A High Speed Wearable System for Body Coupled Communication

by

Devon Rosner

S.B., E.E., M.I.T. (2013)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014

Author ..... Signature redacted .....
Department of Electrical Engineering and Computer Science
August 18, 2014

Certified by.. Signature redacted .....
Professor Charles G. Sodini
Clarence J. LeBel Professor
Thesis Supervisor

Accepted by ..... Signature redacted .....
Albert R. Meyer
Chairman, Master of Engineering Thesis Committee

A High Speed Wearable System for Body Coupled Communication

by

Devon Rosner

Submitted to the Department of Electrical Engineering and Computer Science
on August 18, 2014, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

There are currently no ideal methods by which doctors can read bodily signals detected by implanted devices. Methods are either too high power for long-term implants, such as radio transmission, or pose health threats to the patient, such as connection ports piercing the skin. However, a novel method of transmitting and receiving electronic sensor data is emerging known as body coupled communication (BCC). This method of communication utilizes the inside of the body's low impedance at frequencies on the order of 100 MHz to send signals over that channel and receive the signals at another location on the body. It is also a lower power and more secure wireless option than radio transmission. This thesis presents a 3 Mbps wearable receiver and transmitter system for BCC that was developed from commercially available electrical components and a custom PCB. Both receiver and transmitter are on the same PCB. They share a digital FPGA system, but have separate analog signal conditioning sections on the board.

Thesis Supervisor: Professor Charles G. Sodini
Title: Clarence J. LeBel Professor

Acknowledgments

Over the course of my five years at MIT, there have been far too many to people that deserve my gratitude for all of the help, tutelage, mentorship, opportunities, friendship, and brotherhood. There is simply not enough space here to let you all know how influential you have been, but you deserve to know that I appreciate all of you, and thank you.

Charlie Sodini, my MIT thesis advisor, was a consistent source of positive reinforcement during my time researching with him. He always knew the right questions to ask, provided direction to my work, and made his expectations of me as clear as possible. In times of unconventional circumstances, Charlie remained supportive and gave me the confidence that this was all still achievable. For all of this, I am immensely grateful.

This thesis would not have been what it is without the mentorship and guidance of Grant Anderson. Thank you, Grant, for mentoring me from my time as a UROP to the end of my time at MIT.

For my final three years at MIT, I have had the pleasure of being the student, lab assistant, co-lab assistant, and teaching assistant of Gim Hom. Thank you for giving me all of the opportunities to grow as an engineer through teaching others. The debugging skills I learned from you have played an important role in where I am today, and hardly a day goes by now where I do not find myself using them.

A very special thanks goes out to Heidi Jackson and everyone at Cirtech Technology for generously fixing a hard-to-reach mistake made in the first board revision. Your timeliness and skillful circuit repairs made this thesis possible.

Finally, I would like to give my most sincere thanks to my parents, Brett and Maryann Rosner. Their love and unrelenting support has played an enormous role in my education and has always given me the strength to push myself past my preconceived limits. There is no doubt that this would not be here without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To test the high speed feasibility of a new communication method that transmits signals across the human body, a wearable receiver and transmitter has been designed and built. This method of signal transmission is known as body coupled communication (BCC). Presented in this chapter is the motivation for such a system, the findings and differing results of pre-existing research, and the varying designs used by these research groups [1–9].

1.1 Motivation

As medical research progresses, new information arises that attempts to relate patient health to physiological signals generated by the human body that are transduced to electrical signals. Data extracted from the heart (ECG), brain (EEG), muscles (EMG), etc. is being found to potentially reveal much more information than what was previously thought. However, collecting much of this new data requires sensors and instruments that are faster, more sensitive, more precise, and can store larger sets of data. Some of these signals may even be too small in magnitude to read from outside of the body and may require an implanted device for adequate data collection. This signal detection now requires that the sensors and instruments be low power and can communicate with the outside world.

In the medical field, there are currently two popular methods by which doctors

can read electronic sensor data from body connected devices. The most common is with wires running from sensor to transceiver. This method of communication is low power and reliable, except it can cause unwanted physical discomfort to the patient, as well as interference on the wires from crosstalk. Another drawback to physical wires is they cannot be used in implanted devices, as connection ports piercing the skin have negative sanitary consequences [10].

The second method of transmitting and receiving signals from the body is by radio (i.e. Wi-fi, Bluetooth, radio, etc.). This method alleviates patients and medical staff of the interference from the wires and eliminates the need for a physical port to an implanted device. The power consumption of radio transmission is of greatest concern, as it requires much more energy to send a signal over radio than it does with a wire. This issue concerning power is especially significant since implanted devices operate with a battery, so more energy consumed to transmit data leads to shorter battery life, and results in more frequent surgery just to replace a battery.

For these reasons, systems that transmit and receive electronic sensor data across the body are emerging known as body area networks (BANs). A common implementation of a BAN consists of low power sensors located on or in the body that transmit their collected data to a higher power receiver that can store large amounts of the collected data. An example of such a system can be found in Figure 1-1 [1].

Since signals in BANs are typically confined to the human body, a novel method of intra-body data transfer has been developed known as body coupled communication (BCC) [2]. The inspiration behind this method of communication is that the body has a low impedance channel around 100 MHz. This transfer function is used to model a communication channel across the body, such that a signal can be sent from one point on the body to another. To communicate from the outside of the body to the inside and vice versa involves capacitively coupling to the body. There is conductive tissue beneath the skin connecting the entire body. If a conductive material is placed on the skin outside of the body, a capacitor is formed between the conductive tissue and the outside conductive material. The skin can be thought of as a dielectric. With a capacitive link between the outside and inside of the body, AC signals can be sent

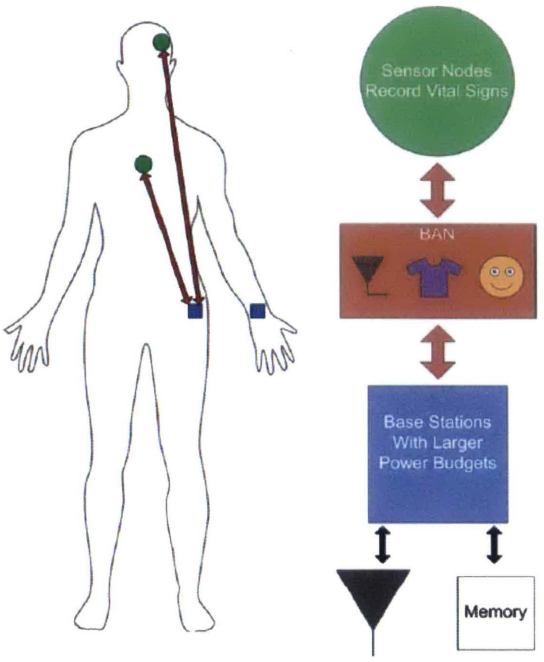

Figure 1-1: An example of a body area network [1]

between the two realms. This is a lower power communication option than radio, and it is wireless.

1.2 Previous Work

Multiple groups have experimented with BCC systems [1-9]. However, this method of communication is still new enough such that these groups all use a variety of body channel models and data transfer schemes.

1.2.1 The Channel

Papers have been published that model the human body's channel magnitude and phase over a range of frequencies [1, 3, 4]. These groups all agree that the appropriate carrier frequency range lies between 1 and 150 MHz, as the channel attenuation is at its minimum in this range. Each system, however, had a different experimental test setup that resulted in discrepancies in the specific optimal range. They also disagree

on the total channel attenuation in their optimal ranges.

The variance in data is perplexing until one observes the experimental test setups of each group. Cho, Yoo, Song, and Lee claim that the maximum channel gain occurs in the 10 to 60 MHz range [4], but Anderson's method claims a range of 70 to 150 MHz [1]. Figure 1-2 shows the setup from [4]. For this thesis, the latter range is appropriate because it is taken from the system with the most isolation between transmitter and receiver.

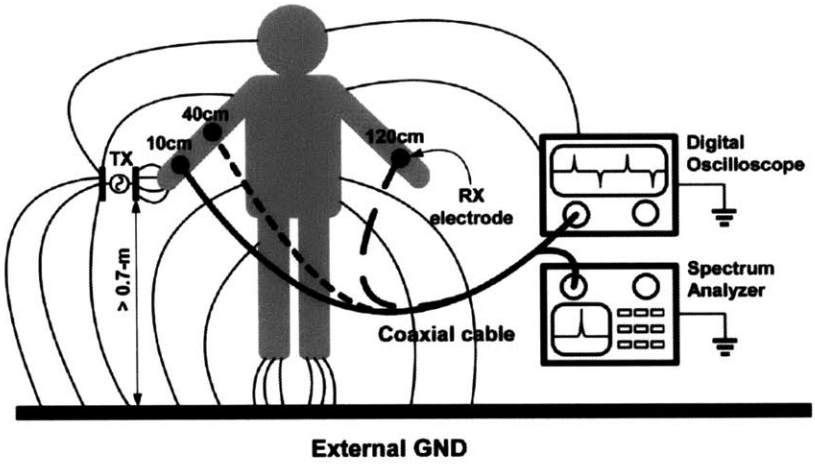

Figure 1-2: A BCC experiment setup with grounded equipment [4]

Looking at the setup in Figure 1-2, it is seen that the setup ties the grounds from transmitter and receiver together via the testing equipment. Figure 1-3 provides a simplified model of a BCC system. For the frequency range of 10MHz-150MHz, the body's conductive channel can be modeled as a spreading resistance. This spreading resistance model is shown in Figure 1-3 as the resistor ladder between the transmitting amplifier and receiving amplifier.

The highlighted ground node connected to the system via dashed lines shows the result of using shared ground testing equipment on both receiver and transmitter. Figure 1-4 shows a simulation of the attenuation across different lengths of a ground isolated spreading resistance, compared with different lengths of a ground connected spreading resistance. This simulation shows that ground connected BCC systems have a significant reduction in channel attenuation. Figure 1-5 shows similar results

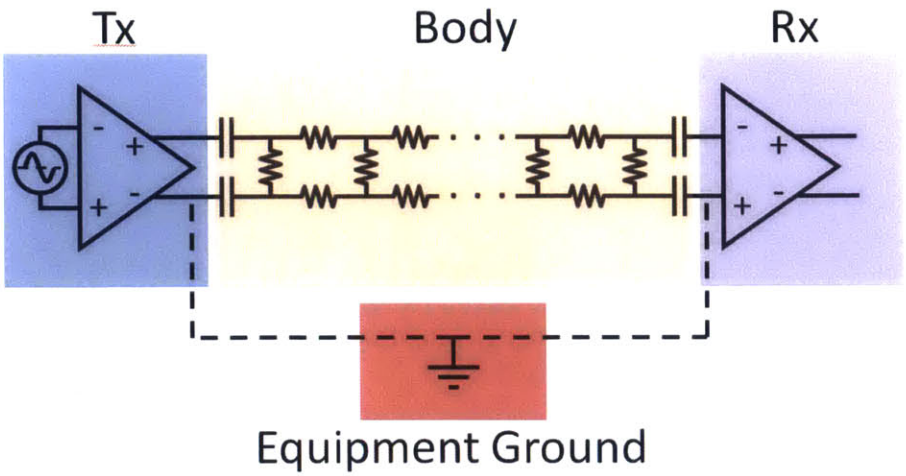

Figure 1-3: A high level BCC system

with two physically tested systems versus frequency [1].

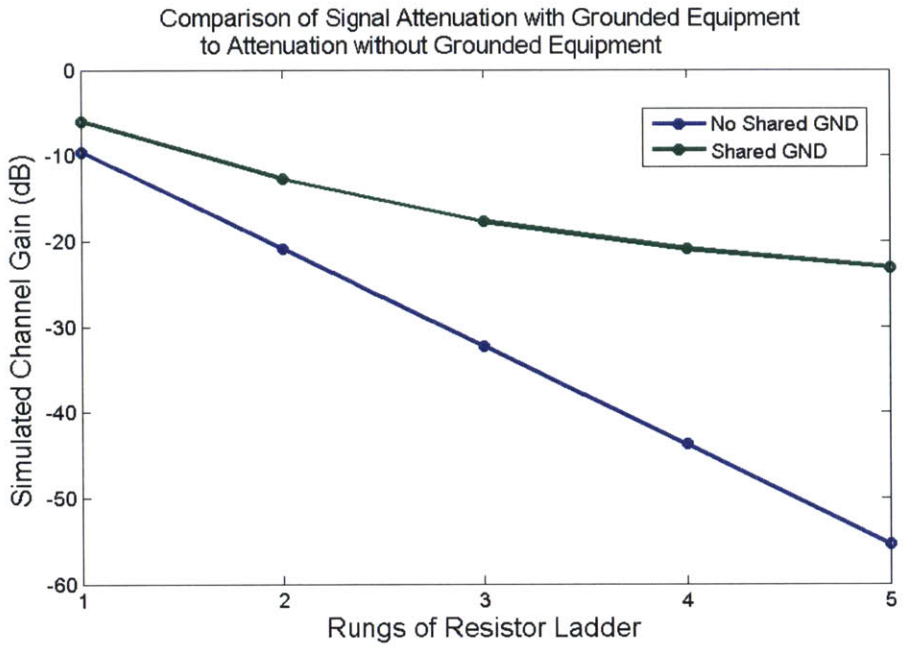

Figure 1-4: Simulation showing increased attenuation of ground isolated BCC systems

Though there is much less attenuation when using external measurement devices connected to both receiver and transmitter, this will not work in a medical BCC implementation where the transmitter is implanted beneath the patient's skin. For

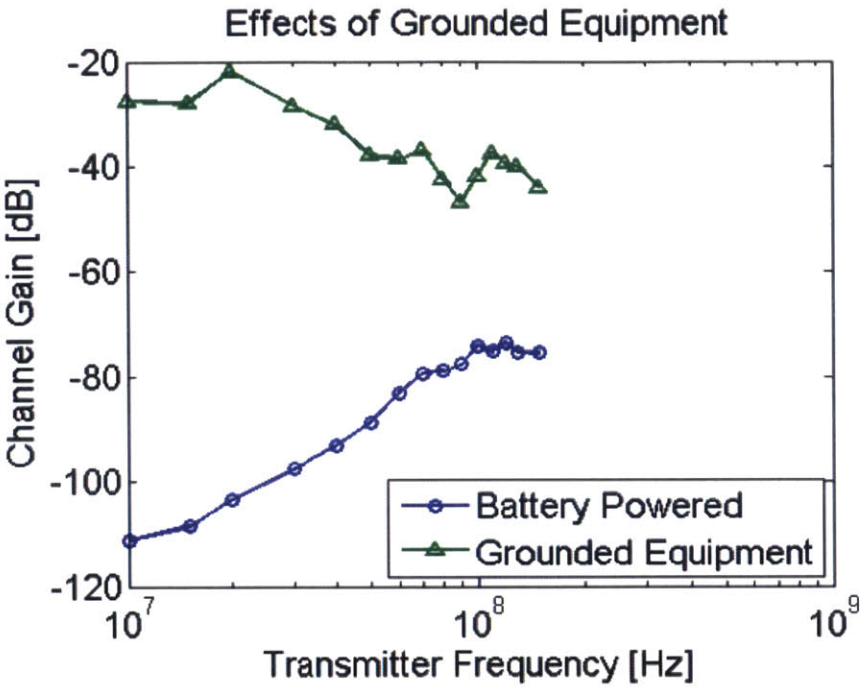

Figure 1-5: Experimental results showing increased attenuation of ground isolated BCC systems [1]

this reason, this design will completely isolate the two grounds, and will transmit within the 60 to 120 MHz range, unless another range with less attenuation is found.

The groups that attempted to transmit data achieved data rates ranging from 172 kbps to 8 Mbps [1,5]. Both of these groups used carrier frequencies in the 10-150 MHz range. What set these groups apart is the 172 kbps system was made of commercially available parts, and the 8 Mbps system was a custom ASIC.

1.2.2 Communication Schemes

The basic data transfer and detection implementation that can be found in almost every BCC system can be seen in Figure 1-6. A simple breakdown of this scheme starts with encoded data being transmitted across the body using carrier frequencies in the range of 10-150 MHz. Unavoidable signal attenuation across the body requires that the receiver have an analog amplifier front end. This is the point in which systems

start to differ. Designs such as Harikumar's [6] and the design in this thesis convert this data to a discrete 1-bit signal. Another method is to use a well-established analog data transfer technique such as FSK [1], however, these systems have maximum data rates below 1 Mbps. A method that seems to be gaining popularity is to mix the incoming signal down to a baseband and process it from there [7]. This method seems promising because after the data has been mixed down, there is a greater abundance of analog systems that can handle those frequencies compared to the 100 MHz range. After mixing the signal down to baseband, the signal is digitally processed and stored to memory. The digital processing depends on the data encoding method, which varies across most designs. Song, for example, employs a synchronous Non-Return-to-Zero (NZR) encoding [8]. The design in this thesis employs an asynchronous multi-bit per sample encoding that will be explained in Chapter 3.2.

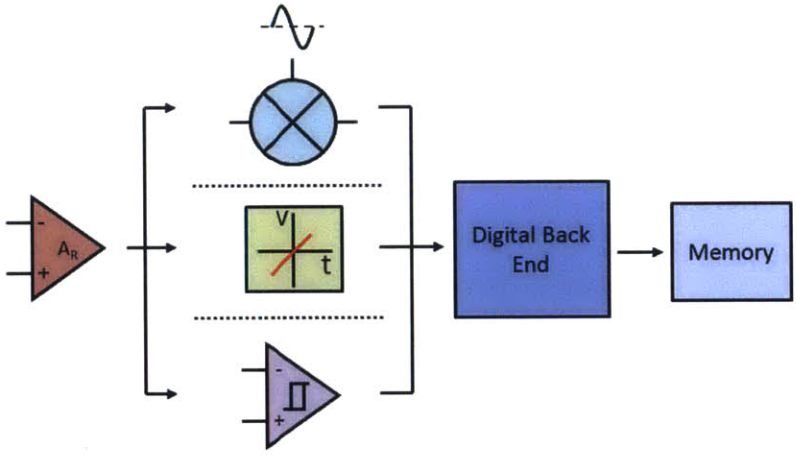

Figure 1-6: A collection of basic receiver architectures. All systems start with an analog amplifier front end and end with a digital processing back end

1.3 Objectives

Since the receiver and transmitter are made of only commercially available parts and a custom PCB, the goal for the system is to have a data capture rate of 5 Mbps. A BCC system like this would likely be used for a single day testing session lasting less than 8 hours. This means that the board must run for at least 8 hours on a single battery

charge, and that it can store data being captured at 5 Mbps for at least 8 hours. Eight hours of 5 Mbps data storage amounts to 18 GB of data. Mechanically, the system needs to be wearable and secure. An unsecured attachment to the body could cause stress on the data collection and transfer electrodes, potentially leaving artifacts in the data or creating a less conductive attachment to the body. To measure the data communication abilities of this system, it will be tested on a human volunteer. Data will be sent as "symbols" that contain multiple bits, so the metric for how the system performs will be Symbol-Error-Rate (SER). This metric is simply the number of symbol errors in a collected data set divided by the total number of received symbols.

Another quality of BCC is that data is transferred across the body, and only the body. This is an important distinction because at 100 MHz, the quarter wavelength is only 2.3 feet, which happens to be in the range of a human limb length. To ensure that the human body is not acting as an antenna, the receiver and transmitter must be shielded and have shielded wires connecting the boards to their respective electrodes. When shielded, there should be no significant data transfer between boards when not connected to the body. The potential radio transfer of data between boards will be tested in addition to the SER.

The following chapters will explain the implementation of a high speed, wearable BCC data communication system. The necessary electrical, software, and mechanical specifications for a successful system will be presented in Chapter 2. Chapter 3 will explain the implementation of the system. Simulation and testing methods, as well as experimental results, will be presented in Chapter 4. Chapter 5 will evaluate the system performance, discuss what was learned from this thesis, and present ideas for future changes to the system that can improve the overall performance.

Chapter 2

Design

Before parts can be purchased and assembled into a system, there must be an end goal, architecture, and list of specifications for said system. The previous chapter discussed the end goal of the BCC system, as well as some previous system architectures. This chapter will discuss the electrical, software, and mechanical requirements of a high speed BCC system, as well as present the basic architecture of the finished device.

Table 2.1 displays the system level specifications of the BCC transmitter and receiver. All specifications were decided on at the beginning of this project as the outline of the system.

Specification	**Target**
Power supply	Battery
Single charge on time	8 hours
Memory	18 GB
Data transfer rate	5 Mbps

Table 2.1: System Specifications

2.1 Electrical Signal Specifications

For this system to succeed, a signal must be able to be sent through the body's channel and recovered by a receiver with enough detail to extract the information from the transmission. This requirement means the characteristics of the body's

channel, which can be seen in Figure 2-1 dictate the entirety of the electrical signal specifications. The entire transmitter system and the front end of the receiver are responsible for the signal conditioning, so the electrical signal specifications will be given for these two sections of the system.

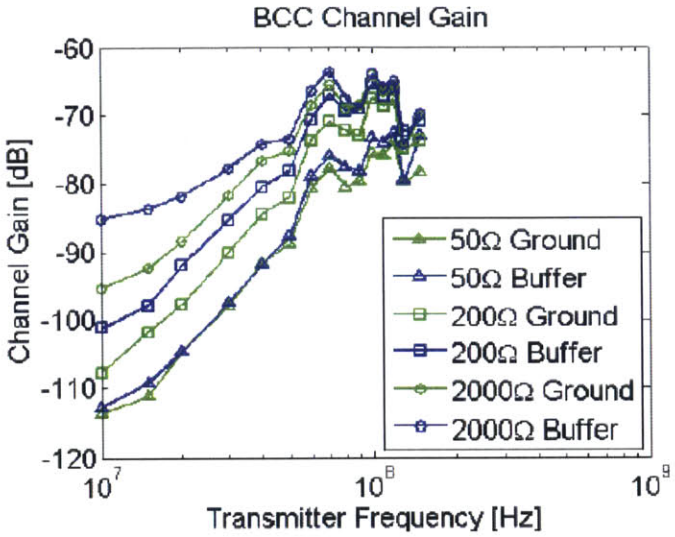

Figure 2-1: Experimentally measured data of the channel gain vs the frequency of a transmitted signal in the body [1]

2.1.1 BCC Transmitter Signal Requirements

Table 2.2 displays the important electrical specifications of the transmitter that relate to the signal quality. All specifications assume a 2 V transmitted AC signal, which will be explained in the paragraphs below along with discussions of each specifications.

Specification	Target
Output pk-pk	2 V
Output current	40 mA
Slew rate	2000 V/μs
Frequencies range	60 MHz - 100 MHz

Table 2.2: Transmitter Specifications

Output pk-pk and Transmission Frequency Range

The goal of the transmitter is to provide the receiver with a signal that closely resembles what was transmitted. With the channel model shown in Figure 2-1, it appears that there is significantly less attenuation between 60 MHz and 100 MHz. This was used as a starting point in the design, and explains why the authors of previous works also chose to operate in approximately this range. Though there is significantly less attenuation, there is will still be between 80 dB and 60 dB of attenuation, so the transmitted signal should be large enough such that the signal to noise ratio of the signal once it reaches the receiver is not too low to recover. The dominant noise source will come from the first stage of the analog front end of the receiver. Some quick market research shows that amplifiers that operate in this frequency range and are "very low noise" have an input referred noise of between 1 and 5 nV/$\sqrt{Hz}$. Assuming the system has a bandwidth of 200 MHz, the noise floor at the input of the receiver will be:

$$\begin{aligned} V_{noise} &= 5nV/\sqrt{Hz} \times \sqrt{200MHz} \\ &= 71\mu V \end{aligned} \tag{2.1}$$

Estimating a channel attenuation of 70 dB, the transmitted signal would have to be 210 mV to match the noise floor. However, an SNR of 0 dB would make signal extraction difficult, so a pk-pk amplitude of 2 V was chosen to give an SNR of 20 dB with these approximations.

Output Current and Slew Rate

Capacitively coupling to the human body provides another design challenge. As shown in Figure 2-2, the path from electrode to conductive channel is a resistor in parallel with a capacitor. The resistor is reported as being in the range of hundreds of kΩ, and the capacitance from the arm, which is where this system will be connected, is approximately 48 pF [9]. In the MHz frequency range, the parallel capacitor has a much lower impedance than the resistor, so the load on the output of the transmitter in this range is equivlent to a 48 pF capacitor. Driving this capacitor 2 V pk-pk at

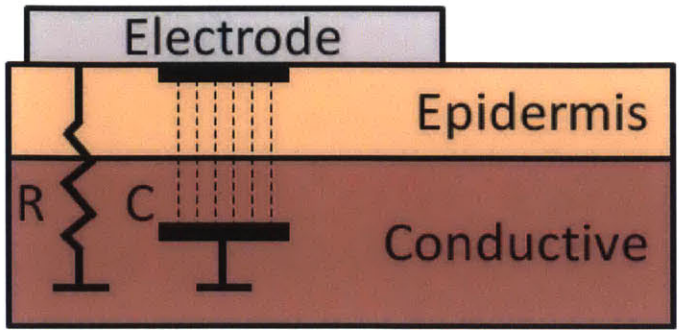

Figure 2-2: A resistor-capacitor model of body coupling

100 MHz will require a minimum output current of:

$$\begin{aligned} I &= C\frac{\mathrm{d}V}{\mathrm{d}t} \\ &= 48pF \times \frac{2V}{5ns} \\ &= 19mA \end{aligned} \tag{2.2}$$

This, however, is the minimum current to generate a triangular signal at 100 MHz. At the bare minimum, the output of the transmitter should provide 50 mA to generate an adequate square wave at 100 MHz. Another specification that limits the voltage swing speed is the slew rate. Since a triangular 100 MHz wave was not desired, it was decided that the slew rate of the transmitter's output should swing 2 V in 1 ns for a slew rate of 2000 V/μs.

2.1.2 BCC Receiver Signal Requirements

Table 2.3 displays the important electrical specifications of the receiver that relate to the signal quality. Each specification will be explained in this section. All specifications assume transmitted data matches the specifications in Table 2.2.

Gain and Bandwidth

With the specified frequency range, Figure 2-1 shows an attenuation of 60 dB to 80 dB. The receiver architecture does not need to completely recover the signal, but it should be amplified to at least the 100 mVpk-pk range. This minimum amplitude is

Specification	Target
Bandwidth	150 MHz
Gain	1000+
Input referred noise	<0.8 nV/$\sqrt{Hz}$
Output noise rejection	100mV
Digital sampling rate	>200MHz

Table 2.3: Receiver Specifications

required because the signal will need to be converted to a digital signal before reaching the digital system. To perform a single bit analog-to-digital conversion, a comparator will be used. These comparators slew the fastest when there is a sizable difference in voltage between the positive and negative terminals. The voltage difference between these rails is referred to as the overdrive voltage. This 100 mVpk-pk range is enough to give a typical comparators an overdrive voltage that allows the system to perform at maximum speed. With a transmitter sending 2 Vpk-pk information across a channel with as much as 80 dB attenuation, a 200 μVpk-pk received signal must be amplified to at least 100 mVpk-pk. This sets the minimum gain to 500. To ensure that the data will be amplified to the desired magnitude and account for some variation in the channel model measurements, the target gain is set to 1000.

The bandwidth of the analog front end of the receiver should match the bandwidth of the transmitted frequencies (100 MHz). However, when selecting parts it is necessary to give the bandwidth some margin of error because of IC process variation and potentially incorrect datasheet information. A part that reports 100 MHz bandwidth could potentially only perform as desired up to 95 MHz, for example. As a conservative measure, the required bandwidth of components in the analog front end of the receiver should be greater than 150 MHz.

The target gain and bandwidth of 1000 and 150 MHz, respectively, now dictates part of the design of the analog front end. Using a single, commercially available, modern-day IC amplifier would require a gain-bandwidth product (GBWP) of 150 GHz. These amplifiers do not exist. In contrast, there is an abundance of IC amplifiers with GBWP's greater than 1.5 GHz. Each of these amplifiers can provide a gain of 10 and a bandwidth of 150 MHz, but connecting three in series gives the desired gain

of 1000 and 150 MHz bandwidth.

Input Referred Noise and Output Noise Rejection

The input referred noise of the receiver determines the minimum magnitude of the received signal. Working backwards from the specifications that have already been determined, the minimum post-amplified signal should be 100 mVpk-pk, and the gain of the system should be greater than 1000. This results in a minimum detectable input signal of 100 μV, unless the noise floor of the input is too high. As discussed in Section 2.1.1, it is desirable to have an SNR of 20 dB, so the target noise floor should be 20 dB less than 100 μV (or 10 μV). From this value, a conservative estimate of the maximum input referred noise of the first stage amplifier can be determined by dividing the noise floor by the square-root of the receiver's bandwidth. Assuming 150 MHz bandwidth and a noise floor of 10 μV, the maximum input referred noise of the system must be less than 0.8 nV/$\sqrt{Hz}$.

If all discussed specifications are followed, the system should only have 10 μV of noise floor at the analog to digital conversion stage. This is simplifying the system, so there may be some more noise that has not been determined. The noise that appears at the analog to digital conversion stage is important to consider even if it is an order of magnitude less than the signal. When the signal is close to the tripping point, the noise on the signal could be enough to trip the comparator, possibly more than once depending on the sensitivity and speed of the comparator. To prevent this from occurring, the comparator will be implemented with 100 mV of hysteresis. This hysteresis will prevent false trips because once the comparator changes state, the tripping point will move 100 mV closer to the old state of the comparator, requiring that the signal swing be larger than 100 mV to trip the comparator again.

2.2 Digital Specifications

Table 2.4 displays the target specifications of the digital back end of the receiver architecture. Each specification will be explained in this section. All specifications

assume transmitted data matches the specifications in Table 2.2.

Specification	Target
Sampling rate	>200 MHz
Frequency detection rate	5e6 frequencies/second per frequencies/bit
Memory read/write rate	> 5 Mbps (dependent on architecture)
Code complexity	Maximize use of logic gates, minimize use of multiplies

Table 2.4: Digital Specifications

Frequency Detection Rate and Memory Read/Write Rate

The target transmission speed of this system is 5 Mbps. Since the data will be encoded in frequencies ranging from 60 MHz to 100 MHz, the receiver will need to recognize what frequency has been sent fast enough to meet the 5 Mbps data rate. This data rate does not necessarily mean that frequencies must be detected at 5 Mbps. For example, if every detected frequency contains two bits of information, then a frequency detection rate of 2.5 million frequencies/s will deliver 5 Mbps.

Memory read/write rates depend on the architecture of the system and whether external memory is required. If external memory is not required, then the memory needs the capability to be written at the data rate of the incoming signal. Preferably, the memory should respond faster than the data rate in case two frequencies are detected slightly closer together in time. This case is not as important for synchronous systems because of the shared clock enforcing data rates, but since the transmitter and receiver can only communicate via one signal, the system is inherently dependent on the reliability of the frequency detection speed and reliability. A system with external memory further complicates this specification because most memory devices cannot be written in blocks of data less than a byte (8 bits). The minimum block size of the external memory will dictate the minimum buffer size of the frequency detection system. This system now must have the capability to either write and read the buffer at twice the speed of the incoming data rate, *or* be simultaneously written and read, be larger than the minimum buffer size, and transmit data to the external memory faster than the data is overwritten. The exact system cannot be completely

determined until available parts are selected and analyzed for potential bottlenecks.

Sampling Rate

Following simple Nyquist sampling theory, the sampling rate of a digital system must be at least twice the maximum frequency of the sampled signal. This prevents aliasing from occurring in the sampled signal. The maximum frequency from the incoming digital signal depends on the preceding system architecture and the overall frequency detection scheme. If the preceding analog architecture does the frequency detection, then the digital system only needs to sample at twice the frequency detection rate. However, if the digital system is to perform the frequency detection, then it must sample the data at twice the maximum frequency of the received signal. For this system, the data directly preceding the digital back end can be low pass filtered to 100 MHz, so the sampling rate needs to be at least 200 MHz in order to prevent aliasing.

A lower sampling rate of around 5-10 MHz would be ideal, but the feasibility of this architecture is dependent on how fast commercially available ICs can detect 60 MHz to 100 MHz. Commercial PLL's and mixers were researched and judged on their maximum required frequency detection time and power dissipation, but none were found that could meet the specifications of this system. The typical trend was that the PLL's ran with slightly less power than the mixers, but they required greater than 1 μs for the output to settle. This would decrease frequency detection to less than 1 Msps. Mixing the incoming signal down to baseband would be a faster method of frequency detection, however, 100 MHz range mixer ICs consume hundreds of milli-Amps quiescent current with a 5V rail (almost 1 W per IC). Multiple of these would be required on the board to detect more than one frequency, so the total power loss from this method would either dwarf the on-time specification, or require an uncomfortably sizable battery pack to be worn by the system tester. This much power dissipation could also make thermal dissipation an issue for a small PCB, which is another issue that should be avoided for a wearable device.

No commercially available frequency detection IC's were determined to be suit-

able for this system. This requires that the digital back end perform the frequency detection, and the sampling rate to be greater than 200 MHz as mentioned above.

Code Complexity

Performing digital frequency detection for the frequencies of this BCC system requires both greater speeds and digital complexity than systems that perform frequency detection before the digital back end. These two requirements will present a significant design challenge because most frequency detection algorithms are computation heavy (requiring many multiplications). The speed of digital systems are commonly limited by the time it takes to perform large computation(s) in one clock cycle. Digital architectures that depend mostly on digital logic are typically capable of much faster speeds than those that are heavy on arithmetic computation. For this reason, frequency detection algorithms that can be implemented with digital logic will be favored over those that require heavy computation, such as the Fast Fourier Transform (FFT) or Frequency Impulse Response (FIR) filters with lots of multi-bit coefficients.

A sampling rate and computation clock of 200 MHz greatly diminishes the available digital processing units that can both be placed on a wearable device and not destroy the power consumption specifications of the system. This operating frequency makes any microcontroller implementation unfeasible because the set architecture of microcontrollers provides far too much overhead for it to operate at low power and fast enough for a monolithic task such as detecting these high frequencies. A highly customizable system such as a Field Programmable Gate Array (FPGA) is much more well suited to this kind of function. FPGA's have no preset system architecture. They consist of an array of logic gates that can be programmed, allowing for simple tasks such as running a single algorithm to be much faster than a system with a set architecture not optimized for such a task. FPGA's typically have some added cores that specialize in fast arithmetic computation and specialized inputs that multiply the speed of an input clock. A quick market investigation found low power, commercially available FPGA's that boast maximum clock speeds at upwards of 500 MHz, making an FPGA the choice for the digital back end to this system.

2.3 Mechanical Specifications

Table 2.5 displays the important mechanical specifications of the device that relate to the system's performance and wearability. All specifications will be explained in the paragraphs below along with discussions of each specifications.

Specification	Target
Electrodes	2, shielded
Device casing	metal shielded box
Attachment to body	straps around wrists
Board size	3" x 2"

Table 2.5: Mechanical Specifications

Electrodes and Device Casing

The signal being sent through the body must be differential, so two electrodes attached to the body are required to send the signal. To receive the signal, two electrodes are also required. This is because the receiver and transmitter do not share a common node, so both received and transmitted signals are floating differential. As mentioned in Chapter 1, the system cannot be susceptible to wireless interference. For this reason, the circuit must be enclosed in a metal shielded container. The electrodes must be wired to the circuit through the box, so the wires connecting the box to the electrodes must also be shielded.

Attachment to the Body and 3" x 2" PCB

To allow for an easy connection to the body, the circuit will be attached to the tester's wrists, and the electrodes will be connected to the forearms. The wrist is a common attachment location for devices and the device can be held in place by some straps that wrap around the wrist. The device's PCB should be of minimal size to provide the easiest and most comfortable means of attachment to the wrist. Minimizing the size should also reduce the unwanted capacitive coupling from the board to the body. The PCB of an older BCC device made of commercial parts was measured to be

4.5" x 2.5". Using the older board as a sizing reference, this circuit should meet a specification of 3" x 2".

2.4 Basic Architecture

Determining the BCC system's specifications and feasibility of each specification led to a basic architecture for both the receiver and transmitter. The receiver will have an analog front end consisting of three low noise amplifiers of gain 10 and GBWP 1.5 GHz, followed by a comparator with 100 mV hysteresis to both reject noise and convert the signal into a digitally compatible format. For the digital back end, the digital processing will be performed on an FPGA, where the computed data will be buffered and stored on an external memory card. The external memory card was deemed necessary because of the 18 GB system level specification. Figure 2-3 shows the basic architecture for the receiver.

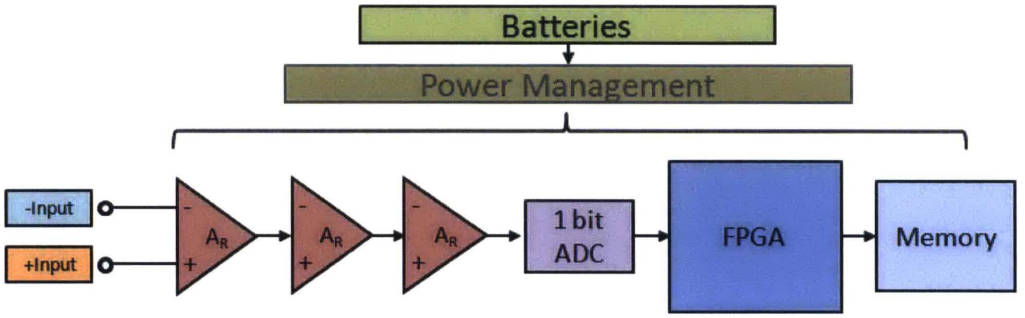

Figure 2-3: Basic architecture for the BCC receiver

For cost and ease of prototyping, it was decided to put the transmitter and receiver on the same board. This means there is an FPGA and memory available to the transmitter. A known transmission signal can be stored in either FPGA block RAM (BRAM) or the external memory, read from the FPGA, and sent to an output buffer amplifier. The basic architecture for the transmitter is shown in Figure 2-4.

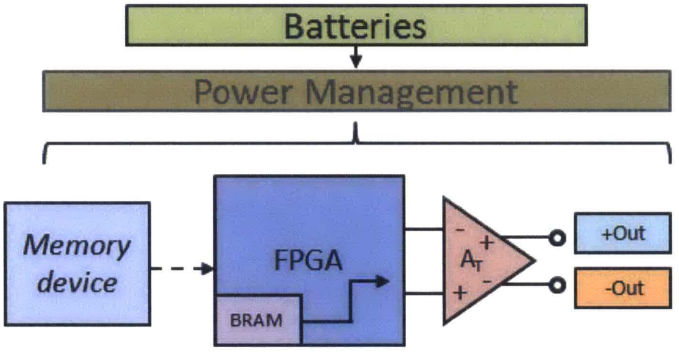

Figure 2-4: Basic architecture for the BCC transmitter

Chapter 3

Implementation

In this chapter, the final implementation of the BCC system is presented. First, the finalized hardware will be shown, and hardware topics that have not been covered in previous chapters, such as the power management and PCB layout techniques, will be explained. Following the hardware, the software system will be described in detail.

3.1 Electrical Hardware

This section describes the hardware selection process and some non-signal conditioning design. To follow what portion of the board each section is describing, Figure 3-1 displays the finished PCB with each subsection highlighted and labeled.

3.1.1 Analog Front End

The most important decision for the analog front end was the first input amplifier. If this amplifier does not have enough bandwidth, or has too much input referred noise, then the rest of the system will not work up to specification. Most amplifiers with a GBWP of 1.5 GHz that are stable for a gain of 10 for frequencies up to 100 MHz have a reported input referred noise above 5 nV/$\sqrt{Hz}$. This is far too high considering the specification asks for the input referred noise to be less than 0.8 nV/$\sqrt{Hz}$. Only one amplifier had the necessary performance to be used in this system, and that amplifier

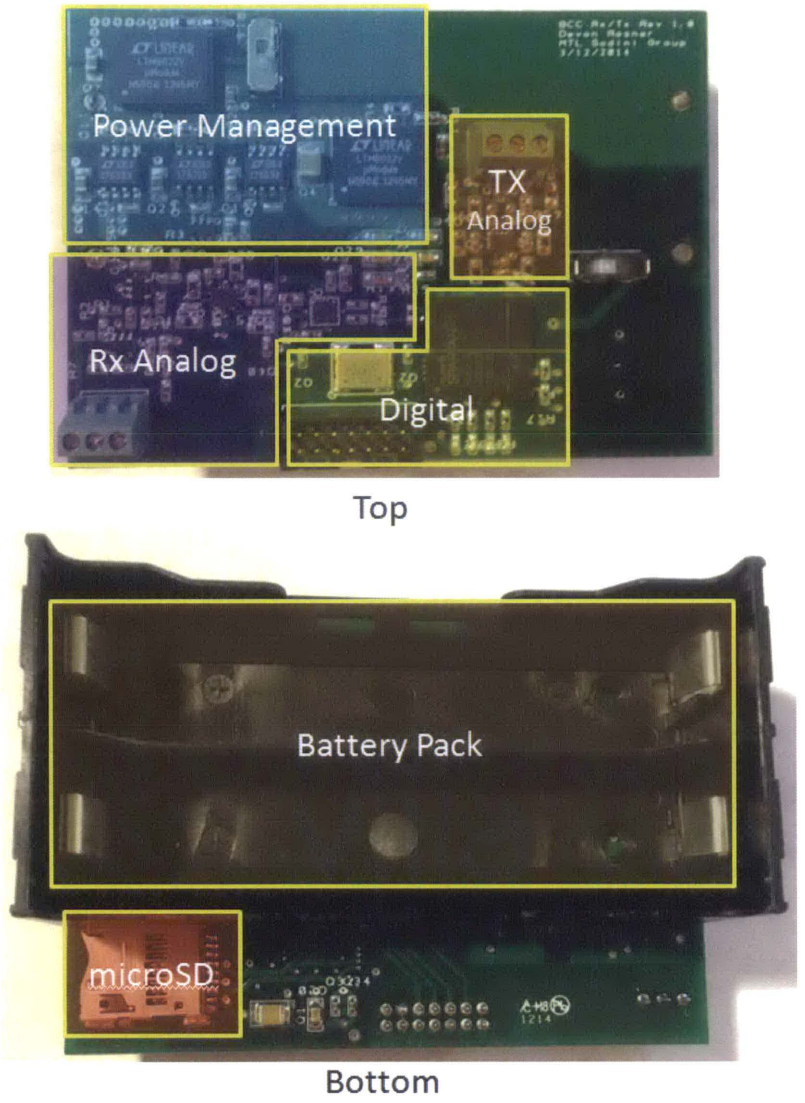

Figure 3-1: Top and bottom of BCC rx/tx PCB with labeled sections

is the LMH6629 from Texas Instruments (originally National Semiconductor). This amplifier boasts a full gain of 10 up to 300 MHz with a 3.3 V single ended input voltage, and 0.69 nV/$\sqrt{Hz}$ of input referred noise. The drawback to using this amplifier is that it only has a slew rate of 1100 V/μs, which is half of the necessary slew rate to swing 2 V at 100 MHz. However, this is not a reason to eliminate the LMH6629 from the entire design. It is still the perfect amplifier for the first and second stage of the three stages of the amplifier, since the attenuated input signal will be only 200 mVpk-pk after the second stage if the transmitted signal is 2 Vpk-pk and the channel

attenuation is 60 dB.

The requirements for the third amplifier are now a gain of 10 above 100 MHz and a slew rate of at least 2000. The noise is no longer of concern, as the input referred noise has been amplified by 40 dB by the input to the third stage. A high slew rate and bandwidth led to the use of the LTC6409 differential amplifier. This amplifier has a 3300 V/μs slew rate and a 10 GHz GBWP, so its bandwidth is higher than what was needed. The downside to this amplifier was its input current consumption of 52 mA at 3.3 V input. Both LMH6629 input amplifiers use 30 mA total *together*. Even with the extra consumed power, the added slew performance was worth integrating into the system. The LTC6409 was also used as the output buffer of the transmitter section of the board.

Providing the link between the analog front end and the digital back end is the ADCMP605 comparator from Analog Devices. The basic specifications of this comparator are that it has a 500 MHz bandwidth, 600 ps rise and fall time, and 1.6 ns propagation delay. Combined, these values ensure that this comparator meets the speed requirements of the BCC system. In addition to the speed, the ADCMP605 has two other feature that make it valuable. The first is that it has a hysteresis pin where the designer can place a specific resistor value to ground to give the hysteresis up to 250 mV of hysteresis. This system requires 100 mV of hysteresis for noise rejection, so this feature is helpful. Secondly, the output stage is LVDS. This is a specific signaling standard, and is an accepted input to the chosen FPGA (to be explained in chapter 3.1.2). The entire analog front end of the BCC receiver can be seen in Figure 3-2.

The final nuance of the analog front end design is the requirement of a 1.5V reference and virtual ground. All components of the analog front end run off of a single supply 3.3V, so an AC signal must be biased to a mid-rail. From the point at which the signal enters the receiver board, it is high-pass filtered at 1 MHz and biased up to 1.5V. The signal rides this bias voltage until the output of the differential amplifier biases the differential signal at 1V. All 1.5V rails in Figure 3-2 are there for biasing purposes.

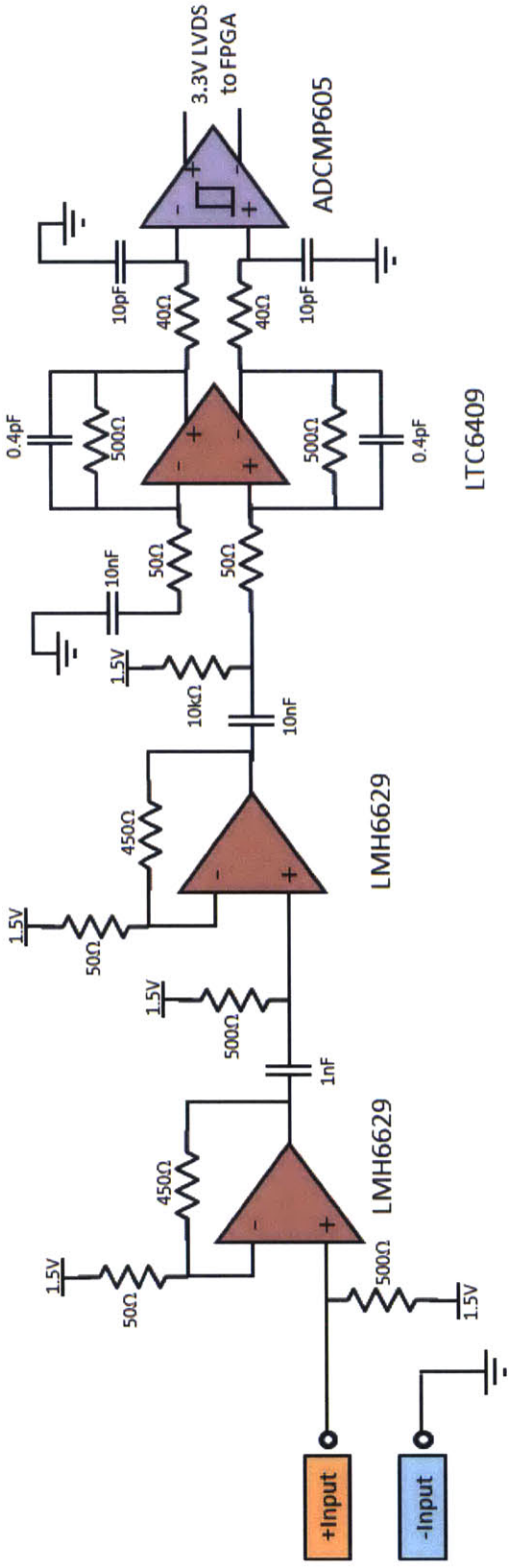

Figure 3-2: Schematic for the analog front end of the BCC receiver

3.1.2 Digital Back End

The first part chosen for the digital back end of the BCC receiver was the FPGA. Major considerations were the size, power consumption, speed, and computation resources. What was found to be the perfect balance of all of these considered qualities was the XC6SLX16 from the Xilinx Spartan 6 family in the CSG225 package. This specific package combination is 13mm x 13mm, making it the largest package on the 3"x2" PCB, but there is still plenty of available board area. The most important quality of this FPGA is that the frequency detection algorithm that was already written synthesized to take up only 30% of this FPGA's resources. It is important to note that the remaining 70% of the resources were not wasted. This code did not include the memory controller module, and FPGA prototypes should, generally, not use much more than 80% of the total resources in order to prevent time consuming compile times. For speed, the FPGA is rated to run at up to 350 MHz clock speeds, but what was discovered later was that this was closer to 200 MHz for computationally intensive systems. Finally, power consumption needed to be addressed. Unlike simple IC's that can list a blanket power consumption, FPGA power consumption is heavily dependent on the application. Using Xilinx's Power Estimator [11], the power consumption for this specific setup is estimated to be 125mW, which is on the same order of magnitude as the other BCC system's devices. An added bonus to using this FPGA is that it has two internal PLL's that can multiply clock speeds. This meant that a 20 MHz clock could be placed in the system instead of a 200 MHz clock that would cost much more, use an order of magnitude more power, and inject significant ground bounce into the PCB ground plane.

The other component in the system's digital back end architecture is the external memory. The four specifications that were considered in choosing external memory were PCB footprint size, amount of memory storage, write/read speed, and the accessibility of data post-testing. These specifications dictated the choice of using a microSD card. Its PCB holder's footprint is approximately the same size as the FPGA, so it can easily fit on the board. The data capacity of microSD cards ranges

from 2 GB to 128 GB, so the 18 GB storage requirement could easily be met. Using the publicly available SPI protocol, these devices can run off of 25 MHz clocks, so data can be transferred at close to 25 Mbps. Finally, these cards can be taken out of their PCB sockets and inserted into a computer for easy data accessibility.

3.1.3 Power Management

As with most mixed signal applications, the BCC receiver and transmitter systems require more than one voltage rail. All IC's in the analog front end of the receiver and the output buffer of the transmitter are powered from 3.3V. The analog front end also requires 1.5V as a reference voltage and virtual ground for AC. The FPGA runs off of a 1.2V core, but its input and output buffers are powered by 3.3V. Both the microSD card and the 20 MHz oscillator are powered by 3.3V. At first glance, there should only be three voltage levels present on the PCB, however, there are actually seven. The first step in power management is to determine how much energy the system consumes. Table 3.1 lists the power consumption of each part in the BCC receiver. Since the transmitter consists of only the FPGA and LTC6409, the receiver is clearly the energy consumption bottleneck.

Part	Power Consumption
LMH6629 x 2	100mW
LTC6409	172mW
ADCMP605	86mW
XC6SLX16	125mW
microSD	100mW
20 MHz Oscillator	20mW
TOTAL	**603mW**

Table 3.1: Power consumption of the BCC receiver architecture

Next, a battery needed to be chosen. Knowing that this device would go through a lot of testing, a lithium-ion rechargeable battery was decided to be the best fit. The first power management design was simply a single cell 3.7V Li-ion battery and a series of linear dropout regulators (LDO's) to provide the necessary voltages. However, as these batteries lose charge, their voltage droops by as much as 0.3V.

At 3.6V, the LDO's will stop working, so this method would not make the most out of the battery's capacity. The final power management system uses two Tenergy 3.7V 2600mAh batteries in series to generate 7.4V. This voltage is buck regulated down to 3.8V and 1.2V using the LTM8022 μmodule from Linear Technology. The 1.2V directly powers the FPGA, but the 3.8V is linearly regulated to a 3.3V analog rail, a 3.3V digital rail, and a 1.5V mid-rail all using variations of the LT1763 from Linear Technology. LDO's are not capable of sinking current, so the 1.5V is used as a reference into a voltage buffer op amp to provide the system with a 1.5V virtual ground rail. In high speed systems with sensitive signals running across the board, separating the digital and analog power supplies is critical. Otherwise, the digital clock edges will appear on the signal lines. This power management hierarchy can be seen in Figure 3-3.

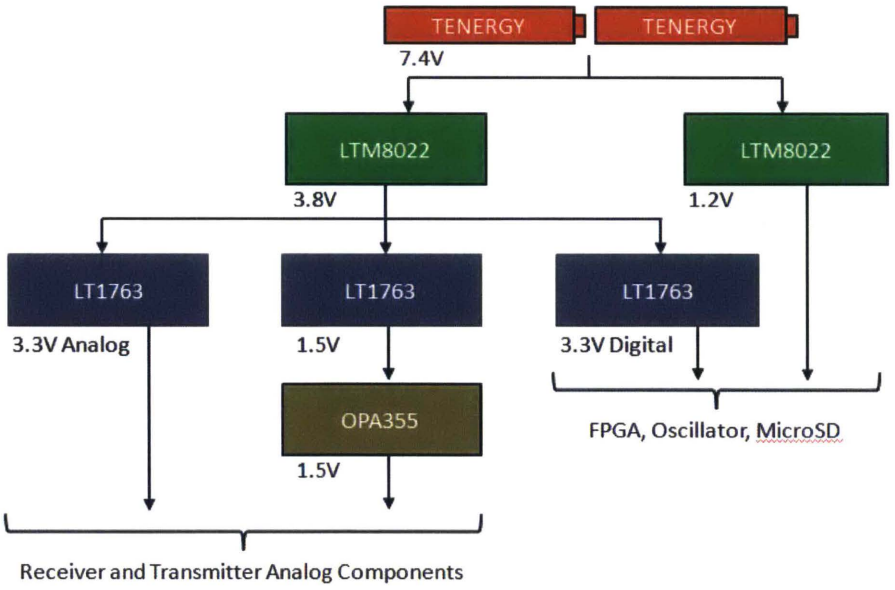

Figure 3-3: BCC system power management hierarchy

One of the system level specifications is that the boards must last for longer than 8 hours. To determine how long the receiver board can run, the total available energy from the two batteries in series must be calculated. Each battery is listed as 3.7V and 2600mAh. This gives a total available energy of $3.7V \times 2600mAh = 19.24Wh$. The efficiency of the LTM8022 stepping from 7.4V to 3.8V with a load drawing 200mA is

85% efficient. The same part stepping from 7.4V to 1.2V with a load drawing 100mA is 80%. Lastly, linear regulators stepping 3.8V to 3.3V are 87% efficient. With the power consumption of each part available, as well as the power converter efficiencies, the total power draw of the system is:

$$\begin{aligned} P_{total} &= \frac{100mW+172mW+86mW+100mW+20mW}{85\%\times 87\%} + \frac{125mW}{80\%} \\ &= 800mW \end{aligned} \tag{3.1}$$

And the total run for the BCC receiver on a single charge is:

$$\begin{aligned} T_{rcvr,on} &= \frac{19.24Wh}{0.8W} \\ &= 24hours \end{aligned} \tag{3.2}$$

In testing, the receiver actually runs out of battery power after 19 hours, but this is still more than twice the minimum specified run time of the system.

3.2 Software

Though the FPGA only takes up $0.25in^2$ on the PCB, it is responsible for the entire frequency detection and bit decoding algorithm, and it acts the memory controller. This section will detail how the FPGA performs these tasks and explain why these methods were chosen. Figure 3-4 provides the simplified digital architecture of the system for reference whilst reading the detailed implementation descriptions. The FPGA system was coded in Verilog HDL.

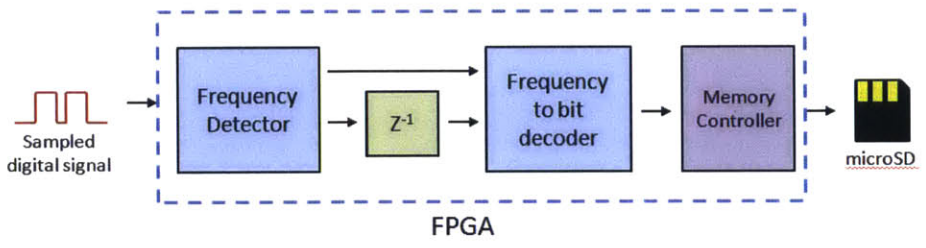

Figure 3-4: Simplified digital architecture of FPGA

3.2.1 Frequency Detection via Autocorrelation

The frequency detection scheme used in this system is based on a well-known fundamental frequency finder scheme that utilizes autocorrelation. The discrete autocorrelation of a function x_n is defined as:

$$R_{xx}(j) = \sum_n x_n \bar{x}_{n-j} \tag{3.3}$$

This is just the convolution of a function with the time-reversed version of itself. The classic fundamental frequency finder scheme works by autocorrelating a recorded signal and then recording the time distance between peaks of the autocorrelation. The autocorrelation has the highest peaks when the time delay j is equal to the fundamental period, so the reciprocal of the distance between peaks of the autocorrelation gives the fundamental frequency. Autocorrelation is used in this system as a means to detect if a specific frequency exists in the received signal. This system needs to detect if one of five frequencies has been received, so there are five banks of 64 individual bits. The bit sequence of each bank is the bit sequence of a specific frequency that is to be detected. Frequencies of interest are 60 MHz, 70 MHz, 80 MHz, 90 MHz, and 100 MHz. The most recent 64 samples of the received signal are convolved with each of the five frequency banks, and if one of the outputs of the convolution peaks above a threshold, then a frequency has been detected. An example system using three frequency banks is shown in Figure 3-5.

When explaining the code complexity specification in Chapter 2.2, the benefits of implementing a digital logic heavy frequency detection scheme over a computationally heavy scheme was explained. Convolution is typically not considered computationally light. If convolution were implemented in its typical fashion in this system, there would be 64 multiplications per bank, per clock cycle. This would amount to 320 multiplications. For this reason, multiplication was replaced with digital logic. Implementing this was not complex because the incoming signal is only 1 bit and it is only multiplied with other 1 bit values. An entire multiplication can be replaced with a single XNOR gate. Better yet, the XNOR gate treats the signals as AC (mean 0),

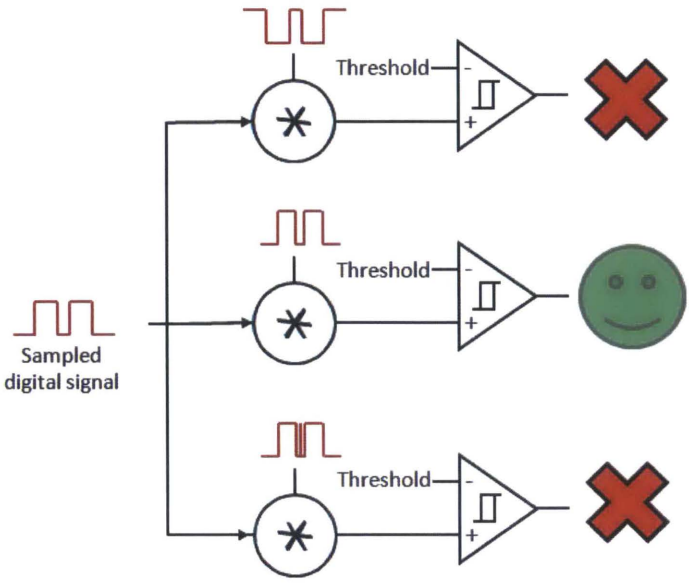

Figure 3-5: A high level example of the frequency detection algorithm using three frequency banks. State machine implementation only allows one frequency to be considered detected at a given instance.

where a "1" is a +1 and a "0" is a -1. This analogy is presented in Table 3.2. Each convolution per bank is performed with 64 XNOR gates that have one of their inputs assigned to an entry in the 64 sample frequency bank, and the other input connected to one of the 64 stored input bits. The 64 outputs of the XNOR gates are summed, and the digital convolution is complete. A graphical representation of this process can be seen in Figure 3-6. Following convolution, the result is compared to a threshold value, and if the convolution produced a value greater than the threshold, the frequency of the convolved bank was detected in the input signal. Using a frequency bank of 64 samples can yields a maximum output value of 64.

A	B	Out	Equivalent Expression	Result
0	0	1	-1×-1	$+1$
0	1	0	-1×1	-1
1	0	0	1×-1	-1
1	1	1	1×1	$+1$

Table 3.2: XNOR logic table and multiplication equivalence

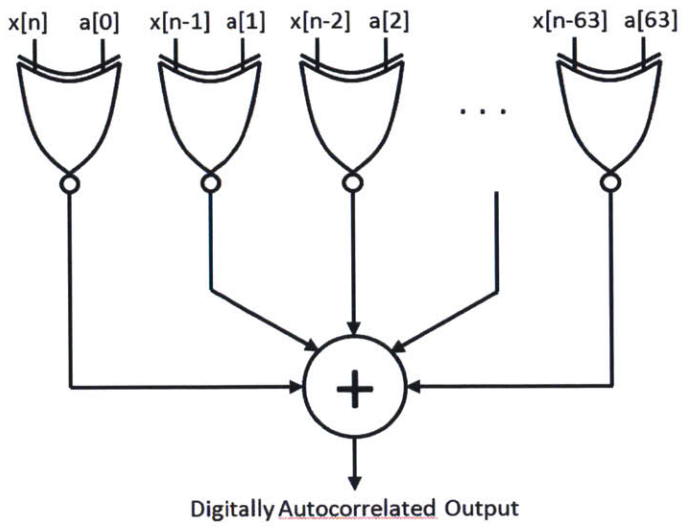

Figure 3-6: Digital implementation of autocorrelation. a[63:0] is the frequency bank, and x[n] through x[n-63] is the input signal

3.2.2 Asynchronous Frequency to Bits Algorithm

Once a new frequency is detected, the digital system must convert that information into bits to store on the microSD card. There is no clock shared between the receiver and transmitter, so the receiver triggers when it detects a change in frequency, then stores the information to memory. Using this system's algorithm, the number of bits per sample depends on the number of total frequencies that are detectable in the system. To create a system with n bits per sample, the system must be able to detect f unique frequencies, where

$$f = 2^n + 1 \tag{3.4}$$

For the system used in this thesis, there are five detectable frequencies, which results in every newly detected frequency carrying two bits of information. The algorithm used to extract these two bits treats the five frequencies as if they are placed in numerically ascending circle where the fifth frequency wraps around to the first frequency. When a new frequency is detected, its distance from the previous frequency (counting down) and subtracting one gives the two bit code. For example, if the third frequency (80 MHz) were detected after the fifth frequency (100 MHz) was the last one to be detected, the fifth frequency had to traverse three frequencies

forward to reach the third frequency. Subtract one from this, and the result is "2". The formula for this algorithm is

$$s = (f_{new} - f_{old})\%5 - 1 \tag{3.5}$$

where f_{new} is the recently detected frequency, f_{old} is the previously detected frequency, and s is the resulting two bits of information. Figure 3-7 shows an example that covers all five frequencies and all four two bit codes.

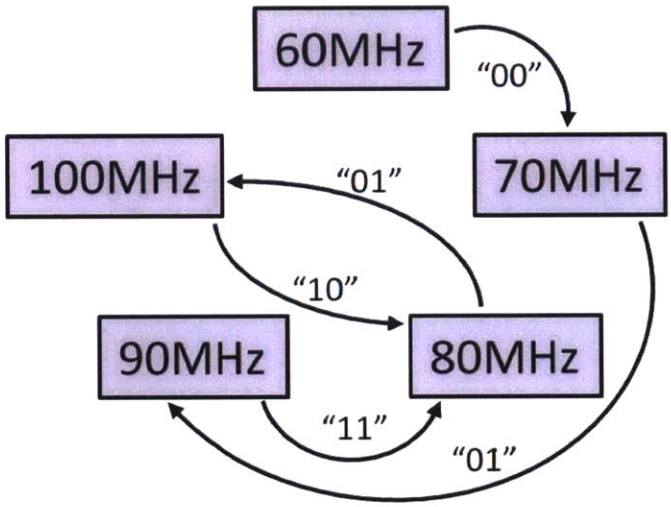

Figure 3-7: Example of asynchronous frequency to bits algorithm

Δ Frequency Steps (Clockwise)	Bits Sent
1	00
2	01
3	10
4	11

Table 3.3: Reference to be used with Figure 3-7 to explain frequency to bits algorithm

3.2.3 Writing Data to the microSD

Two alternating buffers and a memory controller were implemented to write data to the microSD card. The purpose of the two buffers was to allow data to be received while transmitting data to the microSD card at the same time. This means the

transmitter does not need to do anything more complicated than send data at a constant rate. MicroSD cards require data to be written in "pages" of 512 kB, which is equivalent to 4096 bits, and 2048 received samples. The communication protocol is SPI, so all data transfer is serial. This means only one bit can be transferred per clock cycle. Both buffers were implemented as BRAMs with 2048 addresses, with 2 bits per address. At a given time, one buffer is assigned to receive incoming data, and the other is assigned to transmit data or do nothing. The most important requirement for the success of this system is that the buffer writing to the microSD card finishes before the receive buffer is full. For this reason, when the data rate was set to 5 Mbps, the microSD clock was set to 20 MHz, such that the write buffer should finish emptying its contents before the receiver buffer is more than 25% full. The operation of this double buffer is shown in Figure 3-8.

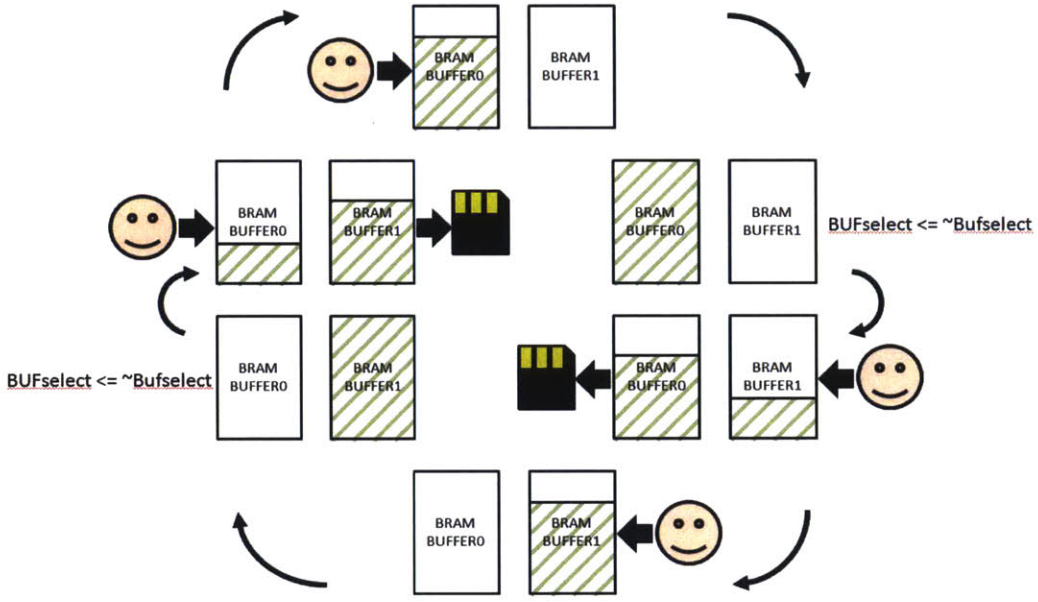

Figure 3-8: Diagram of memory controller states. BUFselect is the binary switch that sets one buffer to write mode and the other buffer to read mode. Green slashes represent amount of data in the buffer

3.2.4 Digital System Initialization

In this section the initialization of the FPGA will be explained. First, the frequency detection and frequency to bit initialization will be covered, then the microSD ini-

tialization.

Frequency Detection and Frequency to Bit Initialization

The initialization of the frequency detection and frequency to bit modules was mostly basic. Necessary registers were either initialized to zero by system default, or initialized explicitly in the Verilog code with a simple value given to the register at its variable declaration. However, the five frequency banks start as empty 64 bit registers, and read their coefficients from five BRAMs that initialize themselves from a set of .COE files. This was done because the five frequency banks of information changed frequently over the course of the system's development, and writing MATLAB code to generate coefficients and write them to a .COE file was a much faster process than writing MATLAB code to format the coefficients such that they can be copy/pasted into the Verilog code directly.

MicroSD Card Initialization

Unlike the rest of the FPGA system, the microSD card initialization has numerous steps and requires more code for initialization than for its main function of sending data. This section will explain the basics of the microSD card and the necessary steps for a successful setup for multi-block write in SPI mode.

The microSD card has 8 pins, as shown in Figure 3-9 [12]. For SPI mode, only 6 of these pins are of concern. These pins consist of Vss, Vdd, DO, DI, SCLK and CS. Vss and Vdd are the power and ground pins, DO is the data out of the microSD (*into* the FPGA), DI is the data in to the microSD (*out of* the FPGA), SCLK is the clock input, and CS is the chip select.

When the system is powered on, the FPGA must wait at least 1ms before communicating with the microSD. The SCLK should also be set to a speed less than 400kHz. For this system, 200kHz was used. After 1ms, CS and DI should be set "high", and the FPGA should stall for 80 clock cycles. At this point, CS can be set "low" and the CMD0 command can be sent to the microSD. The exact bit sequence of any command mentioned in this section can be found in the microSD code section

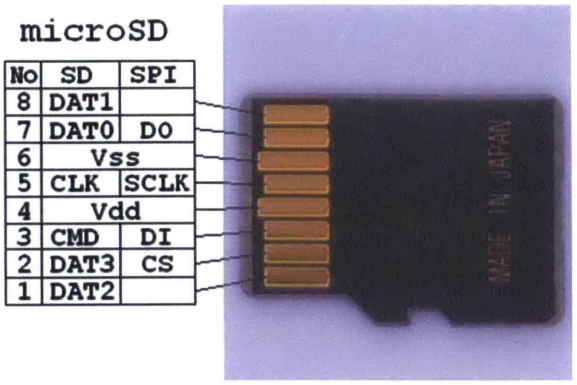

Figure 3-9: MicroSD pin locations

of Appendix A, at the top of the module where the constants are initialized. If the microSD sets the DO pin "low" at any point after CMD0 is sent, then the microSD has responded to the CMD0 command. The first "low" detected after sending a command is the start of a byte response. Once the 8 bits of the byte response are received, CMD8 can be sent. After its byte response is received, this procedure can be repeated for CMD55 and ACMD41. If the response to ACMD41 is anything besides 8 successive 0's, CMD55 followed by ACMD41 must be resent. This loop repeats until the response to ACMD41 is 8 successive 0's. At this point, the SCLK can be set to a speed in the MHz range. For this system, the SCLK was set to 20MHz. After sending CMD58 with the fast clock, the microSD card is ready to be set to multi-block write. This procedure can be followed in Figure 3-10 [13].

The multi-block write setting keeps the microSD card in write mode, and after every 512 kB page is received, responds within 8 clock cycles that it is ready for more data. To set the microSD card to this mode, CMD25 is sent. After the microSD responds and 8 clock cycles have passed, a data packet can be sent. Once the packet is sent, the microSD will respond, and when it is finished, another data packet can be sent. This can be repeated until the microSD card is full. Specifications for multi-block write mode can be found in Figure 3-11 [12].

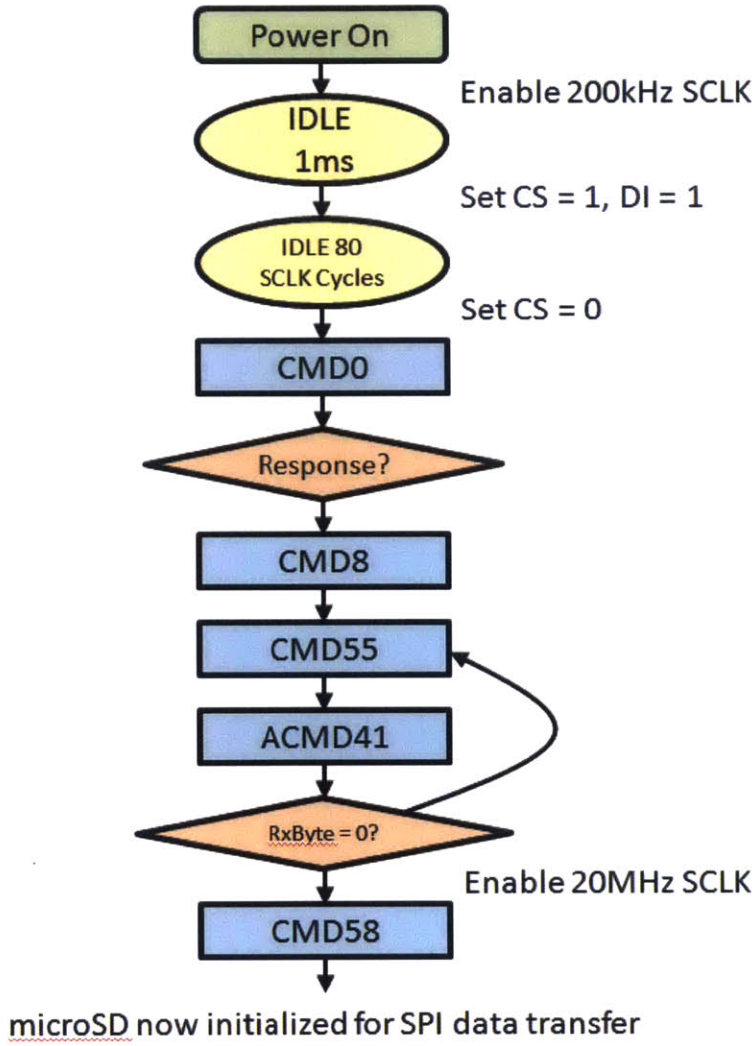

Figure 3-10: MicroSD SPI initialization procedure [13]

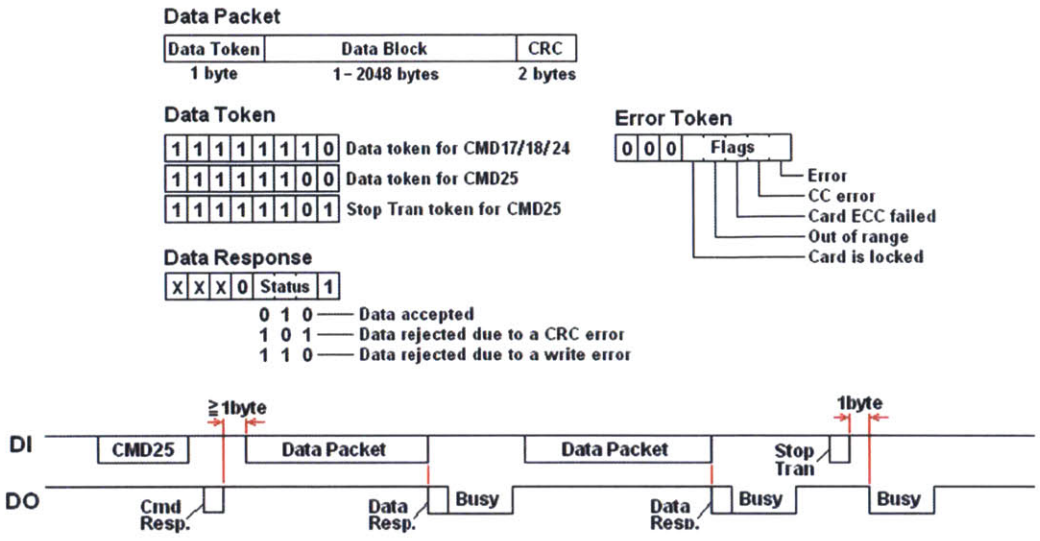

Figure 3-11: MicroSD multi-block write protocol

3.2.5 Importing Data

Reading data from the microSD is not as simple as inserting it into a computer and opening the related drive. The data written to the microSD is unformatted, which ruins the formatting and makes the card invisible to Windows environments. Fortunately, the software Hex Workshop [14] can still detect the card and display the raw 512 kB pages. From this program, data from specified addresses can be saved to .bin files. Once saved, this data can be parsed by any programming language capable of parsing binary files. For this project, files were parsed with Python.

3.3 Mechanical

To summarize the mechanical specifications of the system, there must be two electrode connections to the receiver, two electrode connections to the transmitter, a 3"x2" PCB, straps holding the system to the wrist, and everything must be shielded. The 3"x2" specification for the PCB was broken by 1/4" to support the batter y pack's width. Had the battery pack been separate from the PCB, the size could have dropped by at least 1" in each dimension. This is because, as seen in Figure 3-1, 3/4" of empty space exists on the top right of the PCB to support the battery pack. In

addition, the majority of the PCB's bottom side was covered, so it was unusable by anything more than the microSD card. However, keeping the battery pack on the PCB made the handling of the boards much easier, and simplified the prototyping process. A pre-made aluminum box was purchased for the shielding. It adds an extra 1/2" and 1" to the length and width of the board, but it was the smallest sized standard metal box that could fit this system. Two holes were drilled in the box for SMA connection headers. Two shielded half-SMA-half-button-connection cables attach to these headers (via SMA) and can directly connect to two 3M^{TM} 2560 Red Dot electrodes. Velcro straps hold the system securely to the wrist. Figure 3-12 shows the final mechanical system.

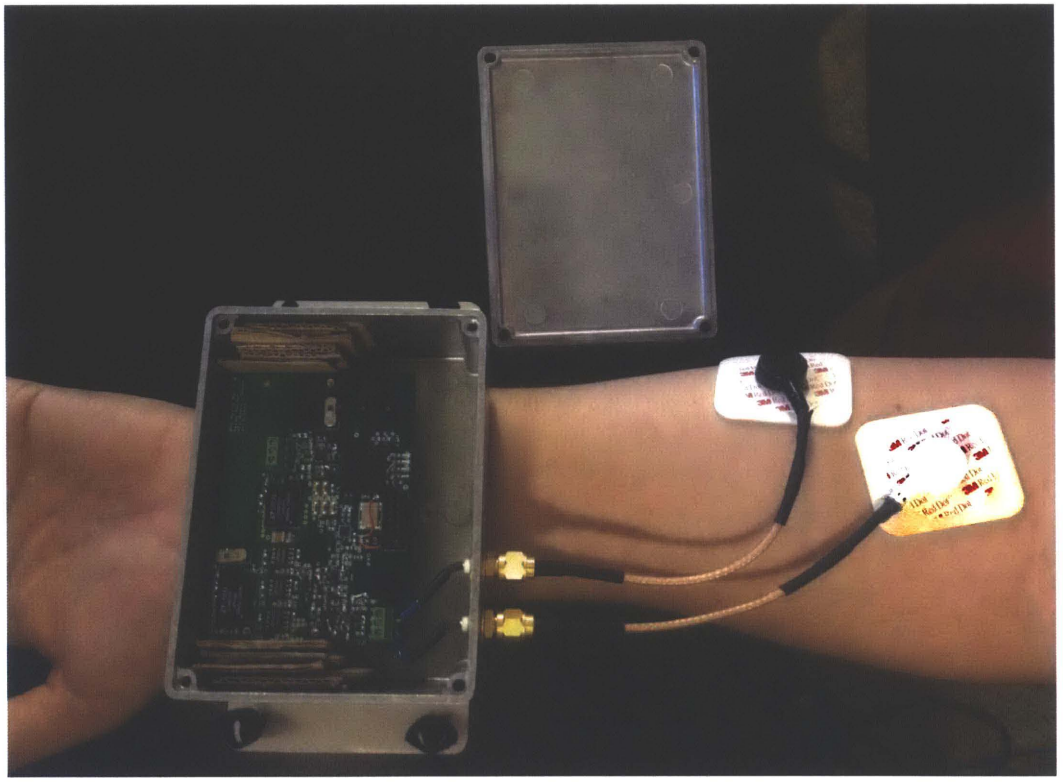

Figure 3-12: Combined mechanical system. Note that the metal cover to the box is attached for all testing

Chapter 4

Testing and Results

Before testing the entire integrated BCC system, every individual section needed to be tested for functionality. This chapter will begin by explaining and presenting the results of the verification of these sections. Then, the final system test will be explained, and results will be shown and summarized.

4.1 Analog Front End SPICE Simulation

Prior to purchasing and building the PCBs, the analog system was developed in simulation with the exception of the ADCMP605 comparator. Linear Technology and Texas Instruments provided detailed device models for the LTC6409 and the LMH6629, respectively. The models were complete with noise specifications, full pin-out functionality, bias voltages and currents, frequency stability, etc. The analog front end simulation was setup to simulate the same circuit as Figure 3-2 up to the comparator. Transient, AC, DC, and noise simulations were observed, but the most important information was gained from the AC and noise simulations as shown in Figure 4-1and Figure 4-2. The AC simulation shows the magnitude and phase of the differential output of the LTC6409 with a -60dB AC input. This simulation shows the circuit providing greater than 60dB of gain for the required frequency range of 60 MHz to 100 MHz. It also shows that the system is acceptably high pass filtered so the dominant frequencies are above 500 kHz. The noise simulation was actually

used in design to determine the input termination resistor. Since this resistor is the first noise source in the system, it is responsible for almost the entire system's noise. Figure 4-2 shows the noise density of five different input resistor values, and their associated gains across frequency. The tradeoff shown is that higher resistor values provide higher gain, but also inject more noise into the system. It should be noted that all gains shown are half of the total gain because the analysis only accounts for one end of the differential output. The final design decision used 500 Ω because of its low noise performance. Also, the gain is approximately equal to the higher resistor values for the frequency range of importance. It is actually *better* that it has less gain for lower frequencies because this attenuates undesired signals more.

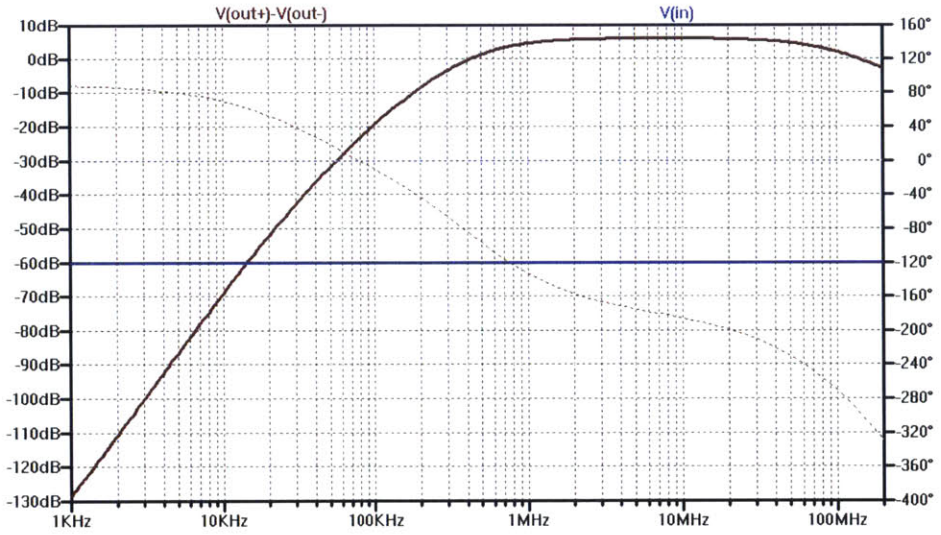

Figure 4-1: AC SPICE simulation of analog front end

4.2 FPGA Behavioral Simulation

An FPGA behavioral simulation was designed for pre-synthesis testing of the digital back end system. This simulation tests everything on the FPGA from input frequency signal to output to the microSD card. The one component not simulated was the microSD card initialization procedure, whose testing will be explained later. The simulation works by using the BCC transmitter code to act as the input to the BCC receiver simulation. This was used to demonstrate the frequency detection actually

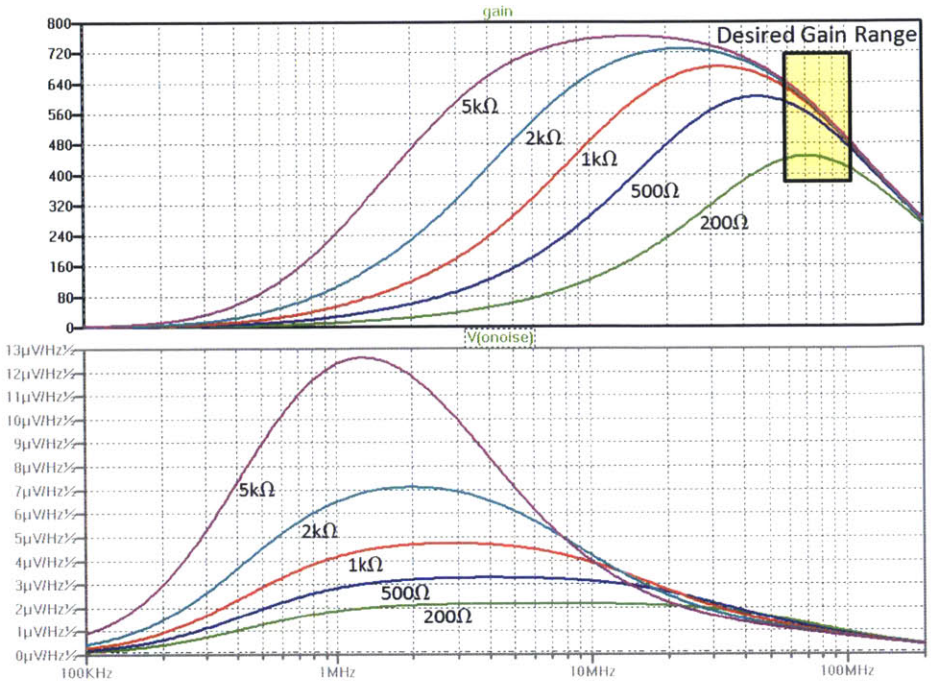

Figure 4-2: Gain and noise spectral density SPICE simulation of analog front end

detecting frequencies, then converting them to bits, and finally storing them into a switching double buffer. Figure 4-3 shows the result of the transmitted frequency changing and how the system responds, where Figure 4-4 shows how the system responds to a recently full buffer.

4.3 Test Strategies

Following the simulations of the system, the PCB's were populated and programmed to be tested for their hardware performance.

4.3.1 Frequency Detection

Isolating individual parts of the system and testing them was a challenge. This was especially the case of the analog front end because of its sensitivity to parasitic capacitances, such as the capacitance of a passive oscilloscope probe. For this reason, an active probe with a capacitance of 3pF was required to attain accurate gain measurements. Before testing the analog front end, the synthesized FPGA system was

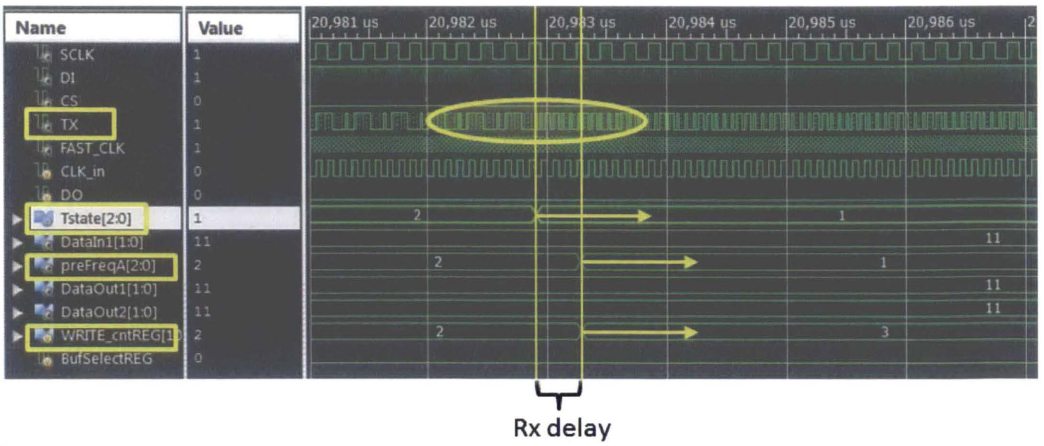

Figure 4-3: Behavioral simulation of frequency detection in FPGA. TX is the transmitted signal clearly changing frequencies. Tstate is the variable that controls what frequency TX is sending. preFreqA represents the most recently detected frequency. WRITE_cntREG counts the number of received frequencies since the last buffer switch

validated, and then the full system was tested with signals of varying magnitude and frequency input to the analog front end.

To test the FPGA receiver alone, the output of the tested transmitter board was wired directly to an available input pin of the FPGA receiver. The first test was strictly visual. Three of the four indicator lights on the PCB were used to display which of the five frequencies has been detected. For this test, the transmitter was set to alternate frequencies every 1 second, and clean frequency detection was visible. Then, the transmitter was set to change frequency at a rate of 5 Mbps. An Intronix Logicport 500 MHz logic analyzer was connected to the three test outputs and was used to analyze frequency detection at data transfer speeds. It was mostly helpful with fine-tuning the frequency detection threshold values.

Once the digital system was tested to be a reliable frequency detector, the analog front end was programmed to be the new input to the FPGA. A signal generator was connected to the analog front end via SMA, and varying magnitudes were tested across the 60 MHz to 100 MHz range. The results of this testing can be seen in Figure 4-5. The output frequency detection was displayed on three indicator lights on the PCB. For the entire frequency range, the output of the frequency detection was clean

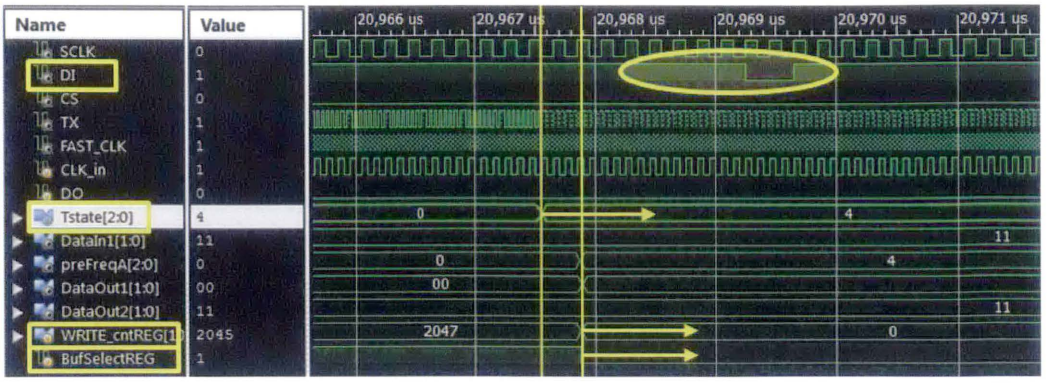

Figure 4-4: Behavioral simulation of buffer switching in FPGA. DI is the signal out to the microSD. Staying low represents a sent start token for a new round of data transmission. WRITE_cntREG is shown overflowing and signaling BufSelectREG to flip. This reverses the roles of the two buffers

down to a 600 μV pk-pk. Below this value, other frequencies began creeping into the output until 400 μV pk-pk where the system was displaying random results. This did not meet the specification of working down to 100μV signals, but with an input 2 V pk-pk to the body, the system should still work because the attenuation should only be 80 dB at max. An input signal of 100μV would be the result of the channel attenuating 86 dB, which is approximately 6 dB more attenuation than the reported channel model in Figure 2-1 for the 60 MHz to 100 MHz range.

4.3.2 The MicroSD Card

As mentioned earlier, the microSD card initialization procedure was not simulated. This was because doing so would require creating an entire microSD simulation brain to simulate the behavior of an actual microSD card. Instead, a short simulation was first built to ensure that the correct signals were being sent in the correct order. Then, the microSD card system was tested on hardware. The four LED indicator lights were used to display which state the initialization procedure was in. If the system was programmed incorrectly, this would report a constant state, showing where the system failed. The first card this was tested on was a SanDisk 4 GB card that never worked. This was because SanDisk removed SPI communication ability from some of its microSD cards, making debugging incredibly difficult until this fact

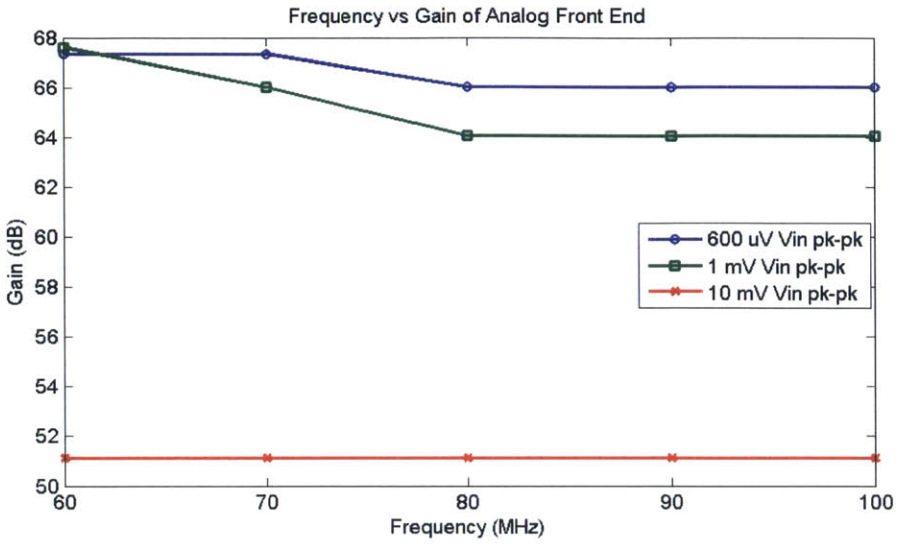

Figure 4-5: Gain of analog front end with varying input voltages across frequency. Note that the gain of the 10 mV input voltage is lower than the others because the output voltage saturates at 1.8 V, only allowing a maximum differential gain of 360 (51 dB)

was discovered. Once a Kingston 8 GB card was obtained, the system worked, and a constant stream of "AA" bytes were written to the card at 20 Mbps.

4.4 BCC Full System Test

After reviewing the results of the individual section tests and running some small fully integrated system tests, the BCC system was determined to be ready for its final data collection. The result of the small tests showed a drastic improvement to BER when the data transfer speed was 3 Mbps as opposed to 5 Mbps, so all tests are performed at 3 Mbps. This section will explain the setup of each test and present the results.

4.4.1 BCC BER Testing Procedure

BER was tested with the subject in three degrees of movement: resting, light movement (walking), and moderate movement (jogging). The receiver board and transmitter board were strapped to the body at each wrist. Both boards were isolated from

the body with the exception of one ground wire and one receive/transmit wire per board for a total of four wires on the body. These wires were attached to the body via 3M^{TM} Red Dot Monitoring Electrodes. Electrodes were placed approximately 1.5" apart. This setup can be seen in Figure 4-6.

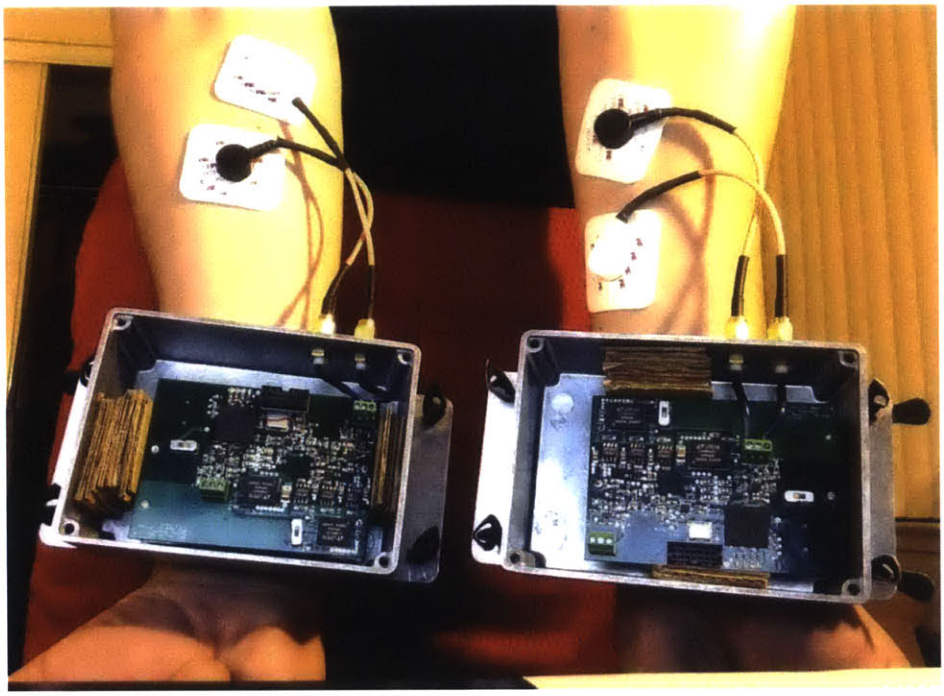

Figure 4-6: Setup for BCC system BER testing

Once the transmitter board was programmed to send a repetitious pattern that spanned all five frequencies, the subject engaged in one of the three specified levels of movement for approximately 3 minutes. At this time the receiver was programmed, and data was collected for 5 minutes. The test was stopped and data was analyzed from the SD memory card located on the receiver board to determine BER. This test was repeated for the three degrees of movement each three times for a total of 9 measurements.

4.4.2 BCC Wireless Security Testing Procedure

For the wireless security test, the transmitter board was connected to the body exactly as described in the BCC BER Testing Procedure. The receiver was strapped to the

body, but did not have its receive and ground wires attached to the body. Once programmed, the transmitter started sending the same data from the BER test, and after programming the receiver, data was collected for 5 minutes. Data was then analyzed from the SD memory card to determine if significant data is being transmitted via radio and not solely via body coupled communication. This test was repeated three times.

4.4.3 BCC BER and Wireless Security Results

The BER was measured in each of the 12 resulting test data sets. Each data set was 100 MB. The results are shown in Table 4.1

	Resting	**Light Movement**	**Moderate Movement**	**Wireless Security**
Trial 1	0.157	0.32	0.379	0.996
Trial 2	0.036	0.314	0.241	0.997
Trial 3	0.153	0.276	0.218	0.996
Average	0.115	0.303	0.279	0.996

Table 4.1: Final BCC system results for BER and wireless security

4.4.4 Summary

As seen in Table 4.1, the system clearly performs its best when the subject is resting, giving a BER of 0.115. Once the subject started moving, results averaged approximately the same between light and moderate movement, with moderate movement actually outperforming light movement at an average of 0.279 compared to 0.303. A possible explanation for this is the subject was swinging their arms when walking in light movement, but holding their arms in positions while running in moderate movement. With all three wireless security tests having less than 0.4% success rates, the test clearly suggests that the data in the BER tests was actually received through BCC and not through the air. To compare this system with previous works, a 172 kbps system made of commercially available parts performed at 1e-4 BER, beating this system's best BER by two orders of magnitude [1]. However, that system also

performed an order of magnitude slower. There do not appear to be any other readily available BCC systems made of only commercially available parts that operate at this thesis' speed, so it is difficult to make a direct comparison to another system. However, the fact that another system consisting of a custom IC layout could run at 2 Mbps with a BER of 1e-7 [8] suggests that this system is not competitive. Though not particularly competitive, the system still displayed the ability to transmit information at 3 Mbps across the human body via BCC with some success.

Chapter 5

Conclusions

A high speed wearable system for body coupled communication has been designed, implemented, and tested. This final chapter will discuss the results and what they predict for future success of this system. Ideas for future improvements to the system will be suggested, and the thesis will conclude with a brief summary.

5.1 Discussion

The BCC receiver and transmitter system designed in this thesis could not be used as an alternative to pre-existing wireless technology in the medical field. A communication link with an average BER with the subject at rest of 0.11 is not reliable enough for a professional setting. Though not yet operating at a professional standard, successful data was transmitted and received, so there is a significant possibility for success in the future. Throughout the design, implementation, and testing of this thesis, many practical lessons were learned, and a list of significant future improvements to the board has been compiled that could result in a much higher performing high speed, wearable BCC system.

Most important to the future success of this project is the ability to isolate as many individual components as possible and test them. Arguably the largest fault in the design of this system's PCB was the lack of test points and broken-out FPGA pins. With the exception of the third amplifier and ADC outputs, the current analog

system could not be fully isolated and tested. Every other signal node was so sensitive to stray additional capacitance that the addition of a probe of any type ruined the performance and prevented an accurate measurement and diagnosis. For example, the ability to measure the output of the third amplifier only gave the gain and bandwidth of the system. This demonstrated that the system gains were greater than 64 dB for all frequencies up to 100 MHz, meaning that any signal down to 100 μV should be amplified to a voltage swing greater than the 100 mV hysteresis band. However, it was also observed that there was some lower frequency content that started to interfere with the transmitted signal for input amplitudes of less than 250 μV. This evidence suggests there is interference generated from somewhere on the board. Since the only way to find where this interference is generated is to track it down via probing the system, the lack of locations able to be probed without causing instability to the system becomes an issue. To remedy this situation, high speed buffers can be placed on the nodes of interest for each component of the analog front end. Measuring the output of these buffers would not affect the actual signal, and could lead directly to the source(s) of interference. A similar case can be made for the layout of the FPGA. It would have been a significant time saver to have more broken-out pins to export test signals to. When laying out this system's PCB, four output test pins seemed like enough, but were quickly discovered to be about 10 too few.

Another set of changes need to be made for adequate diagnosis of the system's final data output. Currently, data is collected and written directly to the microSD card with no extra information of the test environment and situation. This issue was realized when the moderate exercise testing, on average, outperformed the light exercise testing. If the board had some sort of movement detector, such as an accelerometer, and a method of time stamping data, results could be compared with how much the actual boards were moving. This could be used to explain the sporadic nature of the collected data. For mega-bytes at a given time, the system would deliver results with a BER of less than 0.001, then produce another set of mega-bytes of clearly random data. Diagnosing this behavior is necessary to prevent or predict the random data collection cases. Related to this is the reduction in overall system size.

As mentioned, the PCB was much larger than needed. With some smarter layout, removing the battery pack, and utilizing the back of the board, this system could fit in a much smaller form factor. This would improve the mechanical connection to the body, greatly reducing the movement of the system when attached to the body, and removing possible motion artifacts.

Improvement to the power management is another topic worth mentioning. This system technically outperformed the specifications by greater than a factor of two. However, this was only because two, large, energy-dense batteries were used. The system consumes 600 mW, which is not low-power. Two changes to improve the power consumption are to replace the differential amplifiers with much lower power and higher input referred noise amplifiers. There are plenty of commercially available amplifiers that work in the 60 MHz to 100 MHz range, but almost every one of them was eliminated as a consideration because less than 0.69 $nV/\sqrt{Hz}$ was required. This is definitely still necessary for the first amplification stage, but the second and third amplifier stages could support an extra one and two orders of magnitude higher input referred noise. Relaxing the noise specification on the second and third amplifiers would greatly increase the number of available high speed amplifiers, including many low power options. With less power being consumed, the large Tenergy 3.7 V 18605 batteries could be replaced with a smaller battery of lower form factor. At the very least, the second Tenergy battery could be eliminated from the system. The voltage droop issue could be resolved with a buck-boost converter that simply regulates the battery voltage to 3.7 V regardless of droop. This voltage can then be linearly regulated to the lower voltages.

Finally, what the author believes to be the best system revision is to move the frequency detection responsibilities from the digital back end to the analog front end. The current digital frequency detection scheme worked, but only marginally. To minimize complexity, the algorithm was implemented in such a way to reduce the number of multiplications to zero. The problem with this method is that it only selects for one specific signal input, so its susceptibility to interference is immense. If *any* additional signals or noise find their way onto the transmitted signal and are

not completely removed by the time the signal is read into the FPGA, the digital frequency detection's performance is greatly compromised. This is not a desirable quality in this kind of frequency detection scheme. The ideal system would be one that can accept a noisy signal, full of interference, and still report whether the signal has energy at a specific frequency. This is why mixing the input signal to baseband is such a popular method of frequency detection. An 80 MHz signal could be filled with interference at other frequencies, but if it is mixed down, it can easily be low pass filtered and measured. If the system in this thesis were revised, this method of frequency detection would be heavily considered.

5.2 Summary

Finding a wireless, wearable, low-power, high speed, secure, and reliable data transfer scheme for communication with implanted devices and other monitoring devices would provide a great service to the medical field. Implanted sensors could last longer on a single battery charge, collect more data, and sample higher frequency content at good resolutions. A secure wireless transfer scheme would prevent interference between nearby patients using the same devices, without the need of a software intensive bandwidth sharing scheme. These improvements appear possible with systems that utilize body coupled communications. In this thesis, a wearable, high speed BCC receiver and transmitter was developed using only commercially available components. It can be worn by a patient for up to 19 hours on a single charge, and transmit data across the body at 3 Mbps. Testing suggests that this implementation does not perform at a professional communications standard, but it does, however, suggest the feasibility of a wearable, high speed, secure BCC system without the need for a custom IC.

Appendix A

Frequency Detection Code

```
module BCC_Rx_Top(
    input CLK_in,
    input DO,
    input SIGGY_P,
    input SIGGY_N,
    output SCLK,
    output DI,
    output CS,
    output [3:0] LED,
    );

  //DECLARE VARIABLES
  IBUFDS IBUFDS_inst (
    .O(SIGGY), // Clock buffer output
    .I(SIGGY_P), // Diff_p clock buffer input
    .IB(SIGGY_N) // Diff_n clock buffer input
      );
  reg SIGD1 = 0;
  reg SIGD2 = 0;

  reg [5:0] counter = 0;
  reg [63:0]regbuff = 0;

  reg [63:0]f60array;
  reg [63:0]f70array;
  reg [63:0]f80array;
  reg [63:0]f90array;
  reg [63:0]f100array;

  reg [63:0]f60arr;
```

```
31      reg [63:0]f70arr;
        reg [63:0]f80arr;
33      reg [63:0]f90arr;
        reg [63:0]f100arr;
35
        wire f60coeff;
37      wire f70coeff;
        wire f80coeff;
39      wire f90coeff;
        wire f100coeff;
41
        wire [5:0] f60arraySum;
43      wire [5:0] f70arraySum;
        wire [5:0] f80arraySum;
45      wire [5:0] f90arraySum;
        wire [5:0] f100arraySum;
47
        integer x;
49
        reg [23:0] Tcounter = 0;
51      reg [2:0] Tstate = 0;

53        always @(posedge FAST_CLK) begin

55         counter <= counter + 6'd1;

57         //LOAD FREQUENCY BANKS
           f60arr[counter]  <= f60coeff;
59         f70arr[counter]  <= f70coeff;
           f80arr[counter]  <= f80coeff;
61         f90arr[counter] <= f90coeff;
           f100arr[counter] <= f100coeff;
63
           //XNORs
65         for (x=0; x<64; x = x+1) begin
            f60array[x] <= ~(f60arr[x] ^ regbuff[63-x]);
67          f70array[x] <= ~(f70arr[x] ^ regbuff[63-x]);
            f80array[x] <= ~(f80arr[x] ^ regbuff[63-x]);
69          f90array[x] <= ~(f90arr[x] ^ regbuff[63-x]);
            f100array[x] <= ~(f100arr[x] ^ regbuff[63-x]);
71         end
          // DFFs for METASTABILITY
73        SIGD1 <= SIGGY;
          SIGD2 <= SIGD1;
75        regbuff[0] <= SIGD2;
```

```
77      //LOAD CIRCULAR BUFFER
        for (x=1; x<64; x = x+1) begin
79      regbuff[x] <= regbuff[x-1];
         end
81
        end
83
    CLK_MULT clockGen(.CLK_IN1(CLK_in), .CLK_OUT1(FAST_CLK), .CLK_OUT2(CLK));
85
    fir60MHzcoeffs f60(.index({1'b0,counter}), .coeff(f60coeff));
87  fir70MHzcoeffs f70(.index({1'b0,counter}), .coeff(f70coeff));
    fir80MHzcoeffs f80(.index({1'b0,counter}), .coeff(f80coeff));
89  fir900MHzcoeffs f90(.index({1'b0,counter}), .coeff(f90coeff));
    fir100MHzcoeffs f100(.index({1'b0,counter}), .coeff(f100coeff));
91
    Array_Sum #(.ArrayLength(64),.Bits(1)) f60sum(.clk(FAST_CLK),
93      .inArray(f60array), .SummedArray(f60arraySum));
    Array_Sum #(.ArrayLength(64),.Bits(1)) f70sum(.clk(FAST_CLK),
95      .inArray(f70array), .SummedArray(f70arraySum));
    Array_Sum #(.ArrayLength(64),.Bits(1)) f80sum(.clk(FAST_CLK),
97      .inArray(f80array), .SummedArray(f80arraySum));
    Array_Sum #(.ArrayLength(64),.Bits(1)) f90sum(.clk(FAST_CLK),
99      .inArray(f90array), .SummedArray(f90arraySum));
    Array_Sum #(.ArrayLength(64),.Bits(1)) f100sum(.clk(FAST_CLK),
101     .inArray(f100array), .SummedArray(f100arraySum));

103 // RECEIVE DATA TRIGGERING
    wire [4:0]TestOut;
105 wire [2:0]preFreq;
    reg [2:0]preFreqD;
107 reg [10:0] WRITE_cntREG = 0;
    wire [10:0] WRITE_cnt = WRITE_cntREG;
109 wire [1:0]DataIn;
    reg BufSelectREG = 0;
111 wire BufSelect = BufSelectREG;

113 // LOGIC FOR SWITCHING BUFFERS
    always @(posedge FAST_CLK) begin
115   preFreqD <= preFreq;
      if (preFreqD != preFreq) begin
117     WRITE_cntREG <= WRITE_cntREG + 11'd1;
        if (WRITE_cntREG == 11'd2047) BufSelectREG <= ~BufSelectREG;
119     end
      end
```

```
121
        wire [5:0]THRESH = 6'd50;
123     freqLogic fLog(.TestOut(TestOut), .preFreqOut(preFreq), .clk(FAST_CLK),
          .threshold1(6'd50), .threshold2(6'd52),.threshold3(6'd50),
125       .threshold4(6'd50),.threshold5(6'd48),.freq1(f60arraySum),
          .freq2(f70arraySum), .freq3(f80arraySum), .freq4(f90arraySum),
127       .freq5(f100arraySum), .final_out(DataIn));

129     // DOUBLE BUFFER BRAMs
        wire [1:0]DataOut1;
131     wire [1:0]DataOut2;
        wire [11:0] READ_cnt;
133
        BRAM2buff DataOutBRAM1(.addra((BufSelect) ? READ_cnt[11:1] : WRITE_cnt),
135         .dina(DataIn),.wea(~BufSelect),.clka(FAST_CLK),.douta(DataOut1));

137     BRAM2buff DataOutBRAM2(.addra((~BufSelect) ? READ_cnt[11:1] : WRITE_cnt),
            .dina(DataIn),.wea(BufSelect),.clka(FAST_CLK),.douta(DataOut2));
139
        wire [4:0]SD_state;
141
        SD_SPI  SD_card(.CLK_in(CLK),
143             .DO(DO),
                .READ_cntW(READ_cnt),
145             .BufSelect(BufSelect),
                .DataOut1(DataOut1),
147             .DataOut2(DataOut2),
                .SCLK(SCLK),
149             .DI(DI),
                .CS(CS),
151             .SD_state(SD_state)
                );
153
    endmodule
155
    //MODULE FOR SUMMING OUTPUT OF XNORs
157 //PIPELINED INTO 16 BIT SECTIONS
    module Array_Sum #(parameter ArrayLength = 128, Bits = 1)
159             (input clk,
                input [Bits*ArrayLength-1:0] inArray,
161             output reg [5:0] SummedArray);

163     reg [5:0]SummedArray1a;
        reg [5:0]SummedArray1b;
165     reg [5:0]SummedArray1c;
```

```
    reg [5:0]SummedArray1d;
167
    reg [5:0]SummedArray2a;
169 reg [5:0]SummedArray2b;

171   always @(posedge clk) begin
      SummedArray1a[5:0] <= inArray[ArrayLength-1] +
173       inArray[ArrayLength-2] +
          inArray[ArrayLength-3] +
175       inArray[ArrayLength-4] +
          inArray[ArrayLength-5] +
177       inArray[ArrayLength-6] +
          inArray[ArrayLength-7] +
179       inArray[ArrayLength-8] +
          inArray[ArrayLength-9] +
181       inArray[ArrayLength-10] +
          inArray[ArrayLength-11] +
183       inArray[ArrayLength-12] +
          inArray[ArrayLength-13] +
185       inArray[ArrayLength-14] +
          inArray[ArrayLength-15] +
187       inArray[ArrayLength-16];

189   SummedArray1b[5:0] <= inArray[ArrayLength-17] +
          inArray[ArrayLength-18] +
191       inArray[ArrayLength-19] +
          inArray[ArrayLength-20] +
193       inArray[ArrayLength-21] +
          inArray[ArrayLength-22] +
195       inArray[ArrayLength-23] +
          inArray[ArrayLength-24] +
197       inArray[ArrayLength-25] +
          inArray[ArrayLength-26] +
199       inArray[ArrayLength-27] +
          inArray[ArrayLength-28] +
201       inArray[ArrayLength-29] +
          inArray[ArrayLength-30] +
203       inArray[ArrayLength-31] +
          inArray[ArrayLength-32];
205
      SummedArray1c[5:0] <= inArray[ArrayLength-33] +
207       inArray[ArrayLength-34] +
          inArray[ArrayLength-35] +
209       inArray[ArrayLength-36] +
          inArray[ArrayLength-37] +
```

```
211             inArray[ArrayLength-38] +
                inArray[ArrayLength-39] +
213             inArray[ArrayLength-40] +
                inArray[ArrayLength-41] +
215             inArray[ArrayLength-42] +
                inArray[ArrayLength-43] +
217             inArray[ArrayLength-44] +
                inArray[ArrayLength-45] +
219             inArray[ArrayLength-46] +
                inArray[ArrayLength-47] +
221             inArray[ArrayLength-48];

223         SummedArray1d[5:0] <= inArray[ArrayLength-49] +
                inArray[ArrayLength-50] +
225             inArray[ArrayLength-51] +
                inArray[ArrayLength-52] +
227             inArray[ArrayLength-53] +
                inArray[ArrayLength-54] +
229             inArray[ArrayLength-55] +
                inArray[ArrayLength-56] +
231             inArray[ArrayLength-57] +
                inArray[ArrayLength-58] +
233             inArray[ArrayLength-59] +
                inArray[ArrayLength-60] +
235             inArray[ArrayLength-61] +
                inArray[ArrayLength-62] +
237             inArray[ArrayLength-63] +
                inArray[ArrayLength-64];
239
            SummedArray2a[5:0] <= SummedArray1a + SummedArray1b;
241
            SummedArray2b[5:0] <= SummedArray1c + SummedArray1d;
243
            SummedArray[5:0] <= SummedArray2a + SummedArray2b;
245          end
        endmodule
247
        //MODULE FOR FREQUENCY TO BITS CALCULATION
249     module freqLogic(input clk,
                input [5:0]threshold1,
251             input [5:0]threshold2,
                input [5:0]threshold3,
253             input [5:0]threshold4,
                input [5:0]threshold5,
255             input [5:0]freq1,
```

```
        input [5:0]freq2,
257     input [5:0]freq3,
        input [5:0]freq4,
259     input [5:0]freq5,
        output [1:0]final_out,
261     output [4:0] TestOut,
        output [2:0] preFreqOut
263     );

265   reg [3:0]bits_out=0;
      reg  [2:0]preFreq = 0;
267   assign preFreqOut = preFreq;

269   //DETERMINE IF OUTPUT OF CONVOLUTION IS GREATER THAN THRESHOLD
      assign TestOut =
271     { (~freq1[5]?(freq1<= (7'd63 - threshold1)):(freq1>=threshold1)),
      (~freq2[5]? (freq2<= (7'd63 - threshold2)):(freq2>=threshold2)),
273   (~freq3[5]? (freq3<= (7'd63 - threshold3)):(freq3>=threshold3)),
      (~freq4[5]? (freq4<= (7'd63 - threshold4)):(freq4>=threshold4)),
275   (~freq5[5]? (freq5<= (7'd63 - threshold5)):(freq5>=threshold5))
      };
277

    //LOGIC TO COMPARE PREVIOUS FREQUENCY TO NEW FREQUENCY
279   always@(posedge clk)begin
       casex(TestOut)
281      5'b1XXXX:   begin
         preFreq  <= 3'd0;
283      if (preFreq > 3'd0) begin
           bits_out <= 3'd4 - preFreq;
285        preFreq  <= 3'd0;
           end
287      end
         5'b01XXX:   begin
289      preFreq  <= 3'd1;
         if (preFreq > 3'd1) begin
291        bits_out <= 3'd5 - preFreq;
           preFreq  <= 3'd1;
293        end
         else if (preFreq < 3'd1) begin
295        bits_out <= 3'd0 - preFreq;
           preFreq  <= 3'd1;
297        end
         end
299      5'b001XX:   begin
         preFreq  <= 3'd2;
```

```
301         if (preFreq > 3'd2) begin
              bits_out <= 3'd6 - preFreq;
303           preFreq  <= 3'd2;
              end
305         else if (preFreq < 3'd2) begin
              bits_out <= 3'd1 - preFreq;
307           preFreq  <= 3'd2;
              end
309         end
            5'b0001X:    begin
311         preFreq  <= 3'd3;
            if (preFreq > 3'd3) begin
313           bits_out <= 3'd7 - preFreq;
              preFreq  <= 3'd3;
315           end
            else if (preFreq < 3'd3) begin
317           bits_out <= 3'd2 - preFreq;
              preFreq  <= 3'd3;
319           end
            end
321         5'b00001:    begin
            preFreq  <= 3'd4;
323         if (preFreq < 3'd4) begin
              bits_out <= 3'd3 - preFreq;
325           preFreq  <= 3'd4;
              end
327         end
            5'b00000: preFreq <= preFreq;
329         default:         preFreq <= 0;
            endcase
331         end

333         assign final_out = bits_out[1:0];
    endmodule
```

Appendix B

MicroSD Initialization and Data Transfer Code

```
1  module SD_SPI(
     input CLK_in,
3    input DO,
     input BufSelect,
5    input [1:0]DataOut1,
     input [1:0]DataOut2,
7    output SCLK,
     output [11:0]READ_cntW,
9    output DI,
     output CS,
11   output reg [4:0]SD_state
     );
13
     reg [11:0]READ_cnt = 0;
15   assign READ_cntW = READ_cnt;

17   reg CLK_200k = 0;
     reg CLK_5M = 0;
19   reg CLK_10M = 0;
     reg [5:0] clk_counter = 0;
21   reg [0:0] FM_counter = 0;
     //reg [4:0] SD_state = 0;
23   reg ChipSel = 0;
     reg DataOut = 0;
25   reg [10:0] IDLE_cnt = 0;
     reg [12:0] BIT_cnt = 0;
27   reg [7:0] BYTE_rec;
```

```
    reg [39:0] R7_rec;
29  reg D_Tx = 0;
    reg BufSelectD = 0;
31
    reg [31:0]SentBlocks = 0; //Holds SD card address
33
    //Initialization commands
35  wire [47:0] CMD0;
    assign CMD0 = {8'h40,32'd0,8'h95};
37  reg [47:0] CMD0reg;

39  wire [47:0] CMD8;
    assign CMD8 = {8'h48,16'd0,8'd1,8'hAA,8'h0F};
41  reg [47:0] CMD8reg;

43  wire [47:0] CMD58;
    assign CMD58 = {8'h7A,32'd0,8'h00};
45  reg [47:0] CMD58reg;

47  wire [47:0] CMD55;
    assign CMD55 = {8'h77,32'd0,8'h00};
49  reg [47:0] CMD55reg;

51  wire [47:0] ACMD41;
    assign ACMD41 = {8'h69,32'h40000000,8'h00};
53  reg [47:0] ACMD41reg;

55  wire [47:0] CMD25;
    assign CMD25 = {8'h59,32'h00000000,8'h00};
57  reg [47:0] CMD25reg;

59  wire [7:0] sTOKEN;
    assign sTOKEN = {8'b11111100};
61  reg [7:0] sTOKENreg;

63  //Output assignments
    assign CS = ChipSel;
65  assign DI = DataOut;

67  //SCLK is 200 kHz for initialization, then 10 MHz for data transfer
    assign SCLK = (D_Tx) ? CLK_5M : CLK_200k;
69

71  //CLK DIVIDER: Need 200 kHz for SD initialization
    always @(posedge CLK_in) begin
```

```
73          FM_counter <= FM_counter + 1'd1;
            clk_counter <= clk_counter + 5'd1;
75          if (clk_counter == 0) CLK_200k <= ~CLK_200k;
            CLK_5M <= ~CLK_5M;
77          end

79      parameter IDLE0 = 5'd0;
        parameter IDLE1 = 5'd1;
81
        parameter CMD0init = 5'd2;
83      parameter CMD0send = 5'd3;
        parameter CMD0recWAIT = 5'd4;
85      parameter CMD0rec = 5'd5;

87      parameter CMD8init = 5'd6;
        parameter CMD8send = 5'd7;
89      parameter CMD8recWAIT = 5'd8;
        parameter CMD8rec = 5'd9;
91
        parameter CMD55init = 5'd10;
93      parameter CMD55send = 5'd11;
        parameter CMD55recWAIT = 5'd12;
95      parameter CMD55rec = 5'd13;

97      parameter ACMD41init = 5'd14;
        parameter ACMD41send = 5'd15;
99      parameter ACMD41recWAIT = 5'd16;
        parameter ACMD41rec = 5'd17;
101
        parameter CMD58init = 5'd18;
103     parameter CMD58send = 5'd19;
        parameter CMD58recWAIT = 5'd20;
105     parameter CMD58rec = 5'd21;

107     parameter CMD25init = 5'd22;
        parameter CMD25send = 5'd23;
109     parameter CMD25recWAIT = 5'd24;
        parameter CMD25rec = 5'd25;
111
        parameter recBUSY = 5'd26;
113     parameter SendTOKENinit = 5'd27;
        parameter SendTOKENsend = 5'd28;
115     parameter SendDATA = 5'd29;
        parameter recTOKENrecWAIT = 5'd30;
117     parameter recTOKENrec = 5'd31;
```

```
119

121     //SD Card State Machine
        always @(posedge SCLK) begin
123       case(SD_state)
            //Count to > 1ms
125         IDLE0: begin
              IDLE_cnt <= IDLE_cnt + 10'd1;
127           if (IDLE_cnt == 11'b11111111111) SD_state <= IDLE1;
              end
129         //Set CS and DI high for 80 clock cycles
            IDLE1: begin
131           if (IDLE_cnt < 10'd80) begin
                ChipSel <= 1'b1;
133             DataOut <= 1'b1;
                IDLE_cnt <= IDLE_cnt + 10'd1; end
135           else begin
                ChipSel <= 0;
137             SD_state <= CMD0init; end
              end
139         //Initialize CMD0
            CMD0init: begin
141           CMD0reg <= CMD0;
              SD_state <= CMD0send;
143           end
            //Send CMD0
145         CMD0send: begin
              DataOut <= CMD0reg[47];
147           CMD0reg <= {CMD0reg[46:0], 1'b1};
              if (BIT_cnt <= 10'd46) BIT_cnt <= BIT_cnt + 10'd1;
149           else begin
                BIT_cnt <= 0;
151             SD_state <= CMD0recWAIT; end
              end
153         //Wait for a 0 on DO line
            CMD0recWAIT: begin
155           DataOut <= 1'b1;
              if (DO == 0) begin
157             BIT_cnt <= BIT_cnt + 10'd1;
                BYTE_rec <= {BYTE_rec[6:0],DO};
159             SD_state <= CMD0rec; end
              end
161         CMD0rec: begin
              BYTE_rec <= {BYTE_rec[6:0],DO};
```

```
163             if (BIT_cnt <= 10'd6) BIT_cnt <= BIT_cnt + 10'd1;
                else begin
165               BIT_cnt <= 0;
                  SD_state <= CMD8init; end
167             end
              //Initialize CMD8
169           CMD8init: begin
                CMD8reg <= CMD8;
171             SD_state <= CMD8send;
                end
173           //Send CMD8
              CMD8send: begin
175             DataOut <= CMD8reg[47];
                CMD8reg <= {CMD8reg[46:0], 1'b1};
177             if (BIT_cnt <= 10'd46) BIT_cnt <= BIT_cnt + 10'd1;
                else begin
179               BIT_cnt <= 0;
                  SD_state <= CMD8recWAIT; end
181             end
              //Wait for a 0 on DO line
183           CMD8recWAIT: begin
                DataOut <= 1'b1;
185             if (DO == 0) begin
                  BIT_cnt <= BIT_cnt + 10'd1;
187               R7_rec <= {R7_rec[38:0],DO};
                  SD_state <= CMD8rec; end
189             end
              CMD8rec: begin
191             R7_rec <= {R7_rec[38:0],DO};
                if (BIT_cnt <= 10'd38) BIT_cnt <= BIT_cnt + 10'd1;
193             else begin
                  BIT_cnt <= 0;
195               SD_state <= CMD55init; end
                end
197           //Initialize CMD55
              CMD55init: begin
199             CMD55reg <= CMD55;
                SD_state <= CMD55send;
201             end
              //Send CMD55
203           CMD55send: begin
                DataOut <= CMD55reg[47];
205             CMD55reg <= {CMD55reg[46:0], 1'b1};
                if (BIT_cnt <= 10'd46) BIT_cnt <= BIT_cnt + 10'd1;
207             else begin
```

```
            BIT_cnt <= 0;
209         SD_state <= CMD55recWAIT; end
          end
211       //Wait for a 0 on DO line
          CMD55recWAIT: begin
213         DataOut <= 1'b1;
            if (DO == 0) begin
215           BIT_cnt <= BIT_cnt + 10'd1;
              BYTE_rec <= {BYTE_rec[6:0],DO};
217           SD_state <= CMD55rec; end
            end
219       CMD55rec: begin
            BYTE_rec <= {BYTE_rec[6:0],DO};
221         if (BIT_cnt <= 10'd6) BIT_cnt <= BIT_cnt + 10'd1;
            else begin
223           BIT_cnt <= 0;
              SD_state <= ACMD41init; end
225         end
          //Initialize ACMD41
227       ACMD41init: begin
            ACMD41reg <= ACMD41;
229         SD_state <= ACMD41send;
            end
231       //Send ACMD41
          ACMD41send: begin
233         DataOut <= ACMD41reg[47];
            ACMD41reg <= {ACMD41reg[46:0], 1'b1};
235         if (BIT_cnt <= 10'd46) BIT_cnt <= BIT_cnt + 10'd1;
            else begin
237           BIT_cnt <= 0;
              SD_state <= ACMD41recWAIT; end
239         end
          //Wait for a 0 on DO line
241       ACMD41recWAIT: begin
            DataOut <= 1'b1;
243         if (DO == 0) begin
              BIT_cnt <= BIT_cnt + 10'd1;
245           BYTE_rec <= {BYTE_rec[6:0],DO};
              SD_state <= ACMD41rec; end
247         end
          ACMD41rec: begin
249         BYTE_rec <= {BYTE_rec[6:0],DO};
            if (BIT_cnt <= 10'd7) BIT_cnt <= BIT_cnt + 10'd1;
251         else if (BYTE_rec[0] == 1'b1) begin
              BIT_cnt <= 0;
```

```
253             SD_state <= CMD55init; end
                else begin
255             BIT_cnt <= 0;
                SD_state <= CMD58init; end
257             end
              //Initialize CMD58
259           CMD58init: begin
                CMD58reg <= CMD58;
261             SD_state <= CMD58send;
                end
263           //Send CMD58
              CMD58send: begin
265             DataOut <= CMD58reg[47];
                CMD58reg <= {CMD58reg[46:0], 1'b1};
267             if (BIT_cnt <= 10'd46) BIT_cnt <= BIT_cnt + 10'd1;
                else begin
269             BIT_cnt <= 0;
                SD_state <= CMD58recWAIT; end
271             end
              //Wait for a 0 on DO line
273           CMD58recWAIT: begin
                DataOut <= 1'b1;
275             if (DO == 0) begin
                BIT_cnt <= BIT_cnt + 10'd1;
277             R7_rec <= {R7_rec[38:0],DO};
                SD_state <= CMD58rec; end
279             end
              CMD58rec: begin
281             R7_rec <= {R7_rec[38:0],DO};
                if (BIT_cnt <= 10'd38) BIT_cnt <= BIT_cnt + 10'd1;
283             else begin
                BIT_cnt <= 0;
285             D_Tx <= 1'b1; //Out of IDLE, increase CLK to 10MHz
                SD_state <= CMD25init; end
287             end
              //Initialize CMD25
289           CMD25init: begin
                if (BIT_cnt < 100) BIT_cnt <= BIT_cnt + 10'd1;
291             else if (DO) begin
                CMD25reg <= CMD25 + {8'd0,SentBlocks,8'd0};
293             SD_state <= CMD25send;
                BIT_cnt <= 0; end
295             end
              //Send CMD25
297           CMD25send: begin
```

```
            DataOut <= CMD25reg[47];
299         CMD25reg <= {CMD25reg[46:0], 1'b1};
            if (BIT_cnt <= 10'd46) BIT_cnt <= BIT_cnt + 10'd1;
301         else begin
              BIT_cnt <= 0;
303           SD_state <= CMD25recWAIT; end
            end
305       //Wait for a 0 on DO line
          CMD25recWAIT: begin
307         DataOut <= 1'b1;
            if (DO == 0) begin
309           BIT_cnt <= BIT_cnt + 10'd1;
              BYTE_rec <= {BYTE_rec[6:0],DO};
311           SD_state <= CMD25rec; end
            end
313       CMD25rec: begin
            BYTE_rec <= {BYTE_rec[6:0],DO};
315         if (BIT_cnt <= 10'd6) BIT_cnt <= BIT_cnt + 10'd1;
            else begin
317           BIT_cnt <= 0;
              SD_state <= recBUSY; end
319         end
          recBUSY: begin
321         //To send data, send TOKEN
            if (DO==1 && (BufSelectD != BufSelect)) begin
323           SD_state <= SendTOKENinit;
              end
325         end
          SendTOKENinit: begin
327         sTOKENreg <= sTOKEN;
            SD_state <= SendTOKENsend;
329         end
          SendTOKENsend: begin
331         DataOut <= sTOKENreg[7];
            sTOKENreg <= {sTOKENreg[6:0], 1'b1};
333         if (BIT_cnt <= 10'd6) BIT_cnt <= BIT_cnt + 10'd1;
            else begin
335           BIT_cnt <= 0;
              SD_state <= SendDATA; end
337         end
          SendDATA: begin
339         if (READ_cnt[0] == 0)
              DataOut <= (BufSelect) ? DataOut1[1] : DataOut2[1];
341         else
              DataOut <= (BufSelect) ? DataOut1[0] : DataOut2[0];
```

```
343
            if (READ_cnt < 12'd4095) READ_cnt <= READ_cnt + 12'd1; //XX
345         else begin
              READ_cnt <= 0;
347           SD_state <= recTOKENrecWAIT;
              BufSelectD <= BufSelect; end  //STORES CURRENT BUFSELECT
349                   //TO COMPARE WITH NEW ONE LATER
            end          //ASSUMES RECEIVING SLOWER THAN SD WRITING
351       //Wait for a 0 on DO line
          recTOKENrecWAIT: begin
353         DataOut <= 1'b1;
            if (DO == 0) begin
355           BIT_cnt <= BIT_cnt + 10'd1;
              BYTE_rec <= {BYTE_rec[6:0],DO};
357           SD_state <= recTOKENrec; end
            end
359       recTOKENrec: begin
            BYTE_rec <= {BYTE_rec[6:0],DO};
361         if (BIT_cnt <= 10'd6) BIT_cnt <= BIT_cnt + 10'd1;
            else begin
363           BIT_cnt <= 0;
              SentBlocks <= SentBlocks + 32'd1;
365           SD_state <= recBUSY; end
            end
367       default: SD_state <= 0;
        endcase
369     end

371 endmodule
```

Bibliography

[1] G S. Anderson and C. Sodini. Body Coupled Communication: The Channel and Implantable Sensors. *2013 IEEE International Conference On Body Sensor Networks*, May 2013.

[2] T. G. Zimmerman. Personal Area Networks (PAN): Near-field Intrabody Communication. Master's thesis, MIT Media Laboratory, 1995.

[3] T. Schenk, N. Mazloum, and P. Rutten L. Tan. Experimental Characterization of the Body-Coupled Communications Channel. *2008 IEEE International Symposium on Wireless Communication Systems*, 2008.

[4] N. Cho, J. Yoo, S. Song, and J. Lee. The Human Body Characteristics as a Signal Transmission Medium for Intrabody Communication. *IEEE Transactions on Microwave Theory and Techniques*, May 2007.

[5] A. Fazzi, S. Ouzounov, and J. van den Homberg. A 2.75mw Wideband Correlation-Based Transceiver for Body-Coupled Communication. *IEEE International Sold-State Circuits Conference*, 2009.

[6] P. Harikumar, M. Kazim, and J. Wikner. An Analog Receiver Front-End for Capacitive Body-Coupled Communication. *NORCHIP*, 2012.

[7] N. Cho, J. Bae, and H. Yoo. An Interference-Resilient Body Channel Transceiver for Wearable Body Sensor Network. *Biomedical Circuits and Systems Conference*, 2008.

[8] S. Song, N. Cho, S. Kim, J. Yoo, and H. Yoo. A 2Mb/s Wideband Pulse Transceiver with Direct-Coupled Interface for Human Body Communications. *Solid-State Circuits Conference*, 2006.

[9] H. Maruf. An Input Amplifier for Body-Channel Communication. Master's thesis, Linkoping University, 2013.

[10] F. Victor, C. De Place, C. Camus, H. Le Breton, C. Leclercq, D. Pavin, P. Mabo, and C. Daubert. Pacemaker Lead Infection: Echocardiographic Features, Management, and Outcome. *Heart*, 1999.

[11] http://www.xilinx.com/products/design_tools/logic_design/xpe.htm.

[12] http://elm chan.org/docs/mmc/mmc_e.html.

[13] https://www.sdcard.org/downloads/pls/.

[14] http://www.hexworkshop.com/.