 spikes per second	16	22	71
100 spikes per second	64	25	125

a spike firing rate of 10 spikes/sec, and a compression basis dimension of 8, the compression ratio for each channel is given by

$$CR_{channel} = \frac{25000 * 16}{10 * (8 * 8 + 24) + 12504} = 18. \tag{5.4}$$

The different compression ratio with different spike firing rate is described in table 5.1. In another work [24], the author developed a data compressor in BMI based on the Walsh-Hadamard Transform (WHT). The compression ratio in [24] is 60 and 5 when the spike firing rate is 10 spike/sec. and 100 spike/sec. respectively. It should be noted that in [24], the output does not contain the LFP component. Thus, in order to have a fair comparison between [24] and the proposed design, the LFP component should be removed from the calculation of channel compression ratio (Actually the LFP transmission can be turned off by a SPI command, see Appendix 1). Fig. 5.5 shows a clear comparison of compression ratio between these two designs. Both of these two designs are compared with the SNR scenarios of 20. Table 5.1 shows the channel compression ratio with different parameters.

As mentioned before, the entire matrix multiplication (64 samples) is split into 4 small parts and each part contains 16 samples for multiplication (see section 3.4). For each part, it takes (N+3) clock cycles for the compression unit to finish multiplication (see section 3.7.1). N refers to the number of compression basis dimension, as well as the number of compression result which is actually transmitted. Thus, the processing time of spike compression is given by (N+3)*4. According to this formula, if spike compression ratio CR_{spike} is 64 (the compression basis dimension will be 1 according to

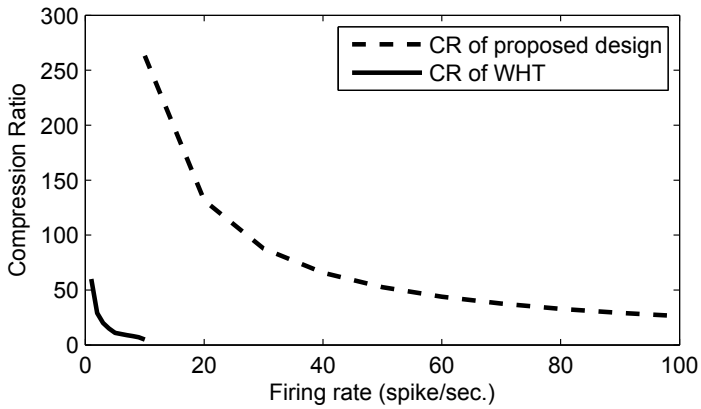


Fig. 5.5: Compression ratio as a function of spike firing rate

(5.2)), the processing time is (1+3)*4=16 clock cycles. In the same way, if spike compression ratio CR_{spike} is 4 (the compression basis dimension is 16), the processing time is (16+3)*4=76 clock cycles.

In summary, the processing time will decrease as the compression ratio increases and the overall memory usage is not influenced by the compression ratio in the proposed design.

5.3 Synthesis Results

In previous research [25], different synthesis strategies for sub-V_T system design using commercial standard-cell libraries (SCLs) and commercial logic synthesis as well as place-and-route tools were analysed. In this project, the synthesis was performed by Synopsys Design Compiler with the low leakage standard-threshold (LLSVT) standard-cell libraries which is already discussed in Chapter 4. Since this design is not timing-critical, the above-V_T synthesis with a relax timing constrain and a hard power and area constrain was performed in order to obtain the most energy-efficient result. Moreover, the re-characterized SCL for sub-V_T voltage supply voltages was still needed during the post-layout simulation at the target voltage although it was not needed for synthesis. Besides, the clock gating technology was used in this design to decrease the power consumption. The insertion of clock gating was realized by the automated script during synthesis, which achieves a fast clock gating controlling.

Table 5.2: Percentage of area for the different parts.

	Non-clock gating (%)	Clock gating (%)
SCMs	38.5	38.9
Registers	35.3	35.4
Multipliers	11.8	12.6
Control net	9.2	7.1
Adders	3.3	3.8
ROM	1.9	2.2

The table 5.2 shows the synthesis result of area. It shows that most of the area is occupied by the standard-cell based memories (More detailed information is shown in Chapter 4.2) and the registers, followed by the multipliers and control nets. The registers used in the filters can not be shared among different channels, which occupy large percentage of the area and will grow with the increasing of total number of channels. After using clock gating, the area occupation of the control net is reduced dramatically as most of the control logics of registers will be redundant and can be removed due to the existing control from the clock gating latches. Together with multipliers and adders, the total area of filters (lowpass filters, highpass filters and exponential filters) is more than 40% while the area of data compressor is less than 15 %. Moreover, the area of filters will increase when increasing the amount of channel but the area of data compressor will keep the same since it is shared for all the channels and independent of the amount of channel. Hence, optimizing and minimizing the filters can be taken to reduce the total area effectively in the future work.

The Table 5.3 shows the synthesis results for the design with clock gating and without clock gating. It is obvious that the design with clock gating is more area efficient and energy efficient. However, the simulation and test with the clock gating should be very careful in a sub-V_T region since the clock gating might be a risk for a sub-V_T system. Moreover, the maximum achievable clock frequency of the design with the clock gating is also higher than the one without the clock gating due to the shorter critical path by removing some control logics of registers. Here the clock frequency only means the frequency of clock 1 (i.e., the clock used by data compressor, SPI, etc), since clock 2 (i.e., the clock used by filters and spike detectors)

Table 5.3: Synthesis results using LLSVT SCLs.

		Non-clock gating	Clock gating
Core area(mm^2)		0.184	0.167
Gate count		88562	80107
Max clock frequency(MHz)	1.2V	396.8	480.8
	0.4V	5.0	5.3
Energy/cycle(pJ)	1.2V	30.51	16.43
	0.4V	1.89	1.03

is just considered as a generated clock from clock 1. When the supply voltage downs to 0.4 V, the maximum achievable clock frequency is still slightly higher than the required value 3.8 MHz (calculated from Chapter 4.6). Although the real measurement of the maximum clock frequency from the chip might be a little different from this simulation result, the supply voltage can be slightly increased or decreased to reach the required clock frequency. Moreover, the energy dissipation of the design without clock gating at supply voltage of 0.4 V is merely 1.89 pJ/cycle, and has a 94% reduction compared to the one at 1.2 V, which will achieve more than 50% reduction if applying clock gating.

5.4 Layout

The final layout is shown in Fig. 5.6, which was realized by Cadence SoCEncounter. As a part of a multi-project die, the size is 900x500 um^2, occupying almost half of the 1 mm^2 die with some space of right hand side remaining for the pads placement of other projects. There are 4 power domains with a separated VDD contact for each and two shared GND contacts. These two GND pads were placed near right top and left bottom respectively, in order to spread out the power distribution for minimizing the dynamic voltage (IR) drop [26] (IR-drop is the voltage drop on the supply lines, caused by the large current spikes at the rising edge and falling edge of the clock due to the share of flip-flops. Increasing the numbers of supply contacts and the width of power stripes are effective methods to reduce the IR-drop.). Table 5.4 shows the density of cell placement of different power domains. The overall cell placement density achieves 70.55%, which is quite compact and

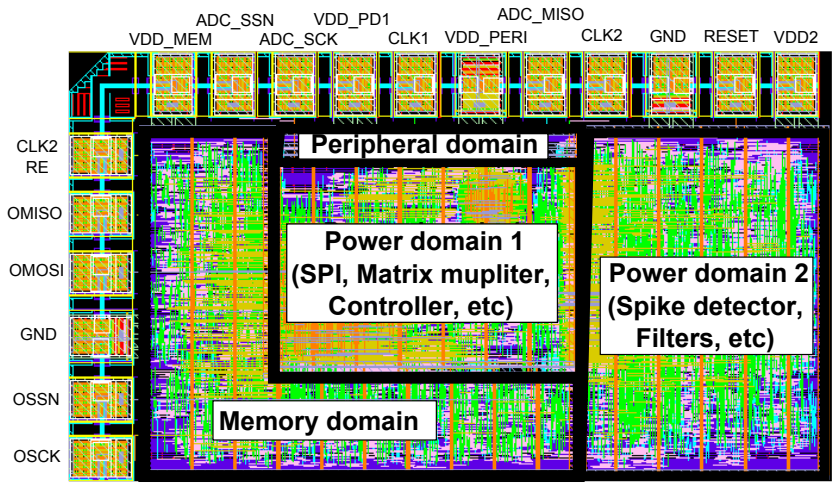


Fig. 5.6: Layout of the overall design.

Table 5.4: Area occupation of cells in different parts of layout.

Part	area occupation(%)
Power domain 1	72.31
Power domain 2	69.81
Power domain memory	71.34
Overall placement	70.55

reasonable. It is still possible to reach a higher placement density if required, but that is not a requirement in this project. Once the post-layout simulation was performed successfully, fully design rule check (DRC) was applied before fabrication. It should be noted that the re-characterized SCLs with the target sub-V_T supply voltage are needed to generate the sub-V_T timing standard delay format (SDF) files for the static timing analysis of sub-V_T timing.

CHAPTER 6

Conclusion

This thesis project realized the data compressor for wireless brain machine interface with an extremely low energy cost targeting in 65 nm CMOS technology. The design consists of pre-processing filters, spike detectors, spike compressor and the SPI IO protocol. The total die area is 900×500 μm^2. Area and energy dissipation have been dramatically reduced by sharing hardware resources for different channels, optimizing the matrix multiplier and utilizing a standard-cell based latch memories. Moreover, aggressive voltage scaling, clock gating and multiple clock domains have been performed resulting in an even less energy dissipation. The simulated energy dissipation is 1.03 pJ/clock cycle if applying clock gating. It is nearly 30 times less than the standard super-V_T implementation without clock gating. At the same time, the precision loss of the fixed point hardware performance is less than 0.1% compared to the full precision Matlab model. The reconstruction accuracy achieved more than 95% for different input SNR scenarios and the different dimension of compression coefficients. Furthermore, high flexibility of compression ratio has been achieved by the adjustable dimensions of the compression coefficients. It can be adjusted via SPI commands according to the signal quality of the channel and the requirement of the spike reconstruction accuracy.

6.1 Future Work

This design has been sent for fabrication targeting in 65 nm CMOS technology. As the first generation, this design provides a very good example and idea. It can be extended for building up the entire wireless BMI system, especially the implantable device where energy dissipation and area become more significant. Hence, some methods can be taken in account regarding to several aspects of the ASIC part of the BMI system:

- Future research on the highpass and lowpass filters as well as the threshold generator (exponential filter) can be investigated for decreasing the area. For example, using direct form II structure of filter will reduce the number of delay buffer. Moreover, the possibility for the use of analogue filter can also be analysed in terms of energy, area and performance.

- Compression ratio should be increased by improving the algorithm or estimating other algorithms. However it should not increase the overhead much in terms of area and power when increasing the compression ratio.

- Customized ultra low power A/D converter should be developed, which can provide the best performance and minimize area by adjusting the dynamic range to the neural recording signal.

- Customized design for RF transmitter/receiver can be implemented in order to build up the most energy and area efficient hardware without any redundant unit.

APPENDIX A

Appendix 1

A.1 User Controlled SPI Command Words

Command: INTERRUPT

MSB 7	6	5	4	3	2	1	LSB 0
0	0	1	1	1	1	1	1

Comments:
This command interrupts the current activities and sets the system state to "IDLE". When the system is working in "WORK MODE", it is necessary to execute this command when you want to modify the parameter registers.

Command: WRITE(C, N-1) - Write the parameter N(Number of compression coefficients per detected spike) to channel C

MSB 7	6	5	4	3	2	1	LSB 0
0	1	0	C[4]	C[3]	C[2]	C[1]	C[0]

MSB 7	6	5	4	3	2	1	LSB 0
X	X	X	X	N[3]	N[2]	N[1]	N[0]

Comments:
This is a two-section command. The first section is to select the target channel and the second section is to specify the amount of coefficients need to be transmitted after spike compression, which is referred to N. The range of N is from 1 to 16 and $N-1$ is from 0 to 15 (”0000” to ”1111” in binary). The default value of $N-1$ is ”1111”. The selection of this value is based on the channel quality. If the channel quality is bad, it is possible to increase the amount of compression coefficients to be transmitted to increase the precision of the reconstructed spike waveform. Besides, this value is independent for each channel which means that it is possible to choose different value for different channel depending on the channel quality.

Command: WRITE(C, α, β) - Write the parameter α and β to the exponential filter in channel C

MSB 7	6	5	4	3	2	1	LSB 0
0	0	1	C[4]	C[3]	C[2]	C[1]	C[0]

MSB 7	6	5	4	3	2	1	LSB 0
α[1]	α[0]	X	X	X	β[2]	β[1]	β[0]

Comments:
This is a two-section command. The first section is to select the target channel and the second section is to specify the new value of α and β. More information about these two parameters is introduced in section 3.2.2. The default value for α and β is "11" and "110" respectively. Moreover, the "X" in the second section indicates that it is unused.

Command: START 1 - Reset threshold registers for ALL channels and then start to do spike detection and compression

MSB 7	6	5	4	3	2	1	LSB 0
0	1	1	0	0	0	0	0

Comments:
This command firstly resets the content of threshold registers of all channels to zero (the reason is detail in 3.2.2). After the settling time of the threshold generator is set, the system will start to do spike detection and compression (move to "WORK MODE" state).

Command: START 2 - Reset threshold registers for channel C and then start to do spike detection and compression

MSB 7	6	5	4	3	2	1	LSB 0
1	0	0	C[4]	C[3]	C[2]	C[1]	C[0]

Comments:
This command is similar to "START 1" where the only difference is that it only resets the threshold register of ONE specific channel. Normally this command is executed to restart the spike detection and compression for one channel without influencing the current state of other channels. Hence, it is convenient to use this command to modify the system parameters of a specific channel.

Command: LFP ENABLE - Turn off/on the LFP transmission

MSB 7	6	5	4	3	2	1	LSB 0
1	0	1	A[4]	A[3]	A[2]	A[1]	A[0]

Comments:
This command disables the LFP transmission (when "A" = "11111") or enables the LFP transmission (when "A" /= "11111"). The default configuration of LFP transmission is ON.

Command: TEST - Start the test mode for fast functional testing

MSB 7	6	5	4	3	2	1	LSB 0
1	1	1	0	0	0	0	0

Comments:
This command sets the system to the "TEST MODE" state for testing and debugging the functionality quickly and efficiently.

References

[1] W. Commons, "Anatomy and physiology," 2007. [Online]. Available: http://commons.wikimedia.org/wiki/File:Neuron.svg

[2] P. T. Thorbergsson, *Signal Modeling and Data Reduction for Wireless Brain-Machine Interfaces*. Lund University, Dept. of Electrical and Information Technology, 2012.

[3] L. R. Hochberg, D. Chen, R. D. Penn, and Donoghue, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," *Nature*, vol. 442, no. 7099, p. 164, 2006.

[4] M.Vellist, S.Perel, M. Spalding, A.Whitford, and A. Schwartz, "Cortical control of a prosthetic arm for self-feeding," *Nature*, vol. 453, no. 7198, p. 1098, 2008.

[5] J. Csicsvari, D. A. Hen, B.Jamieson, K. D. Harri, A. Sirota, and G. Buzsáki, "Massively parallel recording of unit and local field potentials with silicon-based electrodes," *J. Neurophysiol.*, vol. 90, no. 2, p. 1314, 2003.

[6] P. T. Thorbergsson, M. Garwicz, J. Schouenborg, and A. J. Johansson, "Minimizing data transfer with sustained performance in wireless brain-machine interfaces," *Journal of Neural Engineering*, vol. 9, no. 3, p. 036005, 2012.

[7] S. F. Lempka, M. D. Johnson, M. A. Moffitt, K. J. Otto, D. R. Kipke, and C. C. McIntyre, "Theoretical analysis of intracortical microelectrode recordings," *Journal of neural engineering*, vol. 8, no. 4, p. 045006, 2011.

[8] N. Z. Quian Quiroga R and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Computation*, vol. 16, no. 8, p. 1661, 2004.

[9] S. Mukhopadhyay and G. C. Ray, "A new interpretation of nonlinear energy operator and its efficacy in spike detection," *Biomedical Engineering, IEEE Transactions on*, vol. 45, no. 2, pp. 180–187, 1998.

[10] 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," *Biomedical Engineering, IEEE Transactions on*, vol. 50, no. 8, pp. 999–1011, 2003.

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

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

[13] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[14] P. T. Thorbergsson, M. Garwicz, J. Schouenborg, and A. Johansson, "Statistical modelling of spike libraries for simulation of extracellular recordings in the cerebellum," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 2010, pp. 4250–4253.

[15] P. T. Thorbergsson, H. Jorntell, F. Bengtsson, M. Garwicz, J. Schouenborg, and A. Johansson, "Spike library based simulator for extracellular single unit neuronal signals," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, 2009, pp. 6998–7001.

[16] P. T. Thorbergsson, M. Garwicz, J. Schouenborg, and A. J. Johansson, "Computationally efficient simulation of extracellular recordings with multielectrode arrays," *Journal of Neuroscience Methods*, 2012.

[17] D. Heeger, "Poisson model of spike generation," *Handout, University of Standford*, vol. 5, pp. 1–13, 2000.

[18] xilinx, "Coolrunner-ii serial peripheral interface master," 2009.

[19] M. Alioto, "Ultra-low power vlsi circuit design demystified and explained: A tutorial," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 1, pp. 3–29, 2012.

[20] E. Vittoz, *Low-Power Electronics Design*, ch. 16 ed. CRC Press, 2004.

[21] O. Akgun, J. Rodrigues, Y. Leblebici, and V. Öwall, "High-level energy estimation in the sub-VT domain: simulation and measurement of a cardiac event detector," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 1, pp. 15–27, 2012.

[22] P. P. Chu, *RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability.* Wiley-Interscience, Jan., no. 3, pp. 477+.

[23] O. Andersson, B. Mohammadi, P. Meinerzhagen, A. Burg, and J. N. Rodrigues, "Dual-vt 4kb sub-vt memories with $<$ 1 pw/bit leakage in 65 nm cmos," in *ESSCIRC (ESSCIRC), 2013 Proceedings of the*, 2013, pp. 197–200.

[24] H. Hosseini-Nejad, A. Jannesari, and A. Sodagar, "Data compression in brain-machine/computer interfaces based on the walsh hadamard transform," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[25] P. Meinerzhagen, O. Andersson, Y. Sherazi, A. Burg, and J. Rodrigues, "Synthesis strategies for sub-vt systems," in *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, 2011, pp. 552–555.

[26] A. Karlsson, O. Andersson, J. Sparso, and J. Rodrigues, "Ir-drop reduction in sub-vt circuits by de-synchronization," pp. 1–3, 2012.

LUND
UNIVERSITY

http://www.eit.lth.se