IMPLANTABLE VLSI SYSTEMS FOR COMPRESSION AND COMMUNICATION IN WIRELESS BIOSENSOR RECORDING ARRAYS

By

Awais Mehmood Kamboh

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Electrical Engineering

2010

ABSTRACT

IMPLANTABLE VLSI SYSTEMS FOR COMPRESSION AND COMMUNICATION IN WIRELESS BIOSENSOR RECORDING ARRAYS

By
Awais Mehmood Kamboh

Successful use of microelectrode arrays to record neural activity in the cortex has opened new opportunities for scientists to decode the intricate functionality of the human brain and the behavior of neurons that enable its complex operation. The resulting brain-machine interface devices play a critical role in enabling patients with neural disorders to achieve a better lifestyle. Such interfaces provide a direct interface to the brain and show great promise in many biomedical applications.

This thesis explores some of the major obstacles impeding the advance of wireless neural implants and addresses them through development of highly efficient algorithms and implantable hardware. An overwhelming amount of data is generated by the microelectrode arrays, resulting in a data bandwidth bottleneck. To overcome this problem, an implantable system has been devised to enable control over the amount of data that must be transmitted without compromising the information contained in the array of neural signals. Furthermore, the nature of the wireless communication channel across the skin tissue is not well characterized. In this thesis, solutions have been developed to maximize that data throughput and enable unfailing yet low-power communication of bidirectional data between the implanted device and the external world. Finally, a unified energy-efficient, implantable CMOS integrated circuit was developed to address these two critical problems. The resulting integrated solution ensures seamless multi-modal operation, and thus establishes a pathway to the design of next-generation neuroprosthetics devices. Although the motivation for this thesis comes from the field of neuroprosthetics, the solutions devised are pertinent to a wide range of implantable applications.

Dedicated to my parents

ACKNOWLEDGEMENTS

"*In the name of Allah, the Beneficent, the Merciful*"

The Quran (Chapter 1: Verse 1)

I would like to thank my advisor Professor Andrew Mason for his continued guidance and support through my PhD research. He has been a true mentor to me. I would also like to acknowledge my dissertation committee including Professor Hayder Radha, Professor Richard Enbody and Professor Ramakrishna Mukkamala for their valuable feedback that made this dissertation possible.

I must also thank my colleagues at AMSaC lab who helped me put my ideas to practice. Special thanks to Chao, Yue and Daniel whose technical insights helped improve my work. Matthew, Garrett and Leslie also helped me achieve my research goals. Waqar, Xiaowen, Xiaoyi, Yuning, Stefan, James, Haitao, Lin and Liya made my stay at MSU wonderful. I am also grateful to Khawar, Saad, Jawad, Khurram, Usman, Sajjad, Attaullah, Qaiser, Abdul Rehman, Ahmed, Aqeel and Zubair for their friendship.

My research would not have been possible without the support from National University of Sciences and Technology, Islamabad, and the Higher Education Commission of Pakistan. My research was also funded in part by the National Institutes of Health, USA.

Finally and most importantly, I want to thank my parents and siblings for believing in me, and my wife and daughter for supporting me every step of the way.

Awais Mehmood Kamboh

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

1.1 Motivation

How the brain functions is one of the questions that have puzzled scientists and philosophers alike for centuries. Modern scientists have been making significant advances towards solving this mystery. Apart from just satiating an intellectual desire, a better understanding of brain functions has been helping neurophysiologists develop treatments for a variety of disorders including Parkinson's disease or several auditory disorders. It also helps us understand better the physiological changes during various seizures, migraines and epilepsy. Restoration of motor and sensory capability for the disabled such as quadriplegic, blind, deaf or persons with missing or damaged limbs is one of the main reasons for continued interest in a better understanding of working of the brain. Deep brain stimulation and visual implants are some treatments that benefit from the partial understanding of the brain functionality.

Ability to monitor, record and process neural signals is thus essential to the development of treatments for any of the numerous neural disorders. Neuroprosthetics devices and Brain Machine Interfaces (BMIs) are increasingly playing a vital role in helping patients with severe motor disorders achieve a better lifestyle by enabling direct interface to the central nervous system. Thus far, it has only been possible to monitor limited brain activity observable above the scalp, however, scientists need to explore cortical signals observable only under the skull to find solutions to above mentioned problems. Any advancement in the ability to monitor those signals will help millions of individuals with different neural disorders.

Recent advances in biomedical research have enabled recording of neural signals from large populations of cortical cells under the skull. However, lack of a wireless data acquisitioning system hinders their use in brain machine interfaces for freely moving subjects. The research presented in this dissertation addresses this problem by developing an implantable hardware employing on-chip signal processing and enabling wireless data telemetry for further processing of neural signals.

1.2 Neural Recording Techniques

There are different ways of recording neural activity currently in use, Electroencephalography (EEG), Magnetoencephalography (MEG), functional Magnetic Resonance Imaging (fMRI) and Positron Emission Tomography (PET), all of which are non-invasive. In several situations it is desirable to have high spatial resolution as well as high time resolution for better understanding of intra-cranial connectivity and neural activity than provided by the above mentioned recording techniques. This can only be achieved with highly invasive methods and introduction of strip electrodes or microelectrode arrays. Electrocorticography (ECoG) and Intracranial EEG are two such invasive techniques. An overview listing pros and cons of these techniques can be found in the Appendix.

1.2.1 Intracranial EEG

Intracranial EEG has shown the capability of recording activity of individual cells, and has recently been made possible by the use of microelectrode arrays. The signals are recorded by implanting electrodes into the cortex. Recording the activity of cortical neurons with microelectrode arrays was shown to be essential to quantify the degree of involvement of each neuron in encoding movement parameters. Use of microelectrodes allows extremely high spatial and temporal resolution and allows recording from single cells at arbitrary sampling rates, thereby enabling decoding of the neural signals and eventually control of artificial limbs [1].

There are several challenges associated with this highly invasive procedure. Firstly, as with ECoG, a surgery is required to implant the electrode array and supporting electronics into the cortex. Secondly, such high resolutions on the order of microseconds and micrometers result in tremendous amount of data being generated. Thus far, clinical trials have used wires that extend from the skull to the external machines to record the signals. However, enabling wireless telemetry of these signals will allow free movements to the subject in addition to long term or permanent implants without any chance of infection in the scalp. Such BMIs show great promise in many biomedical applications.

1.3 Characteristics of Neural Signals

Neural signals recorded using high density microelectrode arrays can be subdivided into two kinds of potentials, the action potentials, also known as neural spikes, and the Local Field Potentials (LFP). A summary of salient features of these signals is mentioned below.

1. Action Potentials

Action potentials are the primary signal of interest for implantable neural signal processing systems. They are generated at the cell membrane of the neuron but can be recorded if the electrodes are a few microns away from the cell. Each electrode can thus receive signal from multiple neurons. These neural spikes have a wide frequency bandwidth ranging from 100 Hz to 10 KHz, while a typical spike lasts for about 1.5 ms. Amplitude of these spikes is about 100 mV relative to extracellular fluid at the cell membrane, however it reduces to within a range of 50-200 μV when measured using an electrode a few micron away. The recording electrodes also see a relatively large DC offset of 50 mV at the input. A typical neuron fires between 10 to 100 spikes per second when active.

2. Local Field Potentials

Local Field Potentials (LFP) are low frequency signals less than 100 Hz which are generated as a result of electrical discharges from many neurons at the same time. Thus where action potentials are the signals from individual neurons, the local field potentials are the superimposed signals from a large collection of neurons. The amplitude of LFP is about 5 mV. The LFPs contain useful information about collective activity of the signal, however this information does not change drastically from one electrode to another in a microelectrode array based data acquisition. All the electrodes in the microelectrode array are likely to record the same signal for LFP. As a result the signal acquired by the array records the high frequency action potentials superimposed with the low frequency LFP where the LFPs can be up to a few thousand times stronger in amplitude than the action potentials. Since action potentials are the primary signal of interest, the LFPs have to be filtered out to allow any further signal processing.

1.4 Source Separation and Spike Sorting

Each electrode can acquire signals from more than one neuron based on its location relative to the cells. The acquired waveform is thus a superposition of a number of spike trains from as many neurons. Since complex brain processes are a result of activity in a large number of neurons, it becomes necessary to be able to separate the activity of each neuron from the other to enable understanding of inter-neuron functional and physical connectivity.

The process of separating spikes from different neurons is called spike sorting. Spikes from these neurons may or may not have different amplitudes based on location of the electrode, however, they almost always tend to have slightly different shapes which are unique to each neuron. This difference in shape can be used to separate spikes from one neuron from the spikes of others. Two parameters are deemed important by the neuroscientists to enable further research, the shape of the spike and its time of arrival. Thus the task of any neural data acquisition system is to maintain those features of the waveform that enable spike sorting.

Fig. 1.1 shows a typical intracranial EEG recording using microelectrodes. Band pass filtering has been applied to remove LFP and high frequency noise. The inset shows a single action potential which is about 1.5ms long. Action potentials from different neurons have slightly different shapes owing to their relative position to the recording electrode. The noise on the signal is not necessarily white Gaussian.

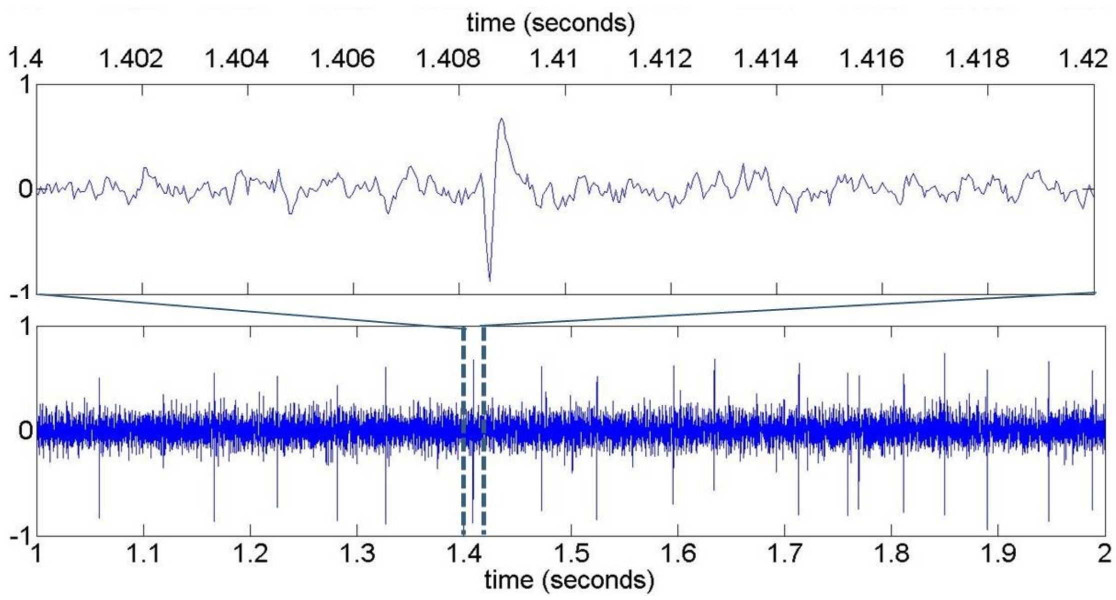


Fig. 1.1. A typical neural signal recording using a microelectrode after band pass filtering to remove LFP and high frequency noise. For Interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

1.5 Challenges Associated with Array Based Recording

There are various challenges associated with wireless recording of multi-channel neural signals using implanted microelectrode arrays. Any system designed to enable wireless neural recording must address these issues.

1. **Data Bandwidth**

One particular challenge with BMI technology is the need to transmit the high bandwidth neural data from the implanted device to the outside world for further analysis. For example, a typical recording experiment with a 100-electrode array sampled at 25 kHz per channel with 12-bit precision yields an aggregate data rate of 30 Mbps, which is well beyond the reach of state-of-the-art wireless telemetry links for biological applications.

2. **Implant Size**

Another significant challenge is the need to fit implanted circuitry within ~1 cm^2 for the entire signal processing system because of surgical reasons. Large size of the system results in physical implantation challenges. There is a small clearance between the cortex and the skull, thus small size is needed to minimize tissue trauma and displacement.

3. **Wireless Power**

Power supply to implantable system is also one of the major challenges as use of a battery provides a highly invasive yet short term solution as batteries need to be replaced and recharged. Inductive power transmission can be used as an alternative to deliver power to an implanted system over short distances from the power source. However, inductive power transmission has its limitations in terms of maximum deliverable power and interruptions in power supply because of coil misalignment due to motion in freely moving subjects.

4. **Implant Power Density**

High power dissipation of the system can result in damage to the neighboring neurons and cortical tissue, in effect killing the neurons that are being recorded from. Studies show that an increase in temperature over one degree Celsius can result in tissue damage. A power density of 0.58 mW/mm^2 results in one degree rise in temperature in the tissue. Thus any design of implantable system must conform to the dissipated power density limitation.

5. **Processing capability**

The above mentioned challenges also directly influence the processing capability of the implantable system as more computations translate to higher power dissipation whereas more memory elements require larger area in silicon. These stringent requirements force the use of application specific integrated circuits over general purpose processors.

6. **Signal Noise**

The recorded signal contains various kinds of noise. Apart from the recording and thermal noise associated with the electronics the acquired signal also contains background noise. Background noise is in effect the attenuated signal from neurons located far from the recording electrode. As a result the background noise is correlated to the signal, occupies the same spectrum as the signal, and also contains similar features as neural spikes. However, the background noise has smaller amplitude than the signal of interest. Cross channel interference between recordings from different electrodes is another source of noise, as well as the interference between the analog, the digital and the wireless transceiver blocks. Physiological noise as a result of LFP, muscle EMG or cardiac activity also damages the signal quality.

1.6 Contemporary Research Groups

1.6.1 Review

The problem of acquiring neural signals from the cortex using an implantable system and transmitting them to external processing units is being explored by a number of research groups. These groups tackle similar problems with slightly different goals and approaches.

1. **Duke University, Durham, North Carolina**

Absolute thresholding is used in time domain to detect and extract spikes [2, 3]. Spikes can be extracted and transmitted for only a subset of channels; as a result spike sorting is possible for only a few channels in parallel. Streaming raw data for only one channel can be transmitted in parallel to the extracted spikes. Another mode transmits the spike count in time bins where a packet is transmitted every 50ms. This loses information about spike shapes and their occurrence times. A commercial 1 Mbps transceiver is used for data transmission.

2. **University of Utah, Salt Lake City, Utah**

Absolute thresholding used in time domain to detect spikes in the analog front end before digitization [4, 5]. Thresholds are programmable for individual channels. A ‘one’ is sent if a spike is detected, a zero is sent otherwise. As a result spike shapes are not preserved. In addition, one channel of raw data (sampled at 15.6KSps) can be transmitted in parallel. Spike sorting not possible on other channels. Data, power and clock are transmitted over the inductively coupled wireless link. A 96 channel prototype using 100 amplifiers with 10-bit ADC is presented, consuming 8mW power.

3. Stanford University, Stanford, California

Raw data can be transmitted for a single channel sampled at 15.6KSps. Recording system is wireless but not implantable [6]. Salient feature is the transmission range of up to 4 meters away from the subject at a power consumption of 63.2mW with a sampling frequency of 15.6KSps. The design has been reported to be scalable and suitable for an increase in the number of channels in future.

4. Ecole Polytechique de Montreal, Montreal, Canada

Spike detection is implemented by applying a threshold to a nonlinear energy operator after digitization [7, 8]. This approach is reported to have better detection results compared to spike detection using thresholding in time domain. This operator uses temporal as well as instantaneous spectral information to detect spikes. 64 8-bit data samples (2ms window @ 30KSps sampling rate) are transmitted for each detected spike. System is reported to handle 16 channels in parallel. The extracted spike samples retain necessary features to allow spike sorting. Data overflow is reported when burst activity is encountered in the neurons.

5. University of Michigan, Ann Arbor, Michigan

Absolute threshold used in time domain after digitization to detect spikes [9-12]. The system can transmit raw data for one channel in scan mode. In a separate mode binary values are transmitted to indicate presence or absence of a spike. Spike shapes are not preserved. Prototype designed to handle 64 channels, where later versions may include neural stimulation in addition to recording. Clock, power and data are communicated through an RF link.

6. Brown University, Providence, Rhode Island

The design uses broadband infrared transmission using vertical- cavity surface-emitting laser (VCSEL) diode to transmit raw data from 16 channels simultaneously [13]. No data rate is provided. Power can be delivered through an inductive RF link or IR link. An important drawback is that the alignment of IR transmitter and receiver is critical for successful broadband transmission.

7. University of California, Santa Cruz, California

Raw data from 128 channels sampled at 40KSps with 9 bit resolution is transmitted using UWB 90Mbps transmission link [14, 15]. Ex vivo experiment results have been presented. The system allows one channel to be processed on-chip to extract spike features for classification using on-chip energy based spike detection, noise shaping filter and min-max feature extraction. No results are provided for the quality of spike sorting achieved based on this feature set. Apart from off chip training required to set detection thresholds, the noise shaping filter (32 tap) also needs offline training and on-chip memory for storing coefficients. A programmable front-end can support various types of biological signals.

Various other groups provide parts of the system e.g. amplifiers or ADC or simply transmit single channel recordings. Some groups present multi-channel systems with discrete components that are not implantable and thus do not have stringent power and bandwidth requirements.

1.6.2 Drawbacks

All of the above mentioned systems try to reduce the amount of data that needs to be transmitted out of the brain. Almost all systems use programmable thresholds to detect spikes; which means offline training is necessary to determine these thresholds. Some systems maintain spike shapes to allow off chip spike sorting while others use binary values to represent the presence of spikes.

Following are some of the drawbacks of these approaches:

1. Most other designs use spike extraction to preserve spike shapes but are constrained to small number of channels because of limited bandwidth. Data overflow is reported when burst activity is encountered.

2. Some systems use binary values to represent the presence or absence of a spike but lose all information that enables spike sorting at later stages.

3. Most designs use thresholding in time domain to detect the spikes. This can lead to a large number of false positives in low SNR conditions, or a large number of spikes can be missed when liberal thresholds are used.

4. False positives result in significant bandwidth loss if complete spike extraction is used, however, these can be detected and ignored in offline processing. Whereas, a false positive will remain undetectable if binary values are used to represent the presence or absence of spikes even in later stage of processing.

5. None of the systems takes into account the unreliability of the communication channel. Although the maximum data rates are sometimes mentioned, the performance of the transceivers has not been discussed thoroughly. Neither any assumptions about the channel have been stated nor any measure of data transfer accuracy has been established.

6. Transmission of raw data has been made possible by the use of UWB transceivers, however, their feasibility for use in freely moving subject is still under investigation since precise alignment of infrared receiver is not possible when the subject is moving.

1.7 Research Goals and Approach

1.7.1 Goals Statement

The goal of this thesis is to solve some of the major obstacles impeding the advance of wireless neural implants. The problems addressed in this thesis lie in the realm of low-power system design and can be solved by developing highly efficient algorithms and hardware suitable for implantation, establishing a pathway to the design of modern neuroprosthetics devices. Based on a review of the state of the art neural recording systems it has been identified that this goal can be achieved if the following objectives are met.

First of the problems is the overwhelming amount of data generated by the microelectrode array resulting in a data bandwidth bottleneck. The objective is to devise a system to enable some degree of control over the amount of data that needs to be transmitted without compromising the information contained in the array of signals. The second problem addressed in this thesis is the unreliable nature of the wireless communication channel across the skin tissue. The objective is to maximize the data throughput and enable unfailing yet low-power communication of bidirectional data between the implanted device and the external world. The final objective is the integration of these respective solutions into a unified energy efficient implantable system for seamless operation of these functions.

Although the motivation for this thesis comes from the field of neuroprosthetics, the aim is to devise solutions that are applicable to a wide range of implantable applications.

1.7.2 Approach

To achieve the above mentioned goals and in contrast to the existing approaches listed above, this research will explore the realization of an implantable low-power on-chip signal processor to acquire and compress the multi-channel neural data before wireless transmission to the external units. The use of high level processing on-chip for lossy as well as lossless compression would allow efficient use of the bandwidth. Configurability built into the system would allow various degrees of signal information retention. The communication channel across the skin tissue would be studied and a communication protocol and hardware will be devised to maximize the data throughput for streaming data communication. Finally, a scheme will be developed for ultra low power hardware-constrained signal compression for scalability and suitability to handle simultaneous recording from thousands of channels.

1.7.3 System Architecture

Based on the above-mentioned system approach, Fig. 1.2 gives the conceptual diagram of the proposed wireless neural recording system. The system consists of two main modules, the implantable Neural Interface Node (NIN), and the extra-cranial Manager Interface Module (MIM).

The NIN is implanted on the cortex under the skull and is connected to one or more microelectrode arrays. The captured signals are amplified and digitized in the analog front end, which relays them to the neural processing unit (NPU). The NPU is responsible for the overall operation of the NIN, including the different modes of operation, compression of signals and error-free bi-directional communication between the NIN and the MIM. The data is transmitted through an RF wireless transceiver, which also receives commands from the MIM. The

NIN is a battery-less module, which receives power and clock through an inductive link from the MIM.

The MIM is placed outside the scalp opposite to the NIN. This provides the best alignment between the NIN and MIM coils for power and data transfer. The two modules are only a few millimeters apart from each other, however, in freely moving subjects the NIN on the cortex cannot remain fixed and thus misalignment of coils is expected during the experiment. The MIM receives the data from the NIN and transmits it to the external processing unit (EPU). The EPU issues appropriate commands to the MIM, which relays them to the NIN to change the operation of the system.

Fig. 1.3 shows the detailed block diagram of the NIN. The neural signal is acquired through the microelectrode array and then amplified and band pass filtered before conversion to digital words. The signals are sampled in a time-multiplexed fashion at a rate of 25KSamples per second per channel. The function of this front-end analog block is controlled by an analog interface controller. The data is then passed to the compression engine, which is responsible for the lossy as well as lossless compression of the multi-channel neural data. Finally, the data is transmitted to the MIM through a communication controller, which handles the outgoing data and incoming commands over a half-duplex wireless link.

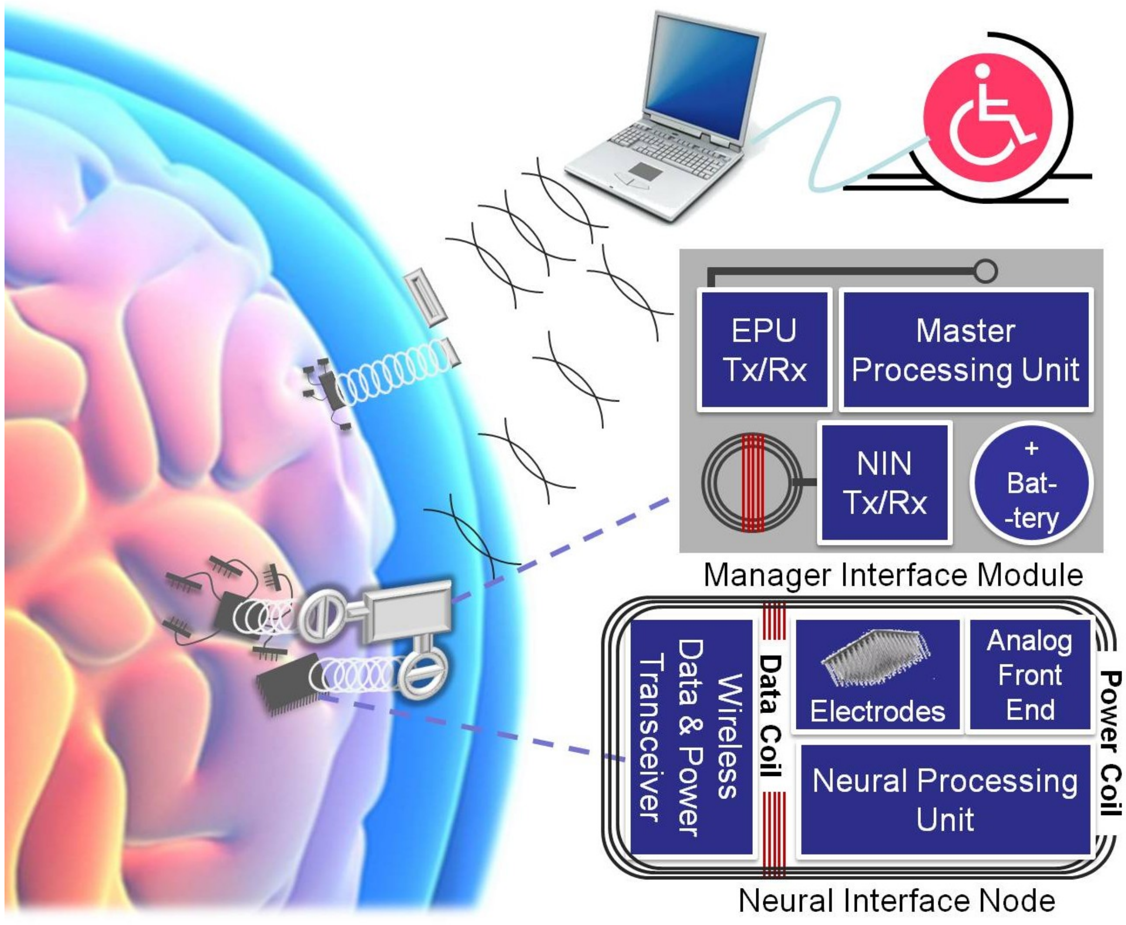


Fig. 1.2. Concept diagram for the proposed wireless neural recording system showing implantable neural interface node and the extra-cranial manager interface module.

The wireless transceiver uses RF circuits to handle bi-directional data transfer. In addition, it is responsible for power transfer from MIM to NIN, as well as extraction of system clock signal from the wireless carrier frequency. A global controller on the NIN unites all these components into a working system. Its most important task is to interpret the commands from the MIM and change the operation of the NIN accordingly. It is also responsible for managing the power and clock for each of the blocks.

All the digital components of the NIN are shown in green in Fig. 1.3. Focus of this thesis is the design and fabrication of these components and the study of algorithms involved.

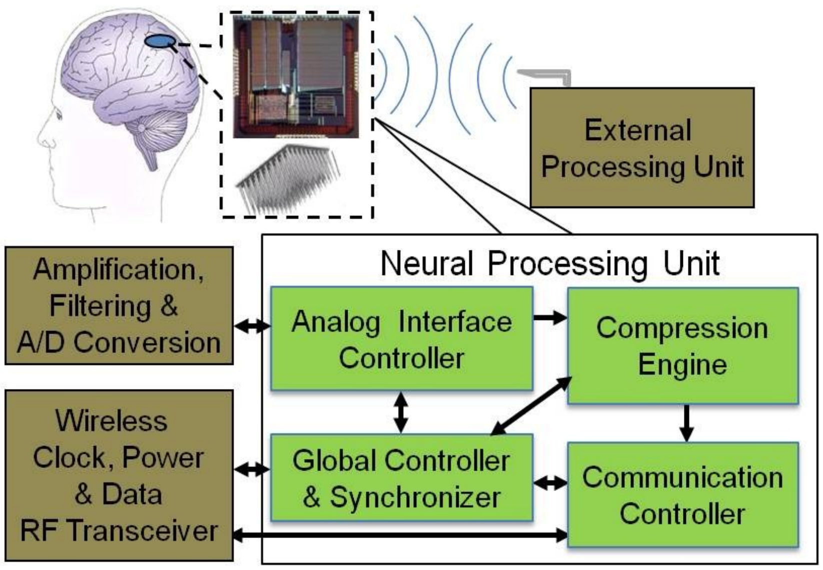

External Processing Unit
Amplification, Filtering & A/D Conversion
Neural Processing Unit
Analog Interface Controller
Compression Engine
Wireless Clock, Power & Data RF Transceiver
Global Controller & Synchronizer
Communication Controller

Fig. 1.3. Block diagram of the implantable neural data compression engine and its position within an implantable neural recording system.

1.8 Challenges

To following challenges need to be managed to achieve the goal and its objectives outlined above.

1. The compression engine has been identified as the most computationally intensive of all the blocks. The main challenge is to identify and model different algorithms and architectures and to achieve the lowest area-power consuming implementation. Intelligent hardware design is required

for area-power efficient circuit. This requires understanding of the hardware requirements of various compression algorithms.

2. A communication protocol is required to be designed that enables error-free half-duplex data communication of the streaming data while maintaining a high data throughput. The challenge is posed by the unavailability of any information about the transceiver performance and channel error profile. Channel characteristics need to be measured and transceiver performance needs to be analyzed and a protocol able to handle this communication needs to be developed.

3. With the rapid increase in the number of channels in a microelectrode array, the challenge is to design an ultra constrained set of features that allows spike sorting for thousands of channels simultaneously.

4. All the components including the analog interface controller, the compression engine and the communication controller have to come together into a single highly configurable system and work seamlessly. The challenge is the design of the system within the ‘implantability’ limitations, i.e. the area, power and power density limitations discussed earlier.

2 Hardware Realization of DWT for Neural Recording

2.1 Discrete Wavelet Transform for Neural Recording

For implanted microelectrode arrays, the power required to wirelessly transmit raw data to extra-cranial processing units is prohibitively large. Likewise, the hardware required to perform full neural data analysis is too complex to be implemented within an implanted system. Data compression before transmission is an attractive alternative, if it can be performed with minimal hardware resources. Discrete wavelet transform (DWT) is a very effective method for compressing neural data [16, 17].

Discrete wavelet transform enables time-frequency domain analysis of neural signals, which manifests more information than either the time domain or the frequency domain analysis with selectable tradeoff between time and frequency resolutions. In addition to the co-existence of time and frequency in wavelet domain, it also provides the flexibility of choice between large set of basis functions, which can result in different numbers of high-energy coefficients after the transform.

For an appropriately selected wavelet basis function, the resulting non-zero DWT coefficients give a sparse representation of the signal that greatly reduces the power required to upload data. The useful information is mostly contained in the short transients, or spikes, above the noise level that result from the activity of an unknown number of neurons. It can be observed that the scarcity introduced by the DWT compaction property enables very few “large” coefficients to capture most of the spikes’ energy, while leaving many “small” coefficients attributed to noise. This property permits the later ones to be thresholded [18], yielding the denoised signal.

2.1.1 Background

VLSI implementation of the discrete wavelet transform (DWT) has been widely explored in the literature as a result of the transform efficiency and applicability to a wide range of signals, particularly image and video [19, 20]. These implementations are generally driven by the need to fulfill certain characteristics such as regularity, smoothness and linear phase of the scaling and wavelet filters, as well as perfect reconstruction of the decomposed signals [21]. In some applications, it is desirable to meet certain design criteria for VLSI implementation to enhance the overall system performance. For example, minimizing area and energy consumption of the DWT chip is highly desirable in wireless sensor applications where resources are very scarce. In addition to miniaturized size, minimizing power dissipation is strongly sought to minimize tissue heating in some biomedical applications where the chip needs to be implanted subcutaneously.

With few exceptions, recent efforts to optimize DWT hardware have concentrated on increasing throughput at the expense of area and power [22, 23]. In contrast, our work tries to identify an optimal implementation of the DWT architecture where chip area and power have priority over speed [24]. The case of computing the DWT for high throughput streaming data has not been fully explored [25]. It has been argued that a lifting scheme [26] provides the fewest arithmetic operations, allowing larger savings in power consumption but at the expense of longer critical path than that of convolution-based ones [26]. Recent work by Huang *et al.*[27] focused on analyzing DWT architectures with respect to tradeoffs between critical path and internal buffer implementations. Such critical path can be shortened using pipelining with additional registers or using a so-called *flipping* structure with fixed number of registers [28]. The *B-spline* approach [29], on the other hand, requires fewer multipliers than lifting, replacing them with adders that may permit a smaller chip area [23]. Nonetheless, most of the reported hardware approaches focus on computational speed and do not adequately address severe power and area constraints. By comparing with other implementations of the DWT in this chapter, we demonstrate that the appropriate compromise among power, size and speed of computations is achieved with a sequential implementation of integer arithmetic lifting approach.

2.1.2 Wavelet Representation of Signals

The classical, convolution-based, dual-band DWT of a given signal involves recursively convolving the signal through two decomposition filters *L(z)* and *H(z)* and decimating the result to obtain the approximation and detail coefficients at every decomposition level *j*. These filters are derived from a scaling function and a wavelet function that satisfy subspace decomposition completeness constraints [16]. A typical FIR low pass and high pass 3-tap filter is expressed as

$$L(z) = l_0 + l_1 z^{-1} + l_2 z^{-2} \tag{2.1}$$

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} \tag{2.2}$$

So that the output approximation and detail coefficients a^j and d^j respectively at the j^{th} level can be computed as

$$a^j(i) = \sum_{k=0}^{K-1} a^{j-1}(2i-k) l_k \tag{2.3}$$

$$d^j(i) = \sum_{k=0}^{K-1} a^{j-1}(2i-k) h_k \tag{2.4}$$

where *K* is the number of filter taps. The obtained coefficient vectors a^j and d^j are $N/2^j$ -dimensional, where *N* is the length of the original single-channel input sequence. Equations (2.3) and (2.4) describe the original pyramidal algorithm reported by Mallat [16]. Reconstruction of the original sequence from the DWT coefficients is achieved through

$$a^{j-1}(2i-1) = \sum_{k=0}^{(K/2)-1} a^j(i-k) \underline{l_k} \tag{2.5}$$

$$d^{j-1}(2i-1) = \sum_{k=0}^{(K/2)-1} a^j(i-k) \underline{h_k} \tag{2.6}$$

where $\underline{l_k}$ and $\underline{h_k}$ are the coefficients of the synthesis filters, respectively.

2.1.3 Basis Selection

For near-optimal data compression, a wavelet basis needs to be selected to best approximate the neural signal waveform with the minimal number of data coefficients. A compromise between signal fidelity and ease of hardware implementation has to be made. A near-optimal choice was proposed in [17] from a compression standpoint and demonstrated that the *biorthogonal* and the *symlet-4* wavelet functions are advantageous over other wavelet basis families for processing neural signals. From a hardware implementation viewpoint, the *symlet-4* family has much smaller support size for similar number of vanishing moments compared to the biorthogonal basis [30]. Fig. 2.1 displays the wavelet basis function of symlet-4.

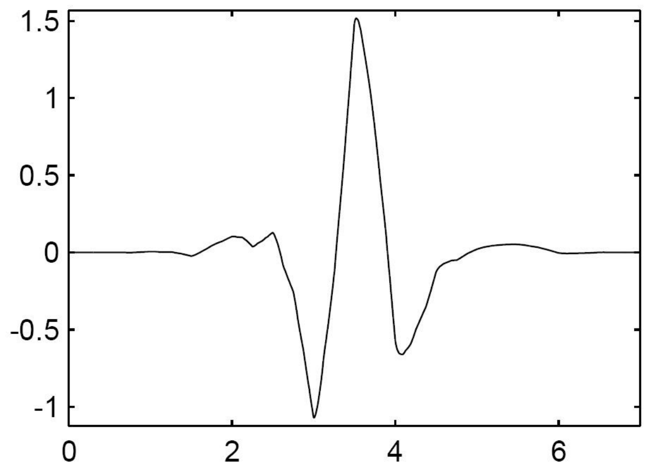


Fig. 2.1: The 'symlet-4' wavelet basis function.

Therefore, we will focus on symlet basis with order 4 throughout the thesis. The high pass and low pass equations of the discrete 'symlet-4' basis are given by

$$\begin{aligned}
&L(z) = -0.076 - 0.030z^{-1} + 0.498z^{-2} + 0.804z^{-3} \\
&+ 0.298z^{-4} - 0.099z^{-5} - 0.013z^{-6} + 0.032z^{-7} \\
&H(z) = -0.032 - 0.013z^{-1} + 0.099z^{-2} + 0.298z^{-3} \\
&- 0.804z^{-4} + 0.498z^{-5} + 0.030z^{-6} - 0.076z^{-7}
\end{aligned} \tag{2.7}$$

2.2 Algorithms for DWT

Mallat's algorithm [16] has been used traditionally for evaluating the wavelet transform of a given signal and involves recursively convolving the signal through two decomposition filters *H(z)* and *L(z)* and down-sampling the result to obtain the approximation and detail coefficients at every decomposition level [25]. Fig. 2.2 shows the multi-level decomposition of input data into DWT coefficients where the blocks represent the filtering and down-sampling operations. The second half of the Fig. 2.2 shows reconstruction of the signal from transmitted coefficients. This chapter concentrates on the decomposition portion that must be implemented within the implantable device, whereas the reconstruction, if needed [31], can be done extra-cutaneously where computational resources are not scarce.

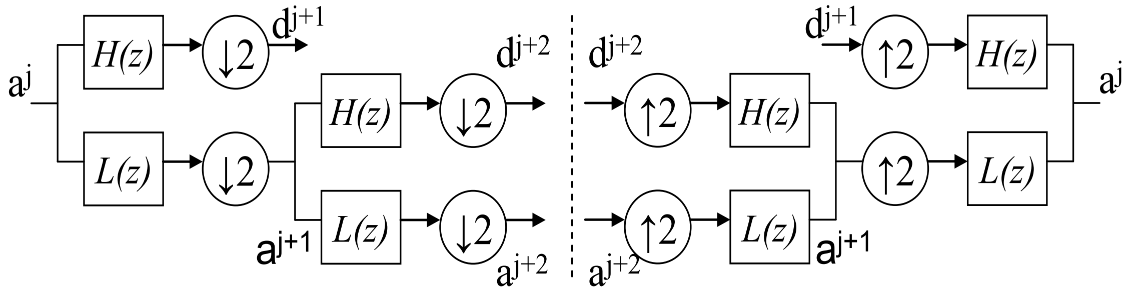


Fig. 2.2: The multilevel discrete wavelet transform decomposition and reconstruction.

2.2.1 Lifting Factorization

Any finite impulse response (FIR) wavelet transform can be expressed in terms of lifting steps [25]. Optimizing the conventional DWT algorithm, the lifting scheme analysis is described with a sequence of “predict” and “update” filters, S_n and T_n, respectively, which form the lifting steps as evident in (2.2). Each lifting step is generally a one or two tap filter, which is computationally more efficient than longer, multiple tap filters.

$$P(z) = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} \prod_{i=1}^{n} \left\{ \begin{pmatrix} 1 & S_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ T_i(z) & 1 \end{pmatrix} \right\} \tag{2.8}$$

Mathematically, lifting DWT is implemented by splitting the data into even and odd samples and applying the S_n and T_n filters simultaneously as shown in Fig. 2.3. The data at each step, after applying the filters, is labeled as $f, f_1, \ldots, f_n$, and $h, h_1, \ldots, h_n$ for update and predict steps, respectively. The coefficient values B_0 to B_7 are listed in Table 2.1. The last step in (2.8) is a multiplication by a scaling factors K_1 and K_2, where $K_1 = 1.571$ and $K_2 = 0.637$. Because this multiplication can be omitted and correspondingly taken care of at the decoding side, the scaling factors have been eliminated in the following discussion. In the case of a multi-level decomposition, the outcome at an arbitrary decomposition level *j* is obtained as the approximation *a* and detail *d*. The former is fed back for the next level of DWT decomposition if required, whereas the latter is stored or transmitted as required.

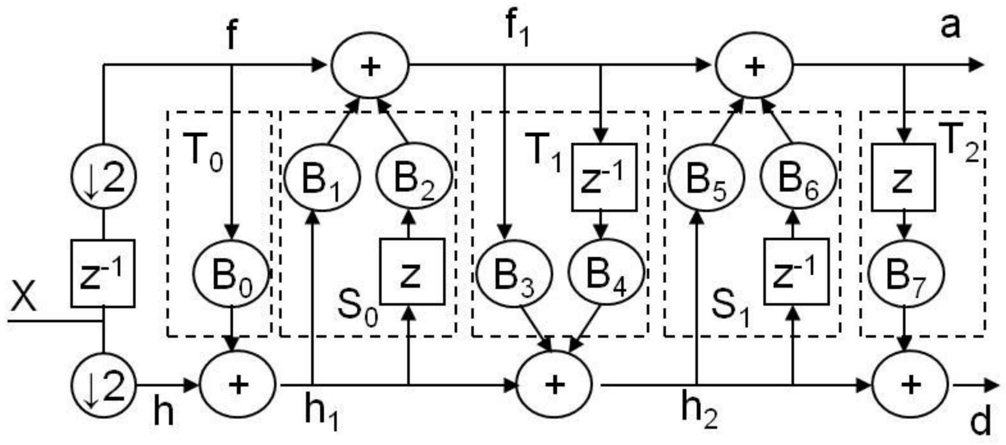


Fig. 2.3: Lifting scheme for DWT.

B_0 to B_7 in Fig 2.3 and Table 2.1 are the coefficients of the lifting and update filter steps for the factorization of symlet-4 filters.

Table 2.1: Coefficients of the symlet-4 lifting factorization.

Coeff.	B_0	B_1	B_2	B_3
Value	0.3911	-0.1243	-0.3392	-1.4195
Coeff.	B_4	B_5	B_6	B_7
Value	0.162	0.4312	0.1459	-1.0492

2.2.2 B-Spline Factorization

It has been shown [23, 29] that wavelet filters can be decomposed as

$$H(z) = (1+z)^M Q(z) \cdot h_o$$
$$L(z) = (1-z)^M R(z) \cdot l_o \tag{2.9}$$

where $(1 \pm z)^M$ is called the B-spline factor and $Q(z)$ and $R(z)$ are the distributed factors. Application of this decomposition to the symlet-4 filters in (2.1), results in $H(z)$ and $L(z)$ given by (2.10).

$$H(z) = (1+z^{-1})^4(1 + A_0 z^{-1} + A_1 z^{-2} + A_2 z^{-3}) \cdot C_1 \tag{2.10}$$
$$L(z) = (1-z^{-1})^4(1 + A_3 z^{-1} + A_4 z^{-2} + A_5 z^{-3}) \cdot C_2$$

where C_1 = -0.076 and C_2 = -0.032 are the gain constants. The coefficients A_0 to A_5 resulting from B-spline factorization of symlet-4 filters are listed in Table 2.2. Expanding the B-spline polynomials in (2.10) results in (2.11)

$$(1 \pm z^{-1})^4 = (1 + 6z^{-2} + z^{-4}) \pm (4z^{-1} + 4z^{-3}) \tag{2.11}$$

which allows all the multiplications required in the B-spline portion of (2.10) to be replaced by less computationally demanding shift and add operations. The polyphase decomposition can therefore be performed on the distributed parts *R(z)* and *Q(z)* [29]. This is achieved by splitting the distributed parts into odd and even components $R_e(z)$ and $R_o(z)$, $Q_e(z)$ and $Q_o(z)$, respectively. For example, the low-pass even distributed part can be represented as $R_e(z) = 1 + A_2 z^{-2} + 0.z^{-4}$ and likewise for the remaining components. Accordingly, the computation gain in the B-spline method is a reduction in the number of floating point multiplications at the expense of more additions [23].

Table 2.2: Coefficients of the B-spline factorization.

H(z)			L(z)		
A_0	A_1	A_2	A_3	A_4	A_5
-3.607	1.867	-0.425	4.391	8.485	2.339

Fig. 2.4 is a Figurative description of DWT implemented using B-spline optimization on the symlet-4 filters. Note that Fig. 2.4 contains two distinct portions. The first portion (left) splits the input stream and implements B-spline portion of (2.10) where the multiplications are implemented using shifters and adders. The model for these shift-and-add multiplications is shown in a blowup box in Fig. 2.4. The second portion (right) implements the distributed factors, where multiplications require a hardware multiplier.

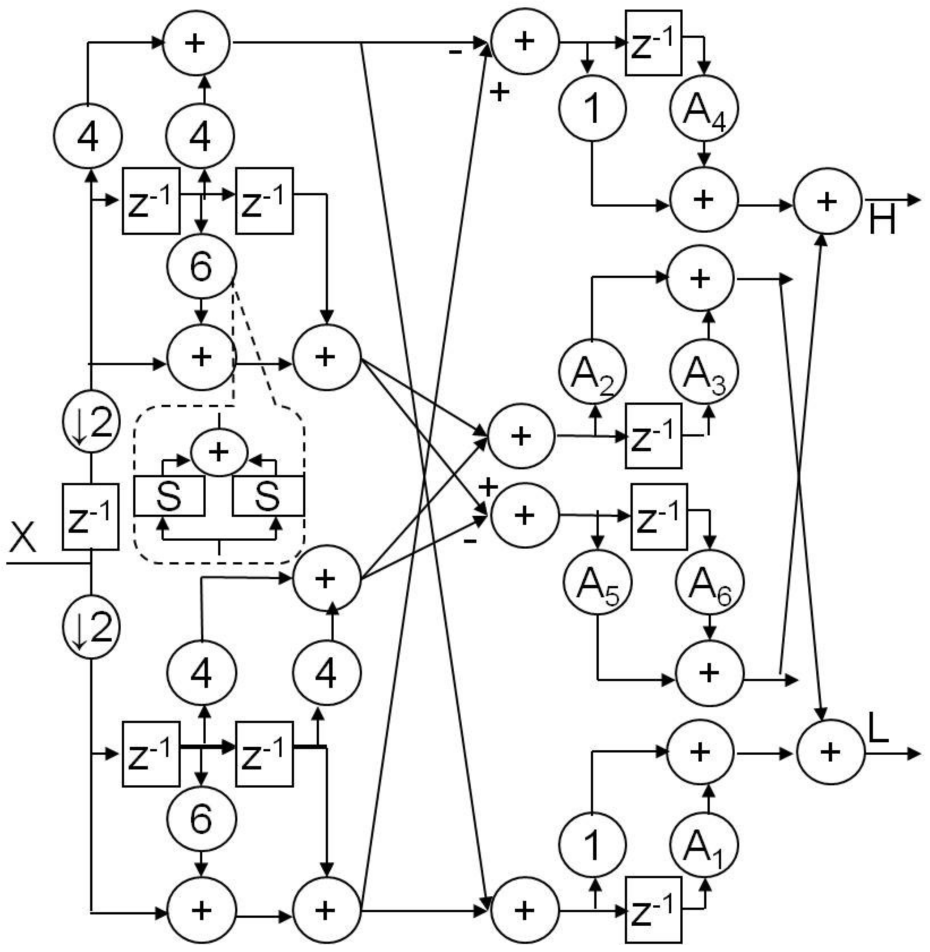


Fig. 2.4: B-spline architecture for computing the symlet-4 based DWT.

2.2.3 Algorithmic Comparison

According to Fig. 2.2, the downsampling operation follows filtering, but this removes half of the samples just calculated and results in wasted computation energy. Lifting and B-spline are both motivated by the idea of moving the downsampling stage before the filtering stage, so as not to compute results that would eventually be thrown away. The resulting filter-downsample operation allows polyphase decomposition [25] to be used, which in this case involves splitting of the input signal into even and odd samples and filtering them with even and odd filter taps, respectively, to minimize hardware resources. This approach results in almost half the number of calculations compared to the standard convolution-based filtering. Both, lifting and B-spline use polyphase decomposition.

Fig. 2.3 illustrates that lifting does not preserve causality, and thus implementation of some of the resulting update and predict filters introduce latency between the input and corresponding output of the DWT block. Although this implementation does not produce results strictly in real-time, this lag is not critical in neuroprosthetics applications where the biologically relevant sampling rate is 25 KHz [32]. B-spline does not have any causality issues, and the only delay between the input and its corresponding output is the computation delay.

Table 2.3 compares the computational requirements of convolution, B-spline and lifting based DWT implementations. The standard convolution based filter requires 16 multiplication and 14 additions. The lifting scheme reduces computation requirements to only eight multiplications and eight additions. For B-spline factorization, the total multiplications required is reduced to 12 and 16 additions are required. Out of these 12 multiplications, six of the multiplications can be implemented using shift-and-add operations. Fig. 2.4 and (2.11) show that four of these multiplications require only shift operations to multiply by the binary factor of four. The remaining two multiplications have six as the multiplicand and require two shifts and one addition operations, as shown in the blowup in Fig. 2.4. As a result, the minimized B-spline implementation requires only six multiplications and 18 additions, as shown in parentheses in Table 2.3. Based on Table 2.3, Relative to lifting, B-spline requires two fewer multiplications at the expense of ten more additions for one level of decomposition. Nevertheless, as the detailed low-power/area DWT implementation below will show, any benefit to B-spline is diminished for multilevel multichannel decomposition. The input data was quantized to 10 bit while the filter coefficients were quantized to 6 bits in sign magnitude form based on an analysis of signal integrity. A detailed discussion can be found in the Appendix.

Table 2.3: Comparison of required computation.

	Convolution	B-spline	Lifting
Multiplications	16	12(6)	8
Additions	14	16(18)	8

2.3 Minimized Hardware Implementations

2.3.1 Lifting Implementation

The lifting scheme is a sequence of predict S_n and update T_n filters as shown in Fig. 2.3. For the symlet-4 basis, the set of lifting filters can be described by

$$\begin{aligned}
P_0 &= h_0 + B_0 f_0 \\
Q_{-1} &= f_{-1} + B_1 P_0 + B_2 P_{-1} \\
R_{-1} &= P_{-1} + B_3 Q_{-1} + B_4 Q_{-2} \\
a_{-1} &= Q_{-1} + B_5 R_{-1} + B_6 R_{-2} \\
d_{-2} &= R_{-2} + B_7 a_{-1}
\end{aligned} \tag{2.12}$$

where *a* and *d* are approximation and detail results, and *P, Q*, and *R* are intermediate results. B_i are the eight constant filter coefficients with index *i* ranging from zero to seven. Each equation represents one filter step, and values of the filter coefficients are given in Table 2.1. The subscripts of variables in (2.12) represent the time sample, where 0 represents the current sample, -1 represents the previous sample, and so on [34, 35].

Analysis of the lifting factorization and resulting equations in (2.12) shows that there is a noticeable regularity in the required computations. All arithmetic operations of the lifting implementation of symlet-4 can be expressed in the general form of

$$W = X + B_i Y + B_j Z \tag{2.13}$$

which permits all filter steps to be written as two multiplications and two additions, with one of the coefficients set to zero for the first and last step in (2.12). This regularity can be exploited to minimize hardware by implementing a standard computation core that executes (2.13), as shown in Fig. 2.5. This single hardware block can be repeatedly used to perform all the computational steps in (2.12) sequentially [36, 37]. Sequential reuse of the same hardware reduces the area required by the overall DWT block without impacting performance in this low bandwidth application [36]. The hardware block in Fig. 2.5 will be called the lifting computation core (CC_L) in the following discussion.

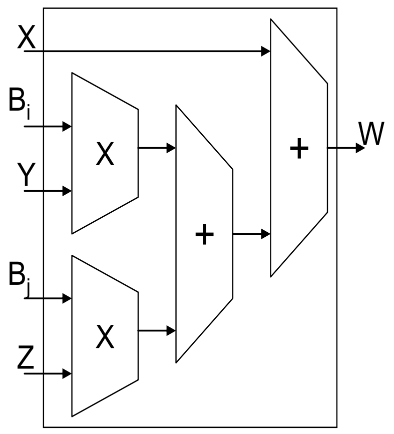


Fig. 2.5: Computation core for lifting DWT [12].

We have shown that fixed-point integer DWT computation using the symlet-4 wavelet with filter coefficients truncated to six bits and data values truncated to 10 bits gives performance comparable to floating-point calculations while significantly reducing computational demand [38]. A customized CC_L block has been designed to support fixed-point multiplications and additions. Using this CC_L to sequentially execute the steps in (2.12) requires five cycles to compute results for one input sample. The critical path for this CC_L is D_m+2D_a, where D_m is the delay of the multiplier and D_a is the delay of an adder.

Applying lifting to a wavelet basis does not guarantee the resulting filters to be causal. In the case of symlet-4, there are two filters that introduce non-causality into the system. These filters can be identified as S_0 and T_2 in Fig. 2.3 and are used for the calculation of Q and d in (2.12). Thus, the current calculations depend on availability of future samples. The corresponding calculations can be delayed to attain causality, resulting in a latency of three samples between input and output.

2.3.2 B-spline Implementation

To implement DWT using B-spline factorization in (2.9), equations corresponding to (2.12) need to be obtained. The time dependent equations governing the calculation of B-spline approximation and detail coefficients of symlet-4 wavelet filters can be expressed by

$$\begin{array}{cc}
J_0 = h_0 + 6h_{-1} + h_{-2} & L_0 = f_0 + 6f_{-1} + f_{-2} \\
K_0 = 4h_{-1} + 4h_{-2} & M_0 = 4f_0 + 4h_{-1} \\
N_0 = J_0 + K_0 & P_0 = J_0 - K_0 \\
Q_0 = L_0 + M_0 & R_0 = L_0 - M_0 \\
S_0 = N_0 + A_0 N_{-1} & T_0 = A_1 P_0 + A_2 P_{-1} \\
U_0 = Q_0 + A_3 Q_{-1} & V_0 = A_4 R_0 + A_5 R_{-1} \\
a_0 = S_0 + T_0 & d_0 = U_0 + V_0
\end{array} \tag{2.14}$$

where *a* and *d* are approximation and detail results and *J, K, L, M, N, P, Q, R, S, T, U,* and *V* are intermediate results. The constant coefficients A_0 to A_5 resulting from this factorization are listed in Table 2.2. The subscripts of variables in (2.14) represent the time samples, where 0 is the current sample, -1 is the previous sample, and so on.

The B-spline equations in (2.14) do not exhibit the regularity of lifting that allowed the general equation of (2.13) to be utilized. However, the required B-spline computations can be generalized into two expressions

$$W = A_i Y + A_j Z \tag{2.15}$$

$$W = Y \pm Z \tag{2.16}$$

where A_i and A_j are constant coefficients. All of the equations in (2.14) can be implemented using (2.15) or (2.16) in one or multiple cycles. Although the CC_L used for the lifting architecture cannot be applied to B-spline, a new computation core suitable for sequential evaluation of the filter steps in (2.14) has been defined to minimize hardware requirements. Fig. 2.6(a) shows the hardware-efficient computation core tailored to the B-spline implementation of (2.15) and (2.16), hereafter called CC_{B1}. Though the B-spline algorithm requires six multiplies and 18 additions, the task can be handled by sequential reuse of two multipliers, an adder and two multiplexers. Since most of the equations do not need multipliers, the inputs are fed directly to the adder stage through multiplexers to conserve power and reduce average delay. In following discussion, the shifters needed to implement binary integer multiplications are ignored because they do not require significant hardware resources.

Using the CC_{B1} block, 18 cycles are required to execute all of the steps in (2.14) because some of the equations cannot be implemented in a single cycle. To reduce calculation delay, notice that the CC_{B1} implementation relies on the fact that (2.16) can be generalized by (2.15) when A_i and A_j are assigned an appropriate unity values (± 1). As a result, some of the steps in (2.14) require two cycles to calculate and no steps can be executed simultaneously using CC_{B1}. The time required to calculate all of the steps in (2.14) can be reduced to 11 cycles by including another adder, as shown in Fig. 2.6(b).

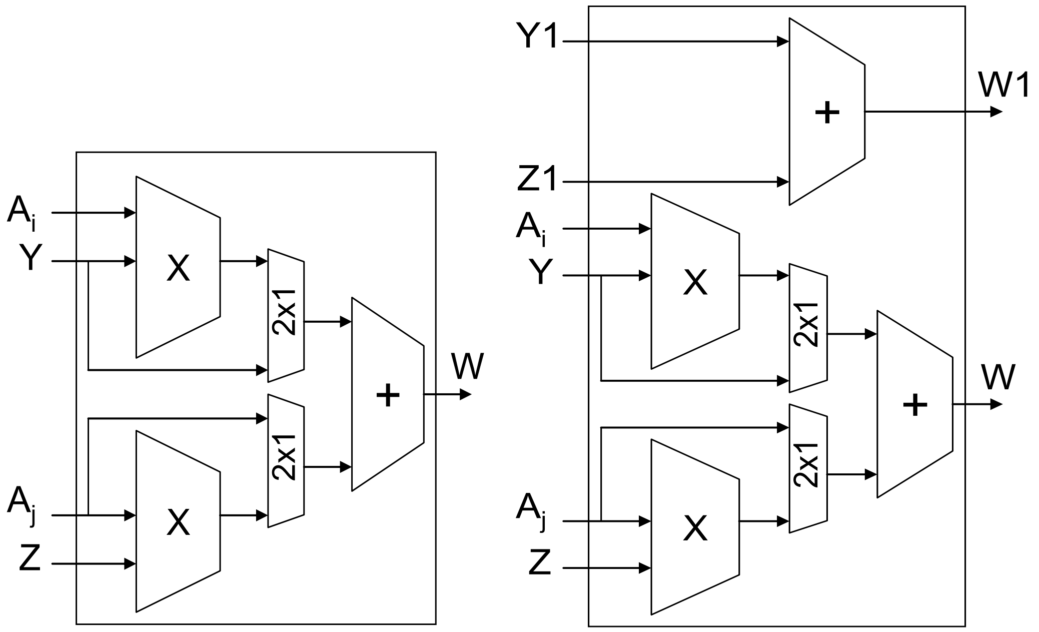


Fig. 2.6: (a) Minimum chip area computation core for B-spline, CC_{B1}. (b) Minimum delay computation core for B-spline, CC_{B2}.

The new computation core, CC_{B2}, allows several computations to be performed in parallel. The delay reduction permitted by CC_{B2} comes at the cost of an increase in chip area and a more complex controller to handle the parallelism in this multiple-input multiple-output block. Additional performance comparisons of CC_{B1} and CC_{B2} are discussed later in the chapter. The critical path for both the

B-spline CCs is $D_m+D_x+D_a$, where D_m and D_a are defined above and D_x is the delay due to the multiplexer.

2.4 Sequential vs. Pipelined Implementations

In a first-order analysis, the area of a CMOS integrated circuit is proportional to the number of transistors required, and power consumption is proportional to the product of the number of transistors and the clocking frequency. Through transistor-level custom circuit design, circuit area and power consumption can be further reduced, with significant improvement in efficiency over held-programmable gate arrays (FPGA) or standard cell ASIC implementations.

Parallel execution of the DWT filter steps using a pipelined implementation is known to provide efficient hardware utilization and fast computation. In fact, a vast majority of the reported hardware implementations for lifting-based DWT rely on pipeline structures [22, 30, 39]. However, these circuits target image and video applications where speed has highest priority and the wavelet basis is chosen to optimize signal representation. A different approach is required to meet the power and area constraints imposed by implantability requirements, the low bandwidth of neural signals, and the type of signals observed. Two different integer lifting DWT implementations, a pipeline approach and a sequential scheme, have been optimized and compared for the symlet-4 factorization. Furthermore, the hardware requirements for lifting DWT have been compared to a B-spline implementation to verify the advantage of lifting in the application at hand.

2.4.1 Pipelined Design for Lifting DWT

The integer DWT filter equations in (2.12) can be implemented simultaneously in a pipeline structure that permits real time, continuous signal processing to take place. Fig. 2.7 illustrates a pipeline structure designed around the customized three-term computation core from Fig. 2.5.

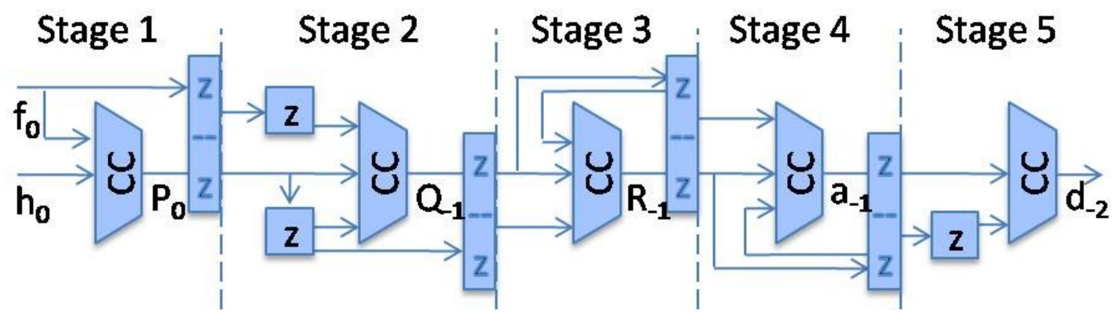


Fig. 2.7: Pipeline structure for integer-lifting wavelet transform with data notations to match filter equations in (2.12) at a single point in time. Boxes marked with ‘z’ represent pipeline stages and registers that need to be stored in memory for multi-channel multi-level processing.

The output of each of the five filter stages is held by a pipeline register, and other registers provide the necessary delays. By clocking all of the registers out of phase from the CC blocks, continuous operation is provided. The computation latency is seven cycles, due to the five pipeline stages and the two delay cycles built into (2.12). The temporal latency for detail *d* and approximation *a* results is 14 samples because each computation cycle operates on a pair of data samples. The overall pipelined *computational node* consists of five CC blocks, 15 10-bit registers, and an 8x6b coefficient ROM. An additional delay phase could be added at the *'d'* output to synchronize the latency of the detail and approximation outputs.

2.4.2 Sequential Design for Lifting DWT

The pipeline structure achieves fast integer DWT processing via a large hardware overhead, and thus well suited for low-power, single channel, neural signal processing. However, as discussed below, scaling the pipeline for multiple data channels and/or multiple decomposition levels begins to break down the efficiency of the pipeline structure. An alternative approach is to process each of the filter steps (or pipeline stages) sequentially using a single CC block and fewer memory registers. This approach takes advantage of the low bandwidth of neural signals that permits the CC to be clocked much faster than the input data sampling frequency (typically in the range of 25–40 kHz).

Sequential processing of the integer DWT filter steps can be achieved when each stage depends only on data from previous cycles or from same-cycle outputs generated in a preceding step. The simplicity of data dependencies relative to the pipeline structure can be observed from Fig. 2.8, which illustrates the sequential structure in a format comparable to the pipeline.

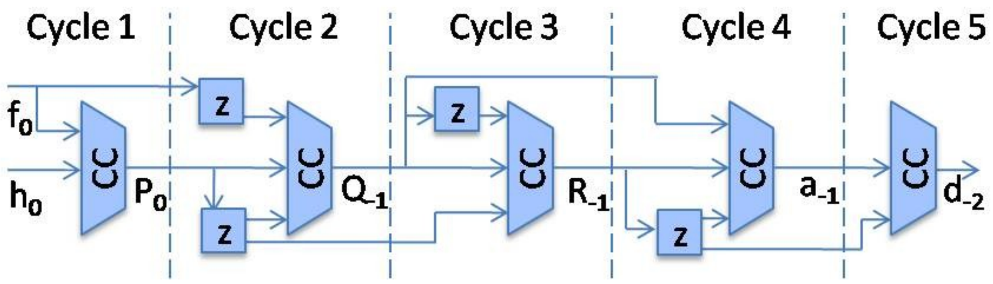


Fig. 2.8: Sequential structure over five operation cycles. Boxes marked with 'z' represent registers that need to be stored in memory for multi-channel multi-level processing.

Here, each section of the circuit represents a temporal phase rather than a physical stage. An important observation is that significantly fewer registers are needed because the inputs of subsequent phases rely largely on preceding outputs from the same computation cycle. Therefore, it can be shown that the overall sequential DWT circuit can be efficiently implemented with six 10-bit registers to manage data flow between computation cycles, a single CC block, an 8 x 6b coefficient ROM, and a simple control block to direct data from memory to the appropriate CC input during each phase of operation. Sequential execution has a computation latency of *two* cycles, and the temporal latency for detail and approximation results is *four* samples.

2.4.3 Pipeline vs. Sequential Lifting DWT

As stated above, the sequential approach requires only one CC unit and six 10-bit memory registers compared to five CC units and 15 registers for the pipeline circuit. The sequential design does, however, require additional multiplexers and control logic to redirect data and coefficients to CC inputs, which are not necessary in the inherently hardware-efficient pipeline design. This added circuitry will make the critical path of the sequential circuit longer than that of the pipeline structure. Furthermore, to maintain the same throughput, the sequential design must be operated at five times the clock rate of the pipeline. Because data is processed in a real-time streaming mode, neither approach requires a large input data buffer.

Both architectures have been thoroughly analyzed to determine which approach is best suited to the power and area requirements of an implantable neural signal processor. To first validate that both approaches can achieve the application speed requirements, a custom computation core has been implemented in CMOS, and analog simulations show the critical path delay is 6.5ns in 0.5μm technology. Thus, approximately 6000 computation cycles could be preformed within a nominal 25-kHz sampling frequency for neural signals. Even if an equally large memory access delay is included, 3000 computations can be completed within a single sampling period. This indicates that speed is not a critical design constraint and that circuit optimization can focus on chip area and power consumption.

Using custom design techniques, the chip area, *A*, required to implement both approaches will be roughly proportional to the number of transistors in the circuit

$$A = T.\sum_i N_i \tag{2.17}$$

where *T* is the average area per transistor and *N* is the number of transistors in ith circuit block. Empirical observations of several custom circuit layouts shows that a single value for T reasonably approximates all of the integer DWT blocks, especially for comparing two similar circuits.

Although absolute power consumption is inherently difficult to estimate, for the purpose of comparing the two similar design alternatives, dynamic power can be determined from

$$P = k \cdot VDD^2 \cdot f_s \cdot \sum_i N_i \cdot s_i \qquad (2.18)$$

where *VDD* is the supply voltage and f_S is the data sampling frequency (nominally 25kHz). The parameter *k* accounts for the average output load capacitance, the average number of transistors operating per output, and the average output transitions per clock cycle. This parameter is a function of both fabrication process and circuit topology and has been derived empirically and set to a value of 3fF and 0.75fF for 0.5μm and 0.13μm technology, respectively. The variable *s* is the clock rate scaling factor relative to f_S for each block *i* such that the clocking frequency of each circuit block is (s_i *x* f_S). For example, in the pipeline configuration, the computation core will be clocked only every other cycle, i.e. 0.5f_S so that the first of the pair of samples to be processed can be acquired in the idle cycle. Correspondingly, because the sequential configuration must be clocked at five times the rate of the pipeline, it will have an average clocking rate of 2.5fs . In the pipeline approach, all of the blocks are clocked at the same frequency, except the coefficient memory, which is static in both designs. Since one of the multipliers is idle during two of the five stages, so we estimate the sequential CC clock-scaling factor to be 2. Similarly, in the sequential controller, most of the circuits are clocked at 2.5f_S while others are clocked at 0.5f_S, so we estimate the clock-scaling factor to be 2 as well.

Table 2.4 lists the total number of transistors in each approach along with the area and power estimated from (2.17) and (2.18) for both 0.5μm and 0.13 μm technology. As expected, the pipeline computation unit requires nearly three times the area of the sequential approach, and it would occupy about 21% of the chip area on a 3x3mm chip in 0.5μm technology or 5% of a 1.5x1.5mm chip in a 0.13μm process. The power model predicts that the sequential approach will consume only 23% more power than the pipeline. The larger power consumption of the sequential approach can be attributed to its requirement for a more complex controller and the need to move more data around within the single computation core. Overall, these results show a tradeoff between area and power consumption between the two approaches.

Conservative values of 80 μm^2 per transistor for 0.13μm technology have been selected to estimate the required chip real estate.

Table 2.4: Characteristics of single-level, single-channel Integer DWT hardware for pipeline and sequential configurations at two technology nodes.

Architecture	# transistors	0.5μm technology		0.13μm technology	
		area [mm^2]	power [μW]	area [mm^2]	power [μW]
pipeline	23,336	1.867	15.10	0.117	0.71
sequential	7,931	0.634	18.55	0.040	0.87

If area-power products are taken as the figure of merit then pipelined design is about 2.5 times more expensive then the sequential design. Moreover, the above analysis does not include static power consumption but since it is directly proportional to the number of transistors, it is expected to be higher for pipelined design.

2.5 Multi-Channel / Multi-Level Sequential DWT

2.5.1 Single Channel / Single Level Architecture

The computation cores described above require supporting memory and a controller to implement the full DWT. The collection of memory, controller and computation core blocks into a larger module that is capable of independently calculating DWT results will be referred to as computational node (CN). The CN requires three different memory modules to store filter coefficients, intermediate results for sequential CC calculations, and intermediate channel/level results as described below. It has been shown previously that quantizing the data to 10 bits and filter coefficients to 6 bits maintains high signal to noise ratio and does not distort the signal of interest when lifting DWT is applied. For proper comparison, the same quantization is assumed for B-spline.

1. **Computation Core Memory**

Due to the prescribed sequential reuse of CC hardware, temporary memory is required to store intermediate calculation results for proper evaluation of (2.12) and (2.14). These 'computation core memory' registers are used over and over again for every cycle of the CC. A careful analysis shows that, for lifting-based DWT, a total of seven 10-bit registers are required while CC is processing one sample. Both CC implementations for B-spline need twelve 10-bit registers to hold temporary values during sequential execution.

2. **Coefficient Memory**

A separate 'coefficient memory' block is required to store the filter coefficients listed in Table 2.1 and Table 2.2. For lifting, the coefficient memory block size is eight 6-bit registers, while for B-spline only six registers are required. The coefficient memory data is constant and can be implemented as ROM.

2.5.2 Multi Channel / Multi Level Architecture

Most neuroprosthetics applications require data from more than one nerve or from spatially scattered locations. As a result, multiple data streams need to be compressed simultaneously in real-time using DWT and transmitted to extra-cutaneous processing units. Because of area constraints in surgically implanted neuroprosthetics, it is desired to process all data channels within a single interface chip. Furthermore, increasing the number of decomposition levels inherently improves the compression ratio of DWT [16, 25], which is vital for reducing the power required to transmit data from the implanted device. Therefore, the design of a computational node that can compute multi-level DWT for multiple channels pseudo-simultaneously, i.e. within a single sampling period is necessary. Recall that the relatively long period between neurologically relevant data samples motivated the design of a sequentially cycled single-stage computation core. Analysis of preliminary DWT implementations [36] indicates that there is sufficient bandwidth between samples to process many data channels sequentially with a single computation core, permitting significant savings in chip area.

1. Channel/Level Memory

Having adopted a pseudo-simultaneous approach for processing multiple input channels, notice that, for each data sample, the CC finishes calculating results for a particular channel and then proceeds to calculate results for next channel. However, the intermediate values, or state, of the current channel need to be stored in memory so they are available when the CC returns to this channel for the next data sample. Similarly, when the CC processes a lower level of decomposition, the current state of the CC needs to be stored. This ‘channel/level memory’ is critical to restoring the state of computation when the CC switches between different channels and levels.

Equations (2.12) and (2.14) define which values need to be stored and made available to process future samples. For each level and channel (beyond level=1 and channel=1), the lifting architecture requires four 10-bit values, f_0, P_0, R_0, and Q_0, to be saved in CN memory. Similarly for B-spline, there are a total of eight 10-bit values, R_0, Q_0, P_0, N_0, f_{-1}, f_0, h_{-1} and h_0, that need to be stored in channel/level memory. Note that these values are set by the algorithm and thus do not change with either of the B-spline CCs presented above. Compared to lifting, B-spline requires four more 10-bit registers for each additional level or channel. Although the number of levels would typically remain small (<5), modern neural recording arrays continue to push the boundary on available channels, and this difference in memory requirements becomes a key factor for large numbers of channels.

Other memory blocks such as pairing memory and input buffer pose equal hardware requirements on both algorithms and are thus discussed in detail in next chapter.

2. DWT Computational Node

Arranging all the above-mentioned blocks, the CC, the controller, and the memories, the general computation unit architecture of Fig. 2.9 is obtained. The size of each block varies with algorithm, lifting or B-spline, as noted in the discussions above. Based on this design, the next section provides a thorough comparison of the VLSI implementations of the two algorithms.

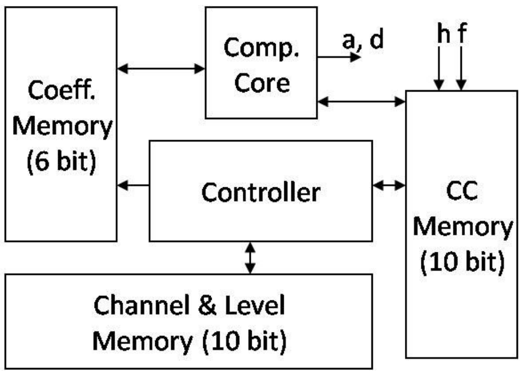


Fig. 2.9: The general computation unit architecture for sequential DWT.

2.6 Comparative Analysis for Multi Channel/Level DWT

2.6.1 Multi Channel / Multi Level Lifting vs. B-spline

For the sake of comparison, the coefficients and data for both the implementations have been quantized with the same precision, i.e. 6 bits and 10 bits, respectively, with negligible loss in signal fidelity. Table 2.5 summarizes the hardware and timing characteristics of both architectures when implemented with the CC modules described above. Multipliers, adders, and memory blocks were fully custom-designed in 0.18μm CMOS and used in all implementations. The multipliers were optimized for 10x6 bits to minimize transistor count and reduce delay compared to a standard square multiplier. The B-spline shifters are listed in Table 2.5, though they require a negligible amount of hardware and contribute very small delay.

To compare critical path delays notice that, for different adder/multiplier implementations, the relative delay of multipliers, D_m, and adders, D_a, could vary significantly. To evaluate the impact of this ratio on computational delays, we defined $D_m \approx \alpha \bullet D_a$ and observed the delays in units of D_a as a function of α. Defining the total critical path delay as the CC critical path delay times the required number of CC cycles per sample, and assuming that the delay due to multiplexers, D_x, in B-spline is negligible, it can be argued that *B-spline 2* always has less delay than both *Lifting* and *B-spline*, indicating the use of CC_{B2} would be preferred where delay poses a critical constraint. With α=3, as it is for our custom designed adder and multiplier, *Lifting* delay is only slightly worse that *B-spline 2.* The computational load per sample listed in Table 2.5 is the measure of computational resources used to calculate results for a single sample and thus directly effects the power dissipation.

Table 2.5: Hardware comparison of lifting and B-Spline architectures.

	Lifting CC_l	B-spline CC_{B1}	B-spline CC_{B2}
Multipliers	2	2	2
Adders	2	1	2
Multiplexers	0	2	2
Shifters	0	4	4
Latency	3	0	0
Cycles per Sample	5	18	11
Coeff. Mem. (6bit)	8	6	6
CC Mem. (10bit)	7	12	12
Per Chn/Lvl Mem.	4	8	8
Critical Path Delay	D_m+2D_a	$D_m+D_x+D_a$	$D_m+D_x+D_a$
Computational Load per Sample	$5D_m+10D_a$	$4D_m+18D_x+18D_a$	$4D_m+11D_x+18D_a$
No. of Tx for CC	1786	1650	1810

The lifting approach results in a highly regular set of equations so that the corresponding CC has a high utilization factor. The equations governing B-spline do not follow a regular pattern, which results in a minimum area CC_{B1} that does not optimally use the capabilities of hardware. As a result, CC_{B1} requires a high number of cycles to process a set of samples. The CC_{B2}, on the other hand, is designed to parallelize the computation and improve the hardware utilization and thus relatively fewer cycles are required for calculations.

The lifting architecture the inputs *h* and *f* lead to P_{-1}, which is required for calculation of Q_{-2}, which is subsequently required for a_{-2}, which finally produces $d_{-3.}$ Thus, we cannot compute d_{-3} until we have received *h* and *f*. Hence, as discussed above, there is a latency of three samples between input and its corresponding output. B-spline does not have any future sample dependence so there is no latency between input and output samples. The cost, however, is that B-spline requires 18 or 11 cycles, depending upon the type of CC used, to process one sample as compared to 5 cycles taken by lifting. This translates to a higher clocking rate for B-spline. Note also that most of the transistors in lifting architecture are active all the time, however, since B-spline does not use the hardware very efficiently, a large fraction of transistors in B-spline remain inactive most of the time, dissipating undesirable static power. By counting the number of transistors active over different calculation cycles for a single sample, power requirements can be modeled by

$$P = K \cdot VDD^2 \cdot f_s \sum_j \sum_i N_i \cdot S_{i,j} \tag{2.19}$$

where the first summation *j* is over the total number of cycles (5, 18 and 11 for lifting, B-spline and B-spline 2, respectively) and *i* refers to the circuit blocks, i.e., computation core, controller, and different memory blocks. N_i is the total number of transistors in ith block and $S_{i,j}$ is the fraction of transistors active in ith block in the jth cycle. *K* is a constant that depends on IC fabrication process parameters. Note that the clock frequency is different for all the three implementations because of the different number of required cycles per sample.

Based on (2.19), and assuming the relative area is directly proportional to transistor count, Fig. 2.10 compares the relative area and power consumption of the B-spline and lifting designs for an increasing number of channels and decomposition levels. The plots show that lifting requires smaller area and consumes less power than both implementations of B-spline. CC_{B1} for B-spline requires one less adder than lifting and thus has fewer transistors. However, in a multi-channel, multi-level implementation the number of transistors for the entire computational node, including memory and controller, increases much more rapidly with B-spline than lifting, thus, offsetting any advantage gained by a smaller CC. The B-spline implementation with CC_{B2} requires less power than CC_{B1} mainly because it requires lower clocking frequency. The lifting implementation shows increasingly superior performance as the number of channels and levels are scaled up, due in large part to its lower channel/memory requirements compared to either B-spline implementation.

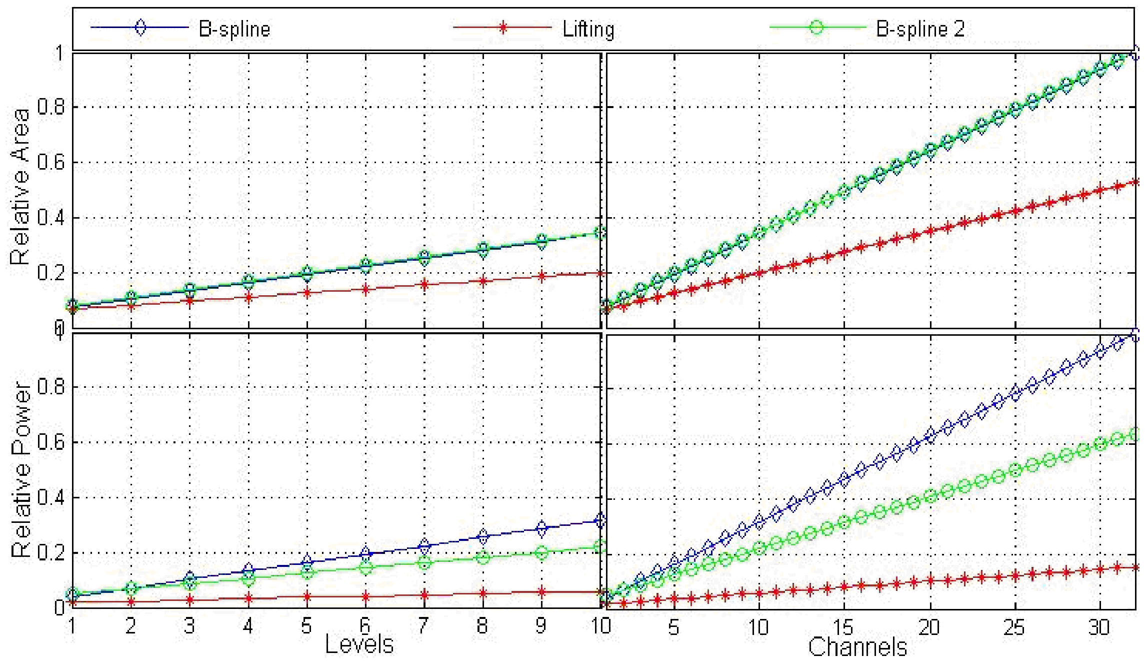


Fig. 2.10: Comparison of relative area and power consumption vs. number of levels and channels for B-Spline, B-Spline 2 and Lifting architectures.

For a single-channel, single-level DWT, no channel/level memory is required. As channels or levels increase, the difference between the two B-spline CC implementations remains constant, however, CC_{B2} retains the advantages of smaller critical path delay and lower power consumption. For every increase in the number of channels or levels, lifting requires significantly fewer transistors than B-spline, making it preferable for multi-channel applications with power and area constraints.

Multiplication-Free Lifting*:* The CC unit proposed for lifting DWT uses two multipliers so that the calculations required per sample are eight multiplications and eight additions that can be completed in. It is noteworthy that a general purpose lifting approach based on only shifts and additions was proposed in [21]. For the sake of completeness, we compared the demands of a CC unit with multipliers to a CC unit without any multiplier, i.e., composed of only a shifter and an adder. The later approach resulted in 12 shift operations and 21 add operations, and required 21 cycles per sample. This is because the equations required to compute multiplication-free lifting DWT did not show any regular structure such as the ones in (2.13). Therefore, substituting another adder and shifter in the data path did not help in reducing the number of cycles required to complete the computation. With respect to area demands, we found that for one sample pair, a CC unit without a multiplier requires 52% less area compared to a CC with multiplier. This obviously translates into large savings in chip area. However, these savings were not substantial when the system is scaled up. For example, a 32-channel/4-level DWT system using a CC with multiplier would occupy 6.5% of the total chip area as opposed to 3.3% using a CC without multiplier. Therefore, the overall savings in chip area are only 3.2%. In contrast, the CC without multiplier requires 13.3% more power than a CC with multiplier for this specification. We therefore concluded that the reduction in area using a shift and add strategy in the lifting approach is overshadowed by the increase in power dissipation when multichannel/ multilevel decomposition is sought.

2.6.2 Multi Channel / Multi Level Sequential vs. Pipeline

Both the pipeline and sequential architectures can be scaled to multiple channels and/or levels by reusing the computational node hardware and increasing the clocking frequency to complete all computations within the input sample period. In both approaches, registers within the computational node hold data necessary for the next cycle's calculation. To sequentially reuse the computational node, some register values for a specific channel/level must be saved so they will be available when that channel/level is next processed in a future cycle. Fig. 2.11 shows the multichannel, multilevel, implementations of the pipeline and sequential configurations.

For multiple channels/levels, the need to copy the entire set of pipeline registers to memory effectively negates one of the primary advantages of the pipeline over the sequential approach. On the other hand, the sequential processing circuit is inherently designed to swap new data in/out each clock cycle. To quantitatively compare these two approaches, circuit models have been developed to describe the power and area for each option as a function of the number of channels and the number of decomposition levels. The following models assume the hardware (including control logic) has been scaled to manage multiple channels and levels, though they are still valid for single channel, single level implementations.

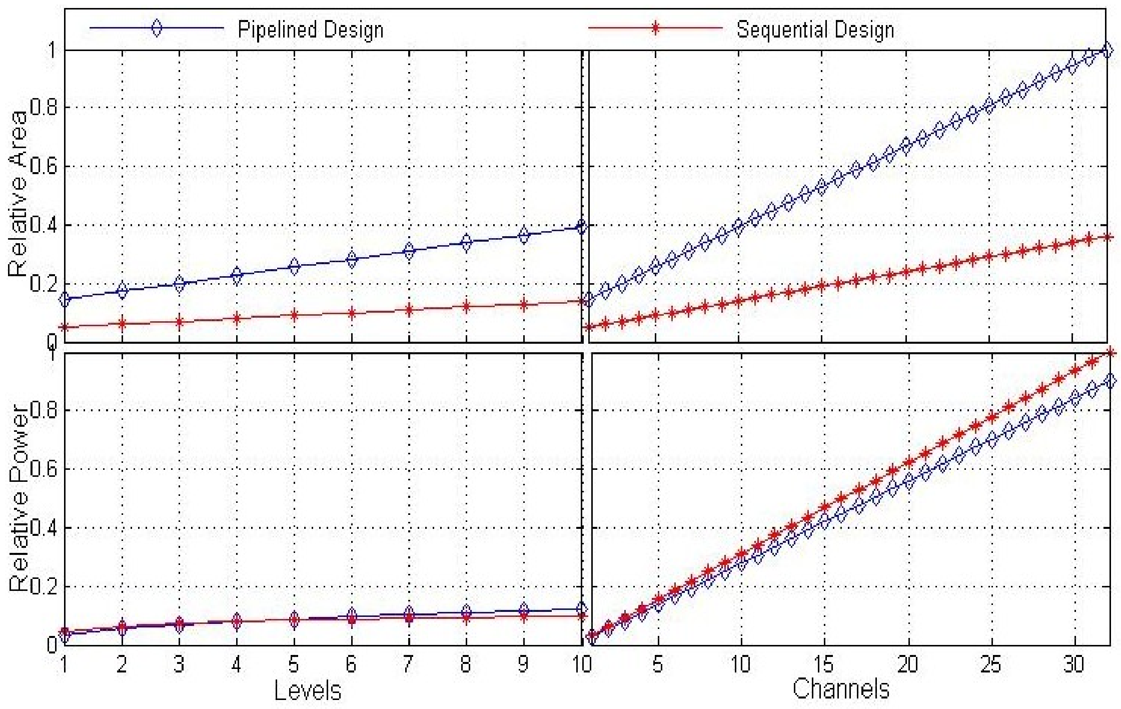


Fig. 2.11. Comparison of multichannel/multilevel pipeline and sequential DWT approaches: relative chip area and relative power consumption versus number of levels and channels.

The relative area for pipeline and sequential architectures as a function of levels and channels is shown in Fig. 2.11. These results demonstrate that the pipeline requires significantly more chip area than the sequential approach and its area needs grow faster with larger number of channels and levels. This is due primarily to the relatively large number of registers that must be stored per channel or level (11 for pipeline compared to 4 for sequential).

Fig. 2.11 also shows the relative power consumption for the two approaches. The linear increase in power per channel is slightly higher with the sequential design than the pipeline. Although there is a sharp jump in power from L = 1 to L = 2, further increases in levels require less and less additional power as the usage rate approaches one. The most important observation from Fig. 2.11 is that the power consumption of the two implementations is almost similar but the sequential design requires significantly less chip area.

Due to size and power constraints in implantable systems, an important figure of merit is the relative area-power product, which is plotted in Fig. 2.12 versus both levels and channels. Fig. 2.12 illustrates that the sequential approach is increasingly preferable as the number of channels or the number of decomposition levels increases. The only significant benefits of the pipeline within the enforced design constraints are that it can be clocked at a higher rate and that it takes fewer clock cycles to complete a computation.

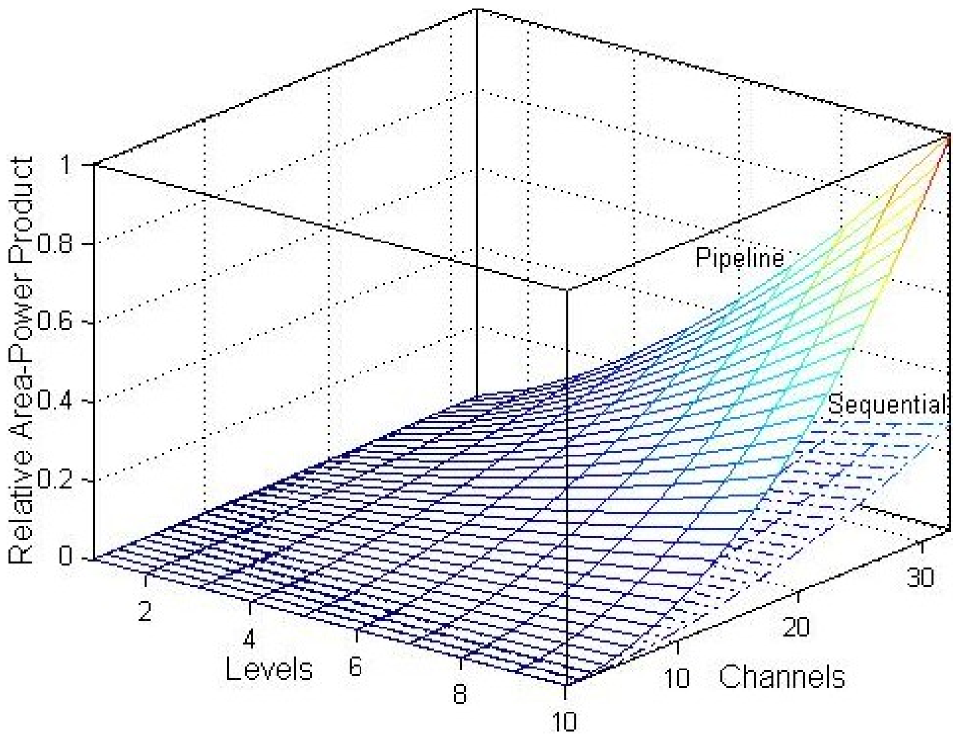


Fig. 2.12. Power-area product versus level and channel for pipeline and sequential approaches.

Both of these factors result in the pipeline having a *higher limit* on the maximum number of channels that can be simultaneously processed. However, based on the parameters defined above, the sequential execution architecture has an estimated maximum of around 500 data channels (at L = 1). Given the chip area limitations, the area-efficient sequential approach is best suited for this application. In an example implementation with 32 channels and 4 levels of decomposition, the models predict that the sequential approach will require 0.692 mm and 50.1µw in 0.13µm CMOS. indicating the feasibility of performing front-end signal processing within the constraints of an implanted device.

Another interesting result of this study is the comparison of the area required by the computational node circuitry versus the area required by the memory that holds register values required for multichannel/multilevel operation. Fig. 2.13 illustrates this result for both sequential and pipeline configurations as a function of channels at L = 4. Increasing the number of levels or channels does not increase the area taken by computation blocks, however, memory requirements scale linearly. With 10-bit data resolution, at L = 4 and C = 32, the pipeline requires over 14 000 bits of SRAM, while the sequential circuit requires only about 5000 bits. Reducing memory requirements becomes increasingly important in multichannel applications, again highlighting the advantage of the sequential approach.

As illustrated in Fig. 2.13, the memory required to store intermediate calculation values will dominate circuit area in multichannel implementations. Careful analysis of an optimized sequential B-spline implementation [24] has shown that eight memory registers are required per channel/level, compared to four for sequential lifting and 11 for pipeline lifting. Based on this information and the comparisons above, B-spline has a slight advantage over pipeline lifting but incurs a significant penalty relative to sequential lifting in terms of area. Furthermore, the sequential lifting implementation requires only about 25% of the dynamic power of sequential B-spline, primarily because B-spline takes 18 cycles to execute sequentially compared to 5 cycles for lifting [24]. The advantage of sequential lifting becomes even more profound when static power is considered, especially in deep submicron technologies.

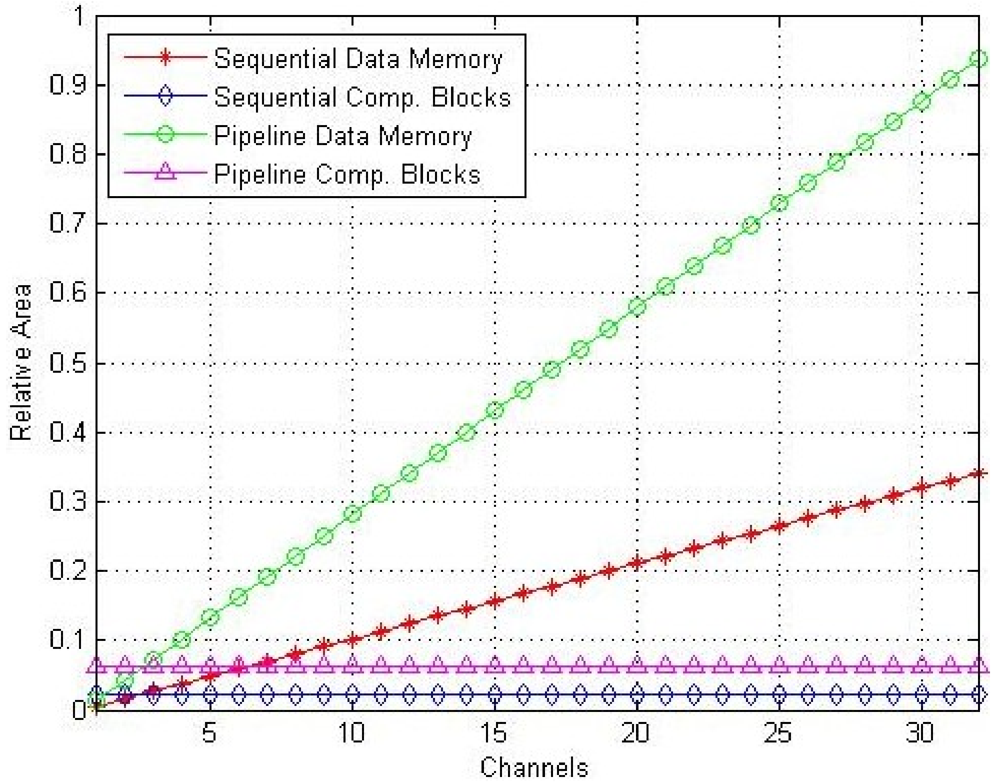


Fig. 2.13. Relative area versus channels of data memory compared to all other blocks for sequential and pipeline designs, at L = 4.

2.7 VLSI Architecture for Multi-Channel DWT

To utilize DWT-based compression with modern neuroprosthetics devices, a multi-channel, multi-level implementation is necessary because microelectrode arrays sample multiple data channels simultaneously and multiple decomposition levels improve signal reproduction accuracy. Thus, area and power efficient hardware that can perform multi-channel, multi-level DWT in real time is highly desirable. In contrast to traditional DWT applications, neuroprosthetics can afford long computation intervals, up to 40μsec [17], permitting hardware to prioritize power and area efficiency over speed. From the hardware point of view, integer-lifting and B-spline DWT factorization schemes have very efficient implementations. The lifting approach to the DWT reduces the required arithmetic operations and, as an in-place implementation, requires less memory at the expense of a longer critical path [22, 23]. The B-spline factorization reduces the critical path delay by converting some of the required multiply operations into less computationally intensive shift-and-add operations [23].

With few exceptions, recent efforts to optimize DWT hardware have concentrated on increasing throughput at the expense of area and power [22, 23]. In contrast, our prior work has identified an optimal implementation of the DWT architecture where chip area and power have priority over speed [24]. The approach relies on a hardware-efficient integer lifting factorization scheme and the 'symlet' family of wavelet basis that has been shown to provide a near optimal compression of neural signals [17]. This chapter describes the design of a highly area and power efficient VLSI circuit that implements multi-channel, multi-level lifting-based DWT in real time.

2.7.1 Architecture Overview

Fig. 2.14 shows the data flow diagram of lifting-based DWT using the symlet-4 bases. If *h* and *f* are two sequential input samples from the same channel, *a* and *d* are approximation and detail results, and *P, Q*, and *R* are intermediate results, the five filter steps in this flow can be expressed by

$$\begin{aligned} P_0 &= h_0 + B_0 f_0 \\ Q_{-1} &= f_{-1} + B_1 P_0 + B_2 P_{-1} \\ R_{-1} &= P_{-1} + B_3 Q_{-1} + B_4 Q_{-2} \\ a_{-1} &= Q_{-1} + B_5 R_{-1} + B_6 R_{-2} \\ d_{-2} &= R_{-2} + B_7 a_{-1} \end{aligned} \qquad (2.20)$$

where B_0-B_7 are the eight constant filter coefficients and subscripts of the other variables represent the time sample, 0 being the current sample, -1 the previous sample, and so on.

To ease hardware requirements, it has been shown that integer lifting DWT with quantized data and filter coefficient values maintains a high signal to noise ratio. Based on this analysis, we have chosen 10-bits data and 6-bit coefficients (including sign bit) for our hardware implementation. Our previous work evaluated two structurally different hardware approaches, namely pipeline and sequential, for suitability in implantable devices based on their resource demands

[35]. Derived from this system-level analysis, we have found that for the single channel case, the pipeline architecture consumes smaller power at the expense of considerably larger area as compared to the sequential architecture, resulting in a higher area-power product. The difference in area-power product increases with the increasing number of channels and levels, making the sequential architecture the better choice of the two especially for higher number of channels.

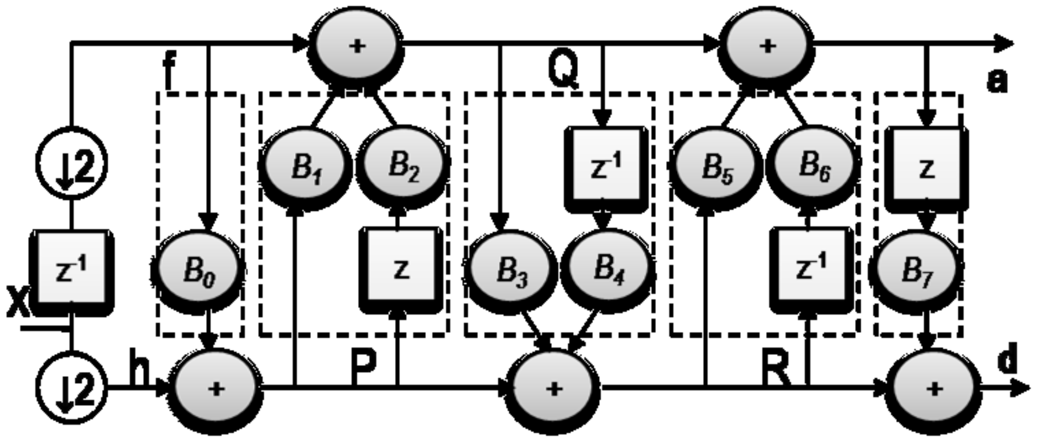


Fig. 2.14: Data flow diagram for lifting DWT using the symlet-4 basis.

Fig. 2.15 describes the DWT architecture resulting from our circuit-level design efforts. It is composed of a customized computation core (CC), a digital controller, and five memory modules for incoming data, filter coefficients, intermediate CC products, and intermediate values necessary to sequentially process multiple levels and channels.

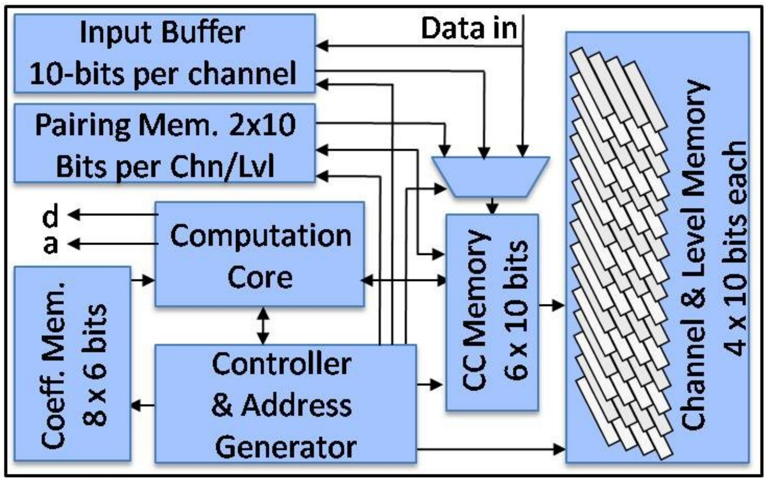

Input Buffer 10-bits per channel
Data in
Pairing Mem. 2x10 Bits per Chn/Lvl
d
a
Computation Core
Coeff. Mem. 8 x 6 bits
Controller & Address Generator
CC Memory 6 x 10 bits
Channel & Level Memory 4 x 10 bits each

Fig. 2.15. Complete system diagram for sequential calculation of DWT.

Sequential data samples (inputs h and f) from a given channel are processed in pairs to generate the approximate and detail DWT coefficients (outputs a and d). To achieve system-level goals of power and area efficiency, each of these architectural blocks has been customized at the circuit level to minimize transistor count and eliminate unnecessary output transitions.

2.7.2 Computation Core

Analysis of the ‘symlet-4’ lifting factorization and resulting equations in (2.20) shows a noticeable computation regularity; all arithmetic operations can be expressed in the general form of W=X+BiY+BjZ. This regularity can be exploited to minimize hardware by implementing a single computation core (CC) that sequentially executes all computational steps in (2.20). Sequential reuse of the same hardware significantly reduces chip real estate requirements without impacting performance in this low bandwidth application.

The merits of using two’s complement arithmetic versus sign-magnitude arithmetic were considered. The 10-bit input data is received in sign magnitude form, as are the constant 6-bit filter coefficients. The CC multipliers are required to perform 10x6 operations. Methods for multiplying sign-magnitude numbers do not work with two's complement numbers without adaptation. We have implemented several combinations of adders and multipliers using sign-magnitude and two’s complement representation and found that it is most efficient to handle multiplication in sign-magnitude form while additions are performed in two’s complement. As shown in Fig. 2.16, the computation core can be implemented using two multipliers and a three-term adder formed by cascading two two-term adders. Using this CC to sequentially execute the steps in (2.20) requires five cycles to compute results for one sample pair. The critical path for this CC is Dm+2Da, where Dm and Da are the delays of the multiplier and the adder, respectively. Overflows that can occur during these computations are handled by replacing the results with appropriate positive or negative saturation values.

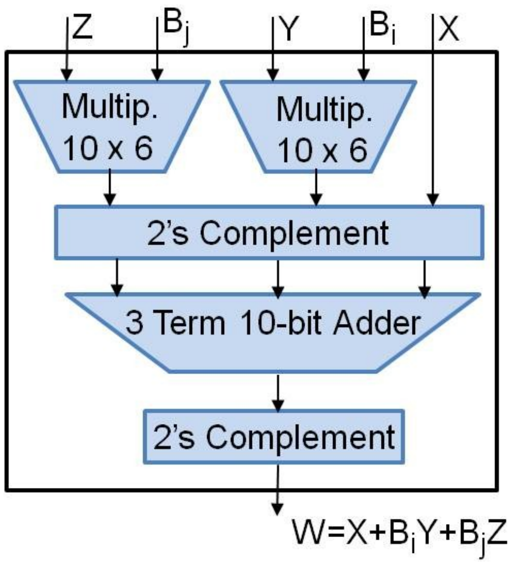


Fig. 2.16. Computation Core architecture for integer lifting DWT.

1. Adders

Ripple carry, carry save and carry look-ahead adders were analyzed for this DWT implementation. Because delay does not pose a bottleneck and area and power are much more important, the lower transistor count required by ripple carry adder was favored. Ripple carry adder performance depends largely upon structure of its individual full-adder cells, whose properties can vary widely to match different applications. Comparative analysis of several full adder structures was performed at the circuit level to take into account transistor count, activity factor, and dynamic power due to parasitics. Based on these results, the pass gate logic 1-bit full adder presented in [40] and shown in Fig. 2.17 was chosen to best suit our DWT application. Unlike some adder cells with slightly lower transistor counts, this cell has no internal direct path between power and ground, eliminating short circuit current during transistor switching. Furthermore, the use of transmission gates limits the output voltage swing and reduces dynamic power consumption without compromising overall speed or reliability. In 0.18μm CMOS, this 16-transistor cell was designed to dissipate 0.4μW power and occupy 41.5μm$^{2.}$ It was utilized to construct 10-bit adders for the three-term adder block and 13-bit adders for the 10x6 multipliers.

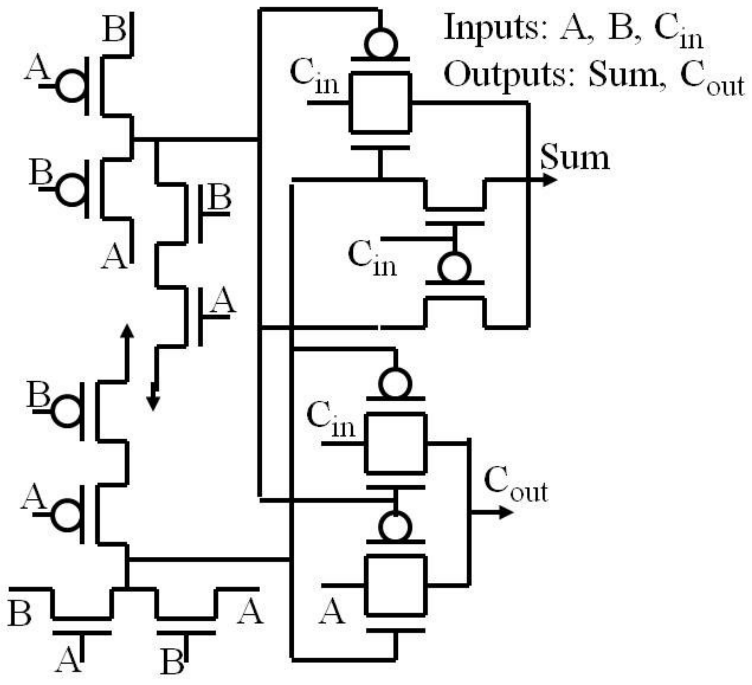


Fig. 2.17. Single bit full adder cell based on pass transistor logic [8].

2. Multipliers

The CC requires two multipliers to perform 10x6 operations. Both Booth and array multiplier structures were optimized to the specific bit resolutions of our DWT application and compared for area and power. The Booth multiplier consists of recoders, partial product generators, 3:2 and 2:1 compressors and a final adder. The recoders and partial product generators were optimized for low power [41]. Two different recoders and partial product generators were considered, both requiring 18 transistors each. The two partial product generator architectures, NPR3a and NPR3b, presented in [41], were tailored to this application and it was determined that, while power consumption is similar, NPR3b needs slightly fewer transistors (14 vs. 16) and is preferred. The 3:2 and 2:1 compressors are full and half adders respectively. In contrast, the array multiplier ANDs every bit of the multiplier with every bit of the multiplicand to create partial products. A Wallace tree structure was implemented to compress the partial products. Both Booth and array multipliers require an adder at the final stage. In comparison, a CC design using a Booth multiplier was shown to occupy 16% more area than a CC design with an array multiplier. Table 2.6 gives area, average power and delay measurements of the custom designed adder and array multiplier sub-modules selected for use in our DWT CC. It also includes the average power, area and delay measurements for the complete computational core including two's complement circuits. The average power includes static and dynamic power dissipation at a clock frequency of 6.4MHz.

Table 2.6: CC Transistor Counts and Worst Case Values

Sub-module	# of TX	Area (μm^2)	Power (μW)	Delay (ns)
10-bit Adder	160	418	2.07	0.41
10x6 Multiplier	733	2858	9.04	2.13
Comp. Core	1854	9828	27.36	6.07

2.7.3 Memory Modules

1. **Multi-channel/level memory module**

Neuroprosthetics applications generally depend upon multiple data streams taken from an array of microelectrodes. Thus, our DWT system has been designed to compress data from multiple channels simultaneously using multi-level decomposition in real time. Analysis of our DWT implementation indicates that there is sufficient bandwidth within a single computation core to process well over 100 data channels sequentially. Sequential processing significantly reduces computational hardware demand; however, the intermediate values, or ‘state’, of the current channel needs to be stored in memory so they are available when the CC later processes the next sample from this channel.

Similarly, memory is needed to store intermediate values while switching between different levels. The memory block required to hold these intermediate values is called the channel/level memory. The values that need to be stored and made available to process future samples are defined by (2.20). For each level and channel (beyond one), the lifting architecture requires five 10-bit values, a0, f0, P0, Q0 and R0, to be saved, four of which are stored in channel/level memory while one is stored in the pairing memory. Every level and every channel requires a corresponding 40-bit memory register. In a 32-channel, 4-level design, this amounts to 128 registers. The channel/level memory was implemented in both an SRAM using standard six transistor cells and a DRAM. For a large number of channels and levels, the channel/level memory module was found to dominate the power and area of the overall DWT system.

For the prototype DWT chip, the channel/level memory was implemented in SRAM to maximize reliability and ease system-level testing. However, a DRAM implementation is particularly efficient in this application; due to the sequential nature of the DWT, the entire memory block is periodically overwritten, eliminating the need for refresh circuitry during normal operation. For example, at a sampling rate of 25 kHz, the least frequently accessed level 4 data is overwritten every 0.64ms, which is much faster than the hold time easily achievable by a DRAM cell implemented in a standard CMOS process. This feature permits significant area savings over the SRAM implementation, as shown in Table 2.7, which compares relative performance criteria for a 32 channel, 4 level memory block.

In a standard 0.18μm CMOS process, use of a discrete DRAM storage capacitor is prohibitively area expensive. Alternatively, the gate capacitance of a large MOSFET has been used to store charge. Assuming that the sense amplifiers need a stored value of at least 1V to reliably read the correct value, a MOSFET with at least 0.2μm^2 gate area is required to realize a dependable hold time. Constructed in this fashion, comparison of two custom designed CMOS memory blocks shows that a DRAM block would permit a 64% area savings over an SRAM block for a 32-channel 4-level implementation.

Table 2.7: Comparison of SRAM and DRAM Memory Implementations

32 Chn/ 4 Level	No. of Tx	Total Area (μm^2)	Bit Density (μm^2/bit)
SRAM	34814	129830	25.36
DRAM	13414	46433	9.07

2. Input, coefficient, pairing and CC memory modules

Because DWT operates on data pairs, there is a hold cycle between each computation of input pairs wherein the first sample of the pair is acquired. Thus, an input data buffer of size equal to the number of channels is required to store input samples during the hold cycles. Note that in this architecture, the hold cycle is used to compute all results for higher (beyond one) levels of decomposition, where the necessary data is available from previous computations.

A separate coefficient memory block is required to store the eight 6-bit filter coefficients. Because this data is static, it is most efficiently implemented using hardwired connections, effectively operating as a ROM structure with only the electronics necessary to switch the proper coefficient values into the CC at the appropriate computation phase.

To accommodate sequential reuse of CC hardware, at the beginning of each computation cycle, four intermediate results from the previous computation cycle for the same channel must be loaded into the CC. These values, loaded from the channel/level memory module, are stored in a block defined as the CC memory. This CC memory block must be capable of loading all four bytes in parallel and moving data to the appropriate CC input as it cycles through the filter steps in

(2.20). As described below, this requires six 10-bit registers that were implemented using flip-flops for parallel load/store and byte-wise shift operations.

When performing calculations for levels 2 and beyond, the CC input data (f and h in Fig. 2.15) does not come from inputs to the DWT circuit. Rather, inputs must be pulled from results calculated previously for a lower level. These values are stored in a pairing memory that must contain two 10-bit values for every channel and level except the highest level. All computation cycles except those of the highest level will generate one 10-bit byte to be written to this pairing memory. During all computation cycles for level two and beyond, two 10-bit values will be read from this block. Due to this unique read/write structure, this block was implemented independent of the channel/level memory using an SRAM with address decoding circuitry that allows us to enable each 10-bit byte block independently.

3. Power saving strategies for memory

To reduce power consumption in the SRAM blocks, a divided bit line structure [42] was adopted, reducing the overall bit line capacitance and eliminating unnecessary dynamic power consumption during read and write cycles. Each of the sub bit lines were connected to only eight SRAM cells, and extra logic within the decoder controls access to these sub bit lines. With reduced bit line capacitance, the SRAM could be implemented without a sense amplifier, eliminating the need to access both sides of the SRAM cell and reducing bit line currents during read operations.

Instead of using a single memory block for all storage, the memory was partitioned into smaller modules based on their different access patterns. As a result, only a comparably small portion of the whole memory is active and undergoes switching at a given time, resulting in saved power.

2.7.4 State Machine Controller

The main functions of the controller are to direct overall system operation and route data between DWT circuit blocks at the proper time [43, 44]. The sequence of actions needed to complete a single DWT computation cycle consists of three different phases over a total of eight clock cycles, as shown in Fig. 2.18. As managed by the DWT controller, one read cycle is followed by five calculation cycles and then two write cycles. During the read cycle, stored values from prior calculations are loaded into the CC memory. During the calculation cycles, the five filter steps in (2.20) are executed. During the write cycles, results are stored onto channel/level and pairing memory blocks. This sequence must be repeated for each channel within the input sampling period. For example, with 32 data channels and a typical neural signal sampling rate of 25 kHz, the eight operation cycles must be clocked at 6.4MHz. For fewer channels, the clock rate can be reduced to ensure power is only consumed when computations are required.

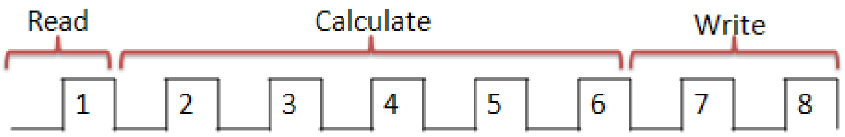
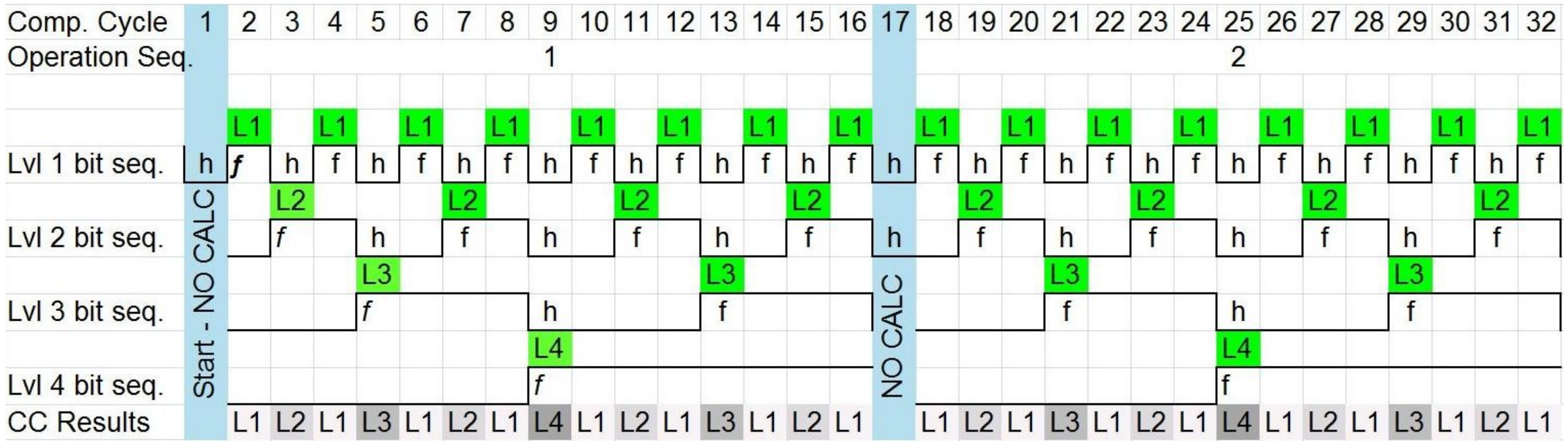


Fig. 2.18: DWT circuit operation phases for a single computation cycle.

Note that the number of levels has no effect on the clock frequency because higher-level results will be computed during intermediate hold cycles while data input pairs are being stored. This is illustrated in Fig. 2.19, which describes the sequence of computation cycles managed by the controller for one channel and four levels. Here, the top line shows the computation cycle count, and the next line counts the cycles within each operation sequence (two shown). A level 1 result is computed when a pair of data samples is received, a level 2 result is computed when two level 1 results are available, and so on. The first results calculated at each level are trash values because the memory initially contains meaningless values. After each 2L computation cycles (where L is the number of levels), there is an idle (no calc) cycle. Each column represents a computation cycle composed of eight clock cycles. The number of computation cycles necessary for a complete, repeatable, DWT operation sequence is 2L, where L is the number of levels. As shown in Fig. 2.19, four levels requires 16 computation cycles for a complete operation sequence. Within each operation sequence (2L computation cycles) one cycle will be an idle cycle where no computations are necessary, as indicated by NO CALC in Fig. 2.19. Notice that, in each odd numbered computation cycle, no level one results are being calculated. These odd cycles correspond to input hold cycles discussed above. While input data pairs are being stored in these cycles (to be processed in the following cycle), the CC hardware can be utilized to process results for all levels greater than one in the sequence defined by Fig. 2.19. Thus, an infinite number of levels could be processed without additional hardware or increasing frequency. The number of channels has no impact on this sequence;

each channel is processed sequentially within a single computation cycle, which can be achieved by increasing clocking frequency by a factor equal to the number of channels. Thus, the only major impact of increasing channels or levels is an increase in dynamic power consumption and the addition of channel/level memory.

To control the sequence and timing of operations within the DWT circuit, both an instruction based microprocessor and a state machine based controller were analyzed. Because of the relatively straightforward and repetitive nature of operations, the efficiency of a state machine design was found to be much more compatible with the power and area goals of this DWT circuit. The inherent tradeoff for this efficiency is limited flexibility, and the maximum number of levels and channels had to be set before implementing the state machine controller. Here we will describe a state machine for 32 channels and 4 levels. This design assumes multi-channel input data is multiplexed into a single 10-bit input bus, and it will output results sequentially onto a single output bus.



Comp. Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Operation Seq.									1																2							
		L1		L1		L1		L1		L1		L1		L1		L1		L1		L1		L1		L1		L1		L1		L1		L1
Lvl 1 bit seq.	h	*f*	h	f	h	f	h	f	h	f	h	f	h	f	h	f	h	f	h	f	h	f	h	f	h	f	h	f	h	f	h	f
			L2				L2				L2				L2				L2				L2				L2				L2	
Lvl 2 bit seq.	Start - NO CALC		*f*		h		f		h		f		h		f		h		f		h		f		h		f		h		f	
					L3								L3				NO CALC				L3								L3			
Lvl 3 bit seq.					*f*				h				f								f				h				f			
									L4																L4							
Lvl 4 bit seq.									*f*																f							
CC Results		L1	L2	L1	L3	L1	L2	L1	L4	L1	L2	L1	L3	L1	L2	L1		L1	L2	L1	L3	L1	L2	L1	L4	L1	L2	L1	L3	L1	L2	L1

Fig. 2.19. Per channel activity of the CC at different levels of decomposition for a 4-level implementation.

For a 32 channel, 4 level DWT system, the state machine controller utilizes a 12-bit counter to keep track of the current state, channel, and level. The three least significant bits of this counter determine the eight clock cycles within a computation cycle (Fig. 2.18) defining the operation phase for the CC and memory blocks, regardless of the channel or level being processed. The next five bits of the counter specify channel being processed. The remaining four bits represent the level being processed, where each bit represents an individual level (Fig. 2.19). The level 1 bit also determines if the input stream should go to the input buffer (hold cycle) or the computation core (compute level 1 results). In this fashion, a level 1 result is computed once a pair of data samples is received; a level 2 result is computed when two level one results are available; and so on. By increasing the width of the counter, more states could be added to support additional channels or levels. Fig. 2.20 describes operations within the CC memory controlled by the three least significant bits of the state machine counter. Here, complex data movement within the CC memory and its interaction with other memory blocks is controlled.

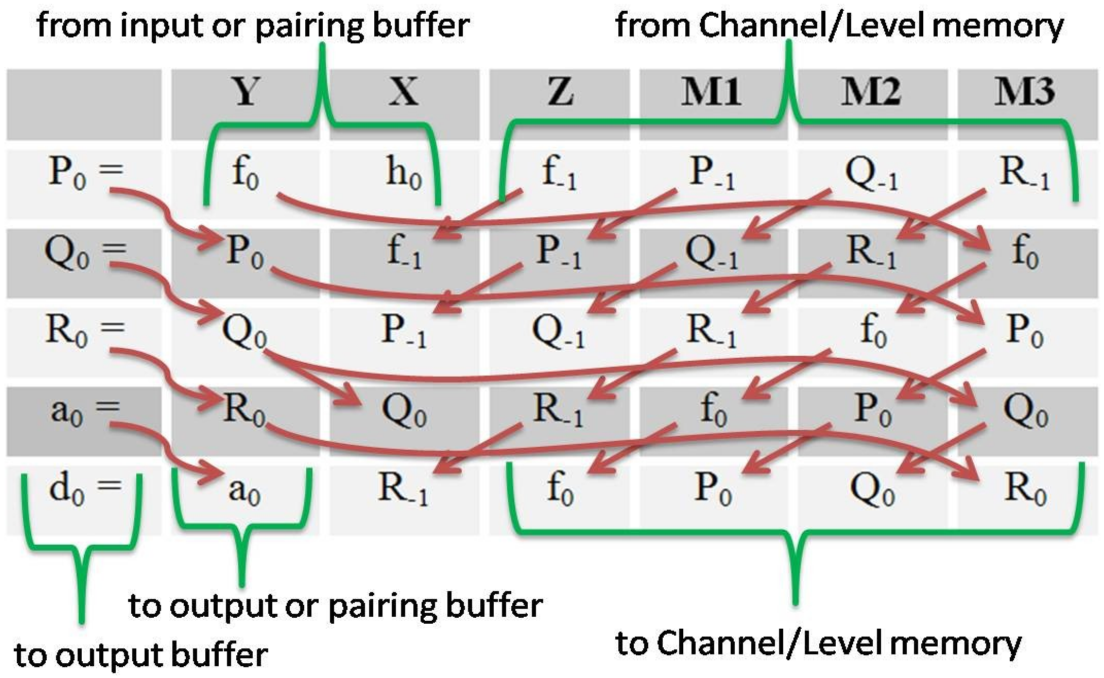


Fig. 2.20. Data movement in CC memory during the five calculation cycles within a computation cycle (one per sample per level). Top line shows the six register names. Subsequent lines define how data must be moved as the steps in (2.20) are processed sequentially.

Fig. 2.20 defines CC memory register names, where X, Y and Z are registers connected to respective inputs of the CC while M1, M2 and M3 store intermediate computation results. Values are read into the CC memory at the beginning of the computation cycle. This is followed by five cycles of calculations and register data shifts. The last calculation cycle produces detail and approximation DWT results. The detail DWT coefficient is sent to the output bus. The approximation DWT coefficient is sent to the output bus and, depending on the level being processed, stored in pairing memory. After the last calculation cycle, four of the values in CC memory, f0, P0, Q0 and R0, are stored to the channel/level memory for use in subsequent calculations of the current channel/level results.

2.8 Simulation Results and Analysis

2.8.1 Hardware Resources

All of the DWT circuit blocks were custom designed and laid out in 0.18μm CMOS. Table 2.8 lists number of transistors required for each module. The corresponding chip area required by each module, including routing, is also shown. The dominance of channel/level memory over other modules is evident, where increasing the number of channels by a factor of four (from 8 to 32) results in roughly a 300% increase in both the number of transistors and the area consumption. Our model 32-channel, 4-level DWT implementation, requires ~54k transistors and occupies roughly 470μm x 470μm. These values were obtained with the major memory blocks implemented with SRAM for reliability and testing purposes. If the input buffer, pairing and channel/level memory were replaced by DRAM blocks, the estimated transistor count for the 32-channel, 4-level circuit would drop to around 25,000 and the total DWT circuit area would reduce to a little more than 100,000μm^2, which is less than 50% of the overall circuit using SRAM. Table 2.8 also gives the area and transistor count precise estimates for 100 channel and 250 channel designs.

A 32-channel system, with a sampling frequency of 25KHz, operating at four levels of DWT decomposition requires a clock frequency of 6.4MHz. The power is managed separately for each block, thus the hardware portions not working at a given time do not consume any power. Operation of the DWT block can be modeled as a state machine which requires 8 cycles to complete its operation, as shown in Fig. 2.18.

Table 2.8: Transistor Count and Area for Hardware Modules

Module	**No. of Tx**	**Area (μm^2)**
Controller	987	6319
Computation Core	1854	9828
Comp. Core Memory (Flip-Flops)	2803	10453
Coefficient Memory (ROM)	122	714
*Pairing Mem. per Chn/Lvl (SRAM)	120*C*(L-1)	Varies
*Input Buffer per Chn (SRAM)	60*C	Varies
*Per Chn or Lvl Mem (SRAM)	240*C*L	Varies
Complete DWT system: 8 Chn, 4 Lvl	17910	74253
Complete DWT system: 32 Chn, 4 Lvl	54339	221643
Complete DWT system: 100 Chn, 4 Lvl	157562	641344
Complete DWT system: 250 Chn, 4 Lvl	385256	1583832

*Counts exclude transistors for address decoding circuitry

Table 2.9 gives the average activity per cycle of each hardware module.If the maximum delay of 10ns is considered to allow for effects of fabrication non-idealities, it allows us time to process 500 channels, at 25KHz sampling frequency, in a time-multiplexed fashion. However, handling of 500 channels does not seem feasible especially at the analog-to-digital converters and the neural signal amplifiers.

Table 2.9: Average Activity per Cycle for Hardware Modules

Module	Average Activity per cycle
Controller	8/8 = 1
Comp Core	5/8
Comp Core Memory (Flip Flops)	8/8 = 1
Coefficient Memory (ROM)	5/8
Pairing Memory per channel/level	2/8 = ¼
Input Buffer per channel (SRAM)	1/8
Per Channel or Level Memory	2/8 = ¼

The overall 32 channel, 4 level system at 0.18 micron process technology with 1.3V VDD and at 6.4MHz of operating frequency consumes an average power of 76μW, including static and dynamic power dissipation, with highest power being consumed by CC memory, the computation core and the controller respectively. Since these three modules are operational most of the time, they account for about 80% of the total power.

2.8.2 Data Compression vs. Reconstruction Error

The DWT circuit outputs interleaved approximate and detail transform coefficients that give a sparse representation of the original neural signal. Coefficient values below a specific threshold can be set to zero to compress the results into a smaller number of bytes. The non-zero coefficients can then be encoded using a lossless encoding scheme and transmitted to the extra-cranial or extra-cutaneous processing units [17, 45]. Choosing the value of the zeroing threshold provides a tradeoff between signal integrity and compression ratio.

To test our DWT circuit implementation on real neural data, a linear-step analog to digital converter was used to convert experimentally obtained neural data into a stream of 10-bit digital values. The data was then processed through the DWT system and results were stored for analysis. In one test, the stored transform coefficients were used to reconstruct the neural signal, and this result was compared to the original signal to measure the quality of reconstruction. This analysis was performed for several different zeroing threshold values to evaluate signal quality verses the amount of compression obtained. The final performance metrics can be defined in terms of three quantities, the root mean squared error (RMS), Shannon's entropy [46], and the assigned threshold.

RMS error is a measure of the average difference between the original and the reconstructed signal. Here the difference between the original signal and the reconstructed signal is comprised of two components: the quantization noise due to finite-word length, and the lossy-compression noise resulting from the thresholding operation. The mathematical representation of RMS error is given by

$$RMS\_error = \sqrt{\frac{1}{N}\sum_{n}^{N}(x_n - \hat{x}_n)^2} \tag{2.21}$$

where x_n is the original signal, $\hat{x}_n$ is the reconstructed signal and N is the length of the signal.

Shannon's entropy is a measure of uncertainty associated with a random signal and can be interpreted as the average minimum message length, represented in bits, which must be transmitted to communicate the exact value of the signal.

Entropy gives the theoretical limit to the best possible lossless data compression for any communication. Entropy can be mathematically represented as

$$H(X) = -\sum_{i=1}^{N} p(x_i)\log_2 p(x_i) \tag{2.22}$$

where N is the total number of possible values (also called symbols in information theory literature) and p(xi) is the probability of occurrence of the ith value.

Increasing the zeroing threshold improves the amount of compression achieved (decreasing entropy) but degrades RMS error. For neural data processed through our DWT circuit, Fig. 2.21 plots the RMS error and entropy as the zeroing threshold increases and confirms the anticipated tradeoff.

Because of this direct tradeoff between RMS error and compression ratio, the zeroing threshold must be chosen to match application requirements; if bandwidth limitations are not severe, a low threshold could maximize signal reconstruction, or if near perfect reconstruction is not needed, a high threshold could reduce power for data transmission.

It is important to note that the measures of entropy and RMS error presented here are also dependent on the signal to noise ratio (SNR) of the input signal. In high SNR signals, most of the data values (which represent noise) have very low amplitude, whereas in low SNR signals the data values are more uniformly distributed. As a result, though the trend of decreasing entropy and increasing

RMS error with threshold holds true for all data sets, the slope of these graphs depends on the particular data set and is dependent on the SNR of the signal.

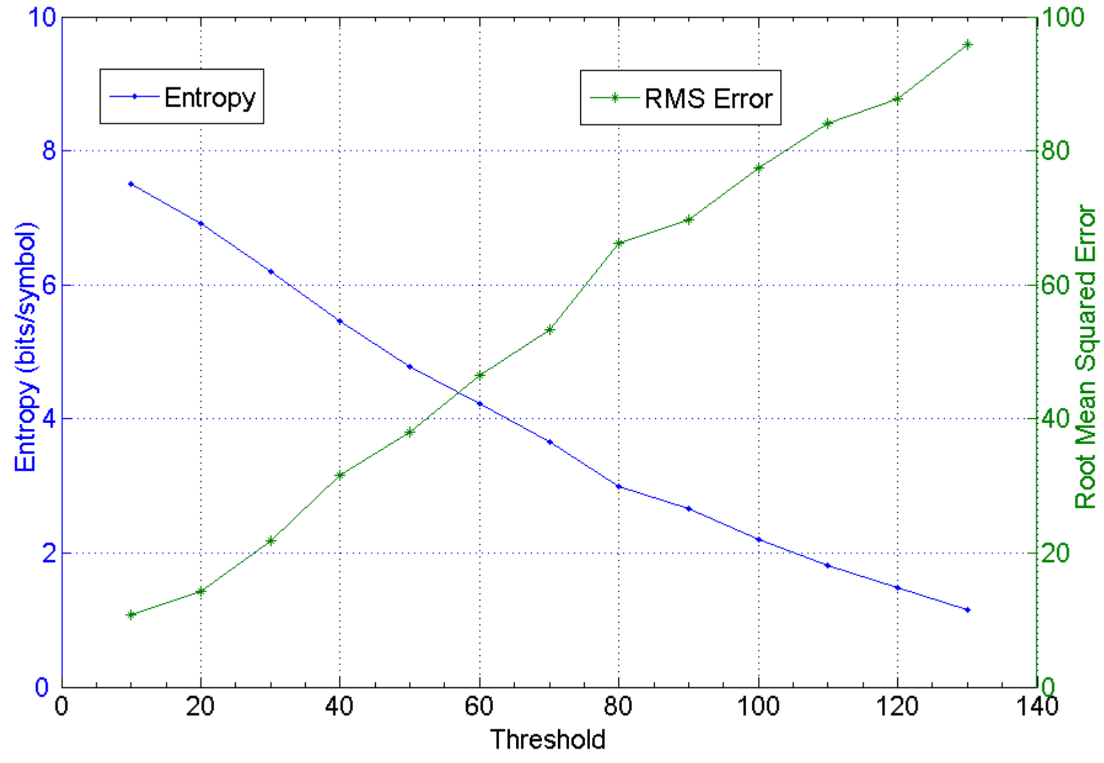


Fig. 2.21. RMS error and Entropy in bits per symbol as a function of threshold value for the neural data set used in our experiments.

Fig. 2.22 and Fig. 2.13 show the original and reconstructed signals for threshold values of 50 and 120, respectively, as obtained in our experiments with real neural data. At a threshold of 50, entropy of 4.78 bits per symbol and an RMS error of 37.95 were obtained. For a threshold value of 120, entropy of 1.48 bits per symbol and an RMS error of 87.69 were measured.

It can be seen that a low threshold value, as in Fig. 2.22 results in a good signal reconstruction so it is suitable for applications that require high quality signal reconstruction. A higher threshold value, as in Fig. 2.23, results in a relatively worse reconstruction; however, it still captures the spikes in the neural signal very effectively and preserves their shapes and sizes.

Since most of the neural signal analysis and applications are based on the information embedded in the spikes only, even this level of reconstruction is highly accurate and useful. Thus depending on the specific application requirements, the threshold value can be chosen to optimally exploit available bandwidth, available power, and desired reconstruction quality.

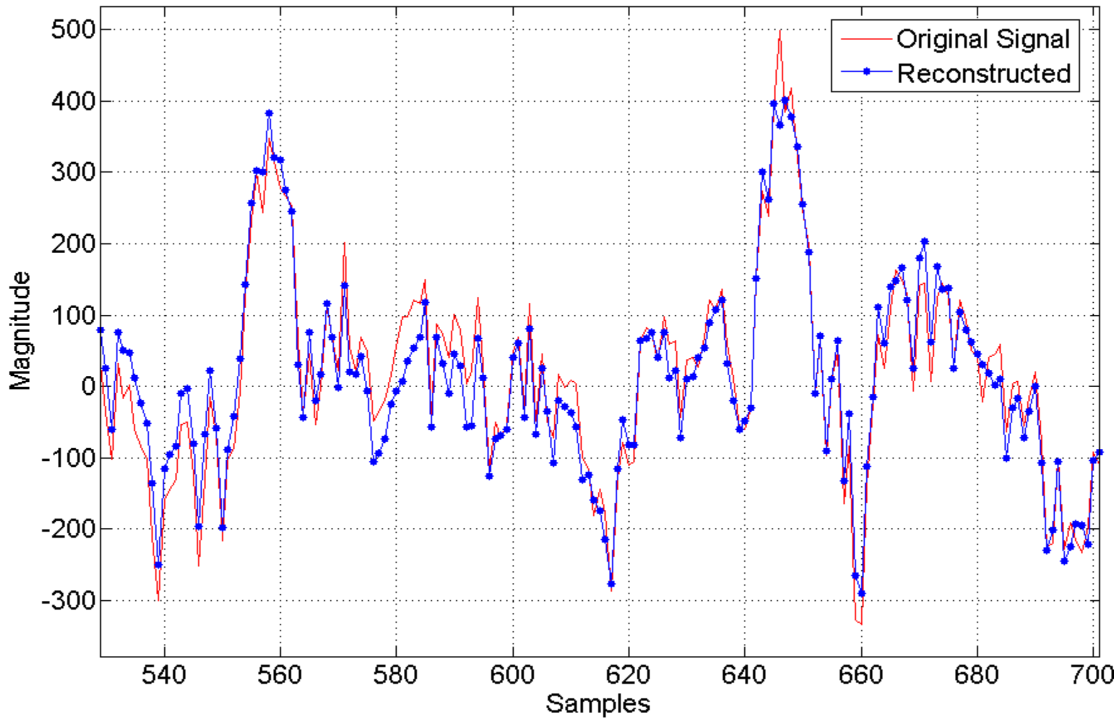


Fig. 2.22 Original and reconstructed signals for Threshold = 50, RMS error = 37.95 and Entropy = 4.78 bits per symbol at 4 levels of decomposition with 10 bits data and 5 bits coefficients.

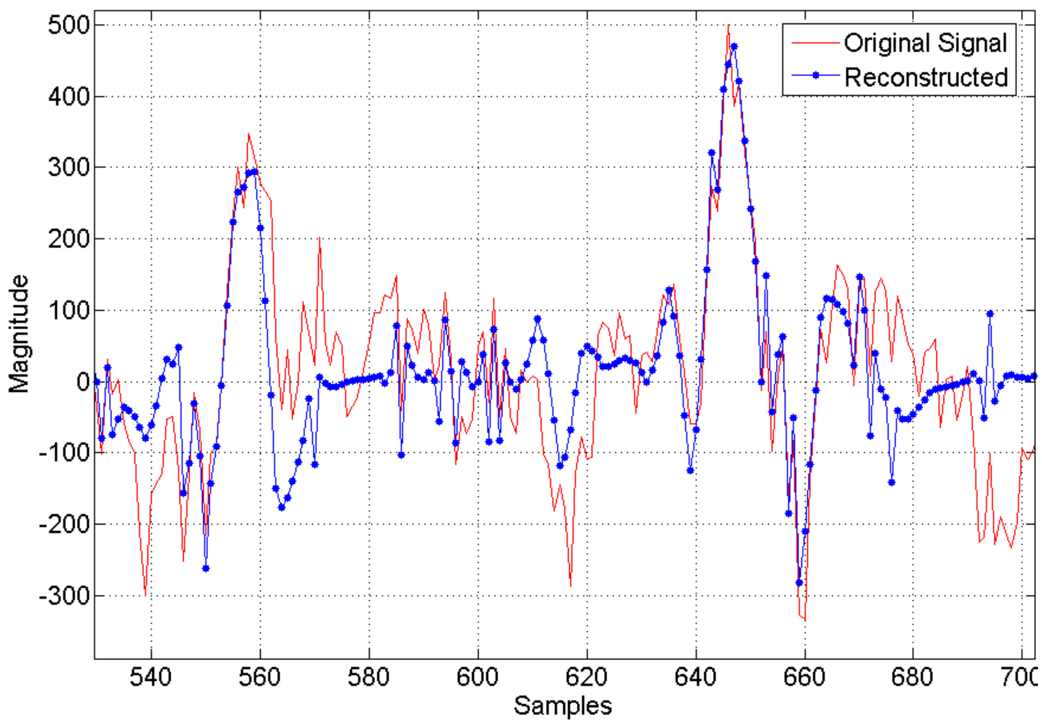


Fig. 2.23. Original and reconstructed signals for Threshold = 120, RMS error = 87.69 and Entropy = 1.48 bits per symbol at 4 levels of decomposition with 10 bits data and 5 bits coefficients.

The resulting DWT circuit can pseudo-simultaneously process multiple channels of incoming neural signals in real time, and its ability to perform multi level DWT enables very high data compression while maintaining signal fidelity. The small size and low power consumption of this DWT VLSI circuit makes it highly suitable for font-end data compression in implantable applications.

2.9 The Design Flow

The design of an ASIC to implement an algorithm in hardware is completed in two main phases, the technology independent phase and the technology dependent phase. The technology dependent phase of the design involves fixed-point algorithm design in Matlab followed by behavioral and structural design in a hardware description language. The ASIC design is verified and prototyped in hardware using an FPGA. Although it is a technology dependent technique, verification using an FPGA has various pros as described in the following section. The FPGA verified hardware description is finally translated to the ASIC design for chip fabrication for a particular technology. Fig. 2.24 gives the high-level steps for the design of proposed ASIC hardware in silicon.

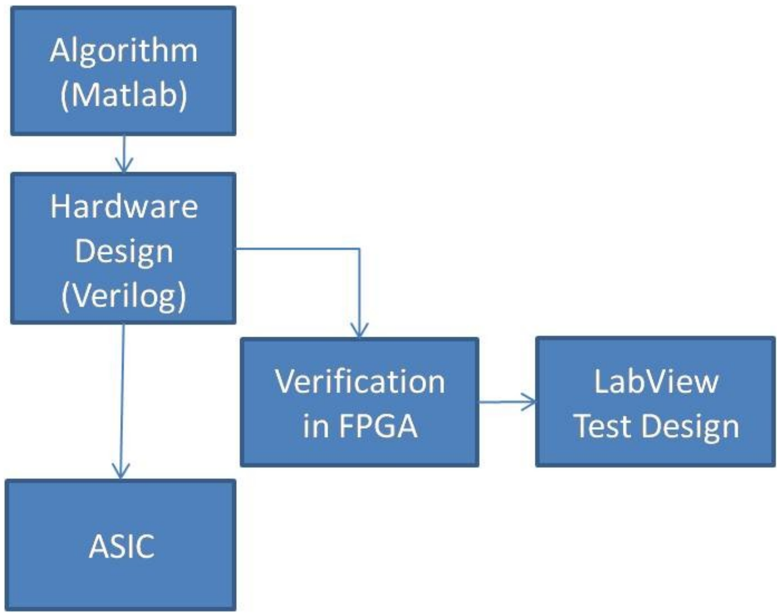


Fig. 2.24: High-level ASIC design flow

The design flow is divided into three phases,

1. Technology Independent Phase
2. Technology Dependent Phase for FPGA based Prototype
3. Technology Dependent Phase for ASIC

These phases are described in detail in the Appendix.

2.10 Integrated Circuit for DWT: Design and Test

2.10.1 Design

A 0.5μm CMOS process was used to design all the blocks of the neural compression engine. The DWT block has been fabricated and the 3mm x 3mm chip is shown in Fig. 2.25. The controller was synthesized using OSU's Standard Cells Library [47], while all other blocks were custom designed for low power and low area. The active components of the prototype 32-channel, 4-level DWT implementation occupy roughly 3.84mm$^{2.}$ The layout for the threshold and RLE blocks requires about 0.95mm^2 of area. Thus the combined compression system is expected to require approximately 5.75mm^2 including global routing for a 0.5μm process. The tested DWT module consumes only 3mW of power while processing 32 channels at 25Ksamples per second, or equivalently 95μW per channel. The power consumption per channel is directly proportional to the neural data sampling frequency.

Power Consumed = 95 μW per channel @ 25KSps per channel

Core Area = 3.84mm^2

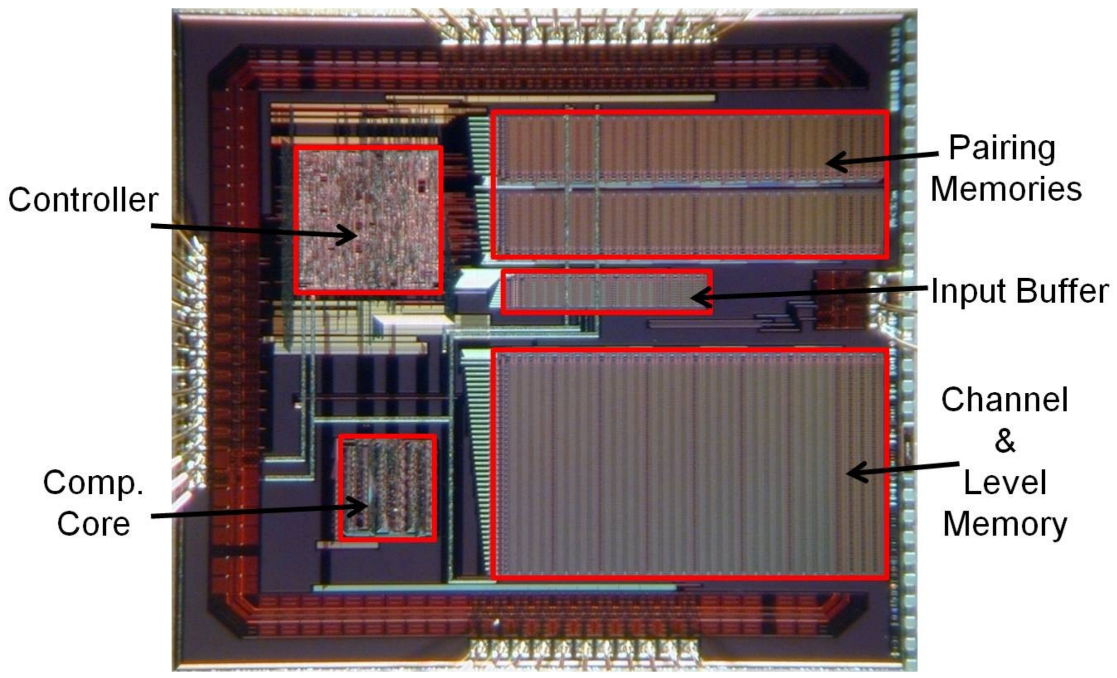


Fig. 2.25. DWT system on chip, fabricated in 0.5μm technology.

2.10.2 Test Setup

To test our designs and algorithms, a stream of experimentally obtained 10-bit neural data was processed through the compression system and the resulting transform coefficients were used to reconstruct the neural signal. The results were compared to the original signal to measure the quality of spike reconstruction versus the compression obtained.

The chip was tested using National Instruments' DAQ card, Labview and an FPGA. Initially Labview and the DAQ card were used to test the functionality of the chip at low speeds. Once verified, the FPGA was used to test the chip at the required frequency of 6.4MHz. The resulting coefficients were an exact match to the results from fixed-point simulations in Matlab.

The PCB shown in Fig. 2.26 was designed and fabricated to allow connection between the ASIC to be tested, DAQ cards and the FPGA. Apart from connecting tips included to provide flexible connections, the PCB also contained several adjustable power supplies to allow chip test at several different supply voltages. In addition, the test board contains voltage level shifters to allow connectivity between different voltage domains. This is necessary since DAQ card uses 5V digital IO, the FPGA uses 3.3V digital IO while the ASIC can operate at a range of supply voltages.

The chip was tested using National Instruments' DAQ card, Labview and an FPGA. The virtual interface used to control the data acquisition is shown in Fig. 2.27. Initially Labview and the DAQ card were used to test the functionality of the chip at low speeds. Once verified, the FPGA was used to test the chip at the required frequency of 6.4MHz. The resulting coefficients were an exact match to the results from fixed-point simulations in Matlab. The same setup is later used to connect the FPGA directly to the DAQ cards during FPGA based prototyping of other modules detailed in next chapter.

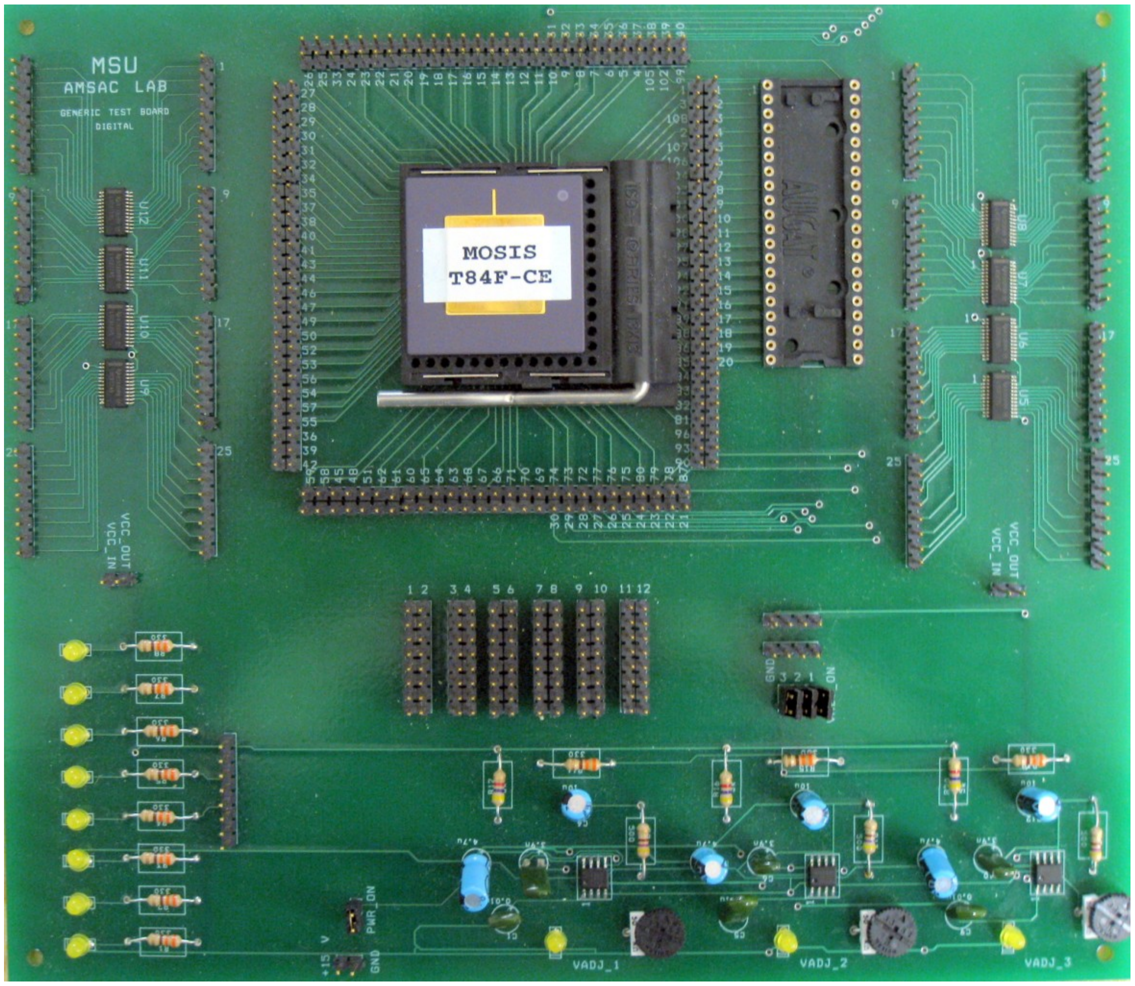
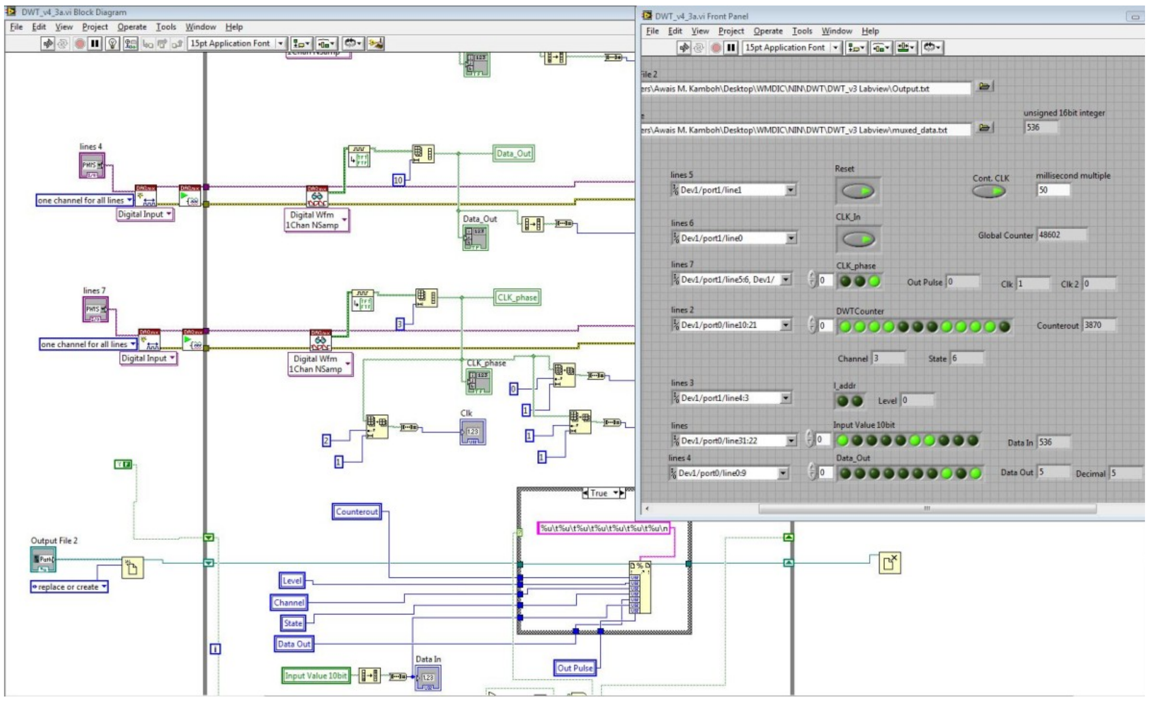


Fig. 2.26. Photograph of Test PCB designed to interface ASIC with FPGA and DAQ cards.



File 2
ers\Awais M. Kamboh\Desktop\WMDIC\NIN\DWT\DWT_v3 Labview\Output.txt

ers\Awais M. Kamboh\Desktop\WMDIC\NIN\DWT\DWT_v3 Labview\muxed_data.txt

unsigned 16bit integer 536

lines 5: Dev1/port1/line1
Reset
Cont. CLK
millisecond multiple 50
lines 6: Dev1/port1/line0
CLK_In
Global Counter 48602
lines 7: Dev1/port1/line5:6, Dev1/ 0
CLK_phase
Out Pulse 0
Clk 1
Clk 2 0
lines 2: Dev1/port0/line10:21 0
DWTCounter
Counterout 3870
Channel 3
State 6
lines 3: Dev1/port1/line4:3
I_addr
Level 0
lines: Dev1/port0/line31:22 0
Input Value 10bit
Data In 536
lines 4: Dev1/port0/line0:9 0
Data_Out
Data Out 5
Decimal 5

Fig. 2.27. Labview interface to control DAQ cards providing input data and recording output data from the ASIC.

2.11 Improvements in DWT

A second version of the DWT block has been designed to improve the functionality and power consumption of the block.

1. Channel Selectivity

When microelectrode arrays are used for recording from the cortex it is not possible to position each electrode independently at the desired location. As a result, in general a significant number of electrodes fail to capture any signal at all. It is thus not useful to waste precious transmission bandwidth to transmit coefficients from those channels. The initial design captured and transformed signals from all the 32 channels in a time-multiplexed sequence. The block has been modified to be able to handle selectable number of channels. The block can now be instructed to compress 32, 16, 8, 4, 2 or 1 channel at a given time.

Since the reduced number of channels does not require as many computations per second as the maximum number of channels, the block can be clocked at slower frequencies to save dynamic power. Dynamic power is linearly dependent on the frequency of the operation.

2. **Overflow**

The second version of the DWT block has been designed to incorporate better handling of data overflow at the output of the computation core. If an overflow is detected at the output of an arithmetic computation, then the sign of the correct result is determined and the output is truncated to the corresponding maximum value. Thus, a negative overflow is replaced by a -511 and a positive overflow is replaced by a +511 value.

3 Design of Implantable Neural Interface Node

3.1 Motivation and Approach

The work in Chapter 2 establishes the design of a highly efficient circuit for computing the multi-level DWT of neural signals. However, for this DWT block to be useful within a neural implant system, it must be combined with several other functional circuit blocks. The DWT does not provide compression on its own, however it enables compression by providing time-frequency transformation and in doing so compresses energy into a few coefficients. Coefficients with ‘significant’ energy are separated from those with insignificant energy by the use of multi-level threshold. The lesser energy coefficients are reduced to zero providing lossy compression. The resulting stream then undergoes lossless compression through a run length encoder block. The DWT, the multi-level threshold and the run length encoder collectively perform neural data compression. Apart from the compression engine, the implantable node needs two more interfaces, the analog frontend interface and the RF transceiver interface. The analog interface is controlled by a dedicated controller while the RF transceiver is interfaced with the communication controller which is also responsible for running the communication protocol and providing error free communication. The aim for the communication controller is to provide maximum data throughput . A global controller is necessary to synchronize the data flow between blocks, to manage various configurations, and also maintains the power management block. This chapter describes the circuits designed to meet the needs of these individual blocks as well as their collective operation under the constraints of area and low power.

3.2 Architecture Overview

Fig. 3.1 provides a system level view of an integrated neural data compression circuit fabricated on the back of a microelectrode array of the NIN that is implanted in the cortex and communicates wirelessly to an EPU. Neural signals from multiple channels are amplified, digitized and fed to the DWT block, which generates a sparse representation of these signals. The threshold block serves a dual purpose of de-noising and spike detection. The Run Length Encoder (RLE) removes the redundancy from the streaming data which is then formatted into packets for wireless transmission and sent to the transceiver. The system employs both lossy and lossless compressions to reduce the data, where the DWT, threshold and RLE blocks form the compression engine.

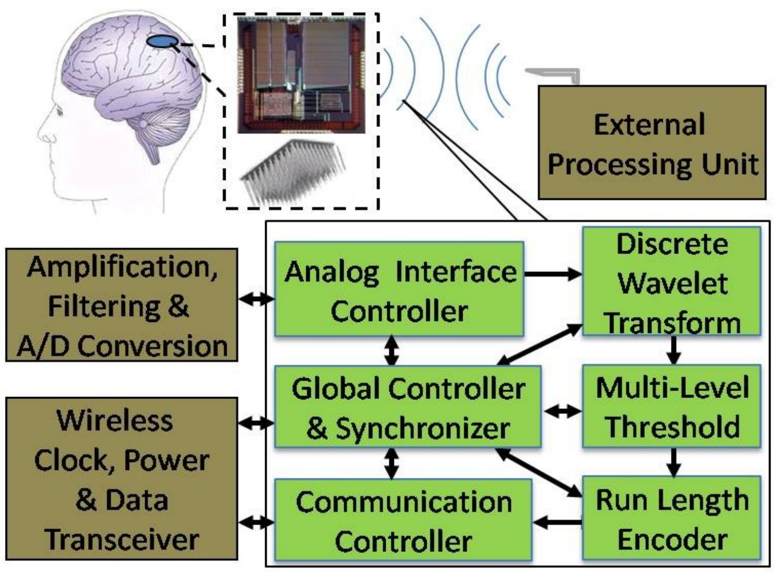

External Processing Unit
Amplification, Filtering & A/D Conversion
Analog Interface Controller
Discrete Wavelet Transform
Global Controller & Synchronizer
Multi-Level Threshold
Wireless Clock, Power & Data Transceiver
Communication Controller
Run Length Encoder

Fig. 3.1. Block diagram of the implantable neural data compression engine and its position within an implantable neural recording system.

3.3 Data Compression Engine at the NIN

3.3.1 Analog Interface Controller

Fig. 3.2 shows the analog front end to receive, amplify, and filter signals from the electrodes that has been assumed in designing the digital control portion of the NIN. The individual signal channels are AC coupled to remove any DC offset and are then passed through the pre-amplifiers which provide much needed buffering at this stage. The channels are then fed to a transmission gate based analog multiplexer which selects one channel at a time. The output of the mux contains a superposition of the LFP and the neural spikes. This signal is then passed through a band pass filter followed by a variable gain amplifier. A number of amplifiers have been reported in literature which satisfy the gain-bandwidth requirements of this study [4, 48]. A modified version of [48] is being designed to allow variable gain amplification. Output of the amplifier contains amplified neural spikes. Depending upon the mode of operation, the amplified spike train or the LFP is then relayed to the ADC which converts the analog voltage into 10-bit digital data. The analog frontend is expected to cover an area of 1.77mm^2 with a power dissipation of 1.1mW.

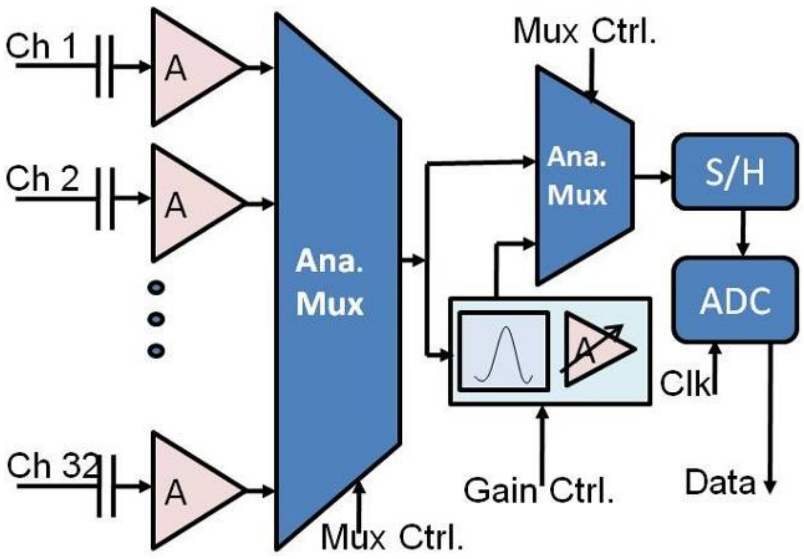


Fig. 3.2. Analog module for multichannel neural data recording with variable gain amplifier.

Operation of this frontend analog module is controlled by a state machine based controller, which depending upon the runtime configuration settings, allows the flexibility to choose individual channels for recording. The controller time multiplexes the channels before the data is handed over to DWT block as well as it selects the data path for different modes of operation selecting between spikes and local field potentials. The controller allows setting individual channel gains, out of four possible gain settings, for each of the channels. It is also responsible of providing appropriate clock to the ADC.

3.3.2 Discrete Wavelet Transform

As detailed in Chapter 2, based on the chip real estate available to this research project, the DWT block shown in Fig. 3.3 was designed to support up to 32 channels of data simultaneously with 4 levels of wavelet decomposition. Multiple decomposition levels generally result in fewer significant coefficients. The relatively long intervals between samples of neural signals allow for computation hardware that prioritizes power and area efficiency over speed [43]. Derived from our prior system-level analysis [35], the power-area product can be minimized by an architecture that sequentially evaluates the DWT of multi-channel data in real time. The lifting scheme is used to compute results; it has been shown to require fewer computations than convolution based filtering [24].

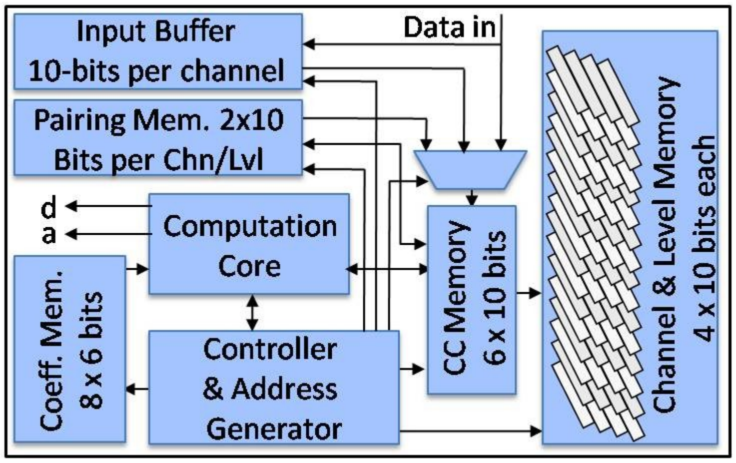


Fig. 3.3. System diagram for sequential calculation of DWT.

The computation core performs sequential calculation of DWT coefficients. The memory blocks store temporary data and intermediate results and have been partitioned based on different access patterns. The controller manages timing and data flow among the blocks. The controller was synthesized from a library while the other blocks were custom designed to minimize power and chip area.

3.3.3 Multi-Level Threshold

Coefficients at the output of the DWT block can be viewed as sparse packets of energy which do not, by themselves, result in any compression. However, following the DWT by a thresholding stage, which reduces the in-significant low-energy coefficients to zero and lets high-energy coefficients pass, does permit compression. The low-energy coefficients have little or no significant information and mainly contribute to noise. The high energy coefficients invariably correspond to different spikes and events in the neural signal, allowing spike sorting even without reconstruction [17]. The values to which the thresholds are set are of critical importance since they determine both the quality of reconstruction and the final rate of compression.

Methods to determine the optimal threshold is an ongoing investigation. However, it has been established that the best compression is achieved by using separate threshold values for each decomposition level of each channel [17]. Thus, DWT results from each level of each channel must be treated separately and will form a stream of data containing information that is virtually independent of the information from other channels and levels within the data stream. The

DWT and the threshold blocks are combined to implement the lossy compression on the input signals. The multi-level thresholds thus result in compression and denoising of the signal at the same time.

Fig. 3.4 shows the main functions of the threshold and RLE stages. The threshold block includes a set of memory registers that contain threshold values for each level of each channel. Since, according to figure 3.6, the last level of DWT produces two separate coefficients streams, an N level system would have N+1 threshold values for each channel. These values would be determined externally and then stored into the memory sequentially before the DWT operation begins. Any of these values can be updated during system operation. The DWT block generates values in signed-magnitude form. The magnitude is compared against the threshold value using a magnitude comparator; if found smaller than the threshold, a 10 bit zero is generated at the output. If equal or greater, the original value is regenerated at the output.

As explained later in this chapter, the overall compression system is designed to operate in two modes: Monitor mode and Compression mode. In Monitor mode, the system bypasses the DWT, threshold and RLE blocks and sends the uncompressed neural signal directly to the transceiver. Monitor mode is used by the EPU to analyze the statistical properties of individual channels and calculate the optimal threshold values for each channel and level. These threshold values are then transmitted back to the implanted system to set compression parameters for use in Compression mode, where all system blocks are activated to compress data.

3.3.4 Run Length Encoder

Several lossless data compressors exist in literature, with varied computational complexity and storage requirements. A few popular techniques are Huffman coders, Lempel Ziv coder, arithmetic encoders and their variants. Most of these algorithms require prior statistical knowledge of the incoming data set, and thus the rate of compression achieved is directly related to the accuracy of this prior information. Since these algorithms are variable length encoders, under certain conditions, they approach the theoretical limits of compression, bounded by the entropy of the incoming signal. However, these algorithms require prohibitively large storage to maintain the dictionary of codes. Use of Huffman coding is not possible, since in this application, the size and statistical properties of the source alphabet (number of possible data values) depends on the threshold values, which are ideally controllable by the neuroscientist.

The only statistical information available for our thresholding compression system is that much of the data stream at the output of the threshold block consists of zeroes. Run Length Encoding is best suited for data with long repetitive strings of values; in addition, it is very conservative in required hardware resources. Because we expect long strings of zeros at the output of the threshold block, RLE is a good lossless compression choice. Though RLE is not an optimal encoding scheme in general, when given very long repetitive sequences it approaches the performance of near-optimal algorithms.

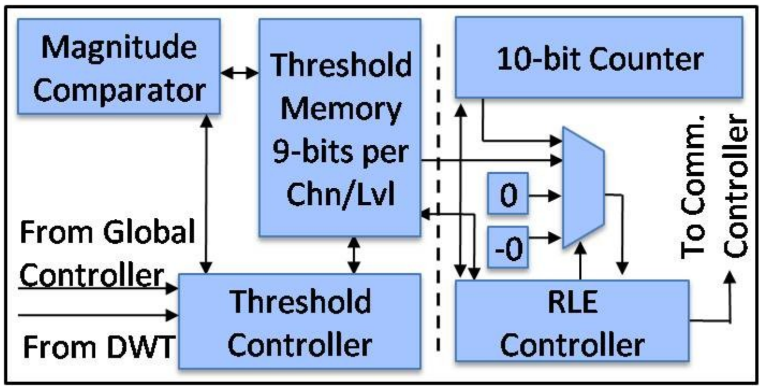
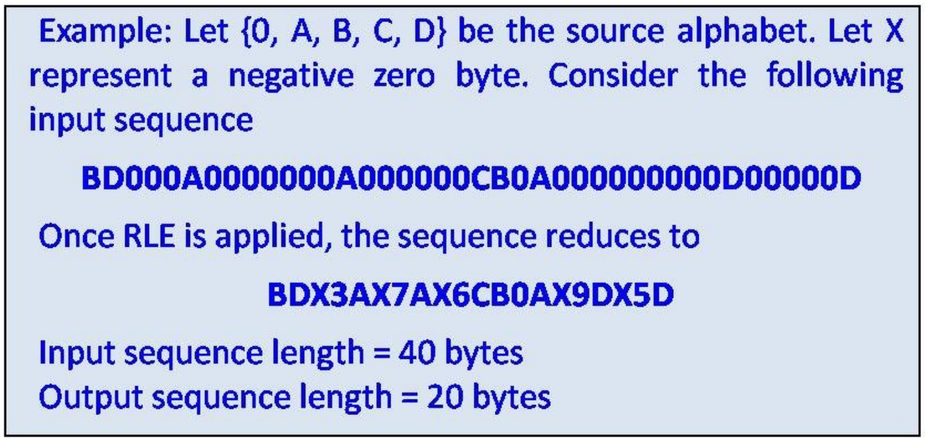


Fig. 3.4 System diagram for the Threshold and the RLE blocks.

Given that, for this implementation, a byte refers to 10-bit values in sign magnitude form ranging from -511 to +511 and that a 10-bit counter can count up to 1023, our implementation of RLE can be summarized by the following rules.

1. Transmit all non-zero bytes as is.
2. Convert all negative zeros (represented by X) to positive zeros.
3. Replace a sequence of zeros (two or more) with an X (negative zero) byte and a zero-count byte.
4. If the zero count reaches 1023, send 1023 and restart a new sequence of zeros.

Following these rules, the example RLE operation of Fig. 4 shows the original 40-byte sequence has been reduced into a 20-byte sequence. Sequences from real neural recordings have been observed to yield much better compression ratios. Since we do not expect long sequences of repeating non-zero values, this implementation compresses only the sequences of zeros. This scheme results in fixed length codes, which have a computational resource advantage.



Example: Let {0, A, B, C, D} be the source alphabet. Let X represent a negative zero byte. Consider the following input sequence

BD000A0000000A000000CB0A000000000D00000D

Once RLE is applied, the sequence reduces to

BDX3AX7AX6CB0AX9DX5D

Input sequence length = 40 bytes
Output sequence length = 20 bytes

Fig. 3.5 Example of the system specific run length encoding.

3.4 Communication Controller and Interface

A communication system is required to communicate neural spike data from the NIN to the MIM and at the same time take configuration settings and commands from the MIM to the NIN. This communication takes place across the skull and the scalp, thus the NIN and MIM are effectively only a few inches apart. A Communication Controller module is designed to control the transmission, reception of the data and general working of the RF transceiver. The controller must adhere to a few requirements for the system to work effectively.

Power dissipation is again a primary concern while designing the communication system. Since the human tissue is known to absorb higher radio frequencies more than lower frequencies, carrier frequencies in the range of a few megahertz are suitable. The system has been designed to abide by the FCC Part 15 recommendation of 13.56MHz.

3.4.1 Packet Structures

The neural data is considered pristine, which means error detection is necessary at both ends of the communication channel so that the packets can be retransmitted until error free data has been received. Two different packet structures are used during communication, one for uplink from NIN to MIM and the other for the downlink from MIM to NIN. As shown in Fig. 3.6, the uplink packet, also called the *Data Packet*, is large and contains eighty 10-bit values representing the compressed neural data. The packet contains 8-bits of channel and DWT level information as the first data value in the packet. This channel and level information is only required for the first data value and subsequent values conform to a predetermined sequence of channels and levels set by the DWT block. The packet contains one bit to represent the current *Packet ID* to establish proper data sequencing at the receiving side. It also contains a *Command Request ID* as an acknowledgement for the last packet received. If the request ID is the same as the ID of the last transmitted packet, the receiver is requesting retransmission of the last packet. Finally, the packet contains 8-bits of *Cyclic Redundancy Check* (CRC) for error detection. A standard 8-bit CRC (CRC-8-

CCITT) is used for this application to detect anticipated error patterns possible during communication. The polynomial for the CRC-8-CCITT is given as $x^8 + x^7 + x^3 + x^2 + 1$

The CRC bits allow error-detection at the receiver side. The choice of CRC depends on two factors; the length of the message, and the desired percentage of error detection. No CRC choice can guarantee absolutely error free communication, however, greater number of CRC bits correspond to better error detection capabilities. These capabilities are achieved, essentially, at the cost of higher redundancy and thus lower throughput. This code guarantees to detect all single bit and double bit errors as well as all odd number of errors. It also guarantees detection for burst errors of up to 8 consecutive bits. Some errors may pass through if the error pattern results in another valid pattern. For the 8-bit CRC there is an average of 1 error pattern which will not be detected for every 255 which would be detected, that is, we should be able to detect 255/256 or 99.6% of all possible 8 bit errors [49]. As a result, the total size of the data packet becomes 818 bits.

The downlink packet, also called the *Command Packet*, is used to send commands from NIN to MIM. This 25-bit packet contains 8 bits for the *Command Code* and 10 bits for corresponding *Command Data*. As in the Data Packet, the Command Packet also contains identification and request bits. A 5-bit CRC is sufficiently powerful for a packet of this length. The polynomial used to generate CRC-5 is $x^5 + x^4 + x^2 + 1$.

Command Code [8-bits]	**Command Data [10-bits]**	**Command Packet ID [1-bit]**	**Data Request ID [1-bit]**	**CRC [5-bits]**

Channel & Level [8-bits]	**Compressed Data [800-bits]**	**Data Packet ID [1-bit]**	**Command Request ID [1-bit]**	**CRC [8-bits]**

Fig. 3.6 (Top) MIM to NIN Command Packet. (Bottom) NIN to MIM Data Packet.

3.4.2 Protocol Design

A major limitation in the protocol design is the use of a half-duplex transceiver, chosen to conserve power and bandwidth. As a result, a single 1Mbps channel is used for both uplink and downlink data transfer. A stop-and-wait based approach has been taken to meet the requirements of the system, including a need for the NIN to wait for a packet to fill before transmission of streaming data. At the same time, the MIM has to transfer various configurations and commands to the NIN. However, the MIM does not have to wait for streaming data to become available. Since the system clock is generated wirelessly, another issue is the transmitter-receiver re-synchronization once a disruption takes place.

Automatic Repeat Request (ARQ) is the error control method based on acknowledgements and timeouts to achieve reliable data transmission. The Stop-and-Wait protocol has been customized and redesigned to cater for the fact that NIN is generating streaming data and thus may have to wait for data to become available for transmission. The protocol in its innate form is known to be not as efficient as other more sophisticated window-based variations of ARQ e.g. Selective Repeat (ARQ) or Hybrid (ARQ) but since the error characteristics of communication across human tissue for such an application are unknown and are highly dependent upon the ambience and transceiver coil alignment, it is not evident if any gain in efficiency would be achieved by switching the protocol to any of the other variations. Several sophisticated window-based protocols have been reported in the literature. However, with streaming data, where the transmitter sometimes has to wait for the data to become available, window-based protocols may take very long to hand channel control back to the MIM. This would preclude instant changes in configuration of the NIN. In addition, the protocol presented here has the obvious hardware area and complexity advantage of not having to maintain a number of previous packets. The other protocols require a large buffer to maintain a 'transmission window'. Since this buffer is based on hundreds of bits of shift registers, the hardware cost in comparison to the other blocks in the system is tremendous. However, improvements can be made to the current protocol or the choice of CRC etc depending upon the profile of error statistics generated and collected from the first generation of design.

Fig. 3.7 shows the flow diagram of the modified protocol. The diagram depicts normal flow of data as well as error handling. Command Packets are represented with a 'C,' Data Packets are represented with a 'D,' and the acknowledgement is represented with an ‘A’. Three important parameters in the flow diagram are the three configurable timeout values described below.

An essential feature included in the protocol is the programmable ‘*Data Not Ready*’ timeout, referred to in Fig. 3.7 as Timeout N1 (T_{N1}). This timeout occurs when the MIM requests new data from NIN but the NIN packet is not filled completely and thus not ready for transmission. T_{N1} is the maximum time that the NIN waits before bit-stuffing and transmitting the packet if it is incompletely filled. This timeout is only relevant when neural activity is low and the spikes are sparse. A packet is transmitted as soon as the buffer fills; however, if a single spike is followed by a very long string of zeros it can take up to approximately 100ms to completely fill the Data Packet. This ‘delay’ in transmission may not be acceptable for certain applications where real-time spike recording and decoding is important. The default value of T_{N1} is set to 8ms, but it can be reconfigured to any temporally relevant value of less than 32ms through a command from the MIM.

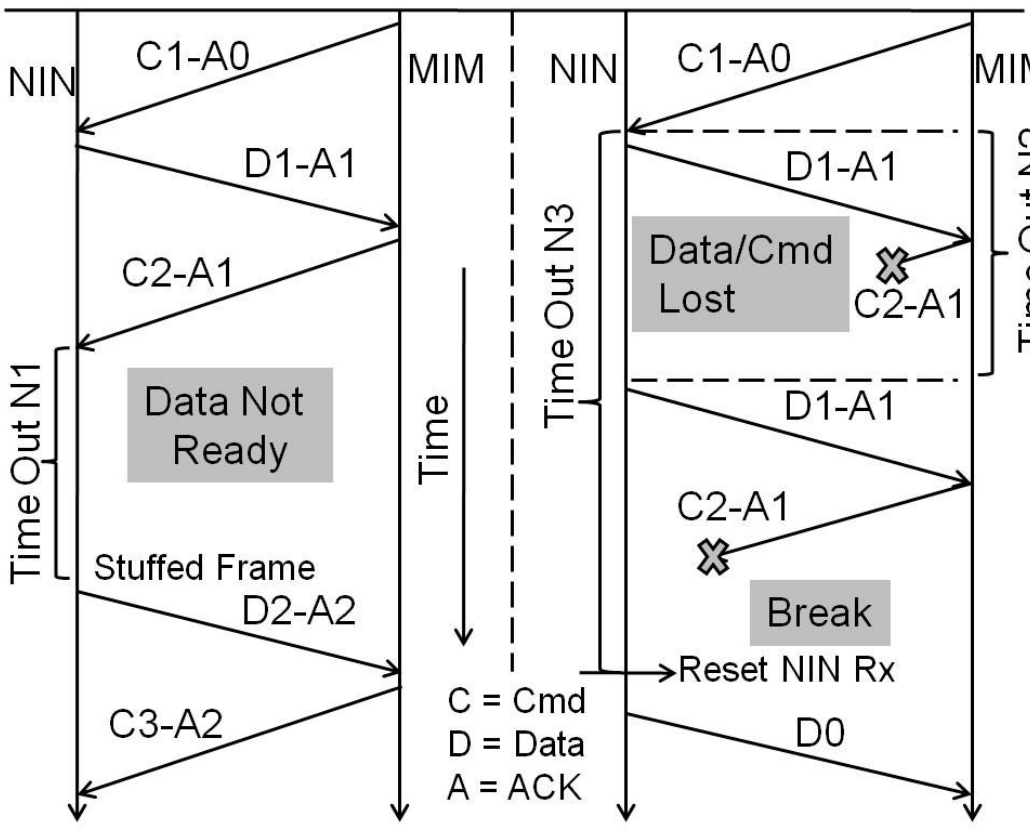


Fig. 3.7. Flow diagrams for the communication protocol. The diagram shows normal flow as well as handling of error conditions.

The MIM transmits a Command Packet as soon as it receives a Data Packet from the NIN; it does not have to wait for a new command to become available and always maintains a preconfigured Command Packet. If the NIN does not receive a Command Packet within a specified time, then Timeout N2 (T_{N2}) occurs. The NIN assumes a *'Data or Command Packet lost'* error and retransmits the Data Packet. Let *'T_D'* be the time to transmit a Data Packet and *'T_C'* be the transmit time of a Command Packet. With a transmission frequency of 1Mbps, the 818-bit Data Packet takes T_D=0.818ms for transmission, and the 25 bit Command Packet takes T_C = 0.025ms. T_{N2} can then be expressed as

$$T_{N2} \geq T_D + T_C + 2T_P \approx 0.85ms \tag{3.1}$$

where T_P is the packet propagation delay, which is negligible due to the close proximity of the NIN and MIM. Similarly, the idle time between reception of a packet and transmission of a preconfigured packet is also negligible.

The reconfigurable Timeout N3 (T_{N3}) acts as a watchdog timer for the communication controller. If the NIN does not receive a packet from the MIM for three consecutive Data Packet transmissions, then the NIN communication controller resets itself assuming a '*Break*' in communication requiring re-synchronization. T_{N3} can be expressed as

$$T_{N3} \geq T_{N1} + 3T_{N2} \tag{3.2}$$

The default value of T_{N1}= 8ms gives $T_{N3}\approx$ 10.6ms. If needed, a reset command can be issued through the MIM that resets the NIN and initializes all settings to their default values.

Fig. 3.8 shows the functional diagram of the communication controller. The transmitter and the receiver use serial data in and out lines in conjunction with corresponding clock signals to communicate with the transceiver. The controller contains two frames. At any given time, one frame is the active frame while the other acts as a reserve frame. The reserve frame is filled while data from the active frame are being transmitted. It is desirable to be able to transmit the active frame twice (in case of an error) before the reserve frame fills and data overflows. As a result, from (3.1), the buffer should be large enough to store incoming data for a time greater than $2^*T_{N2} \approx 1.7$ ms seconds. This can be achieved by settings the appropriate threshold and frame size.

The RF transmitter and the receiver use serial data in and out lines in conjunction with corresponding clock signals to communicate with the communication controller. The NIN controls the select line which enables the transceiver or the receiver according to the protocol presented in Fig 3.7. The transmit and receive clocks have a frequency of 1Mbps where the data-bits are transmitted or received at the positive edge of the clock as shown in Fig 3.9. Tx/Rx Select signal is generated by the communication controller in addition to the Tx clock and Tx data. The Rx clock and Rx data are the signals generated by the RF receiver.

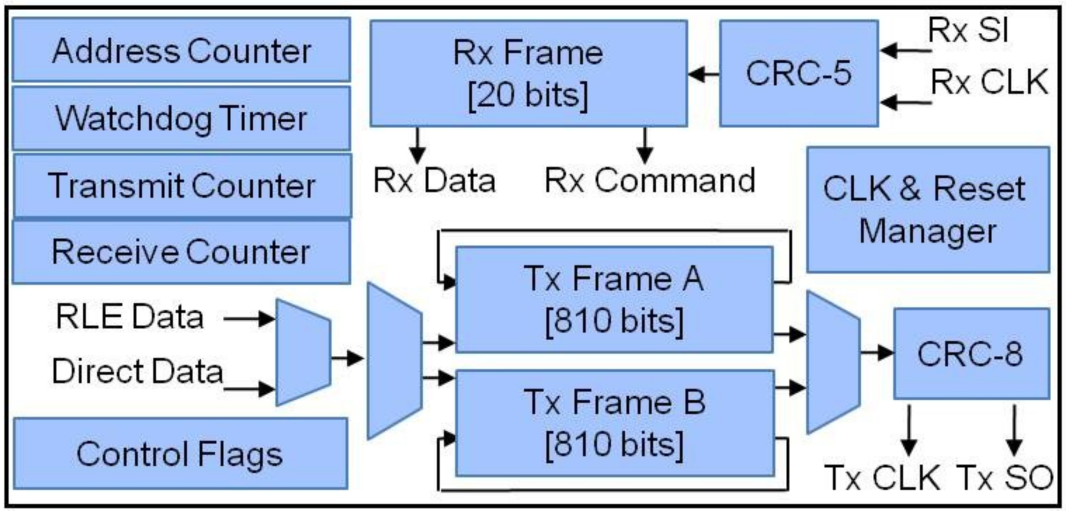


Fig. 3.8. Functional diagram for the communication controller.

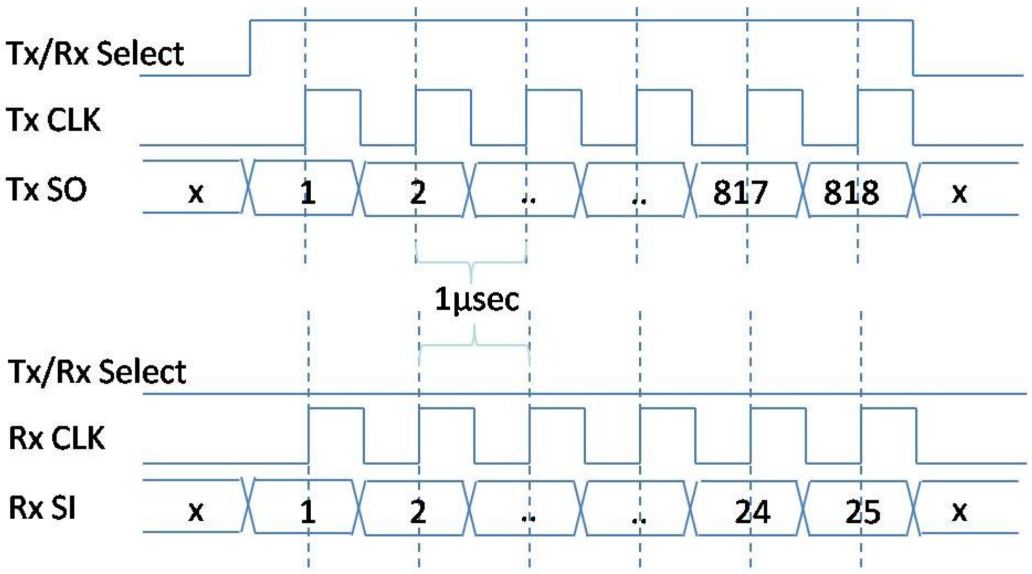


Fig. 3.9. Timing diagrams (Top) Packet transmission from NIN. (Bottom) Packet reception at the NIN.

3.4.3 Efficiency

The analysis and results of the DWT, threshold and RLE blocks have been presented in [50]. The communication system can be seen as operating in two phases. In the beginning, the NIN has data to send to the MIM and the MIM has configuration commands to send to NIN. During this phase the protocol efficiency is maximum and is given by

$$\eta = \frac{T_D + T_C}{T_{N2}} \approx 1 \tag{3.3}$$

Steady state is reached when the initial configuration is complete and the MIM has very few commands for the NIN. During this phase, the efficiency of protocol can be given by

$$\eta = \frac{T_D}{T_{N2}} \approx 97\% \tag{3.4}$$

The efficiency expression in (3.4) is valid when there are no errors during transmission. The probability of error because of a CRC failure or lost packet is denoted by P_e. If T is the amount of time taken to successfully transmit a packet and receive its acknowledgement in the presence of errors, then the average time it takes to successfully transmit a packet in presence of errors, E[T], is given by

$$E[T] = T_{N2}(1 + \frac{P_e}{1 - P_e}) \tag{3.5}$$

During steady state, in the absence of any errors, the data throughput of the compressed coefficients can be given as

$$\text{NIN Throughput} = (\text{No. of Data Bits} / \text{Total Bits}) * \text{Tx. Rate} \qquad (3.6)$$

$$= (800/(818+25)) * 1\text{Mbps} \approx 950\text{ Kbps}$$

Thus, a 1Mbps channel can transfer data generated at rates less than 950Kbps. However, as with other protocols, if successive packets result in failed CRCs or are lost, then multiple retransmissions may be required for each packet. If at the same time there is a burst of activity in the neurons, then this may result in data being generated at a higher rate than the throughput of the channel, raising the possibility of a buffer overflow and thus data loss at the NIN.

3.5 Global Controller

In addition to the local controller for each block, a global controller binds all the components together into a system and is responsible for the correct operation of the NIN. It is responsible for synchronization of data flow between blocks, generation of respective clocks and the engagement or bypass of certain blocks in the data path. The controller also contains a power management module. However, most importantly, the global controller interprets the different commands from the MIM and changes the configuration and operation of the NIN accordingly. The operation of the NIN is controlled by several configuration registers. The values in these registers can be changed by the MIM to control the operation of the NIN.

3.5.1 Modes of Operation

The system operates in three modes controlled by the configuration registers. The first two are monitor modes in which no signal processing is employed and the data is transmitted without compression. These modes are called *monitor spikes* (MS) mode and *monitor LFP* (MF) mode, where LFP refers to local field potential. Because data is uncompressed in monitor modes, the obvious bandwidth limitation dictates that only one selectable channel can be active at a given time. In MF mode, the system bypasses the high gain amplifiers and routes the analog signals directly to the ADC. The signal is transmitted out without any further processing. The transmitted signal thus contains the information embedded in the LFP as well as the neural spikes. In MS mode, the amplifiers and filters are activated resulting in a signal that contains only the neural spikes and is free of LFP effects. Gain of the amplifiers is adjustable and can be increased or decreased to four different settings.

The third mode is called the *compression mode* (CM). This mode engages the digital signal compression blocks in the NIN and is capable of processing up to 32 channels simultaneously. The configuration registers dictate which channels will be sampled and which, if any, will be ignored. The channels are sampled in a time division multiplexed manner maintaining separate gain settings for each channel. The ADC output is then fed to the DWT block which computes the wavelet coefficients up to four levels of scaling. These coefficients are generated in a predefined sequence. They are then passed through the threshold block, which keeps the significant coefficients and discards the insignificant values. The MIM assigns appropriate values to each threshold, which are stored on the NIN. The resulting signals are sent to the RLE block that compresses the data and forwards it to the communication controller for transmission to the MIM.

Upon reset, the system will sample channel one at 25Ksamples/s in MS mode. The MIM receives and processes the uncompressed data to compute appropriate thresholds and transmits the resulting values back to the NIN. Then the NIN is configured to monitor the next channel and the procedure repeats. Once the thresholds for all the channels have been set, the NIN can be reconfigured for operation in CM mode and begin processing all channels simultaneously.

3.5.2 Power Management

The global controller also manages power for the NIN. Two main techniques are used to conserve power. First, dynamic frequency scaling (DFS) is employed whenever the NIN system is not required to function at its maximum capability; the clock frequency is scaled down, which linearly effects the power dissipation. A dedicated block in the global controller is used to generate clock signals for all the NIN blocks. The controller also disables any components not on the signal data-path for a given mode of operation, reducing unnecessary transistor switching. The analog, digital and RF modules are being designed to work with separate supply voltages, each working at the minimum required voltage to conserve power. Reducing supply voltage increases signal propagation delay. However, as the NIN has a pipeline-based architecture, the performance of components is not limited by the signal propagation delays, which allows for the use of dynamic voltage scaling (DVS) in addition to DFS for future generations of the design.

3.6 RF Data and Power Receiver

A transceiver is responsible for inductively transferring power from the MIM to the NIN as well as providing the master clock to the NIN. The transceiver also allows bidirectional data transfer between the two modules. Two pairs of perpendicularly aligned coils are required for these transfers. Since the communication protocol is half-duplex only one module is transmitting at a given time, this allows for the use of one only pair of coils for both directions of communication.

There has been extensive research in designing high data rate, high efficiency RF transceivers and antenna coils, and various designs have been published recently [48, 51]. The design presented in [51] can be modified accordingly and used in our application. Fig. 3.10 shows the functional diagram of the RF power, clock and data transmission circuit from MIM to NIN. It must be noted that the design of the transceiver is out of the scope of this thesis. The design is described here to help understand the blocks the global controller must interface with.

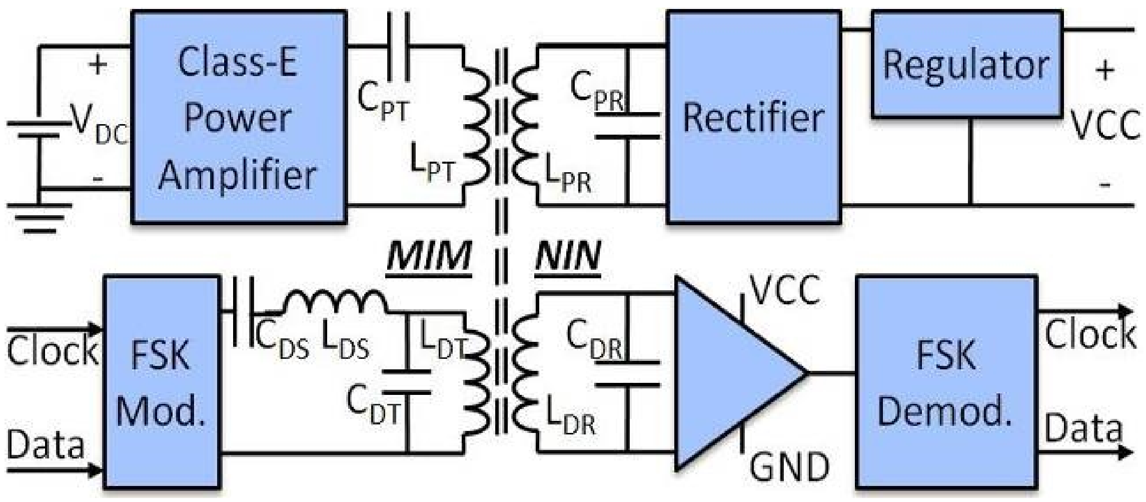


Fig. 3.10 Functional diagram of circuits for inductive power and clock transfer and RF data transmission from MIM to NIN.

The NIN and the MIM are separated by the skull and the scalp and the coils transfer the power and data inductively. An LC tank is used in conjunction with a power amplifier to generate the power carrier which is received at the NIN coil, rectified and regulated to generate a constant and stable power source for other blocks in the NIN. The data is transferred using frequency shift keying (FSK) modulation and high quality LC tank. The receiver amplifies and demodulates the received signal to regenerate the transmitted data bits. The RF transceiver covers an area of 2.2mm^2 and consume power up to 7.3mW. The coils for the power and data transfer, however, are expected to be 10mm x 10mm wide.

3.7 Results & Performance Evaluation

A system enabling data telemetry using very high data compression of neural recording is presented. All the necessary components of the system have been detailed including the analog frontend, the digital neural signal processor and the RF data and power transceiver. The analog frontend interfaces with the electrodes implanted in the cortex. The neural signal processor control the functionality of the NIN and employs the compression engine to compress the neural signals while maintaining high signal fidelity has been described. The processor also manages different modes of operation and various configurations built into the design. Finally, a communication protocol is presented which enables error free transmission of data to the MIM and reception of commands at the NIN.

A 0.5μm CMOS process was used to design all the blocks of the neural compression engine. The DWT block has been fabricated and the 3mm x 3mm chip is shown in Fig. 3.12. The controller was synthesized using OSU's Standard Cells Library [47], while all other blocks were custom designed for low power and low area. The active components of the prototype 32-channel, 4-level DWT implementation occupy roughly 3.84mm^2. The layout for the threshold and RLE blocks requires about 0.95mm^2 of area. Thus the combined compression system is expected to require approximately 5.75mm^2 including global routing for a 0.5μm process. The tested DWT module consumes only 3mW of power while processing 32 channels at 25Ksamples per second, or equivalently 95μW per channel. The power consumption per channel is directly proportional to the neural data sampling frequency.

To test our designs and algorithms for the compression engine, a stream of experimentally obtained 10-bit neural data was processed through the compression system and the resulting transform coefficients were used to reconstruct the neural signal. The results were compared to the original signal to measure the quality of spike reconstruction versus the compression obtained. This analysis was performed for several different zeroing threshold values. For a fair comparison, the same threshold value was used for all channels and levels. As discussed in previous chapter, Root Mean Squared (RMS) error and Entropy are used as the primary measures to evaluate the performance of our system. RMS error is a measure of the average difference between the original and the reconstructed signal. Shannon's entropy gives the theoretical limit to the achievable lossless data compression for a given data set.

For a given spike train, Fig. 3.12 plots the number of detected spikes, RMS error, the entropy of the transmitted sequence and the RLE compression achieved with respect to the threshold. The plots confirm the anticipated tradeoff between compression and RMS error. As expected, when thresholding is not employed, the RLE does not result in any compression, while entropy is at its maximum and RMS error at its minimum. The RMS error never goes to zero because of quantization noise. As the threshold value (and thus the number of zeros) increases, the RLE compression approaches the theoretical limit of entropy, which proves the effectiveness of our design. A threshold increase also results in an increase in the RMS error. This region of operation removes noise from the signal while preserving all the neural spikes and their shapes. At very high threshold values, the system starts distorting neural spikes, which results in a drop in the number of spikes detected at the output. The point of operation, thus, must be selected just before this region. The optimal point of operation may vary from one application to another depending upon the quality of reconstruction desired. Because of this direct tradeoff between RMS error and compression ratio, the zeroing threshold must be chosen to match application requirements; i.e. available bandwidth, quality of signal reconstruction required, and the power available for data transmission.

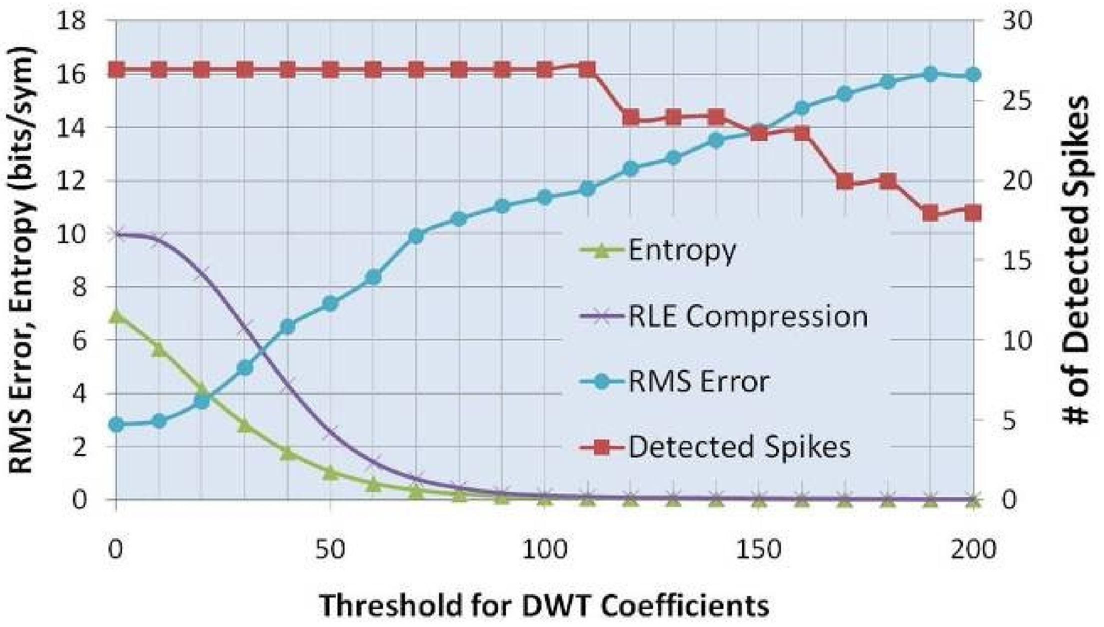


Fig. 3.11 System performance as a function of threshold value for the neural data set used in our experiments.

Fig. 3.11 shows the results for a sample spike train which contains 27 spikes. Almost perfect reconstruction is achieved for a zero threshold, but at the cost of high data rate. Beyond the threshold value of 110 the system starts losing spikes for this particular data set. Thus, the region of operation should be set between the values of 50 and 100 depending on the application and the desired spike reconstruction quality. The measure of quality of reconstruction depends highly on application specific spike detection and classification algorithms employed by the neuroscientist.

The DWT chip has been tested and works as designed with an excellent match between simulated and experimental results. Because the RLE block has not yet been fabricated, data presented in Fig. 3.11 is based on a combination of measured and simulated results. Fig. 3.12 shows the same spike at four different de-noising thresholds and compression ratios. For our prototype 32 channel design, a conservative threshold value of 80 resulted in an output data rate of less than 370Kbps, providing a compression of more than 20 times over 8Mbps for unprocessed data. The authors are not aware of any other publications where the spike shapes have been maintained, thus we are unable to compare our results against other methods of neural signal compression.

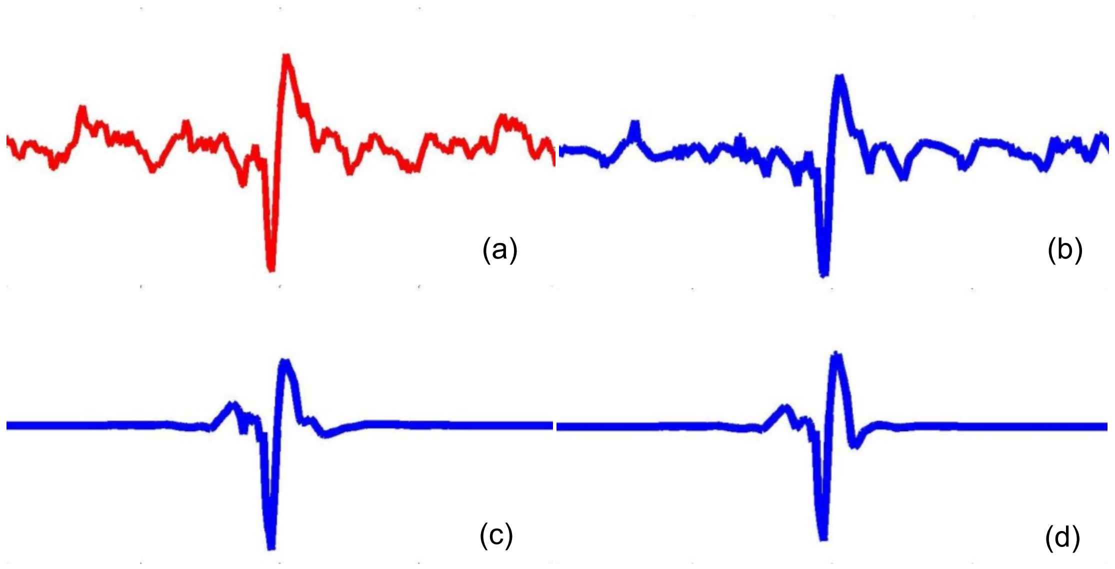


Fig. 3.12 (a) Original signal. (b) 2 times compression. (c) 16 times compression. (d) 62 times compression.

To test the functionality and integrity of the design, 32 channels of neural spike trains were simulated and sampled at 25Ksamples/sec per channel. Each channel contains spikes at a high firing rate of 90 spikes per second which are multiplexed into a single stream and provided to the compression engine. The system was tested in both the monitor and compression modes. The monitor mode resulted in an exact reconstruction of the original signal. For the compression mode, Fig. 3.13 shows a comparison of the original signal with a signal reconstructed by keeping only 2.5% of the coefficients, which means each bit of compressed data represents 40 bits of raw signal. No data loss due to overflows was recorded during the transmission. Same thresholds were used for consistency across channels, resulting in similar quality of reconstruction for all channels.

Fig. 3.14 shows a histogram of the time taken for incoming neural data to completely fill each frame during the experiment. It can be seen that, on average, each packet was filled in about 4ms. Corresponding to the highest level of activity in the neurons, the minimum time taken to fill a packet was recorded as 2.6ms which is within the allowed time limit of 1.7ms to avoid data overflow. However, in experiments using very low threshold values, packets have been observed to fill faster than they can be transmitted, which is expected and demonstrates the need for setting thresholds that result in the greater data compression. Our ongoing work aims at raising the thresholds to their highest possible value to enable spike sorting to simultaneously take place [52].

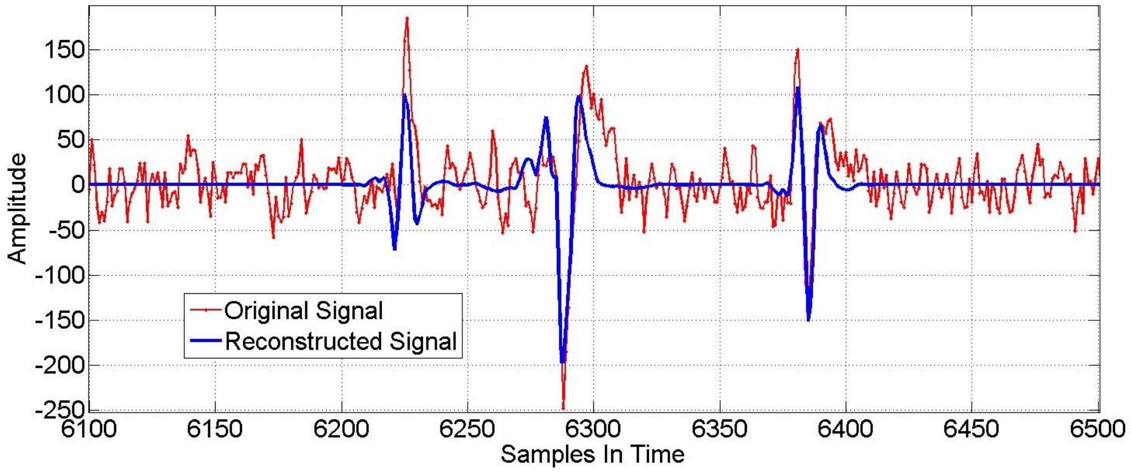


Fig. 3.13. Part of the 32 neural signals received and reconstructed by the MIM, at a compression ratio of 40, in comparison to the original signal acquired by the NIN. Despite some apparent signal loss, spikes are easily discriminated.

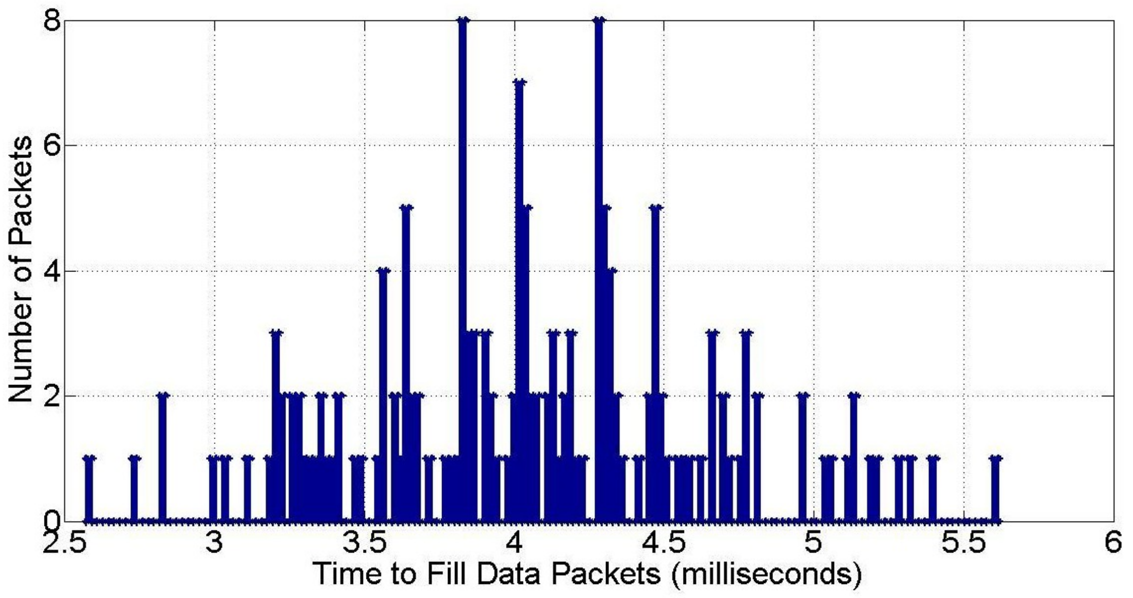


Fig. 3.14. Histogram of time taken in milliseconds by the compressed neural data to completely fill each Data Packet.

Fabricated in 0.5µm CMOS, the DWT block require only 3.84mm^2 of area to process 32 channels of data at 4 levels of in real time, while consuming only 3mW of power. The layout for neural signal processor covers an area of 13.8mm^2 with a power dissipation of 8.8mW. The analog frontend is expected to cover an area of 1.77mm^2 with a power dissipation of 1.1mW. The RF transceiver covers an area of 2.2mm^2 and consume power up to 7.3mW. Thus at peak performance, the NIN module is expected to consume 11.4mW of power covering an area of 17.77mm^2 in 0.5µm technology. The coils for the power and data transfer, however, are expected to be 10mm x 10mm wide. Table 3.1 summarizes the estimated area and power requirements for all the modules in the NIN. Empirical measurements show that this area would be reduced by roughly a factor of 10 if implemented in a 0.18μm process reducing the chip area to 1.8mm^2 with power consumption in the proximity of 5mW. However, this system is sufficiently small for implantation even in 0.5μm CMOS.

Table 3.1. Estimated Area and Power Requirement for Implantable Hardware Modules in 0.5 micron CMOS technology

Module	Area (mm^2)	Power(mW)
Analog Frontend Module	1.77	1.1
Digital Neural Signal Processor	13.8	8.8
*RF Data and Power Transceiver	2.2	7.3
*Complete NIN block for 32 Channels	17.77	11.4

*Area numbers do not include coil area for RF transceiver

This highly configurable area-power efficient hardware results in large compression rates while extending flexibility to the neuroscientists to choose the 'perfectness' of reconstruction of neural data from a multitude of neural signals. The small size and low power consumption of the system makes it highly suitable for implantable high-density microelectrode array devices.

3.8 Hardware Implementation and FPGA Test

Once the algorithms were designed and tested in Matlab; the hardware for the complete NIN digital system including the Multi-Level Threshold block, the Run Length Encoder the Communication Controller, the Analog Frontend Controller and the Global Controller was designed in HDL using Verilog. In addition, the design of DWT was improved to include channel selectivity and dynamic frequency scaling. To enable high degree of testability, the analog frontend controller was modified to receive input from two different sources, one serial and one parallel. Test points were inserted in the design to enable testing of data outputs from each block.

To test the design of digital system in absence of the prototype analog frontend and RF transceivers, the system was mapped on an Altera Cyclone III FPGA and interfaced with a custom designed printed circuit board containing the analog frontend components. The amplified signals were sent to the PCB designed with discrete components containing an analog multiplexer and an analog to digital converter. The control signals for the A/D were generated by the analog frontend controller present in the FPGA. The time-multiplexed outputs from the board are received by the FPGA which processes the signals as per the settings. The input data was received at the analog frontend, passed through various blocks and observed at the output of the communication controller. The RF interface was replaced with a wired connection between the NIN-FPGA and a host computer acting as the MIM. The control signals for the transceiver were generated by the communication controller in the FPGA. The signals were then received by the MIM computer through a data acquisitioning system. The MIM completes the necessary processing and generates the command packets which are finally received by the NIN in the FPGA. The basic functionality of the NIN was tested and verified, however, since the MIM emulation was very rudimentary, the elaborate testing of NIN functionality, its various configurations and modes of operations was not possible. These tests allowed us to test the system using real input signals. In the process the system allowed testing for situations that were ignored or not represented accurately during simulations.

Although the basic functionality was tested, the power consumption and area requirements of the system implemented on FPGA could not be used to estimate the requirements for the system implemented in the ASIC because of the difference in fabrication technology and various architectural overheads associated with the FPGA. As a result, the prototype was not be able to give an accurate estimate of the power consumption of the system.

3.9 VLSI Layout of Digital System on Chip in 0.13 Micron CMOS

Once the design was functionally verified on the FPGA, the layout was prepared for fabrication in IBM 8RF 0.13 micron CMOS process. This involved preparation of various components of the ASIC for automatic place and route. This includes synthesis and routing for all the different digital blocks and generation of different memory blocks. Third party Low-Vt Standard Cell Libraries and Low-Vt SRAM generators provided by ARM were used for low power operation. Synopsys Design Vision was used for logic synthesis. Once all the components are available, floorplanning was completed, clock trees were created and the final place and route is completed with the inclusion of pads with electrostatic discharge protection. The layout was passed through the design rule checks (DRC) for the rules outlined from the fabrication foundry. Fig. 3.15 shows the final layout of the system.

The chip layout was generated to enable measurement of chip area and to allow simulations for power measurements. The chip may be fabricated at a future date. Note that for area optimization all the blocks have been merged together into a single unit. The core of the design is 1.1mm x 1.1mm whereas including the IO and power pads the area of the NIN chip becomes 1.9mm x 1.9mm = 3.69mm^2. Simulation results on the core show 800µW of power consumption with a power supply of 1.2V.

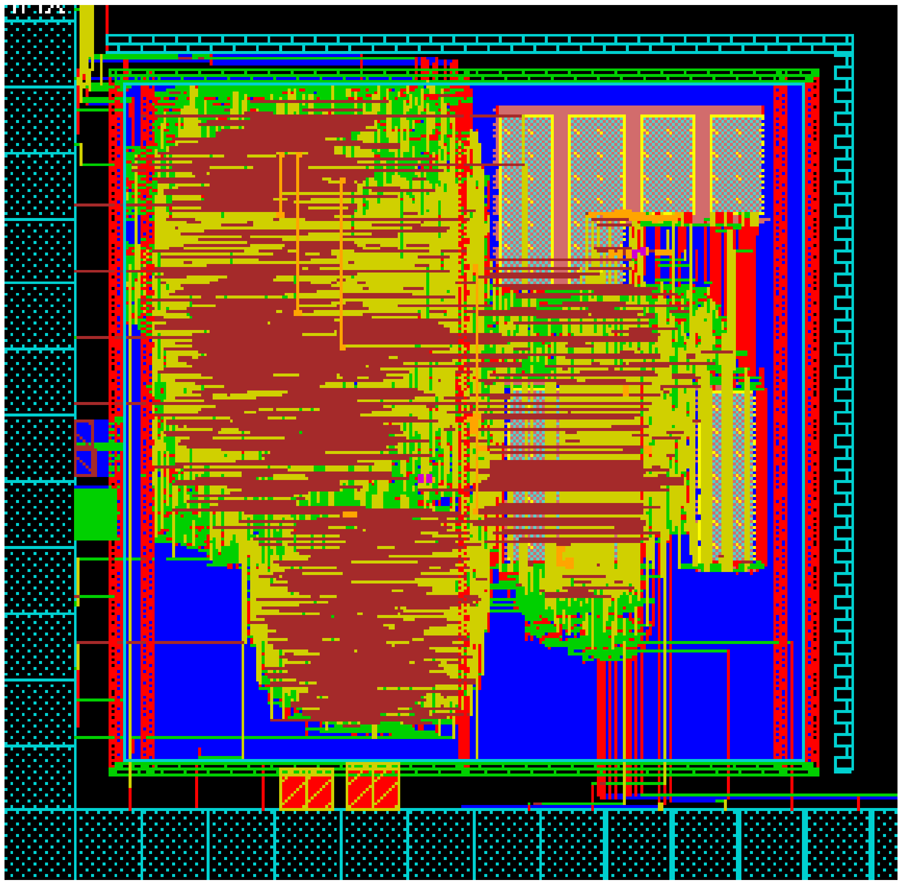


Fig. 3.15: Layout of the complete NIN digital system in 0.13 micron CMOS technology.

3.10 Analysis

The digital neural signal processor described above has exceeded the original design goals for area and power. However, while developing this chip, some limitations and unexpected challenges were observed and are described below. The research described in the next two chapters was undertaken in order to address these challenges and demonstrate how the performance of the neural recording systems can be improved for various applications.

Optimal design of the communication controller requires information about the error characteristics of the channel within its intended environment, including across skin. Although inductively coupled transmitters and receivers have been used in earlier studies, there has been no effort on quantifying the error characteristics of the channel. For example, design of the communication protocol and the packet format rests on the distribution of bit errors in a packet. This affects the decision on the use of error correction vs. error detection and thus affects the total number of retransmissions necessary for error free transmission. The number of parity bits needed is dependent on the maximum possible errors in a single frame. Similarly the number of consecutive packets in error defines the transmission buffer size at the receiver. Because this error profile has never been reported in literature, extensive trace collection and analysis of this error process was undertaken, as described in the next chapter.

The DWT based compression engine design has its own merits and demerits. The DWT based neural recording system developed in this research permits a large amount of processing to be completed on-chip to retain much of the information in the spike train and needing only a fraction of coefficients to be transmitted off-chip. The result is the ability to reconstruct the signal as faithfully as needed for a particular application. This however is achieved at a tremendous cost of processing every sample of the neural channel, which includes 8 additions and 8 multiplications, and then deciding if the resulting coefficients need to be kept or discarded. This is an overkill for applications that do not require signal reconstruction (an issue that is still highly debated in the neural engineering community) or applications that can afford some degree of inaccuracy in spike sorting capability (again, this issue is debatable as no clear physiological need has been established). The research in this chapter has shown that it is possible to process many channels within an implanted chip; however, the presented solution does not scale well to neural systems demanding simultaneous processing of a higher number of channels, say one the order of a few hundred or more. With the DWT-based design, area, power and fabrication costs make it impossible to complete such level of processing on an implantable system. It thus becomes imperative to search for an ultra hardware resource constrained and computationally efficient set of features that can provide high spike sorting performance for a fraction of the area-power cost. A set of features that meet this goal was explored as presented in Chapter 5.

4 Channel Characterization for Implant to Body Surface Communication

4.1 Introduction

To achieve reliable communication between nodes on an implant-to-body-surface biotelemetry link, knowledge of the power and data transfer characteristics of the transceivers and the channel is required. There has been recent interest in the field of Body Area Networks; however, it is still a new field and no standards for the design of power and data transceivers or communication protocols have been adopted. The search for the most suitable transceiver is a topic of continuing research and hence there are no limitations, requirements or capabilities that the transceiver hardware should conform to. Recent research has focused on employing inductive links for data communication [5, 9, 48, 53]. The transceiver design reported in [53] is distinctive as it is capable to transmitting power and data over the same inductive channel using a technique called ‘back telemetry’. For the rest of the chapter we call such inductive power / data transceiver ‘Inductively Coupled Transceivers’ (ICT). Implantable ICTs are of particular interest to the biotelemetry research community because they are based on a wireless communication techniques that allows low power communication at reasonable data rates. It is expected that ICTs will be used extensively in biotelemetry applications. Although various ICT designs have been presented, a concerted effort and detailed experimental analysis is necessary to quantify the performance and limitations of the transceiver before they can be used in implantable applications.

Communication of digital data sees errors at two levels, the bit-level and the packet level. These random errors observed over time can be modeled as a random process. A deeper understanding of this error process is necessary for the optimal design of various components of the communication system including the use of error detection and correction codes, the protocol design, the packet size and the hardware resources required to allow retransmission of erroneous packets.

The purpose of this study is twofold. Firstly, it is to estimate the performance limitations of the ICT presented in [53] under bandwidth and power constraints by studying the effects of various orientations of transmitter and receiver. Secondly, the error patterns for the channel across the skin for implant to body surface power/data communication need to be characterized to enable design of protocols for reliable data communication.

Trace collection has been used to model error processes of various wireless communication standards including 802.11x WLANs [54-55], and 802.15.4 LR-WPANs [56-57]. The same approach will be followed to analyze the ICT link.

4.2 Wireless Power / Data Transceiver

The neural recoding system consists of two modules, the Implanted Module (IM) and the Surface Module (SM). The IM is implanted inside the body, preferably within a couple of centimeters from the body surface. It needs to be powered wirelessly from an external source. The SM is placed on the surface of the body and is supplied with continuous power through a battery. The data transmitter is part of the IM, while the data receiver is part of the SM. On the other hand, the power transmitter is part of SM while the power receiver is part of the IM. Each module contains a flat inductive coil etched on a printed circuit board to allow inductive coupling. The SM generates an AC signal of 13.56MHz which is coupled to the IM coil. This frequency belongs to the FCC mandated ISM band which is allocated for industrial, scientific and medical research. The IM houses a power rectifier that converts the signal received at the coil into a DC voltage. The detailed design is presented in [53]. The rectifier has been designed to provide 20mW of power to the IM, which includes the frontend analog and digital processing parts in addition to the transceiver. The system uses 'passive back telemetry' to transmit data over the same power channel. This is achieved by the use of Load Shift Keying (LSK). Under normal operation, when there is no data to be transmitted, the IM is constantly receiving power from the SM. However, when the IM wants to transmit a 'one' bit, it creates an open circuit at the power receiver coil for 0.2μsec. As a result, the IM stops receiving power from the SM for this duration of time. This change in 'load' is sensed at the SM by a change in current and a 'one' is detected by the data receiver. Thus at the IM the power receiver coil also works as the data transmitter coil, whereas on the SM the power transmitter coils also works as a data receiver coil. The current design is capable of data transfer at a maximum rate of 500kbps. The transmitter power is fixed at 1mW which corresponds to 2nJ energy per bit transmission.

Both the power and data are dependent on a single coil based inductive link. Inductive links are highly directional and the quality of induction depends heavily on the orientation and alignment of the coil. The distance between the two coils also affects the amount of induction achieved. Since in practical situations it is not possible to guarantee a perfect alignment between an implanted coil and a surface coil, it is important to study the effect of relative coil orientation on the quality of data transfer achieved to enable design of suitable communication protocols to maximize data throughput.

4.3 Experimental Setup

4.3.1 Trace Collection

Trace collection refers to transmission of a large number of packets from the transmitter and reception of those packets at the receiver. Since the contents of the transmitted and received packets are known, any errors in transmission can be identified at the bit and packet levels. A bit-wise XOR operation on the transmitted and received packets results in a ‘zero’ if the bits match, and results in a ‘one’ if the bit has been flipped during communication. Thousands of packets need to be transmitted and received in order to formulate an unbiased model. The statistics that can be extracted from these traces are the Packet Error Rates (PER) and the Bit Error Rates (BER). PER is the fraction of transmitted packets that are received with at least one bit in error.

$$PER = \frac{Packet\_Errors}{Total\_Packets} \tag{4.1}$$

BER is the fraction of transmitted bits that are received in error.

$$BER = \frac{Bit\_Errors}{Total\_Bits} \tag{4.2}$$

Some papers report Packet Reception Rate (PRR) instead of PER, which is related to PRR by

$$PRR = 1 - PER \tag{4.3}$$

To our knowledge, this is the first thorough trace collection effort for inductively coupled transceivers for biotelemetry links. For this study, error traces of approximately 3 million packets were collected, where each transmitted and received bit was logged for detailed analysis.

4.3.2 Physical Setup

The trace-collection setup is depicted in Fig. 4.1. It consists of a surface module (SM) and an implantable module (IM). The SM houses a power transmitter in addition to a data receiver, whereas the receiver houses a power receiver in addition to a data transmitter. The power source is only connected to the SM. A data acquisition card (DAQ) card is connected to both the SM and IM to provide the data to be transmitted to the IM and to record the data that is received at the SM. The DAQ card is connected to a host PC running a LabView application that continuously retrieves data from the receiver and logs it. The SM and IM are placed in a Faraday cage to minimize the effect of external electromagnetic noise on the data communication.

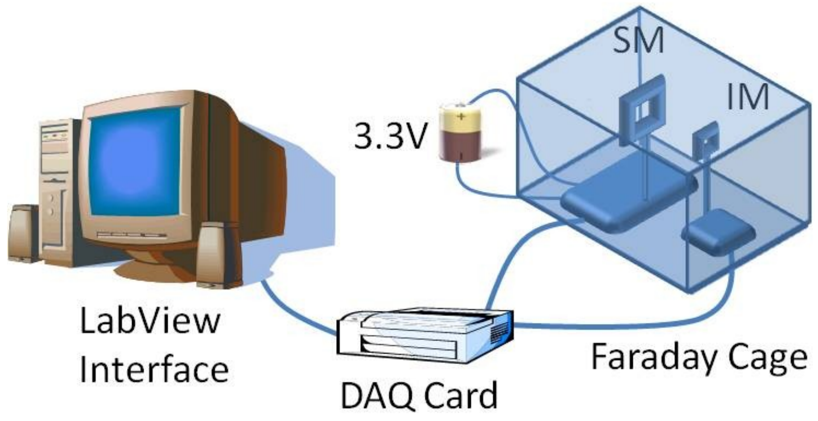


Fig. 4.1. Experimental setup used for transceiver characterization.

The size of the packet is 820 bits, and the packet format is detailed in previous chapters as well as in [58]. The packet is composed of 800 bits of data, 10bits of addressing information and 8 bits of CRC.

4.4 Transceiver Performance Analysis

The performance of the power and data transceiver was examined to determine the operating conditions for which the transceiver provides acceptable performance.

4.4.1 Transceiver Orientation Diversity

Packets were collected over a period of weeks, at different times of the day, in a typical lab environment. Each trace collection was characterized by the location of the transmitter and receiver, separation between them, the tilt of coils, and the misalignment from the center. Fig. 4.2 shows the scaled diagram for different settings used to measure the performance of the transceiver. The amount mis-alignment is represented with ‘z’, the distance is represented with an ‘x’ and the tilt is represented with a θ. For the scaled Fig. 4.2, x=10mm, z=10mm and θ=30^{o}. The transmitter has a fixed transmission power of 1 mW.

For each setting, a total of 100,000 packets were transmitted in segments of 5000 each with two minutes of inactivity between each segment. This corresponds to 200 seconds of packet transmission for each segment. As a result a total of 82 million bits were transmitted for each setting. The data was transmitted at a rate of 500kbps, which corresponds to packets being transmitted at a rate of 25 packets per second.

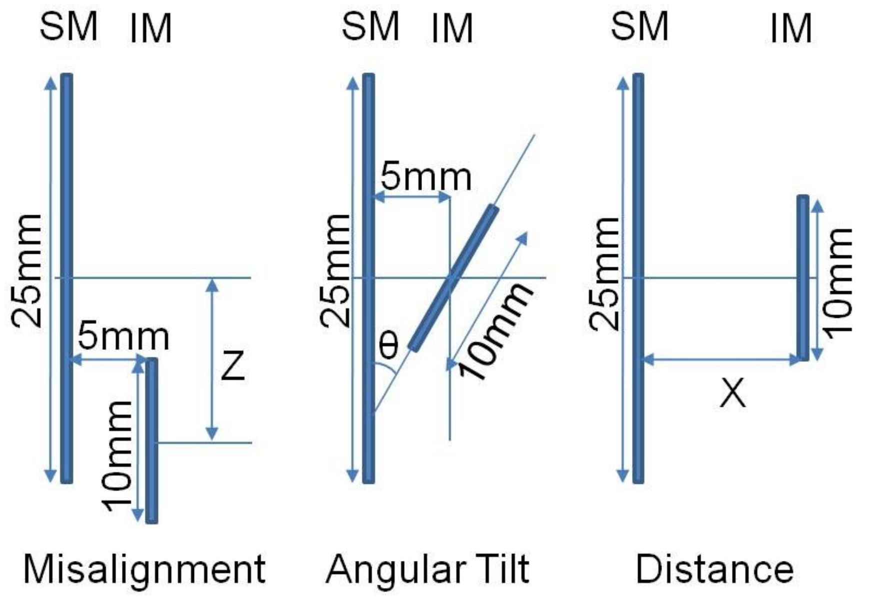


Fig. 4.2. Scaled diagram of different settings used to measure the performance of the transceiver, including the distance, the tilt and the misalignment.

4.4.2 Transceiver Characterization Results

Figures 4.3-4.5 plot the BER and PER vs. the three parameters of distance, mis-alignment and angular tilt. The BER is plotted against a logarithmic scale. As expected, both the BER and PER deteriorate by increasing the values of these parameters. It must be noted that the minimum value plotted in each plot corresponds to zero errors, i.e. no errors were detected for those settings. However, becasue BER can only be detected up to 1E-8 (because of the number of transmitted bits), the value of 1E-8 is plotted for the least value. Similarly, the minimum plotted value for PER is 1E-5.

It must be noted from Fig. 4.3-4.5 that the initial settings of 5mm distance, 11 degrees tilt and 5mm mis-alignment do not result in any transmission errors. These are the typical settings expected in practical situations when the transmitter is implanted into the body and the receiver is placed on the body surface. Fig. 4.3 shows that a nominal acceptable BER value of 1E-4 is achieved at 14.5mm of distance with a PER of about 10%.

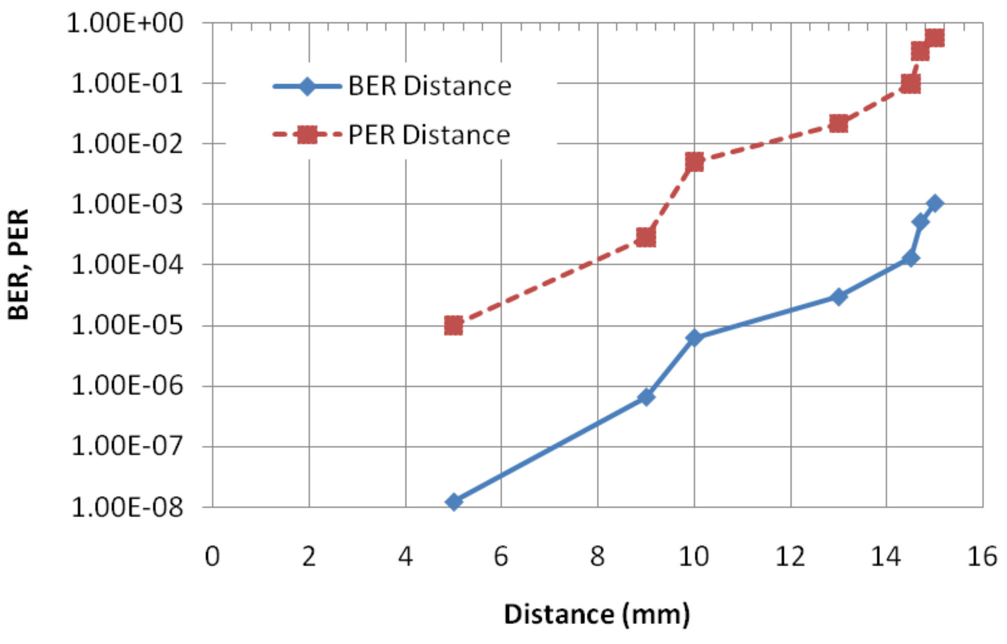


Fig. 4.3. Bit Error Rate and Packet Error Rate versus the Distance in mm.

Fig. 4.4 shows that 11mm of mis-alignment provides a BER of 3E-5, beyond which the BER increases sharply. It is interesting to note that from 9mm to 11mm the BER remains almost constant (from 2.6E-5 to 3.1E-5); however PER increases rapidly (0.4% to 2.5%). This suggests that, although the number of bits in error remain approximately same, the errors are spread over a larger number of packets. Conversely, it can be seen that at 12mm and 13mm misalignment the BER increases sharply (0.25% to 1.4%) while the PER barely increases (84% to 100%), suggesting an increase in number of bit errors per packet.

Fig. 4.5 shows that at 34 degrees of tilt a BER of 2E-4 is achieved with a corresponding PER of 2.3%. Both the BER and the PER show similar amount change with increasing tilt.

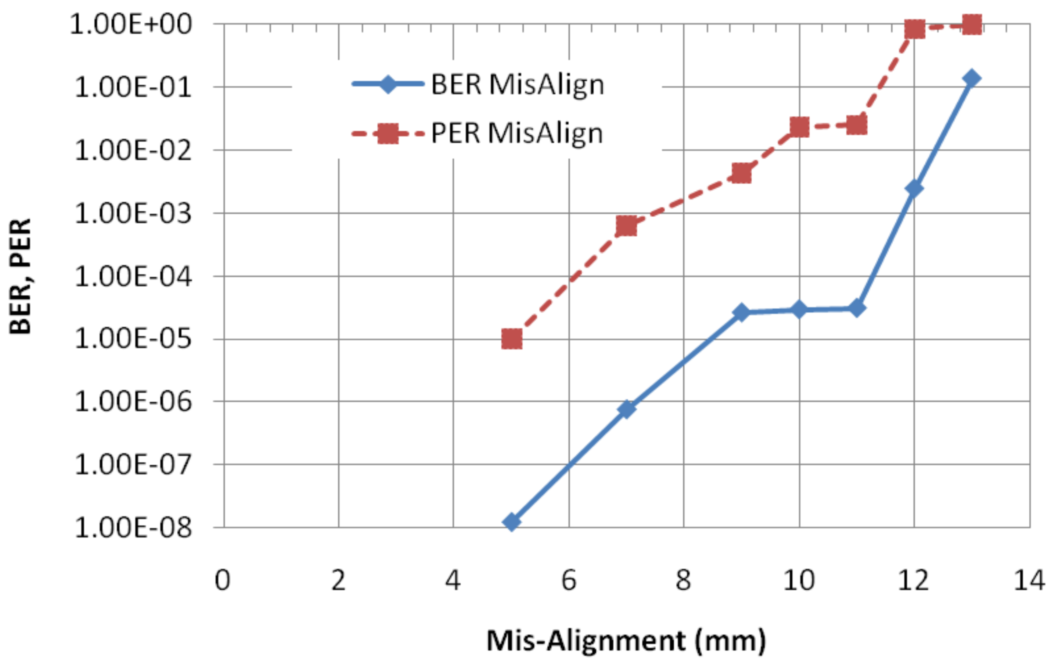


Fig. 4.4. Bit Error Rate and Packet Error Rate versus the Mis-Alignment in mm.

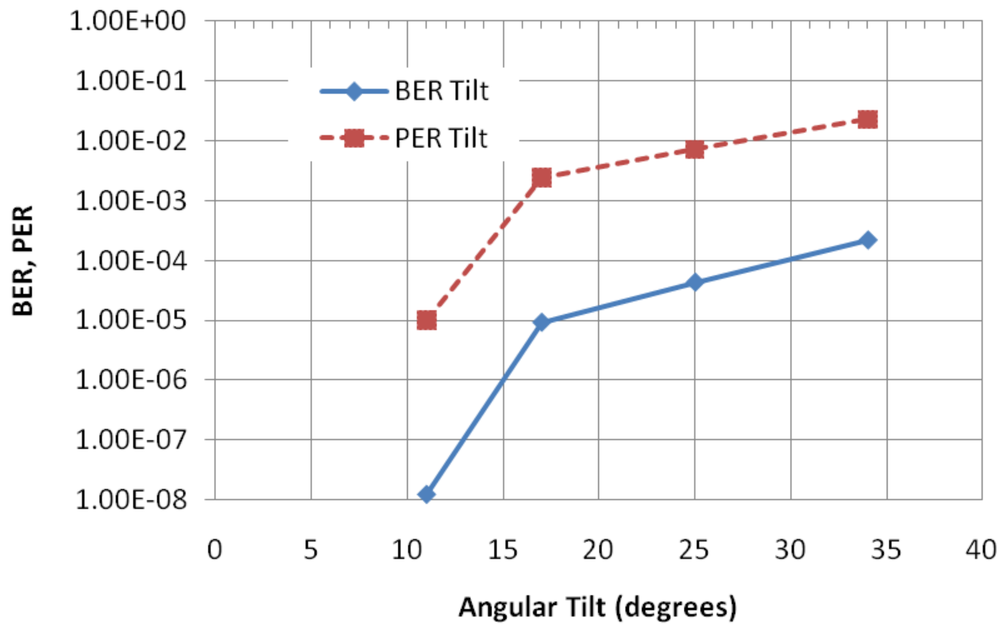


Fig. 4.5. Bit Error Rate and Packet Error Rate versus the Angular Tilt in degrees.

Disturbance Effect: The transmitter and receiver were physically disturbed to simulate the effect of body movement, which is expected to affect all three parameters of distance, tilt and mis-alignment. Disturbance was created as a combination of all six degrees of freedom, include the three translational and three rotational axis. Although the disturbance magnitude could not be measured scientifically, the relative initial displacement of the transceivers was recorded to be ±1.5mm from the center points. The disturbances were created at a frequency of about 2Hz, but each disturbance resulted in higher frequency damped vibrations because the Tx and Rx were suspended in air using copper wires.

Fig. 4.6 shows the number of packets in errors over time for the case when disturbance was added to the initial setting of 5mm distance between transceivers. To plot the results, the 100,000 packets have been divided into bins of 100 packets each. As a result, the 1000 bins show the number of erroneous packets in each bin. Note the ‘burst’ nature of errors, i.e. whenever there is an error it is very likely that there will be more packets with errors in the same vicinity.

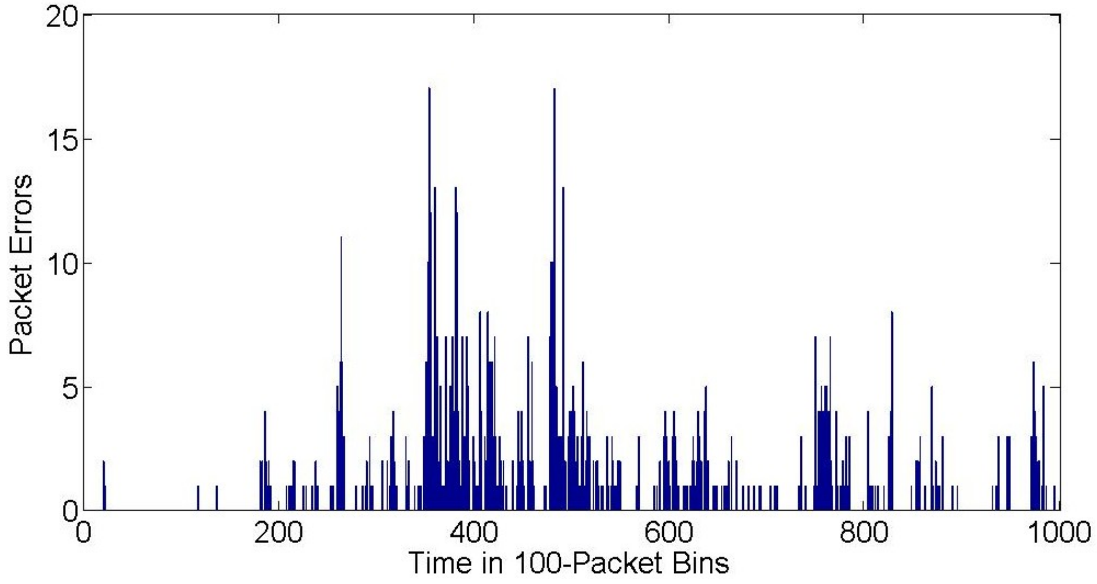


Fig. 4.6. Packet errors versus time for disturbance effect test. The 100,000 packets are divided into bins of 100 packets and show the number of packet errors in each bin

Fig. 4.7 compares the BER and PER at distances of 5mm and 10mm distance with and without disturbance. It can be seen that disturbance has a significant effect on the quality of channel and the total number of errors.

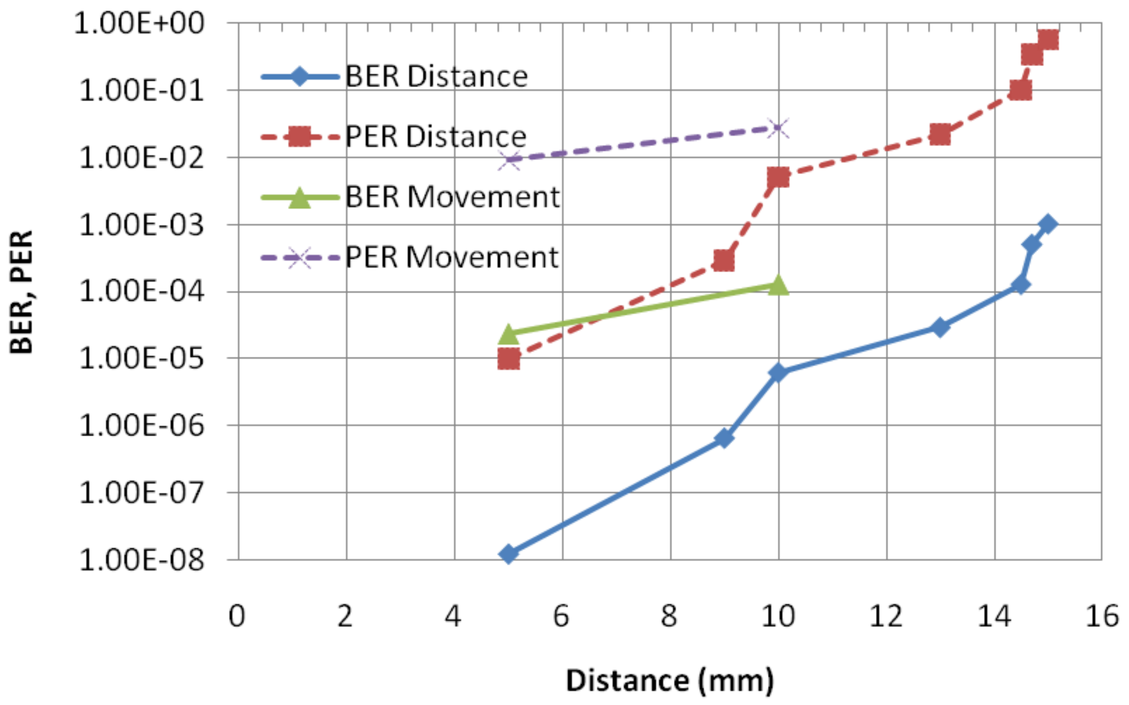


Fig. 4.7. Bit Error Rate and Packet Error Rate versus the Distance in mm showing the effect of physical disturbance on BER and PER.

4.5 Channel Memory

Most wireless channels show ‘burstiness’ in the packet-level and bit-level errors. Thus, if a packet (or bit) error has been seen in transmission, the probability that the subsequent one or more packets (or bits) will be in error increases. This increase in probability is a characteristic of the channel and varies from one channel to the other. This phenomenon is called Channel Memory. The degree to which effects of a particular bit error are carried over time can be quantified by a measurement of the packet level channel memory, P_M, and the bit-level channel memory, B_M, respectively [57]. In essence, channel memory provides an estimate of the average number of consecutive errors expected whenever an error is observed.

4.5.1 Bit Level Memory

Knowledge of the amount of bit-level memory of a channel helps in the design of error detection and correction schemes. Correlograms are commonly used to measure the memory of a process [54-56]. A correlogram is a plot of the auto-correlation function, $R_{XX}(t)$, of the process. In this case, it is a bit level trace consisting of zeros (when the bit is correct) and ones (when the bit is inverted) as a function of time, t.

$$R_{XX}(t) = \frac{E\{(X-\mu)(X_t-\mu)\}}{\sigma^2} \tag{4.4}$$

The correlogram was computed for each erroneous packet over the length of the packet. The lag ‘t’ for which $R_{XX}(t)$ becomes insignificantly small, less than 0.15, was taken as a measure of bit level memory [54]. A histogram of the memories was computed and a cumulative distribution function is presented.

4.5.2 Packet Level Memory

A similar correlogram-based analysis was applied to the packet level, in which each erroneous packet is represented with a one and each correct packet is represented with a zero. In order to observe time-dependent effects on P_M, this measurement was performed separately for all 20 segments for each setting. A histogram of memories of these individual traces was computed and a cumulative distributed is provided to show the trend.

For the test case of 5mm distance with disturbance, the histogram and cumulative distribution of packet level memory is shown in Fig. 4.8, where the red (line) plot shows an 85% chance that the errors will be solitary and a 95% chance that there will be four or fewer consecutive errors. No error correction capability is assumed for memory calculations; however, if error correction were applied then the packet memory is expected to reduce even further. Note that a memory of zero means that either zero packets are in error or solitary packets are in error. Memory of one means that two consecutive packets are in error, a memory of two means that three consecutive packets are in error, and so no.

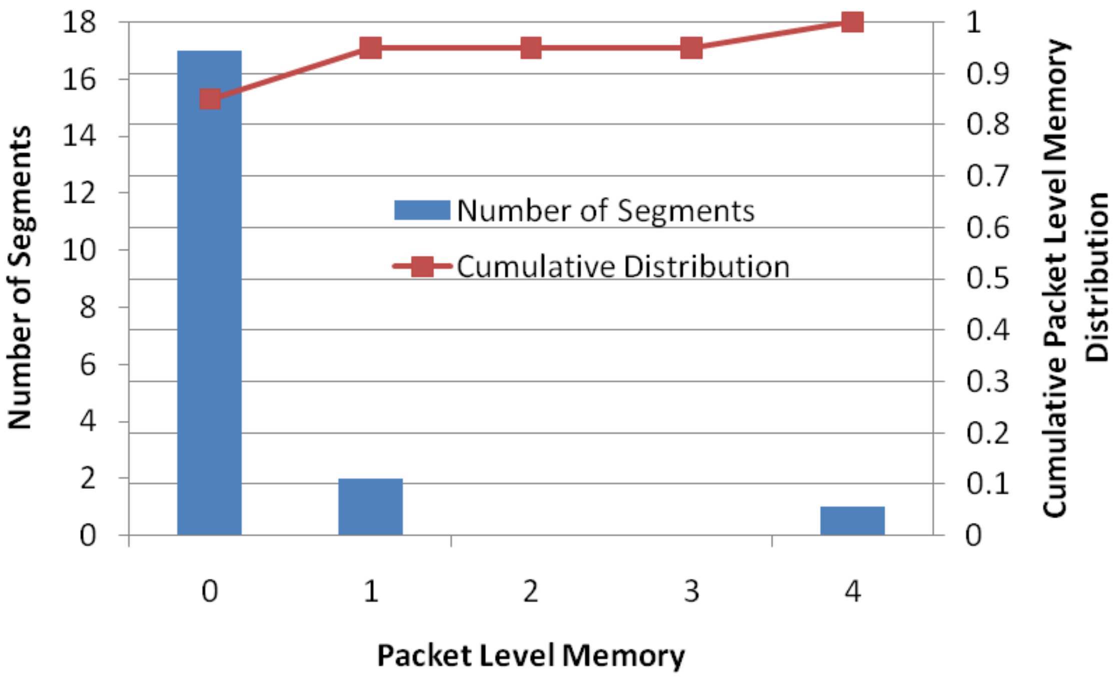


Fig. 4.8. Histogram of packet level memory for 5mm distance with disturbance.

Fig. 4.9 gives the bit-level memory of the channel, showing the consecutive number of bits in error. It can be seen that 79% of the time the bit-errors are solitary, and 98% of times there are four or fewer consecutive bits in error.

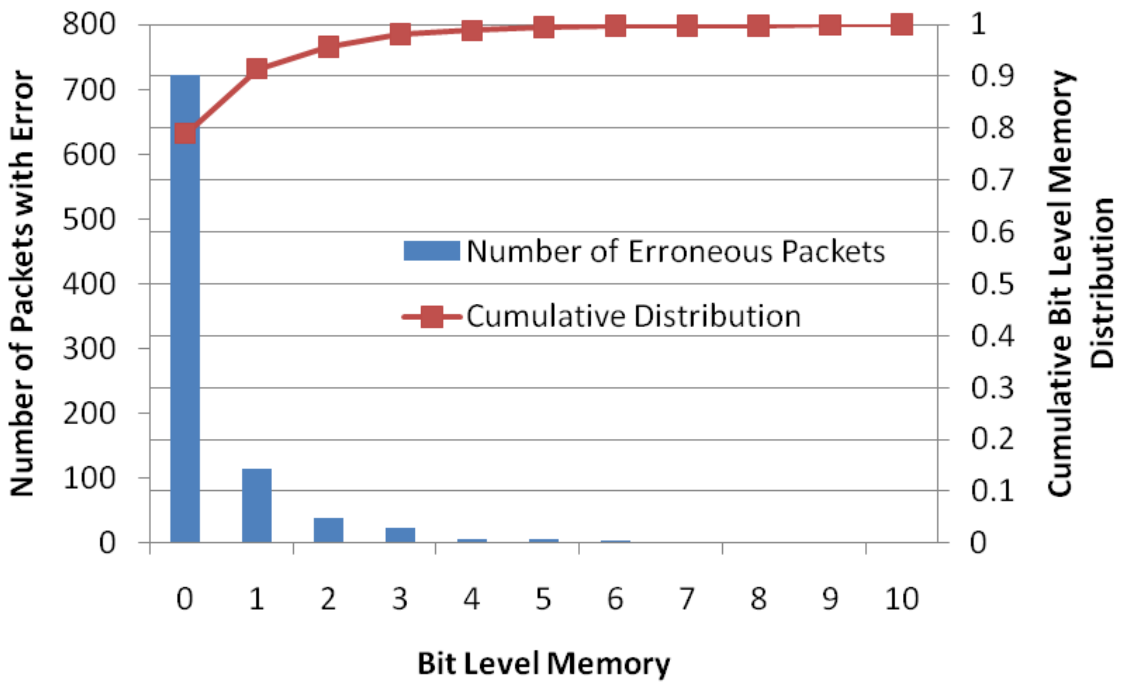


Fig. 4.9. Histogram of bit level memory for 5mm distance with disturbance.

For the movement artifacts with 5mm distance there were 913 erroneous packets out of 100,000 and 1909 bits in error out of 82 million, given the PER = 9.13E-3 and BER = 2.33E-5. Fig. 4.10 shows the histogram of number of bit errors in each erroneous packet. Only the packets containing errors are considered for this plot. It can be seen that most packets have only one bit in error, while 89.4% packets have only three or fewer bits in error. If error correction techniques are applied that can correct up to 3 errors, then retransmissions could be reduced by 90%.

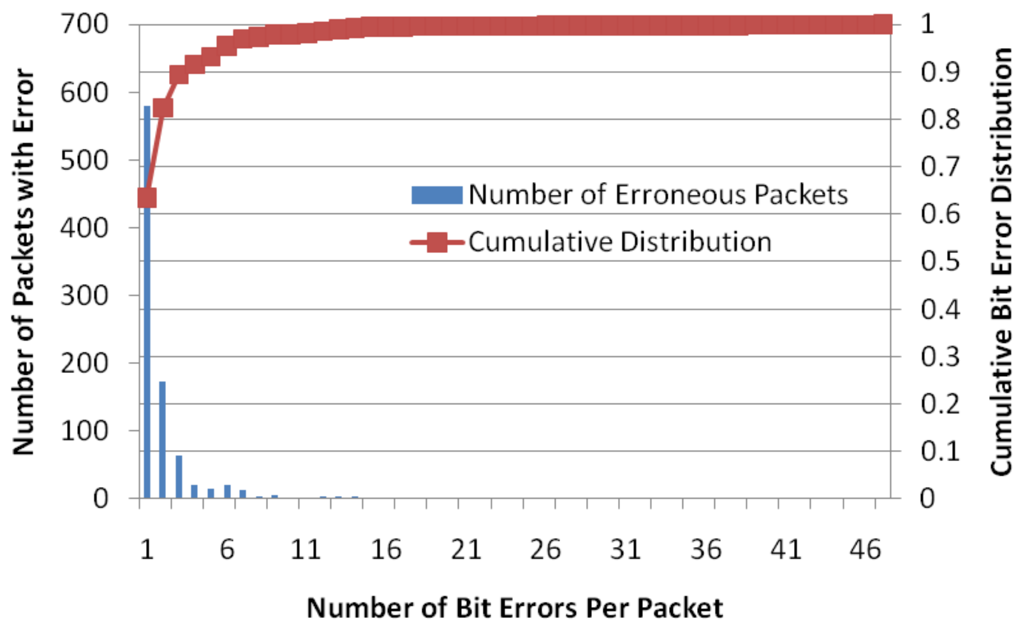


Fig. 4.10. Histogram of number of bit errors per packet form 5mm distance with disturbance.

4.6 **Channel Characterization Across Animal Skin**

4.6.1 Physical Setup

Several experiments were conducted using chicken skin to model the general effects of skin tissue on inductive power transfer and data communication. Experiments were conducted without a Faraday cage in the open lab environment containing many sources of electromagnetic noise. For 5mm distance measurements, a 4mm thick layer of skin was sandwiched between the transmitter and receiver coils. For 10mm distance measurements, layers of skin adding up to 8mm thickness were used. However, tilt and movement effects could not be measured with the skin between the transmitter and receiver. Fig. 4.11 shows the setup used for these measurements.

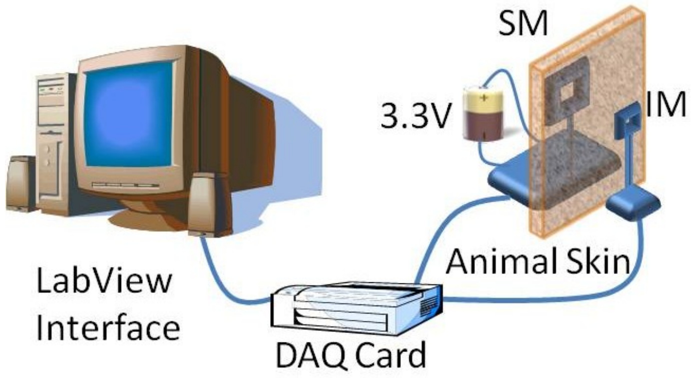


Fig. 4.11. Experimental setup used for channel characterization including animal skin.

4.6.2 Channel Characterization Results

The results for the case of 10mm distance with 8mm thick layer of skin between the transmitter and receiver are discussed in this section. A total of 0.5 million packets were transmitted and a PER = 7.43E-3 was recorded. The 5000 bins of 100 packets each are shown in Fig. 4.12.

The cumulative distribution of packet memory is shown in Fig. 4.13, where the red plot shows 84% chance that the errors will be four or fewer errors and a 89% chance that there will be 7 or fewer consecutive errors.

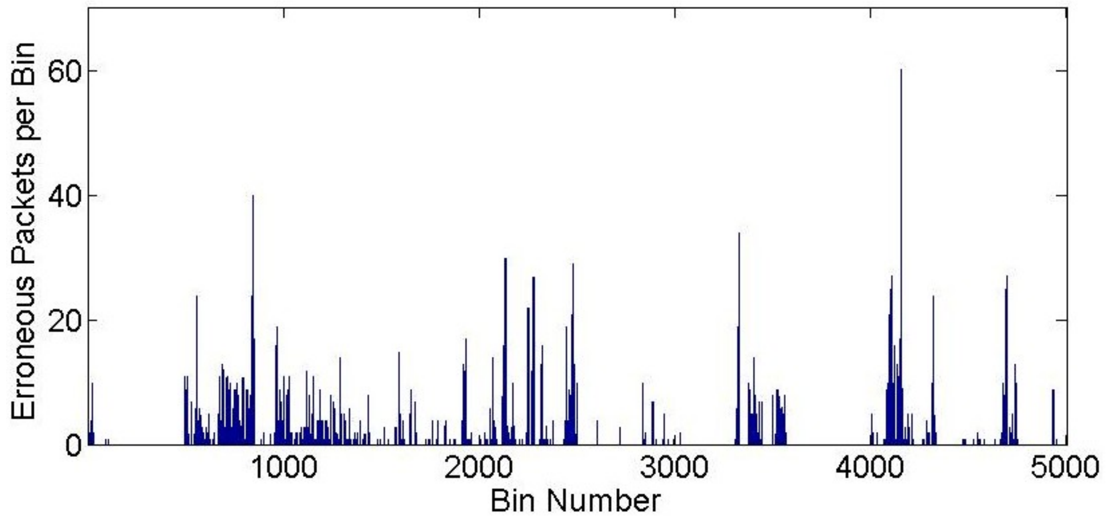


Fig. 4.12. Packet errors versus time for the case of 10mm distance with animal skin. The 500,000 packets are divided into bins of 100 packets and show the number of packet errors in each bin.

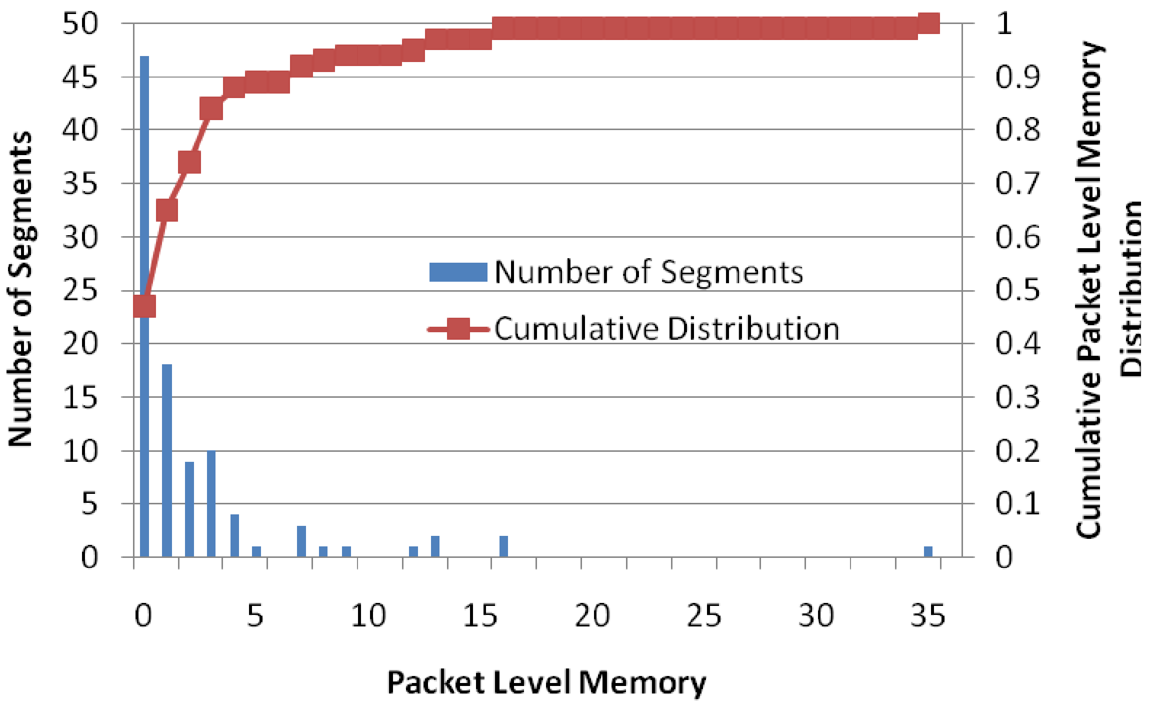


Fig. 4.13. Histogram of packet level memory for 10mm distance through animal skin.

Fig. 4.14 gives the bit-level memory of the channel, showing the consecutive number of bits in error. It can be seen that 68% of the time the bit-errors are solitary, and 97% of times there are 3 or fewer consecutive bits in error.

For the skin effects at 10mm distance, there are 3716 erroneous packets out of 500,000 and 15144 bits in error out of 410 million, giving the PER = 7.43E-3 and BER = 3.69E-5. Fig. 4.15 shows that most packets have only one bit in error, 66% packets have only three or fewer bits in error and 89.8% have 9 or fewer bits in error. If error correction techniques were applied that can correct up to 3 errors, then retransmissions could be reduced by 66%. It must be noted that a packet was found with 69 bits in error, which means the error detection capability should be able to handle such cases of large errors in a packet, even though most of the packet have only a few bits in error.

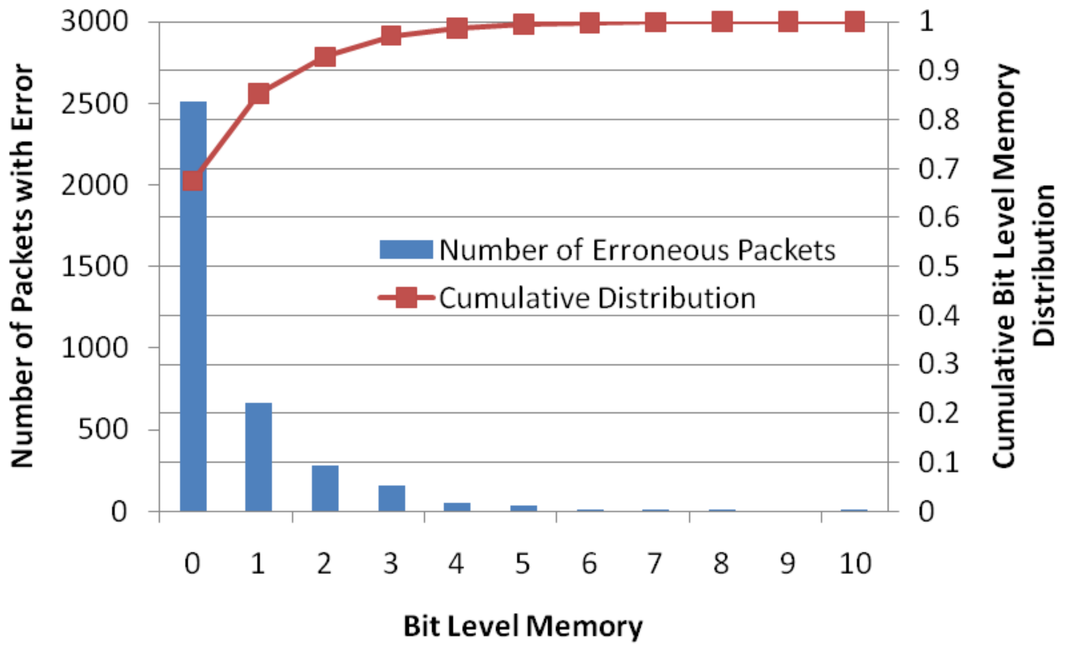


Fig. 4.14. Histogram of bit level memory for 10mm distance through animal skin.

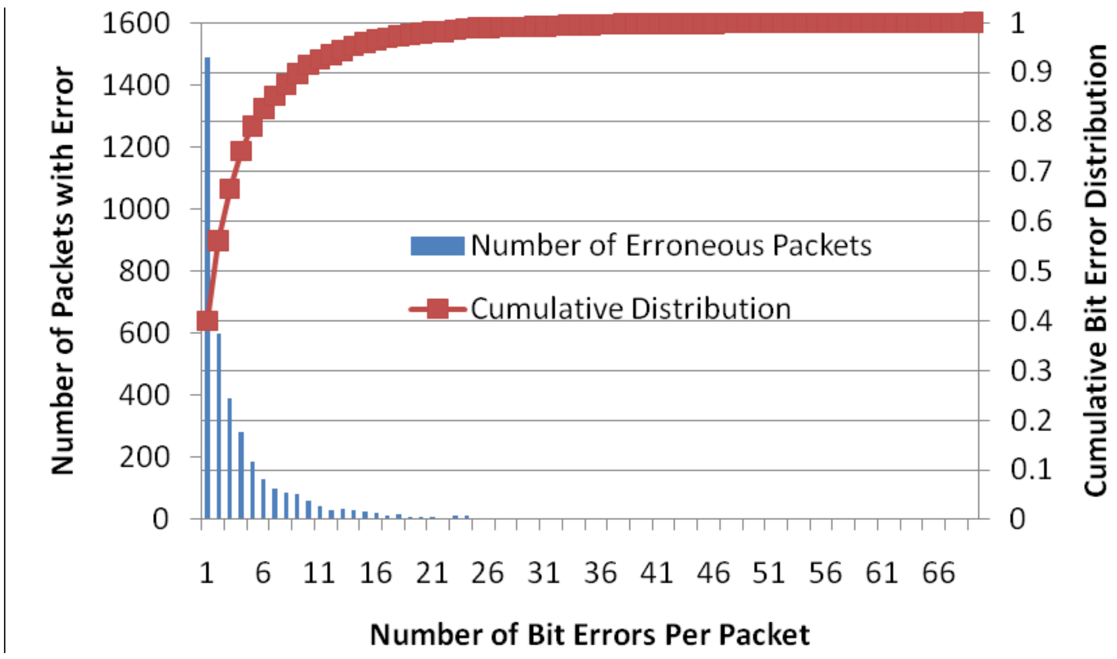


Fig. 4.15. Histogram of number of bit errors per erroneous packet for 10mm distance through animal skin.

Fig. 4.16 shows the comparison of disturbance artifacts for the skin-based channel and the air-based channel. It can be seen that disturbance has the most effect on the performance of the transceiver while the introduction of skin also degrades the error characteristics of the channel in comparison to the air based channel.

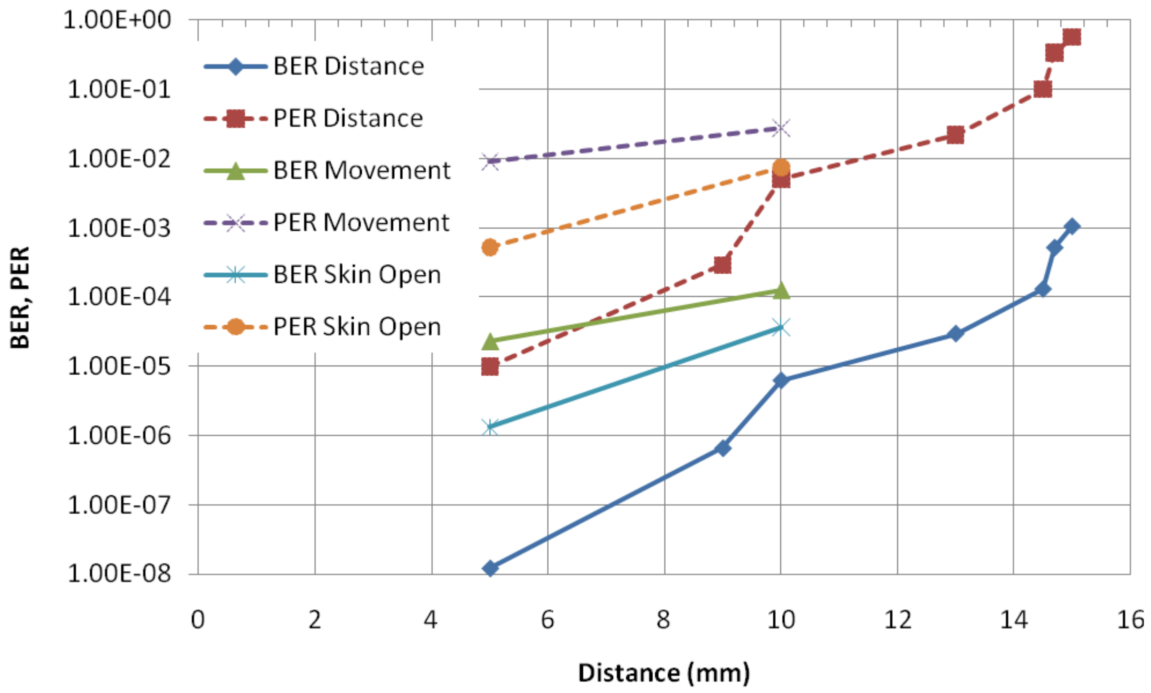


Fig. 4.16. Comparison of BER and PER versus distance for various experimental settings.

4.7 Forward Error Correction

There are several forward error correction (FEC) coding techniques and algorithms which provide various degrees of error correction capability. Also, different codes can handle certain types of errors better than others. Most codes have separate error detection and correction capability. The experiments described above have established that error detection capability is extremely important for the system. The addition of error correction capability can improve the throughput of the system and can also help in reducing the retransmission buffer length at the transmitter, saving hardware resources.

Table 4.1 shows the comparison of BER, PER and packet memory P_M for different settings before and after a 3-bit FEC is applied. The minimum measureable PER is 1.0E-5 and BER is 1.0E-8. Note that apart from the reduction in BER and PER, the FEC also impacts the packet memory, which dictates the amount of data-buffering capability required at the transmitter end. Smaller memory means only a small buffer is required to hold the data until it is successfully transmitted. The cumulative distribution value of 0.9 was chosen to calculate these memories.

Table 4.1: Comparison of BER, PER and Packet Memory for different settings before and after 3-bit FEC is employed.

		Before FEC			After 3-bit FEC		
Distance	Setting	BER	PER	P_M	BER	PER	P_M
5mm	Air	1.00E-08	1.00E-05	0	1.00E-08	1.00E-05	0
	Skin	1.32E-06	5.22E-04	0	5.64E-07	8.75E-05	0
	Disturbance	2.33E-05	9.13E-03	1	9.66E-06	9.70E-04	1
10mm	Air	6.27E-06	5.13E-03	0	1.00E-08	1.00E-05	0
	Skin	3.69E-05	7.43E-03	5	2.75E-05	2.48E-03	2
	Distrubance	1.28E-04	2.74E-02	3	9.17E-05	7.52E-03	1

Fig. 4.17 shows the comparison of PER before and after 3-bit FEC is applied. It can be seen that using as few as 3-bits of FEC reduces the PER greatly, which in turn reduces the number of retransmission necessary for error free data transmission.

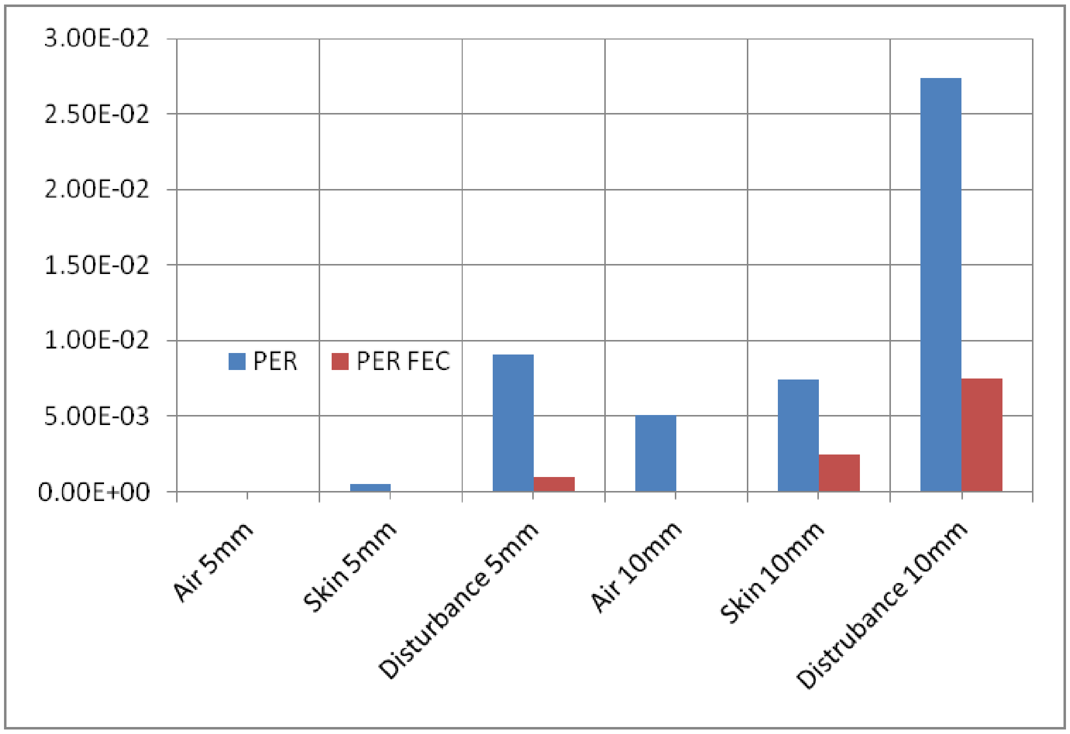


Fig. 4.17. Comparison of PER before and after 3-bit FEC for different settings.

Fig. 4.18 shows the comparison of BER before and after 3-bit FEC is applied. The effect of FEC is not very profound on BER which means that, although most of the packets can be corrected using FEC, the BER is contributed mostly by a few packets with a large number of bits in errors. It can be concluded that, although BER may appear high, the system needs only a few packets of retransmission to maintain a high data throughput.

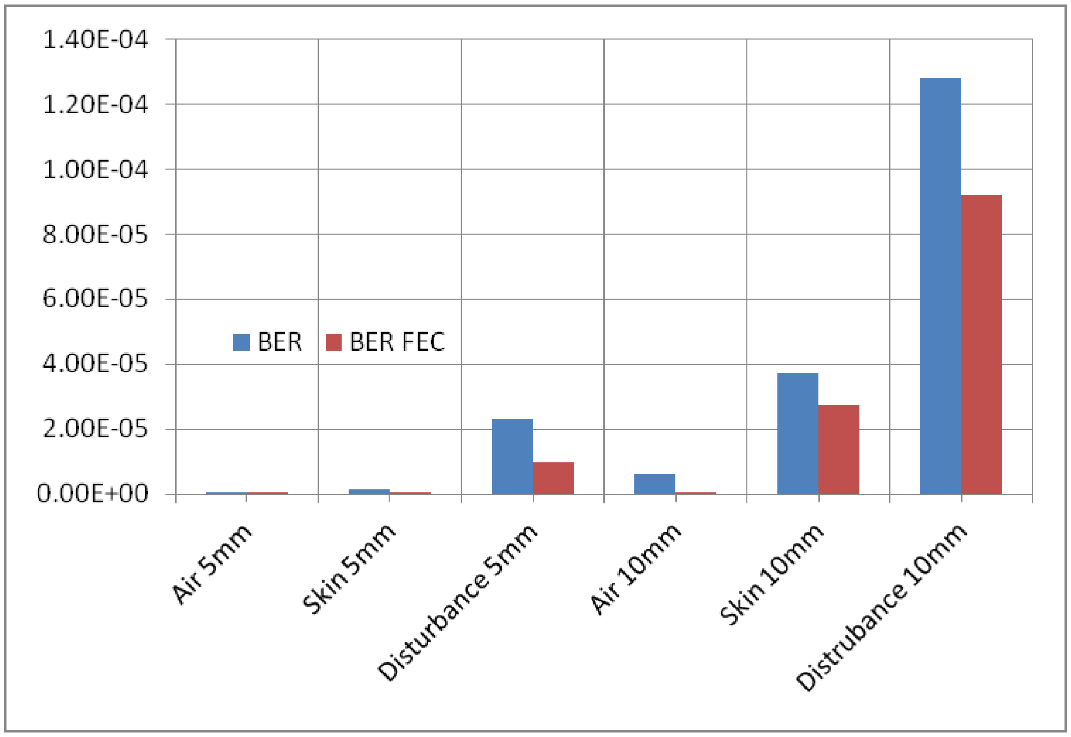


Fig. 4.18. Comparison of BER before and after 3-bit FEC for different settings.

4.8 Conclusions

From the above results it can be concluded that, although it is important to keep the BER to a minimum, it is the PER and the packet memory that most affect the throughput of the system (by forcing packet retransmissions) and the hardware resources required for error-free communication. For the typical setting of 5mm distance between transceiver and receiver with no disturbance, the communication controller developed by this research and presented in previous chapters is capable of handling the communication task effectively. A departure from the typical channel parameters will still work correctly as long as the transceiver settings result in a small packet level channel memory and the compression engine produces a reasonable output data rate. However, if the channel has a greater packet level memory, then the transmission buffer may overflow. This, however, can be avoided by increasing the size of the transmission buffer and making appropriate changes to the packet structure. These changes are discussed in the section for future work.

5 High Density Feature Extraction

5.1 Introduction

Advances in microelectronics and nanostructures have enabled scientists to combine thousands of electrodes into microelectrode arrays [59], permitting capture of neural signals from hundreds of neurons simultaneously for use in neuroprosthetics applications and neuroscience research. However, existing implantable wireless transceivers lack the capability to transmit the large amount of data generated by these electrodes. This mandates the use of implantable signal processing systems that enable on-chip data reduction while maintaining necessary information embedded in the neural spikes. On-chip feature extraction for spike sorting has been identified as a way to reduce transmission bandwidth [60].

Existing methods of off-chip feature extraction for spike sorting, e.g. template matching, principal component analysis (PCA), and time-frequency transforms, are computationally too demanding to be realized in implantable hardware, as reported in the comprehensive survey [61]. For implantable systems, the area required by circuitry is also a major concern in addition to the power consumption during computations. It has been shown that memory is the largest block in neural recording systems incorporating on-chip signal processing [43]. Thus, it is highly desired to reduce the number of computations while also using a minimum number of memory elements. Most high performance algorithms are not scalable to 1024 channels, keeping within the area and power limitations, due to the incurred hardware cost of complex processing [43,61]. An implanted system has to be low-power, low-area, highly accurate, automatic, and able to operate in real-time [61].

This section presents a new set of features for spike sorting, named Zero-Crossing Features (ZCF), and compares their sorting accuracy against PCA features for several data sets. PCA has been known to provide good separability between spikes for channels with reasonable signal to noise ratio. The ZCF have been explicitly framed to be extremely computationally efficient and to not require any offline training. Figures of merit for different stages of the system have been established. The effect of non-ideal spike detection on the performance of ZCF and PCA was studied. In addition, the computational complexity was analyzed and a scalable hardware architecture suitable for processing thousands of channels in an implantable system is presented.

5.2 Theory

Fig. 5.1. shows the general data flow of a spike sorting system. Input signal X[n] is the recording from electrodes containing spikes from multiple neurons. S[k] is the result of spike detection where only spikes samples are kept while noise is discarded, as a result reducing the number of samples. The spikes are fed to a feature extractor which generates F[p], where the number of values generated per spike equals the number of features.

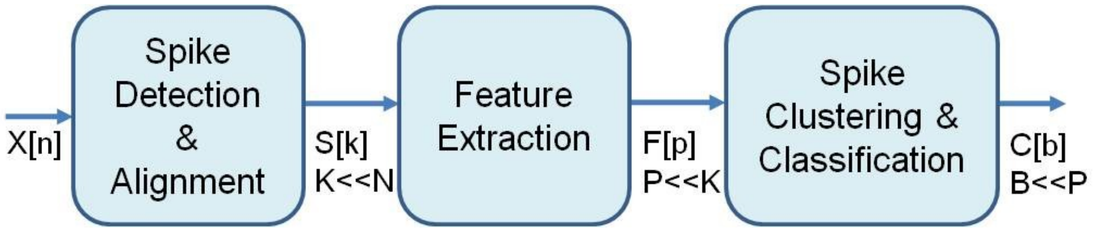

X[n]
Spike Detection & Alignment
S[k]
K<<N
Feature Extraction
F[p]
P<<K
Spike Clustering & Classification
C[b]
B<<P

Fig. 5.1. General data flow and typical components of a neural recording and classification. system.

The features are then used either on-chip or off-chip to classify the incoming spike as belonging to one of the possible neurons on that channel. All multi-channel systems keep track of the spikes by recording a ‘spike arrival time’ and a ‘channel identifier’ in addition to the features. Most systems have two phases of operation. The ‘training’ phase followed by the ‘acquisition’ phase [61-62]. During the training phase, the neural data is transmitted uncompressed and is used by external processors to calculate different parameters, e.g. detection thresholds and feature vectors. These parameters are then relayed back to the implanted module where they are used in the acquisition phase. These steps are repeated for all the channels. Nevertheless, ideally, a system suitable for recording from hundreds of channels should require minimum user intervention.

It is important to note that the ZCF feature extraction presented in this section does not require any training. Throughout this section, the ‘training’ phase is only used to train detection thresholds and classification algorithms. Although this section primarily discusses feature extraction, the performance of feature extraction is also dependent on the quality of the spike detector used. In addition, it is also dependent on the quality of spike classifier in the system.

5.2.1 Spike Detection and Classification

Several methods for spike detection have been reported in literature including data transformations, derivatives and template matching [62]. All methods employ a thresholding step for spike detection at the front-end or later in the system. Most high performance methods are computationally very intense and require offline training in addition to a large number of memory elements to maintain templates or temporary data. Thus, time-domain front-end thresholding is the most suitable option for high density multi-channel implanted systems [61], where the thresholds are computed off chip in the training phase. Using two independent thresholds, one positive and one negative, referred to herein as Dual Thresholds (DT), provides significantly improved performance as compared to the ‘absolute’ threshold. ‘Absolute’ threshold is a specific instance of DT where a single magnitude is compared to the incoming sample and the polarity of the sample is ignored. As a result the DT always performs equally good or better than absolute threshold. DT is chosen as the spike detector for this study because of its good detection performance.

Once the features have been extracted, several algorithms are available for off-chip spike classification. K-Means and Mahalanobis distance based classification were considered for this study. Mahalanobis classification was found to perform better for almost all tests and was used for classification henceforth. Off-chip Mahalanobis clustering also allows for adaptability in clustering as channel statistics change and spike amplitudes vary over time.

5.2.2 Feature Selection and Extraction

PCA and Wavelet Transforms (WT) have traditionally been used for off-chip spike sorting; however, they are deemed unsuitable for implantable on-chip multi-channel feature extraction because of the high computational cost and hardware resources required to map the neural data onto the feature space. Furthermore, both methods require offline training, and neither of the methods provides a generic set of features applicable across channels. This means each channel requires separate training and results in separate sets of features. This high computational cost mandates the design of a new set of features that is explicitly framed to be computationally efficient and do not require any offline training while maintaining good sorting performance.

The most prominent and visually distinctive features of a set of different spikes are the relative amplitude (or energy) of the spikes, the relative position of positive and negative peaks in the spikes, and the widths of the spikes. Each of these features can be used to sort spikes of one kind from the others, however, the ability to sort spikes based on these features falls sharply with an increase in noise level, i.e., in signals with low signal to noise ratio (SNR). However, all these features can be combined together to form a new set of features that are distinctive, resilient to noise and computationally efficient. The mathematical representation of the set of ZCF features can be expressed as

$$ZC1 = \sum_{n=0}^{Z-1} x(n) \qquad ZC2 = \sum_{n=Z}^{K-1} x(n) \tag{5.1}$$

where ZC1 and ZC2 are the two features, K is the number of samples in a spike and Z is the index of first zero crossing after the spike has been detected. Note that in a DT based spike detection system, the spikes are detected when the value of the spike is larger than either of the specified thresholds. The value of Z in (5.1) is thus the first zero crossing after a significant amount of energy in the spike has already been recorded. Features in (5.1) can also be seen as recording the pre-zero-crossing and post-zero-crossing energies of the spike. Collectively, these features are referred to as Zero-Crossing Features (ZCF) for the rest of the section.

The underlying assumption in selection of these quantities as a feature set is that the spikes from different neurons have different ‘area’ before and after the first zero crossing. Though, in theory, it is possible to have multiple neurons generating spikes with same energy before and after zero crossing, in practice, experimentally recorded spike waveforms show considerable differences in these two values allowing for the use of these features for spike sorting. In case the spikes have no zero crossing, ZC1 contains all the area of the spike while ZC2 remains zero. Even if multiple non-zero-crossing spikes are present in the channel, the features still maintain a degree of separability. An average spike is about 1.5ms long, which amounts to 30 samples at a 20Ksps sampling rate. The factor ‘K’ in (5.1) is thus set to 30 and is dependent on the sampling rate.

The ZCF features are graphically represented in the example of Fig. 5.2. When a spike is generated, the electrode potential changes very rapidly. In this example, the spike is detected when the electrode potential crosses the negative threshold. There are still some samples before the detection that belong to the spike and may contain useful information; thus it is desirable to be stored some samples in a buffer. For the sampling rate of 20Ksps, empirical results show that a buffer length of only 3 successfully captures all the useful samples and captures the onset of the spike, as shown in Fig. 5.2.

Buffer length and spike length are the two parameter that effect the amount of required hardware resources. The buffer length is dependent on the type of spike detector used, and the ideal detector is the one that detects the spike as soon as it is generated. Also note that, as in Fig. 5.2, only one spike can be detected within any K samples.

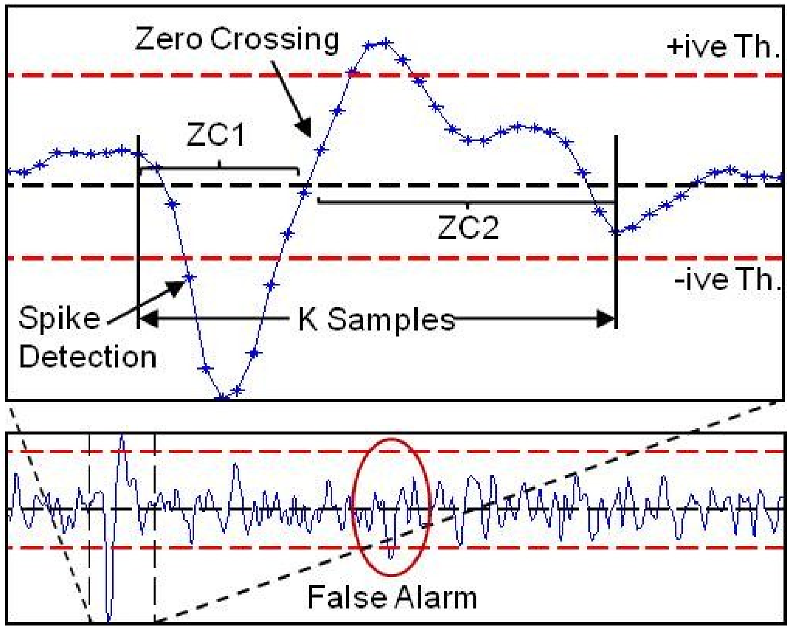


Fig. 5.2. Graphical representation of ZCF features for a spike detected using dual thresholds.

5.2.3 Complexity Calculations and Hardware Resources

The computation complexity of implementing the two features of ZCF was compared to that of implementing the two most significant components of PCA, where each of the PCA components has K dimensions. Computations for spike detection algorithms occur for each input sample, while the computations for feature extraction occur for each detected spike. To compare complexity in terms of addition operations, each multiplication is considered to be equal to 10 additions [62] whereas each 'comparison' operation is considered equal to one addition. For the case of DT spike detection, use of two thresholds means that every sample is compared against either positive or negative threshold, thus only one computation occurs per sample.

From Table I, it can be seen that for each detected spike, the ZCF needs less than 5% of the computations and requires less than 8% of the memory as compared to the PCA.

Table 5.1: Comparison of Complexity and Required Memory for Different Algorithms

	Add.	Mult.	Complexity	Memory
DT	1	0	1	2
ZCF	30*	0	30*	5
PCA	60*	60*	660*	65

* Computations per spike

Fig. 5.3 gives the hardware architecture of the proposed DT signal detection and ZCF feature extraction algorithms. Light colored blocks represent memory elements. A three word FIFO is updated every new sample to implement the buffer that ensures spike data before detection is not lost. When a spike is detected, the control logic uses the data stored in FIFO and the subsequent input values to accumulate in ZC1. When the zero-crossing-detector (ZCD) detects a change in sign, then ZC2 is accumulated until all the 30 samples representing a spike are received.

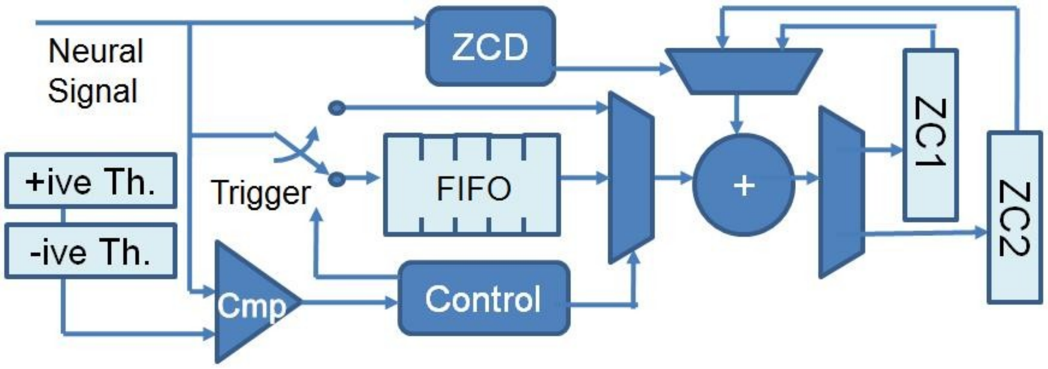


Fig. 5.3. Hardware architecture of proposed signal detection and feature extraction stages units.

It is important to note that, with a system for 1024 channels; one additional byte of storage per channel translates to 1Kbyte of storage for the system. Similarly, each additional computation per channel means 1024 additional system computations.

5.3 Methods

Neural spikes recoded from live experiments were used in conjunction with a neural signal simulator to generate signals that mimic electrode recordings. A detailed discussion of experimental procedures to collect neural data can be found in [52]. A total of ten different spike shapes were used to mimic ten separate neural channels, each consisting of three spikes. Fig. 5.4 shows the 10 spike channels used in these simulations. For each dataset, one second of 'training' phase was completed to determine thresholds for DT, principal components for PCA and initial statistics for Mahalanobis clustering. The signals generated did not contain any overlapped spikes, and are generated for different values of SNR, ranging from 18dB to -5dB, as computed by

$$SNR = 20\log_{10}\left\{\frac{\sigma_x}{\sigma_n}\right\} \tag{5.2}$$

where σ_x and σ_n are the signal and noise standard deviations, respectively.

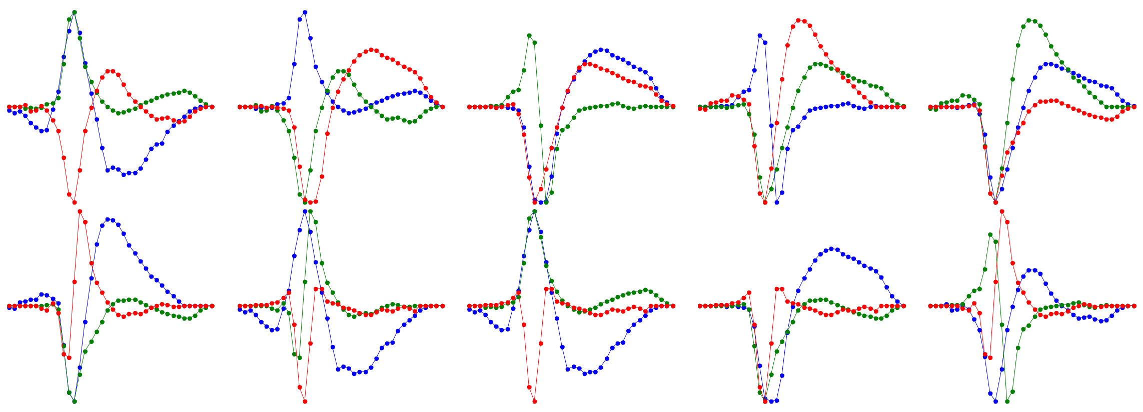


Fig. 5.4. Channels and spikes used in simulations. Ten different channels consisting of ten different spike shapes were used with each channel containing three spike shapes. (Top Row) Channel 1-5 from left to right. (Bottom Row) Channel 5-10 from left to right.

The performance accuracy of a spike sorting system depends on the collective and individual performance of each of the components. Performance accuracy of the detection algorithms is computed based on the number of spikes missed and the false alarms in relation to the number of original spikes generated by the simulator. Detection Error for Dual Threshold, DE_{DT}, is represented by [63].

$$DE_{DT} = \frac{MS + FA}{OS + FA} \times 100 \tag{5.3}$$

where OS is the number of original spikes, FA is the number of false alarms and MS is the number of missed spikes. Performance of ZCF is compared against the performance of PCA; however, PCA requires a training phase to find the two most significant principal components. When 'perfect' detection is used then Classification Error, CE, can be calculated as

$$CE_{P} = \frac{MCS}{OS} \times 100 \tag{5.4}$$

where MCS is the number of mis-classified spikes. Accuracy of the feature extraction algorithm is computed through (5.4) treating misses or false alarms as misclassified spikes. The effects of non-ideal spike detector on classification accuracy are reflected in the classification error formula

$$CE_{DT} = \frac{MCS - FA}{DS - FA} \times 100 \tag{5.5}$$

where DS is the total number of spikes detected. The performance of feature extraction is isolated from the performance of non-ideal spike detector by (5.5). Since different sets of features are effected differently, (5.5) represents the effects of mis-alignment on the classification accuracy caused due to imperfect spike detection. This can happen in instances when the noise is such that it suppresses the onset of the spike creating a slight 'delay', resulting in a spike of shorter duration; or when the noise is such that it creates a high magnitude just before the onset of the spike giving the illusion that the spike started earlier and thus lives longer than other spikes from the same neuron.. Here, the number of misclassified spikes includes all the false alarms, because false alarms are always wrongly classified. The performance of the complete system including the non-ideal DT spike detector, feature extractor and classifier can be computed as

$$DCE_{DT} = \frac{MS + MCS}{OS + FA} \times 100 \tag{5.6}$$

5.4 Results and Discussion

Fig. 5 shows an example of how well the ZCF based clustering performs and when Mahalanobis clustering outperforms K-Means clustering sanctioning the use of Mahalanobis to create clusters .

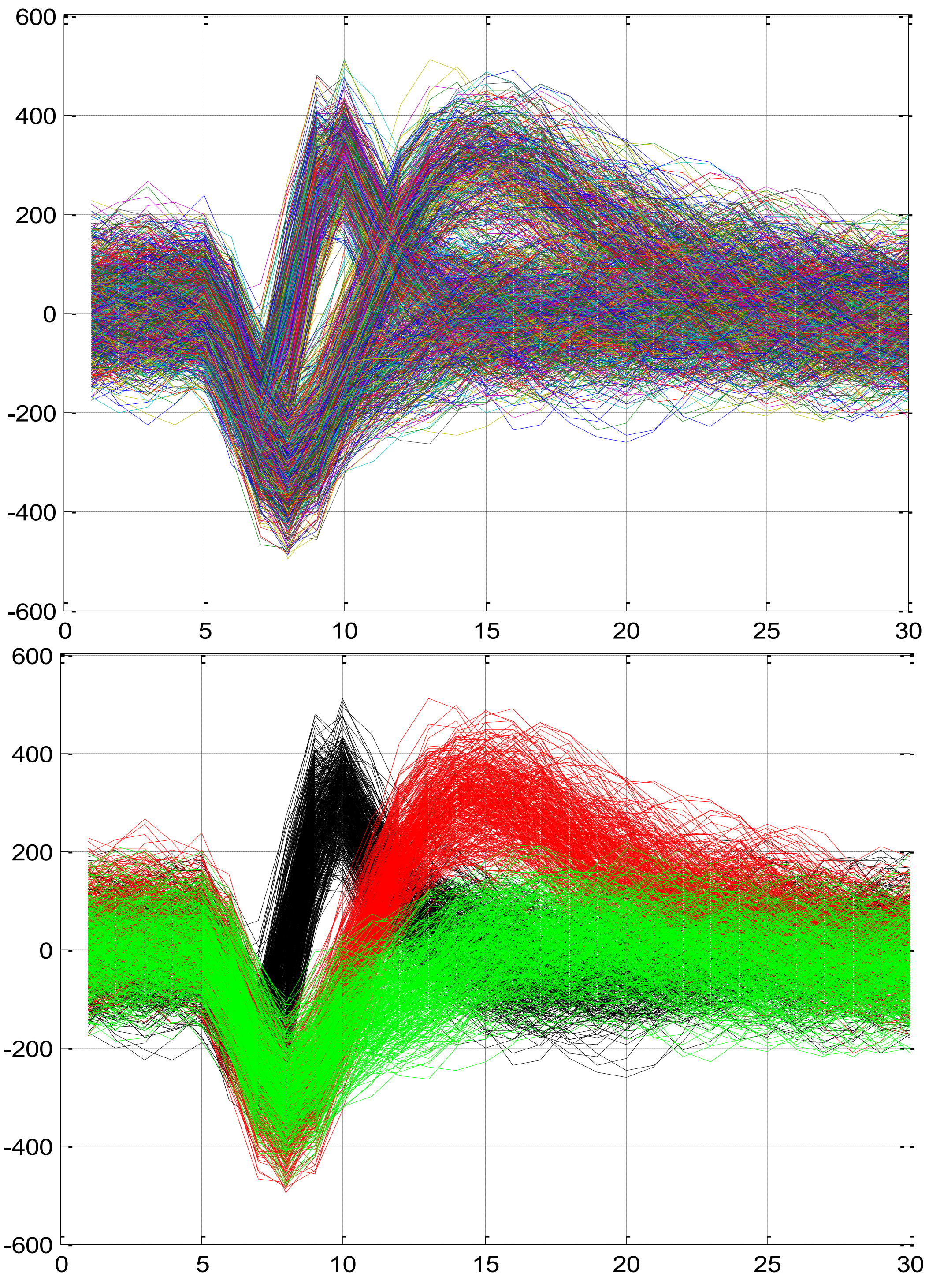
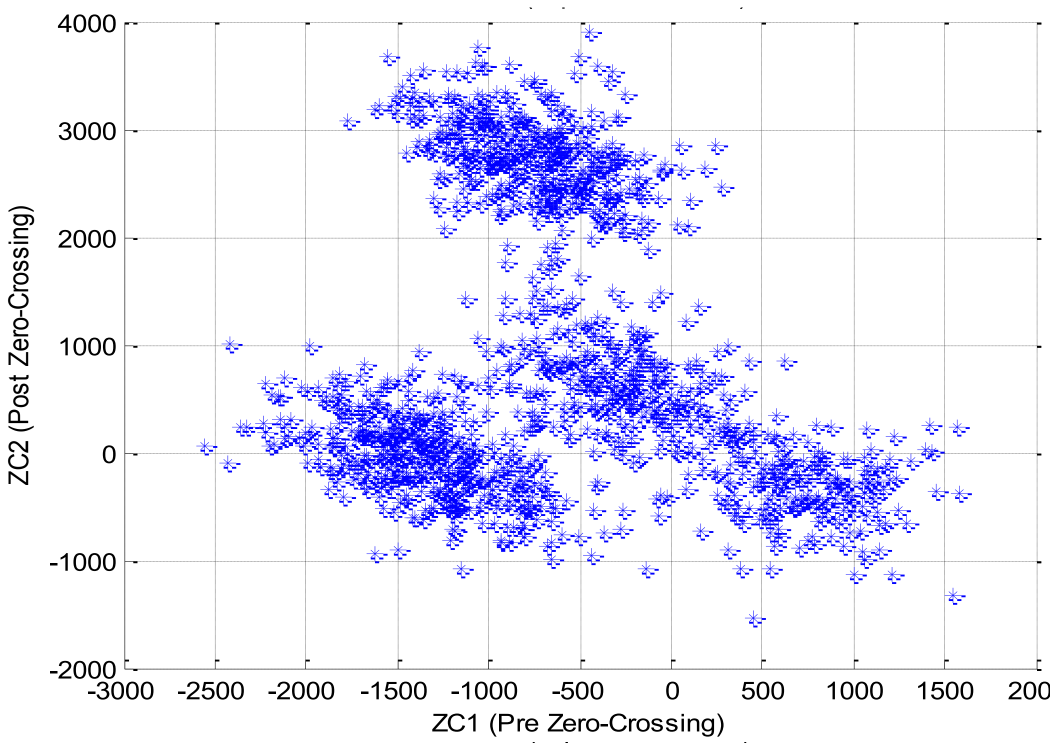


Fig. 5.5a. (Top) Spikes detected on the channel. (Bottom) Color-coded spikes for different neurons. This knowledge is available from the neural signal simulator and is presented here only for reference.



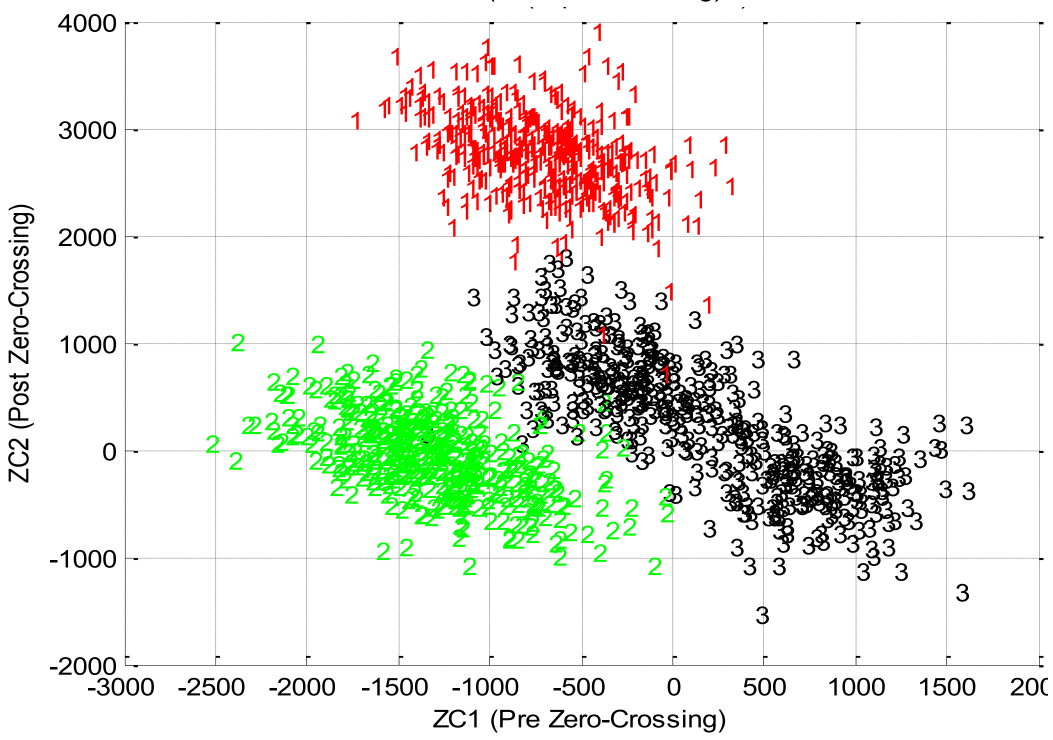
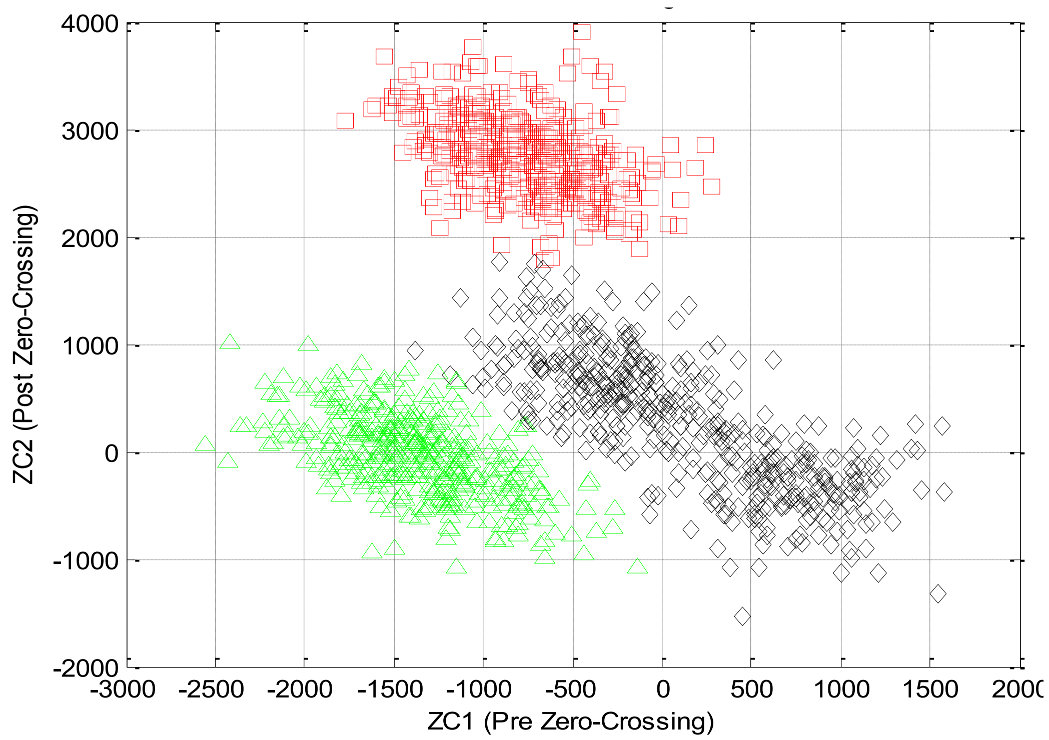


Fig. 5.5b. (Top) ZCF mapping of the spikes on the ZCF feature space as seen by the spike classifier. (Bottom) ZCF mappings are color-coded to distinguish between neurons.



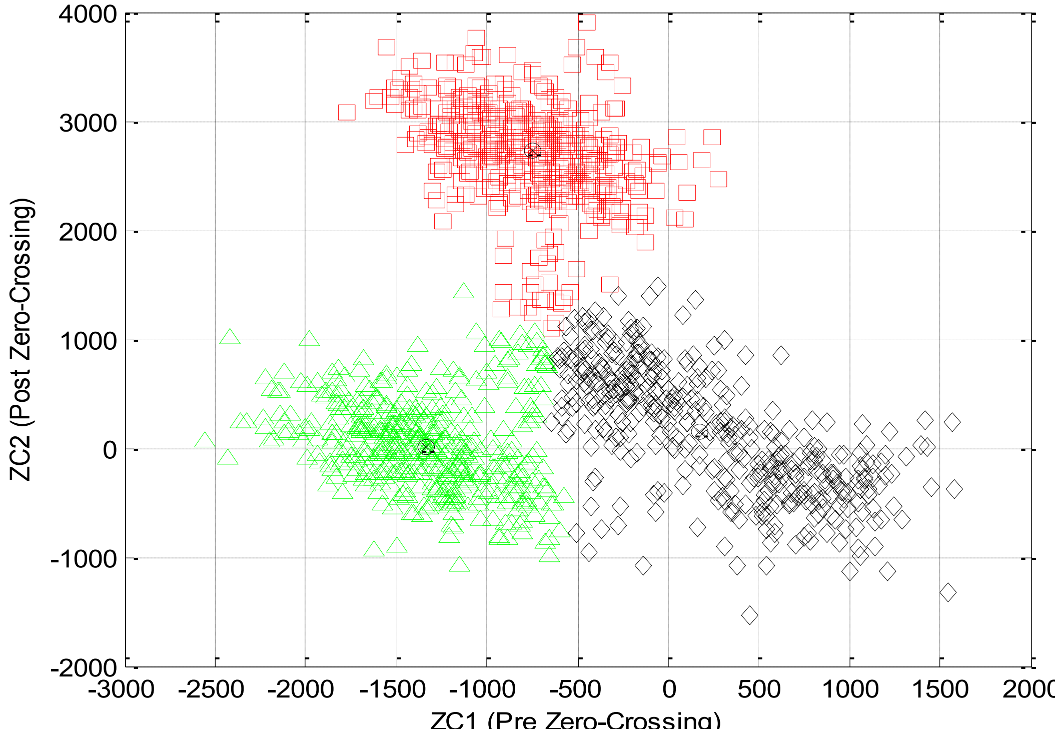


Fig. 5.5c. (Top) ZCF mappings are grouped into three clusters using Mahalanobis distance, 1.71% spikes are mis-classified. (Bottom) ZCF mappings are grouped into clusters using K-Means clustering, 6.14% spikes are mis-classified.

Neural channel 6 from Fig. 5.4 was simulated for an SNR of 3dB. Fig. 5.5a(Top) shows all the spikes that are present on the channel and are detected. Since prior knowledge is available from the neural simulator regarding the true identities of these spikes, the detected spikes are color-coded to distinguish between different neurons in Fig. 5.5a(Bottom). Fig. 5.5b(Top) shows the mapping of the spikes on the ZCF feature space with ZC1 (pre zero-crossing feature) on the x-axis and ZC2 (post zero-crossing feature) on the y-axis, as seen by the classifier. Fig. 5.5b(Bottom) shows the color-coded mappings, using prior knowledge from the neural simulator, as a reference to distinguish between the three neurons. Fig. 5.5c(Top) is the clusters created by the classifier using Mahalanobis distance. Note that, although it looks very similar to the color-coded mapping on center right, there still are some mis-classified spikes. Out of 1286 spikes shown, 22 spikes are misclassified giving an mis-classification ratio of 1.71%. Fig. 5.5c(Bottom) shows the clusters created by the classifier using K-Means clustering. The mis-classified spikes can easily be identified in the middle of the feature space. A mis-classification ratio of 6.14% is recorded. Both classifiers need some training. The K-Means classifier needs sample points to calculate the centeroid of the cluster while the Mahalanobis clustering needs sample points to compute the covariances of the clusters.

Fig. 5.6 shows the performance results of detection and classification stages using the four figures of merit described by (5.3)-(5.6). Ten different data sets were used in simulations, and the resulting averages have been plotted. Fig. 5.6(i) shows that the positive and negative DT threshold values that generate the minimum detection error (DE_{DT}) in the initial training phase result in comparable error in the acquisition phase. The detection error increases sharply for an SNR of 3dB and below where noise power becomes comparable to signal power. Fig. 5.6(ii) compares the performance of ZCF and PCA and represents the classification error when ideal detection is assumed (CE_P). It can be seen that ZCF and PCA perform equally well even under low SNR conditions. Thus, even though ZCF requires only 5% of the resources as PCA, it can perform as well as PCA-based features. Fig. 5.6(iii) shows that, when non-ideal spike detection is employed, the resulting mis-alignment of detected spikes adversely effects the classification performance (CE_{DT}) of both ZCF and PCA algorithms. The errors have increased by about 50% for each SNR. However, it can be seen that there is no performance loss at high SNRs and that both ZCF and PCA still perform equally well for all SNRs. Fig 5.6(iv) shows the system level results including spike detection, feature extraction and classification. The performance of ZCF and PCA are nearly indistinguishable. It can be seen that, even though the ZCF and PCA are capable of performing well under low SNR conditions (e.g. 5% errors at 0dB), the system performance (55% errors at 0dB) is dominated by the perfromance of the spike detector (45% errors at 0dB in addition to the mis-alignment of detected spikes). Note that this analysis does not reject the possibility that if more than two features were used in PCA, then its classification performance may improve, albeit at significant cost of increased hardware resources.

A 1024 channel system with a sampling frequency of 25Ks/sec/channel and 10 bits ADC resolution would generate raw data at about 250Mbits per second. Using ZCF feature extraction, the information to be transmitted could be reduced to a time stamp, a channel ID and the two computed features (around 50 bits per spike). Assuming the average number of neurons on each channel to be three and the neuron firing rate to be 30 spikes per second, data would only need to be transmitted at the rate of ~4.7Mbps using ZCF. Thus, ZCF permits transmission at is less than 2% of the raw data rate. The resulting savings to transmission power is purchased by the cost of ZCF computations, which average to, 2.86 million computations per second during steady state operation. In comparison, PCA provides similar data reduction but costs 60.91 million computations per second, many times the dynamic power of ZCF.

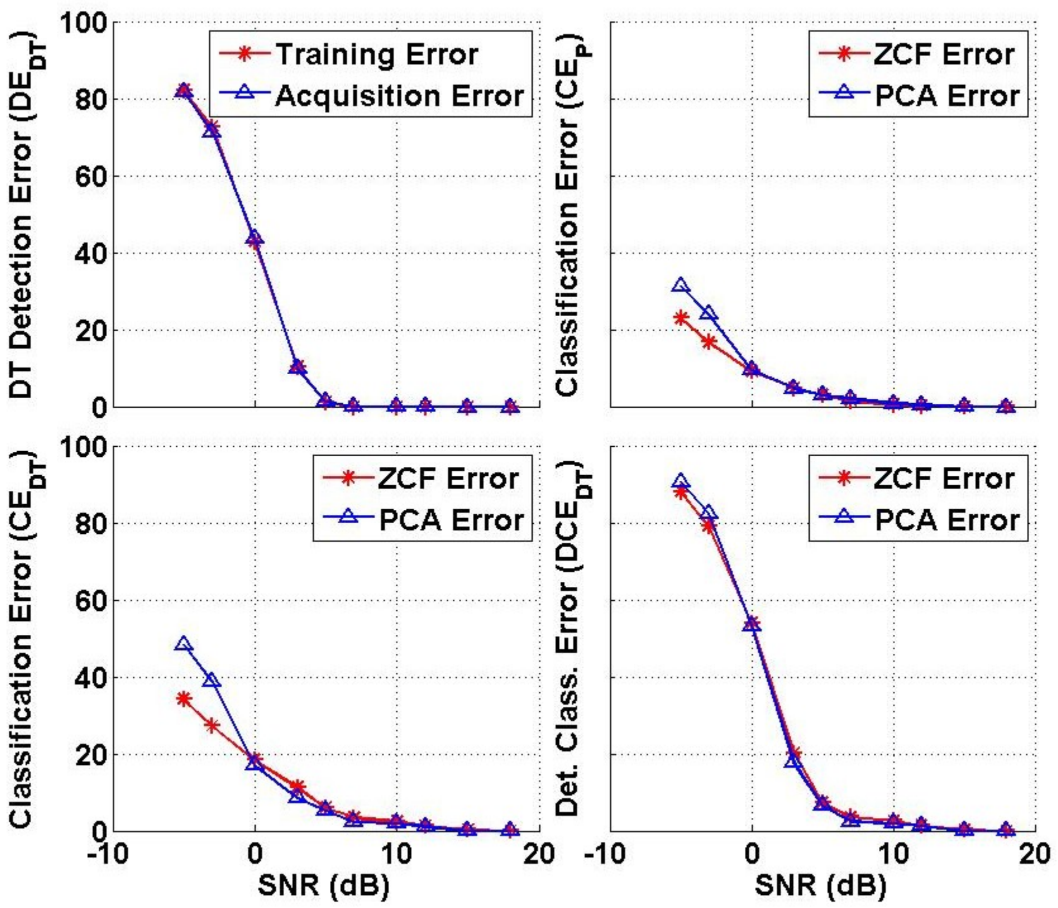


Fig. 5.6. Percentage values of (i) DE_{DT}, (ii) CE_P, (iii) CE_{DT} and (iv) DCE_{DT} versus different values of SNR. Data represents performance results for the DT spike detector, the ZCF and PCA feature extractor and the combined spike sorting system.

5.5 ZCF vs DWT

In this section we present a comparison of a DWT based spike sorting system, henceforth called DWT, against a ZCF based spike sorting system, henceforth called ZCF.

1. **Sorting Performance**: DWT has been shown to have slightly better spike sorting results than PCA. Since ZCF performs as good as PCA, we can assume that DWT would be able to perform better than ZCF.

2. **Reconstruction Flexibility**: DWT is capable of allowing various degrees of reconstruction by changing the multi-level thresholds.As a result the actual spike shapes may be reconstructed if needed. ZCF based features do not allow such reconstruction, however a ‘monitor’ mode can be used to record the spike shapes in the time domain.

3. **Computational Complexity and Power Consumption**: There is no spike detection before DWT; as a result all of the input samples go through the time-frequency transformation, which is computationally hungry. Each input sample requires 8 multiplications and 8 additions for a 4 level transform. ZCF processes ‘detected’ spikes only, and thus computes features only when a spike is detected, which saves a huge amount of power and resources.

4. **Detection performance**: In DWT, detection is applied offline once the data has been transmitted. This allows a higher performance algorithms to be used for spike detection. The performance of ZCF is affected by the performance of an on-chip spike detector which may limit the overall system performance.

5. **Neural Activity**: Power consumed by a ZCF system is proportional to the neural activity on the channels. DWT processes all samples of the channel even if it has low neural activity.

6. **Classifier Complexity**: ZCF requires only a 2 dimensional spike classifier while a 4 level DWT requires a 5 dimensional classifier. However, the complexity of classification may not be directly proportional to the dimensionality of classifier.

7. **Scalability**: DWT requires a lot of memory for computation and thus does not seem scalable for systems with over a thousand channels. ZCF has been designed to require minimum memory, only one adder and no multipliers, and thus is scalable to a very large number of channels.

8. **Training and Feedback**: Although taking the time-frequency transform in DWT does not require any supervision, selection of appropriate thresholds to retain or discard features requires initial training and feedback from an external processors or user. ZCF does not require any initial training, however the preceding spike detector may require some offline training.

9. **Compression**: ZCF produces two features for each detected spike, while DWT can, theoretically, be trained to produce a single feature capable of providing good sorting results in addition to some feature identification information. The resulting throughput of both system appears to be similar.

The comparison presented above highlights the strengths and weaknesses of both the DWT-based and the ZCF-based systems. The DWT has a clear advantage when any level of signal reconstruction is needed after the data has been received at the output. In addition, because of its time-frequency domain processing, DWT-based systems may provide better spike detection and sorting performance as compared to only time-domain based ZCF; however the difference in performance is not clear and depends heavily on the performance of threshold determination algorithm running in the external processing units. On the other hand, though the DWT has been shown to effectively handle multi-channel processing, the ZCF is scalable to a much higher number of channels that can be processed on-chip. This is possible because of the extremely resource efficient implementation of ZCF resulting in smaller IC area and very low power consumption. The power consumption is directly related to the amount of neural activity observed, as opposed to the constant power consumed by the DWT.

It can be concluded that both systems perform well for different kinds of applications. The DWT is suitable when detection and sorting accuracy take precedence over the number of simultaneously processed channels, whereas ZCF is suitable when extremely large number of channels are processed while slight performance loss is acceptable.

6 Summary and Future Work

The summary of this Ph.D. thesis research is provided below followed by a brief description of the contributions made in this thesis and a list of tasks that could further this research in the future.

6.1 Summary

Neuroprosthetics devices and Brain Machine Interfaces (BMIs) are increasingly playing a vital role in helping patients with severe motor disorders achieve a better lifestyle by enabling direct interface to the central nervous system. Such BMIs show great promise in many biomedical applications.

This thesis addresses some of the critical challenges that impede progress in wireless neural implants by developing highly efficient algorithms and hardware suitable for implantation. Employing front-end on-chip signal processing, the system is tailored to enable simultaneous multi-channel neural recording, compression and transmission of 32 neural channels. The system uses an area-power efficient discrete wavelet transform based compression engine to compress the multi-channel data while providing control over the quality of reconstruction needed for the particular application. Lossy as well as lossless compression algorithms combine to provide high compression ratios. A host of low power design techniques have been employed in addition to a power management module to keep the system under the low area, low power and low bandwidth constraints. A communication protocol has been designed to enable error free data communication between the implanted device and the external module on body surface. A salient feature of the system is its ability to compress multi-channel data while preserving the important attributes of the signals, allowing off-chip spike sorting. Moreover, a scalable design enables simultaneous processing of a higher number of channels. In addition to this core functionality, the ASIC allows configurability of the entire system supporting real time operation, different modes of operation and different levels of compression as desired by the neuroscientist.

The design of a reliable communication and power link between an implanted transmitter and a receiver on the body surface depends heavily on the channel characteristics across animal skin and the characteristics of low power inductive links. These error characteristics have never been explored in literature before. This thesis empirically determines the channel characteristics and analyses the performance of the inductively coupled data and power transceiver by presenting results from extensive testing for different settings and orientations of the directional transceivers. The channel characteristics can be used to model the communication link for a inductive power-data transceiver implanted in any part of the body with different thicknesses of skin.

Finally, to match the pace of rapidly increasing number of channels on a microelectrode array, a new set of features named ‘Zero Crossing Features’ has been identified and analyzed for extremely efficient feature extraction in hardware without sacrificing any spike sorting performance. The sorting capability of the features is compared against the features from principal component analysis and shown to perform equally well. A resource efficient hardware architecture is also presented for complexity analysis and proves its feasibility for implantation.

6.2 Contributions

This thesis makes the following contributions to the field of biomedical engineering.

1. The prohibitively large amount of data generated by the microelectrode arrays results in a data communication bandwidth bottleneck. We have designed and developed a new implantable system that enables control over the amount of transmitted data without compromising the vital information contained in the array of signals. As a result, a large number of channels can be acquired simultaneously within the same bandwidth where the scientist controls which channel gets more share of the bandwidth than others do. This will allow the scientists to allow more resources to the channels of interest and is expected to improve neural decoding accuracy in neuroprosthetics applications.

2. We have developed a unique architecture for discrete wavelet transform that is designed to be extremely area-power efficient in addition to being easily adaptable to implement any other wavelet basis with any number of levels of decomposition. Accordingly, the DWT architecture can be used as is or can be adapted for use in a wide variety of sensor arrays for signal compression or pattern recognition applications.

3. The unreliable nature of wireless channels poses a formidable challenge in the pattern of bit and packet errors seen at the receiver. We have created the first ever error profile for inductively coupled transceivers transferring data and power between the implanted device and the external world across the skin tissue. In addition, we have also designed an efficient half-duplex protocol for streaming data to maximize the data throughput and to enable unfailing bidirectional data transfer. This error profile enables scientists to evaluate design tradeoffs for error correction, data throughput, transceiver hardware complexity and power consumption. The general nature of transceiver and skin channel ensures that the profile and the resulting protocol are valid not only for neural but for any implantable system.

4. We have developed the first ever implantable digital system that employs on-chip high level signal processing of multi-channel neural data. Hardware has been designed for all the components including the analog interface controller, the compression engine and the communication controller, which have been combined together into a single system keeping within the implantability limitations of area, power and power density. Although the motivation for this design comes from the field of neuroprosthetics, the system developed can be used in a wide range of implantable applications.

5. Advances in MEMS have allowed thousands of channels to be fabricated on to a microelectrode array; however, simultaneous recording from such a large number of channels has not been possible. To enable recording from such high-density microelectrode arrays we have designed a new ultra-constrained set of spike sorting features named 'Zero-Crossing Features' which allows computationally and hardware efficient online spike sorting without requiring any training or user intervention. Consequently, enabling simultaneous recording from thousands of channels will allow decoding of physical and functional inter-connections between large swathes of neurons, helping our understanding of brain functionality.

6.3 Future Work

Following is a set of tasks that can further this research.

1. **Update the Communication Protocol using Channel Characteristics**

With the channel error characteristics available through this research, it is now possible to improve the communication protocol and optimize for this particular biotelemetry application. The goal is to maximize the throughput while keeping the power consumption and transmission buffer length as small as possible. Based on the results from chapter 4, the number of parity bits included for error detection could also be re-evaluated. The cost to compute and transmit FEC bits needs to be evaluated further for the tradeoff between power consumed in transmission relative to the benefit of increase in throughput. It is evident from Fig 4.10 and Fig 4.15 that the system should have a very strong error detection capability, although only a small error correction capability is sufficient. FEC algorithms need to be evaluated for the amount of error correction and error detection capabilities that they provide relative to what is needed by this system for error free low power data communication.

2. Optimization for Low Power Operation

Once the functional verification of the system is complete, the design can be analyzed to identify areas where power saving techniques can be employed to reduce the power consumption of the system as a whole, including both the digital blocks as well as the analog front end. For the target technology of 0.13 micron CMOS, the focus of this effort is to reduce the combined leakage and dynamic power of the system while keeping a check on the area overhead. These techniques may include Multiple Supply Voltages, Dynamic/Adaptive Voltage and Frequency Scaling, Clock Gating, Transition rate buffering, operand isolation, logic restructuring, memory splitting, substrate biasing, multi threshold voltage and power shutoff.

3. System Design for the Ultra Constrained Spike Sorting

The hardware design of ultra constrained feature extraction can be extended to include spike detection and spike classification. The designs need to be low complexity, low power and resource efficient enough such that their resource consumption does not overshadow the complexity gains achieved by the feature extraction algorithm. Current designs for spike detection are either too resource hungry for an implantable system or do not have good detection performance.

Spike classification can also be performed through several different algorithms; however most high performance classifiers are computationally expensive. It is thus required to evaluate the tradeoff between power consumption for on-chip spike classification or transmission power required to transmit unsorted spike features to be used off chip.

4. **Evaluate the Number of Input Output Pins**

It has been learnt from our experience that for advanced technologies like 0.13 micron CMOS, the IO and power supply pads for various inputs and outputs take up a sizeable amount of overall area, as shown in Fig. 6.1, mainly to enable packaging and for protection against electrostatic discharge. The pads are also the biggest power consuming blocks in the system since they add large capacitances at the interface. Methods to reduce this non-essential power consumption could be explored. One way to reduce the number of IO pads is the use of serial communication between various components of the design. The cost may involve increasing the clock frequency or use of frequency multipliers.

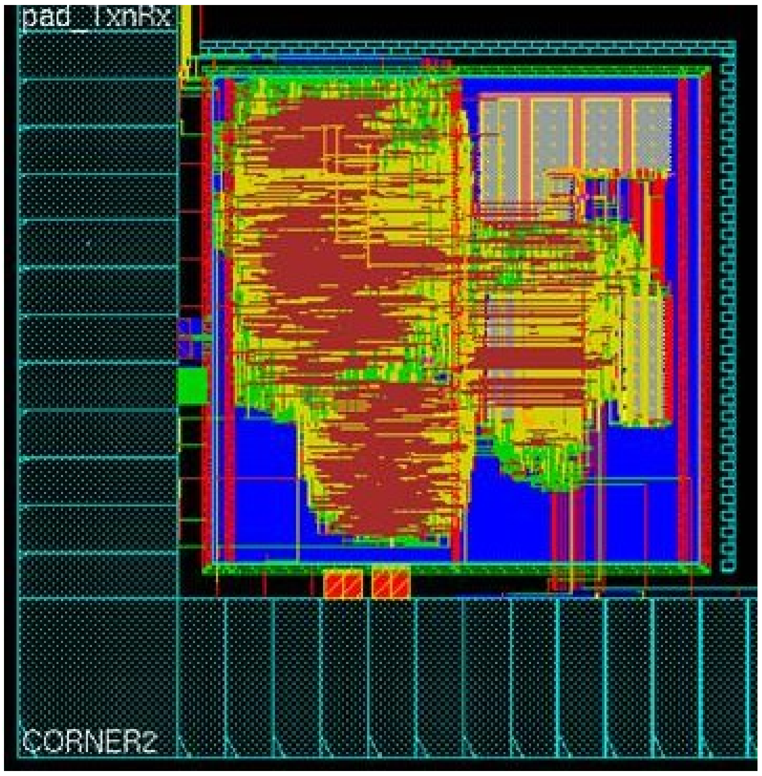


Fig. 6.1. NIN Chip fabricated in 0.13 micron CMOS

5. Test of Fabricated ASIC in Live Animals

Once the system chip presented in Chapter 3 is fabricated it needs to be tested in live animals for two things mainly, functionality and power consumption. The tests will verify the initial choices of appropriate sampling rate and suitable quantization levels, and will verify the assumptions of amount of data generated and the amount of data necessary for off chip spike sorting. Long-term usability of the system could also be evaluated for any unforeseen glitches in hardware implementation. Including the wireless data and power transfer will enable testing of effects of short-time power outages and the functionality of communication controller. Once the testing is complete, a data sheet and user manual could be created to enable transfer of the chip to users unfamiliar with the design.

6.4 Papers Written

Journals

1. A. M. Kamboh, A. Mason, and K. G. Oweiss, “Analysis of Lifting and B-Spline DWT Implementations for Implantable Neuroprosthetics,” Journal of Signal Processing Systems, vol. 52, no. 3, pp. 249–261, September 2008.

2. K. G. Oweiss, A. Mason, Y. Suhail, A. M. Kamboh, and K. E. Thomson, “A Scalable Wavelet Transform VLSI Architecture for Real-Time Signal Processing in High-Density Intra-Cortical Implants,” IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 54, no. 6, pp. 1266–1278, June 2007.

3. A. M. Kamboh, M. Raetz, K. G. Oweiss, and A. Mason, “Area-Power Efficient VLSI Implementation of Multichannel DWT for Data Compression in Implantable Neuroprosthetics,” IEEE Transactions on Biomedical Circuits and Systems, vol. 1, no. 2, pp. 128–135, June 2007.

4. Awais M. Kamboh, and A. J. Mason, “High Fidelity Neural Signal Compression and Wireless Communication,” IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2010 (In Prep).

5. Awais M. Kamboh, and A. J. Mason, “Ultra Constrained On Chip Feature Extraction for Neural Spike Sorting,” IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2010 (In Prep).

6. Awais M. Kamboh, Mehdi Kiani, Maysam Ghovanloo, and Andrew J. Mason, “Performance Analysis of Inductively Coupled Transceivers for Power and Data Transfer in Wireless Implants,” IEEE Transactions on Biomedical Circuits and Systems, 2010 (In Prep).

Conferences

7. A. M. Kamboh, and A. J. Mason, “On Chip Feature Extraction for Spike Sorting in High Density Implantable Neural Recording Systems,” IEEE Conference on Biomedical Circuits and Systems, Paphos, Cyprus, pp. 13-16, Nov 2010.

8. Y. Yang, A. M. Kamboh, and A. J. Mason, “Adaptive Threshold Spike Detection Using Stationary Wavelet Transform for Neural Recording Implants,” IEEE Conference on Biomedical Circuits and Systems, Paphos, Cyprus, pp. 9-12, Nov 2010.

9. A. M. Kamboh, Y. Yang, K. G. Oweiss and A. J. Mason, “Design of a Configurable Neural Data Compression System for Intra-Cortical Implants,” IEEE International Symposium on Circuits and Systems, (ISCAS 2010), pp. 3473-3476, Paris, France, June 2010.

10. A. M. Kamboh, K. G. Oweiss and A. J. Mason, “Resource Constrained VLSI Architecture for Implantable Neural Data Compression Systems,” IEEE International Symposium on Circuits and Systems, (ISCAS 2009), pp. 1481-1484, Taipei, Taiwan, May 2009.

11. F. T. Abu-Nimeh, A. Kamboh, M. Aghagolzadeh, U.-M. Jow, A. Mason, M. Ghovanloo, K. Oweiss, "A Highly Modular, Wireless, Implantable Interface to the Cortex," IEEE/EMBS Conference on Neural Engineering, Antalya, Turkey, pp. 375-378, April 2009.

12. A. M. Kamboh, M. Raetz, A. Mason, and K. Oweiss, “Area Power Efficient Lifting based DWT Hardware for High-Yield Area-Power Efficient DWT Hardware for Implantable Neuroprosthetics,” IEEE International Symposium on Circuits and Systems, (ISCAS), New Orleans, Louisiana, USA, pp. 2371-2374, May 2007.

13. M. A. Shetliffe, A. M. Kamboh, A. Mason, and K. G. Oweiss, “Impact of Lossy Compression on Neural Response Characteristics Extracted From High-Density Intra-Cortical Implant Data,” IEEE/ EMBS International Conference of the IEEE Engineering in Medicine and Biology Society, (EMBS 2007), Lyon, France, pp. 5358–5361, August 2007.

14. A. M. Kamboh, A. Mason, and K. Oweiss, “A High-Yield Area-Power Efficient DWT Hardware for Implantable Neural Interface Applications,” 3rd International IEEE/EMBS Conference on Neural Engineering, (CNE 2007), Kohala Coast, Hawaii, USA, pp. 212–216, May 2007.

15. A. M. Kamboh, A. Mason, “Comparison of Lifting and B-spline DWT Implementations for Implantable Neuroprosthetics,” 5th IEEE Conference on Sensors 2006, Daejeon, South Korea, Pages 811-814, Oct 2006.

APPENDICES

Appendix A: Neural Recording Techniques

Overview

There are different ways of recording neural activity currently in use, Electroencephalography (EEG), Magnetoencephalography (MEG), functional Magnetic Resonance Imaging (fMRI) and Positron Emission Tomography (PET), all of which are non-invasive.

1. EEG:

EEG is the recording of electrical activity along the scalp produced by the firing of neurons in the cortex (brain). These signals are recorded by placing electrodes on the scalp. EEG has been used extensively to test neural activity in different regions of the brain and has been helpful in categorizing seizures, brain death, coma and epilepsy etc. The electrical activity of the brain can be described in spatial scales from the currents within a single neuron to the relatively 'accumulated' potentials that the EEG records from the scalp. Although EEG is the most helpful technique for some of these situations, it has several limitations the most important of them is poor spatial resolution. EEG has very high time resolution, on the order of milliseconds, enabling detection of very small events of interest.

2. MEG:

MEG and EEG are generated by the same neurophysiologic process, where EEG results from accumulated electric fields and MEG results from accumulated magnetic fields. The skull and the scalp do not distort magnetic fields as much as electric fields; thus MEG has better spatial resolution than EEG. MEG has found its uses in the study of epilepsy, however, EEG signal is much richer in information content and is the primary monitoring technique for most neurophysiological processes.

3. **fMRI & PET:**

fMRI and PET both record neural activity directly by monitoring changes in blood flow or metabolic activity in the cortical tissues respectively. Both these techniques have better spatial resolution than EEG in three dimensions, however there are limitations associated with both. Poor time resolution is one of the limitations where successive recordings can contain several minutes long of superimposed information. Another limitation is that the subject cannot move freely (or cannot move at all) for the duration of the recording.

In several situations it is desirable to have high spatial resolution as well as high time resolution for better understanding of intra-cranial connectivity and neural activity than provided by the above mentioned recording techniques. This can only be achieved with highly invasive methods and introduction of strip electrodes or microelectrode arrays.

4. **ECoG:**

Electrocorticography (ECoG) provides better resolution by placing the electrodes on the cortex under the skull. A fine strip with numerous electrodes is placed over the cortex. It records signals which are more pristine and are much more defined than the EEG. These signals are recorded without having to pass through the skull or the scalp, and thus have not been smeared. Though ECoG provides better resolution, it is still not possible to monitor activity of individual cells. A disadvantage of this technique is that a surgery is needed to place the strip of electrodes on the cortex and thus far no system has been developed to wirelessly transmit the recorded data to the external machines. As a result a set of wires extends out of the scalp and feeds signals to the processing machines.

5. **Intracranial EEG:**

Intracranial EEG has shown the capability of recording activity of individual cells, and has recently been made possible by the use of microelectrode arrays. The signals are recorded by implanting electrodes into the cortex. Recording the activity of cortical neurons with microelectrode arrays was shown to be essential to quantify the degree of involvement of each neuron in encoding movement parameters. Use of microelectrodes allows extremely high spatial and temporal resolution and allows recording from single cells at arbitrary sampling rates, thereby enabling decoding of the neural signals and eventually control of artificial limbs [1].

There are several challenges associated with this highly invasive procedure. Firstly, as with ECoG, a surgery is required to implant the electrode array and supporting electronics into the cortex. Secondly, such high resolutions on the order of microseconds and micrometers result in tremendous amount of data being generated. Thus far, clinical trials have used wires that extend from the skull to the external machines to record the signals. However, enabling wireless telemetry of these signals will allow free movements to the subject in addition to long term or permanent implants without any chance of infection in the scalp. Such BMIs show great promise in many biomedical applications.

Appendix B: Quantization

Power and area requirements of the DWT hardware are determined largely by the complexity of the computational circuitry and the required memory. To systematically reduce hardware requirements, we have explored different options to reduce computation and memory requirements at the algorithm level and analyzed their impact on signal integrity to determine an optimal solution. Quantization of input signal and wavelet coefficients into finite width words effects the signal integrity negatively, at the same time, reducing the word lengths has a large effect on the area and power consumed by the processing hardware. This tradeoff is analyzed below.

Signal Quantization

Fixed-point integer approximation limits the range and precision of data values but also greatly reduces the computational demand and memory requirements for processing and storage. The choice of signal width also affects the design of the analog to digital converter, as width of the data path equals the number of bits of the ADC. The input analog signal is first scaled and then digitized to obtain data samples within acceptable signal-to-quantization-noise ratio. Traditionally neural signals are digitized with a word width between 8 to 12 bits. Our analysis shows that a 10-bit quantization provides sufficient signal to noise ratio for most spike sorting algorithms, and increasing the width does not improve sorting performance. As our results in later chapters demonstrate, the data precision is sufficient to maintain signal integrity.

Quantization of the Filter Coefficients

Since the algorithms involve a lot of multiplications and additions, it is important to design the multipliers and adders to be requiring low power and smaller area. Multipliers required to multiply two 10-bit inputs are considerably larger in area than those required to multiply a 10-bit word with a 6-bit word. As a result, wherever possible, using unequal input word lengths has power advantage over using equal word lengths. Another factor that encourages use of unequal word lengths is that the result of a multiply operation consists of a word length equal to the sum of input word lengths. Which means that the dynamic range of the output is different from the dynamic range of the input. In hardware systems where the input and the output of the multiplier are required to have same word length, it is essential to further quantize the output by means of discarding a number of least significant bits, essentially resulting in quantization error. Use of unequal word-length inputs requires less number of bits to be discarded, resulting in smaller quantization error.

We quantified the effect of the round off and quantization errors on the signal fidelity as a function of word-lengths for the wavelet filter coefficients. Our results demonstrate that 6 bits (5 magnitude + 1 sign) for coefficients in conjunction with 10 bits (9 magnitude + 1 sign) for data can adequately preserve signal integrity.

Signal Integrity

We assess the effects of data and filter coefficient approximations on the quality of the signals obtained after reconstruction. We quantified the performance in terms of the complexity of hardware required to implement the Lifting based DWT and illustrate the results in Fig. B.1. The wavelet filter coefficients were quantized to different resolutions ranging from 4 to 12 bits. The data was also quantized in the same range. The root mean squared error (RMS) is illustrated in Fig. B.1 versus data and coefficient word lengths. These results demonstrate that, with sufficient precision, the use of integer computations does not result in significant signal degradation as quantified by the observed output RMS error. It can be noted that there is no significant improvement in SNR when the signal is quantized with greater that 10 bits or when the coefficients are quantized with greater than 6 bits of resolution.

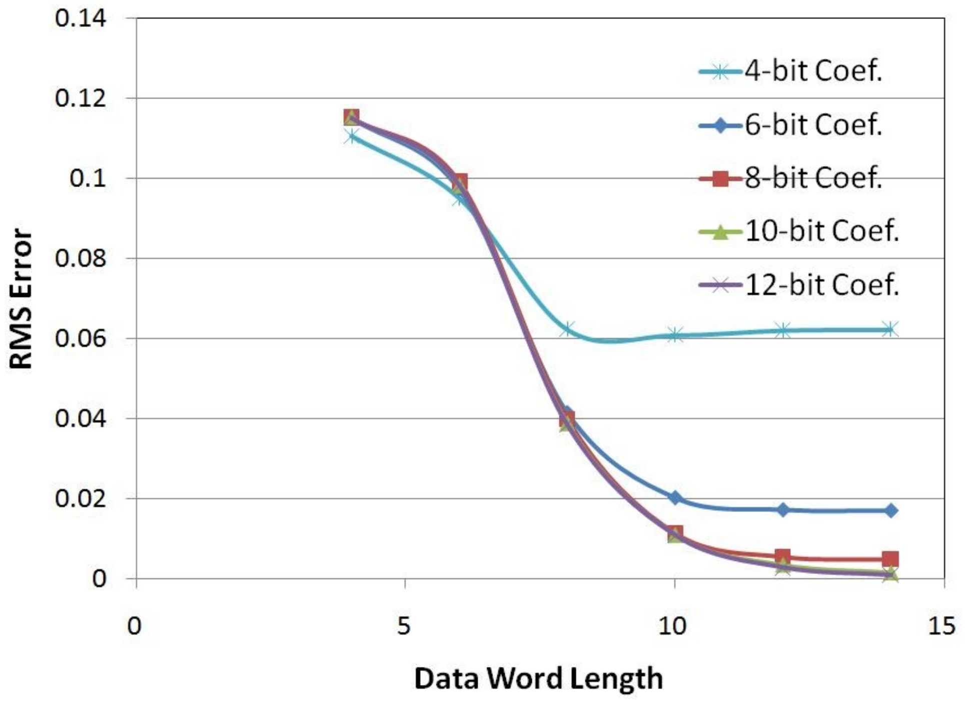


Fig. B.1. Effect of round off and quantization errors on the signal fidelity as a function of quantized data word length and coefficient word length.

Taking these results all together, it is clear that the choice of 6/10-bit coefficient/data quantization offers the best compromise between multiplier complexity and signal fidelity as concluded earlier. We should emphasize that perfect reconstruction of signals off chip may not be always needed. Typically, neural signals contain the activity of multiple neurons that need to be sorted out, and this information remains in the compressed data at the output of the DWT block. It has been shown elsewhere that sorting the multi source neuronal signals can be performed directly on the wavelet transformed data [32, 33], and this topic is outside the scope of this research.

Appendix C: Design Flow

The design of an ASIC to implement an algorithm in hardware is completed in two main phases, the technology independent phase and the technology dependent phase. The technology dependent phase of the design involves fixed-point algorithm design in Matlab followed by behavioral and structural design in a hardware description language. The ASIC design is verified and prototyped in hardware using an FPGA. Although it is a technology dependent technique, verification using an FPGA has various pros as described in the following section. The FPGA verified hardware description is finally translated to the ASIC design for chip fabrication for a particular technology. Fig. C.1 gives the high-level steps for the design of proposed ASIC hardware in silicon.

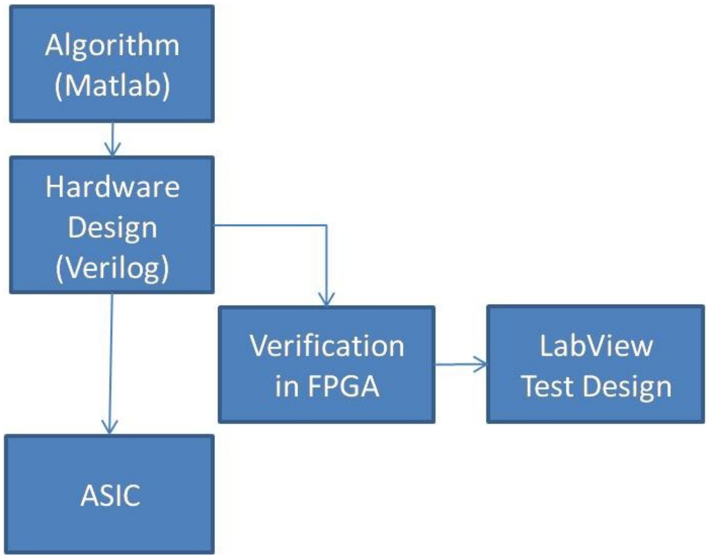


Fig. C.1: High-level ASIC design flow

Technology Independent Phase

The technology independent phase of the design involves the algorithm design and the register transfer level (RTL) description of the hardware in a hardware description language (HDL). This phase has following main components.

1. **Algorithm design**

First step in fabricating an ASIC that implements a signal-processing algorithm is the design of the algorithm in a high-level language. Apart from the desired functionality of the algorithm, design of the algorithm involves calculation of the resources required for processing the data. There are two main resources that need consideration, the number of calculations and the number of memory elements required. As a general rule, higher number of calculations mean higher power dissipation whereas higher number of memory elements mean more silicon area in the final ASIC. However, area and power are not always mutually exclusive as power loss due to leakage has been the dominating reason for power loss lately. Frequency of operation also depends on the degree of parallelism inherent in the algorithm.

Algorithms are generally designed in high-level languages such as C, Python or Matlab. We use Matlab for our designs.

2. **Fixed Point Algorithm**

Hardware designs require a finite word length, which is generally much smaller than the word lengths available in the high-level languages. This introduces two issues, quantization errors and overflows. Both these issues contribute towards degrading the signal to noise ratio at the output of the system. It is generally tried to keep the word length minimum while maintain an acceptable level of signal to noise ratio. This is critical as increasing or decreasing the word length of an 8-bit system by only one bit may result in increase or decrease of the final silicon area by about 10%. Correspondingly, the power consumption of the system also changes. Similarly, it is essential to handle overflows correctly whenever the result of an arithmetic operation exceeds the available word length. These issues make it mandatory to adapt the algorithm for finite precision, fixed-point arithmetic and ascertain the integrity of the algorithm.

Our system is designed with 10 bits of word length. This word length was chosen to keep the quantization error small enough to preserve the integrity of the incoming data and to enable spike sorting in the later stages. Matlab was used to perform the analysis for required bit resolution.

Once the fixed-point simulations have verified the integrity of the algorithm, a “list of specifications” can be created, outlining key requirements for the hardware to be designed. Among other specifications, this list includes a list of inputs and outputs for the system, number of calculations per second of individual components in the system to maintain a desired throughput etc.

3. Register Transfer Level Hardware Description

The list of specifications is then used to design the actual hardware using a hardware description language (HDL). The benefit of using an HDL is that it is easier and faster to create, manage, test and modify larger designs as compared to creating a schematic based design. The register transfer level (RTL) of coding is the level next above gates. Registers are the memory elements in the design. Meaningful designs contain some combinational logic between different registers. RTL thus refers to transfer of signals from one set of registers to the next through the combinational logic. The RTL code contains several high-level language constructs like if-then-else or switch-case, which are later mapped to complex structures like edge-triggered finite state machines or simpler structures like multiplexers and decoders during synthesis.

There are several hardware description languages prevalent in the industry including Verilog, VHDL, Verilog-AMS and System Verilog. We use Verilog to design the hardware as it is one of the most popular and highly developed languages and is supported by majority of vendors to model their technologies and systems. Similarly, there are several software tools available that enable HDL coding e.g. Modelsim, Nc-Verilog and Xilinx ISE.

4. Functional verification & Simulation

The functionality of the designed hardware is verified through simulations. Real data is provided in parallel to the Matlab based algorithm in fixed point and the Verilog based HDL design. The outputs of both the systems are compared for differences. For any given hardware configuration, the outputs of the Verilog design should be an exact match to the outputs of the Matlab code. Simulations at this stage are independent of the fabrication technology and do not cater for the gate delays that are present in the system. As a result, the simulations cannot provide insight into the race conditions or spurious glitches that may arise because of different delays and path lengths. However, functional simulations are still important at this stage. If some problems are diagnosed at later stages, the designed is rolled back to this stage and appropriate changes are made.

5. **Design Constraints File**

The design constraints file is required by the synthesizers to optimize the design and conform to the design requirements such as operating frequencies and various IO delays. The file specifies the names of the clock pins, internally generated clocks, their frequencies and the rise and fall times. The file also specifies other parameters e.g. the amount of uncertainty expected in the input clock or the maximum allowed delay between two registers in the design. Multicycle paths can also be specified here to improve the synthesis results.

Technology Dependent Phase for FPGA Based Prototype

As ASIC designs become more complex, prototyping based on FPGAs has become one of the most reliable techniques for verification. There are several reasons for using FPGAs for prototyping. Bigger designs mean more bugs, similarly higher complexity means the bugs are harder to find. Finding and fixing these bugs in software requires very long verification and simulation times. It is very difficult to develop comprehensive testbenches that can test all possible situations and conditions.

There are several benefits for FPGA-based prototyping including comprehensive verification, software development in parallel and field-testing. First and foremost, it is a way to prove the functionality of the design, by rigorously verifying it physically using real data, before an expensive ASIC fabrication is undertaken. About 70% of ASIC re-spins are a result of functional errors resulting from bugs, which could not be detected during simulations and verification. Since ASIC fabrication is a very expensive and time-consuming processes spanning over several weeks, it is very important to get the design right in the first attempt. Secondly, A hardware prototype permits design and early integration of the complete system, allowing software and board development to progress in parallel to the ASIC manufacturing.

Finally, a functionally verified hardware prototype spawns more confidence in the design, which enables the development of a “right-first-time” ASIC. About 40% of all digital ASIC designs are prototyped using ASICs.

There are various FPGA platforms available for prototyping. We use Altera’s Cyclone III FPGA with sufficient number of logic elements to map the hardware design. Following steps are necessary to ensure proper verification.

1. **Synthesis, Placement and Routing for FPGA**

Verilog files are synthesized to the FPGA specific libraries provided by the FPGA vendor. The synthesis software converts the RTL level Verilog description into gate level circuit. The vendor provided libraries contain information about the delays caused by each cell/gate. Since there are multiple ways of implementing any given function, the synthesizer uses the constraints listed earlier in the Design Constraints File to choose the most suitable topology. This choice results in a tradeoff between area, power and performance (speed). These parameters are compared against the requirements set in the Design Constraints File; in case of any violations, the synthesis is iterated using any of the other synthesis topologies. In case the synthesizer is unsuccessful in creating a gate level circuit, it becomes necessary to modify the high level RTL code at the top of the design flow.

Output of the synthesis step is another Verilog file that contains gate-level description of the circuit. The synthesized output is then provided to the 'place n route' software which maps these gates to the logic elements present in the FPGA. This mapping also takes into account the delays incurred by the long wires between different cells. Apart from mapping the gates to the logic elements, this step generates a file in the Standard Delay Format (SDF) that contains information about the delays resulting from individual wires.

FPGA vendors provide the required software for synthesis and place n route, however, general-purpose synthesizers like Mentor Graphics' Leonardo Spectrum or Synopsys' Design Vision can also be used to synthesize the code for the particular FPGA. We use Altera's Quartus II to perform these steps.

2. Post-Synthesis Timing Simulation

Once the synthesis is complete, the design is simulated and verified again in Modelsim. This simulation is based on gate-level synthesized code and also includes the information present in the SDF file. Consequently, the simulation includes delays experienced by the signals through all the gates and the wires connecting them. This gate and wire delay based simulation is necessary to establish the integrity of the design and gives insight into any race conditions present in the system as well as the effects of any undesirable change in signal value at the output of a gate. Again, the output of the gate-level simulation is compared to the output from fixed-point Matlab based simulation for a perfect match.

Although synthesis results for the FPGA are expected to be different from the synthesis results for the ASIC due to difference in technology and the FPGA architecture; a success at this stage verifies that the hardware has been designed suitably enough such that it can be converted to physical hardware while keeping within the design constraints.

3. Hardware Verification

The simulated and verified code is then ‘burned’ on to the actual FPGA for hardware verification. Hardware verification of the design requires additional hardware and software resources to provide the real data at the inputs of the FPGA and, at the same time, record the outputs from the FPGA. Output of the FPGA needs to be the same as the simulation results, which should be the same as the Matlab based algorithmic simulations.

We use National Instruments’ data acquisition cards in conjunction with LabView and Python programming language to generate the signals required at the input of the FPGA. This setup provides high degree of flexibility in the test design, however, these software-based interfaces are not fast enough to test the hardware at high frequencies. To test the design at appropriate frequencies, additional FPGAs need to be used to provide and record the signal from the FPGA. Though use of FPGA to monitor the signals lacks the flexibility of software based monitoring, nevertheless, it verifies the functionality of the design at high operating frequencies.

As mentioned earlier, due to the possible technology difference from the ASIC, FPGA based prototyping does not provide an accurate estimate of the power consumption or the IO delays.

Technology Dependent Phase for ASIC

Once the design has been verified on an FPGA based prototype, it is ready to be ported to an ASIC. Choice of the fabrication technology for an ASIC depends on many factors. Newer technologies mean smaller feature size, which generally translates to less area consumption, smaller power dissipation and faster designs. However, fabrication costs as well as design complexity increase tremendously with reduced feature size.

We used AMI 0.5 micron CMOS fabrication process to verify the functionality of DWT chip and develop the process design flow. The complete system is to be fabricated on an IBM 0.13 micron CMOS process. The use of this process is mandated by the low area and low power requirements of this research project.

Design of digital ASICs requires various libraries and models provided by the vendor for proper verification and reliable fabrication. These files include various HDL models for simulations, standard cells library files for the synthesizer, and the cell layout files for final layouts. Technology specific memory generators are also required to compile large and reliable memory blocks. The following technology specific steps are being followed to ensure ‘right-first-time’ ASIC design for the algorithm.

1. **Synthesis for ASIC**

The Verilog files containing RTL level design description is imported into the synthesizer along with the design constraints file. It is ensured that the design constraints file does not contain any FPGA specific constraints. Synopsys Design Vision is used to synthesize the design and map it to the IBM 0.13 micron CMOS specific libraries. The synthesizer outputs a Verilog file containing gate-level description of the synthesized block.

The synthesizers are not capable of generating memory blocks as six-transistor SRAM. The synthesizers can only generate flip flop based registers that are about three times larger than the SRAM. As a result, special memory generators are needed to compile the memory blocks while synthesizing other portions of the design using the synthesizer.

Each block can be synthesized independently or the complete system can be synthesized as a large block. We chose to synthesize the blocks independently since it allows the system to be divided into various modules. As different blocks operate at different clock frequencies, a modular approach helps to reduce power consumption at the expense of additional silicon area.

2. **Memory Generation and Custom Blocks**

Required memory blocks are generated using technology specific memory generators provided by the vendor. Memory generators are necessary since change in the depth of memory requires Different memory blocks are generated with different word size and memory depths depending upon their access patterns to reduce the power consumption of memory read or write operations. Once generated the memory is simulated for functional and timing verification. The memory generators also generate the Verilog model files for the memory. We use Cadence systems' Virtuoso to compile the memories.

3. Timing verification for ASIC

The synthesized blocks and the memory models are combined into a system and verified through simulations in Modelsim. These simulations incorporate the IO delays and the delays associated with individual gates but do not incorporate the wire delays introduced due to parasitic capacitances.

4. Abstract Generation

Abstract generation is necessary if the design contains any custom blocks that are not included in the standard cells library. Abstract generation is a process that extracts necessary information from a custom layout block to enable the 'automatic place n route' software to include the custom blocks in the design. Apart from other useful items, this information includes the precise location of input and output pins, as well as the areas occupied by different metals. We use Virtuoso for abstract generation.

5. Automatic Placement and Routing

Automatic Place n Route is a layout generation process that includes automatic placement of different cells representing logic gates on a layout and then routes the inputs and outputs of those gates according to the synthesized netlist generated by the synthesizer. The resulting layout contains all the gates and connections specified in the synthesized Verilog gate level netlist.

There are various steps involved in the generation of layout from automatic place n route. These include floor-planning and generation of clock trees and power rings. Each module is placed n routed individually. Once all the digital blocks have been routed then global routing is performed. All the digital blocks, memory blocks and custom blocks are connected to each other in the global routing stage. The output of this stage is a set of SDF files that contain information about delays introduced by various connecting wires at the block level and at the global level.

6. Layout Integrity Check

Automatic place n route follows several technology specific design rules while routing signals through various cells. These rules are defined in the technology files. However, since various components of the design flow come from different vendors it is still possible that there may be some rules which are overlooked by the router. As a result a Design Rules Check is performed on the layout files to verify the routing. There are some rules that cannot be checked by the router automatically and thus need to be checked manually once the process is complete.

7. Timing verification for ASIC

The gate level netlists and the SDF files generated by the place n route stage are then simulated in modelsim. These simulations include all the system components connected to each other (except the pad frame for input / output pins) and also uses gate delay and wire delay information to generate accurate results. it is highly desirable to simulate at this stage as simulation of moderately sized layouts may take weeks to generate output vectors of interest.

8. Post-Layout Simulations for Timing Verification

In the final step, the layout is completed by including the pad frame for the input, output pins and the power, ground pins. Simulation of the complete layout is time consuming but may be necessary to verify some element of design. Since it is difficult to verify all the test vectors, a selection of test scenarios can be simulated to verify the functionality of the chip. Ocean is a scripting language that can be used in Cadence Systems' Virtuoso to analyze and extract useful information from the multi-gigabytes of data generated during the simulations.

This step concludes the ASIC design process after which the final layout is taped out for fabrication.

Appendix D: Design for Testability

Key signals in the design are included into a scan chain designed to increase the testability of the system in future generations. Fig. D.1 shows a design to include observability and controllability for a particular signal. A standard 4-pin interface is used to operate the scan chain. One pin T/N selects Test or Normal mode of operation. Serial scan in and serial scan out are used to serially input data to the scan chain as well as read register values at the output. T-Clk and U-Clk are the Test and Update clocks. They can be controlled through designated pins or can be controlled through one T-clk and accompanying control logic.

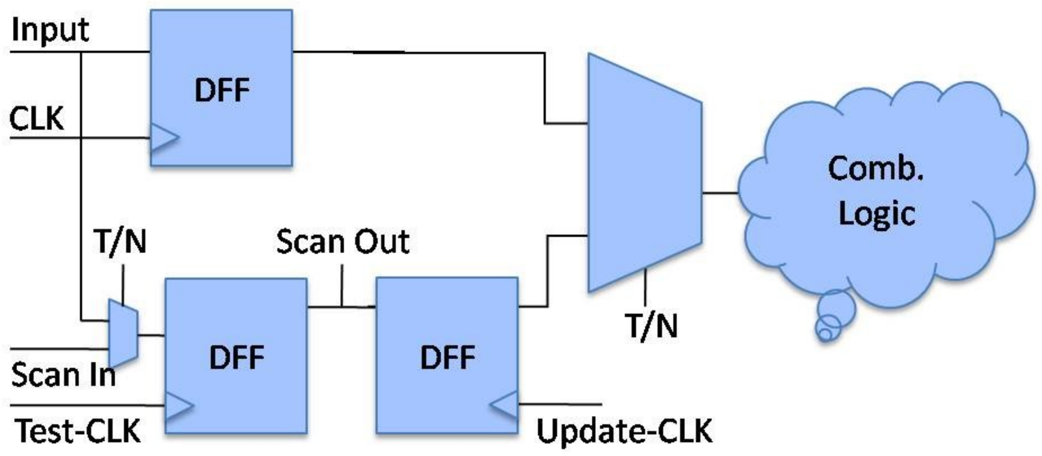


Fig. D.1: Scan chain insertion for testability including both observability and controllability

BIBLIOGRAPHY

[1] M. Nicolelis, "Actions from thoughts," *Nature*, vol. 409, no. 6818, pp. 403-408, 2001.

[2] M. Rizk, C. A. Bossetti, T. A. Jochum, S. H. Callender, M. A. L. Nicolelis, D. A. Turner, P. D. Wolf, "A fully implantable 96-channel neural data acquisition system," *Journal of Neural Engineering*, vol. 6, no. 2, pp. 1-14, 2009.

[3] M. Rizk, I. Obeid, S. Callender, P. Wolf, "A single-chip signal processing and telemetry engine for an implantable 96-channel neural data acquisition system," *Journal of Neural Engineering*, vol. 4, no. 3, pp. 309-321, 2007.

[4] R. Harrison, P. Watkins, R. Kier, R. Lovejoy, D. Black, B. Greger, F. Solzbacher, "A low-power integrated circuit for a wireless 100-electrode neural recording system," *IEEE Journal of Solid State Circuits*, vol. 42, no. 1, p. 123-133, 2007.

[5] R. Harrison, R. Kier, C. Chestek, V. Gilja, P. Nuyujukian, S. Ryu, B. Greger, F. Solzbacher, K. Shenoy, "Wireless Neural Recording With Single Low-Power Integrated Circuit," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, no. 4, pp. 322-329, 2009.

[6] C. Chestek, V. Gilja, P. Nuyujukian, R. Kier, F. Solzbacher, S. Ryu, R. Harrison, K. Shenoy, "HermesC: Low-Power Wireless Neural Recording System for Freely Moving Primates," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, no. 4, pp. 330-338, 2009.

[7] B. Gosselin, M. Sawan, "An Ultra Low-Power CMOS Automatic Action Potential Detector," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, no. 4, pp. 346-353, 2009.

[8] B. Gosselin, A. Ayoub, J. Roy, M. Sawan, F. Lepore, A. Chaudhuri, D. Guitton, "A Mixed-Signal Multichip Neural Recording Interface With Bandwidth Reduction," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 3, no. 3, pp. 129-141, 2009.

[9] A. Sodagar, K. Wise, K. Najafi, "A Wireless Implantable Microsystem for Multichannel Neural Recording," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 57, no. 10, pp. 2565-2573, 2009.

[10] K. Wise, G. Perlin, Y. Yao, K. Najafi, A. Sodagar, "An Implantable 64-Channel Wireless Microsystem for Single-Unit Neural Recording," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 9, pp. 2591-2604, 2009.

[11] K. Wise, A. Sodagar, Y. Yao, M. Gulari, G. Perlin, K. Najafi, "Microelectrodes, Microelectronics, and Implantable Neural Microsystems," *Proceedings of the IEEE*, vol. 96, no. 7, pp. 1184-1202, 2008.

[12] A. Sodagar, K. Wise, K. Najafi, "A fully integrated mixed-signal neural processor for implantable multichannel cortical recording," *Biomedical Engineering*, *IEEE Transactions on,* vol. 54, no. 6, p. 1075-1088, 2007.

[13] Y. Song, D. Borton, S. Park, W. Patterson, C. Bull, F. Laiwalla, J. Mislow, J. Simeral, J. Donoghue, A. Nurmikko, "Active Microelectronic Neurosensor Arrays for Implantable Brain Communication Interfaces," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, no. 4, pp. 339-345, 2009.

[14] M. Yuce, W. Liu, M. S. Chae, J. S. Kim, " A Wideband Telemetry Unit for Multi-Channel Neural Recording Systems," *Ultra-wideband, IEEE international conference on*, pp. 612-617, 2007.

[15] M. S. Chae, Z. Yang, M. Yuce, L. Hoang, W. Liu, "A 128-Channel 6 mW Wireless Neural Recording IC With Spike Feature Extraction and UWB Transmitter," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, no. 4, pp. 312-321, 2009.

[16] S. Mallat, *A wavelet tour of signal processing*, Academic Press, 1999.

[17] K. Oweiss, "A systems approach for data compression and latency reduction in cortically controlled brain machine interfaces," *Biomedical Engineering, IEEE Transactions on*, vol. 53, no. 7, pp. 1364-1377, 2006.

[18] D. Donoho, "Denoising by soft-thresholding," *Information Theory*, *IEEE Transactions on,* vol. 41, no. 3, pp. 613-627, 1995.

[19] K. Parhi, T. Nishitani, "Folded VLSI architectures for discrete wavelet transforms," *1993 IEEE International Symposium on Circuits and Systems*, pp. 1734-1737, 1993.

[20] C. Chakrabarti, M. Vishwanath, R. Owens, "Architectures for wavelet transforms: A survey," *The Journal of VLSI Signal Processing*, vol. 14, no. 2, pp. 171-192, 1996.

[21] H. Olkkonen, J. Olkkonen, P. Pesola, "Efficient lifting wavelet transform for microprocessor and VLSI applications," *IEEE Signal Processing Letters*, vol. 12, no. 2, pp. 120-122, 2005.

[22] H. Liao, M. Mandal, B. Cockburn, "Efficient architectures for 1-D and 2-D lifting-based wavelet transforms," *Signal Processing, IEEE Transactions on*, vol. 52, no. 5, pp. 1315-1326, 2004.

[23] C. Huang, P. Tseng, L. Chen, "VLSI architecture for forward discrete wavelet transform based on B-spline factorization," *The Journal of VLSI Signal Processing*, vol. 40, no. 3, pp. 343-353, 2005.

[24] A. Kamboh, A. Mason, K. Oweiss, "Analysis of Lifting and B-Spline DWT Implementations for Implantable Neuroprosthetics," *Journal of Signal Processing Systems*, vol. 52, no. 3, pp. 249-261, 2008.

[25] I. Daubechies, W. Sweldens, "Factoring wavelet transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247-269, 1998.

[26] K. Kotteri, S. Barua, A. Bell, J. Carletta, "A comparison of hardware implementations of the biorthogonal 9/7 DWT: convolution versus lifting," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, no. 5, pp. 256-260, 2005.

[27] C. Huang, P. Tseng, L. Chen, "Analysis and VLSI architecture for 1-D and 2-D discrete wavelet transform," *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1575-1586, 2005.

[28] C. Huang, P. Tseng, L. Chen, "Flipping structure: An efficient VLSI architecture for lifting-based discrete wavelet transform," *IEEE Transactions on Signal Processing*, vol. 52, no. 4, pp. 1080-1089, 2004.

[29] M. Unser, T. Blu, "Wavelet theory demystified," *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 470-483, 2003.

[30] K. Andra, C. Chakrabarti, T. Acharya, others, "A VLSI architecture for lifting-based forward and inverse wavelet transform," *IEEE Transactions on Signal Processing*, vol. 50, no. 4, pp. 966-977, 2002.

[31] K. Oweiss, "Compressed and Distributed Sensing of Multivariate Neural Point Processes," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 577-580, 2007.

[32] K. Oweiss. Multiresolution analysis of multichannel neural recordings in the context of signal detection, estimation, classification and noise suppression. University of Michigan, Ann Arbor. 2002.

[33] K. Oweiss, "Compressed sensing of large-scale ensemble neural activity with resource-constrained cortical implants," *Society of Neuroscience Abstracts*, 2006.

[34] K. Oweiss, D. Anderson, M. Papaefthymiou, "Optimizing signal coding in neural interface system-on-a-chip modules," *IEEE conference on EMBS*, pp. 2016-2019, 2003.

[35] K. Oweiss, A. Mason, Y. Suhail, A. Kamboh, K. Thomson, "A scalable wavelet transform VLSI architecture for real-time signal processing in high-density intra-cortical implants," *IEEE Transactions on Circuits and Systems Part I*, vol. 54, no. 6, pp. 1266-1278, 2007.

[36] A. Mason, J. Li, K. Thomson, Y. Suhail, K. Oweiss, "Design optimization of integer lifting DWT circuitry for implantable neuroprosthetics," *IEEE/EMBS Special Topics Conference on Microtechnologies in Medicine and Biology*, pp. 136-139, 2005.

[37] K. Thomson, Y. Suhail, K. Oweiss, "A Scalable Architecture for Streaming Neural Information from Implantable Multichannel Neuroprosthetic Devices," *IEEE International Symposium on Circuits and Systems*, pp. 1342-1345, 2005.

[38] Y. Suhail, K. Oweiss, "A reduced complexity integer lifting wavelet based module for real-time processing in implantable neural interface devices," *IEEE International Conference on Engineering in Medicine and Biology*, pp. 4552-4555, 2004.

[39] B. Wu, C. Lin, "A rescheduling and fast pipeline VLSI architecture for lifting-based discrete wavelet transform," *IEEE International Symposium on Circuits and Systems*, pp.732-735, 2003.

[40] A. Shams, T. Darwish, M. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 10, no. 1, pp. 20-29, 2002.

[41] Z. Huang. High-level optimization techniques for low-power multiplier design. University of California, Los Angeles. 2003.

[42] B. Yang, L. Kim, "A low-power sram using hierarchical bit line and local sense amplifiers," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 6, pp. 1366-1376, 2005.

[43] A. Kamboh, M. Raetz, K. Oweiss, A. Mason, "Area-power efficient VLSI implementation of multichannel DWT for data compression in implantable neuroprosthetics," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 1, no. 2, pp. 128-135, 2007.

[44] A. Kamboh, M. Raetz, A. Mason, K. Oweiss, " Area-Power Efficient Lifting-Based DWT Hardware for Implantable Neuroprosthetics," *Circuits and systems. IEEE International Symposium on*, pp. 2371-2374, 2007.

[45] K. Oweiss, Y. Suhail, K. Thomson, J. Li, A. Mason, " Augmenting real-time DSP in implantable high-density neuroprosthetic devices," *Microtechnology in Medicine and Biology, IEEE/EMBS Special Topic Conference on*, pp. 108-111, 2005.

[46] C. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379-423, 1948.

[47] J. Grad, J. Stine, "A standard cell library for student projects," *International conference on microelectronic systems education*, pp. 98-99, 2003.

[48] R. Sarpeshkar, W. Wattanapanitch, B. Rapoport, S. Arfin, M. Baker, S. Mandal, M. Fee, S. Musallam, R. Andersen, "Low-power circuits for brain-machine interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 2, no. 3, pp. 173-183, 2008.

[49] A. Tanenbaum, *Computer Networks*, Prentice Hall, 1981.

[50] A. Kamboh, K. Oweiss, A. Mason, " Resource constrained VLSI architecture for implantable neural data compression systems," *Circuits and systems, IEEE International Symposium on*, pp. 1481-1484, 2009.

[51] M. Ghovanloo, S. Atluri, "A wide-band power-efficient inductive wireless link for implantable microelectronic devices using multiple carriers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 10, pp. 2211-2221, 2007.

[52] M. Aghagolzadeh, K. Oweiss, "Compressed and Distributed Sensing of Neuronal Activity for Real Time Spike Train Decoding," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, no. 2, pp. 116-127, 2009.

[53] M. Ghovanloo, and Suresh Atluri, “An integrated full-wave CMOS rectifier with built-in telemetry for RFID and implantable biomedical applications,” *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 10, pp 3328-3334, Nov 2008.

[54] S. A. Khayam, “Analysis and Modeling of Errors and Losses Over 802.11b Networks,” M.S. Thesis, Dept. Elect. Eng., Michigan State Univ., East Lansing, MI, 2003.

[55] S. Karande, U. Parrikar, K. Misra, and H. Radha, “On Modeling of 802.11b Residual Errors,” *Conference on Information Science and Systems*, pp. 283-288, March 2006.

[56] M. U. Ilyas, and H. Radha, “Measurement Based Analysis and Modeling of the Error Process in IEEE 802.15.4 LR-WPANs,” *IEEE INFOCOM Conf on Computer Communication*, pp. 1274-1282, Apr 2008.

[57] M. U. Ilyas, “Analytical and Quantitative Characterization of Wireless Sensor Networks,” Ph.D. dissertation, Dept. Elect. Eng., Michigan State Univ., East Lansing, MI, 2009.

[58] A. M. Kamboh, Y. Yang, K. G. Oweiss and A. J. Mason, “Design of a Configurable Neural Data Compression System for Intra-Cortical Implants,” *IEEE International Symposium on Circuits and Systems*, pp. 3473-3476, Paris, France, June 2010.

[59] M. Ghovanloo, K. D. Wise, and K. Najafi, “Towards a Button-Sized 1024-Site Wireless Cortical Microstimulating Array,” *IEEE/EMBS Conference on Neural Engineering*, pp. 138-141, March 2003.

[60] Z. Zumsteg, R. Ahmed, G. Santhanam, K. Shenoy, and T. Meng, “Power Feasibility of Implantable Digital Spike-Sorting Circuits for Neural Prosthetic Systems,” *IEEE Proceedings on Engineering in Medicine and Biology Society*, pp. 4237-4240, Sep. 2004.

[61] S. Gibson, J. Judy, and D. Markovic, “Comparison of Spike Sorting Algorithms for Future Hardware Implementation,” *IEEE Proceedings on Engineering in Medicine and Biology Society*, pp. 5015-5020, Aug. 2008.

[62] A. Zviagintsev, Y. Perelman, and R. Ginosar, “Low Power Architectures for Spike Sorting,” *IEEE Conference on Neural Engineering*, pp. 16-19, Mar. 2005.

[63] C. Rogers, J. Harris, J. Principe, and J. Sanchez, “A Pulse Based Feature Extractor for Spike Sorting Neural Signals,” *IEEE Conference on Neural Engineering*, pp. 490-493, May. 2007.