Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project

Problem Definition and Requirements Review

dDOSI Spectrum Analysis Unit (dSAU)

Submitted to

Darren Roblyer
44 Cummington Mall
Boston, MA 02118
617-358-1554
roblyer@bu.edu

by

dDOSI

Team 19
Team dDOSI

Team Members

Caroline Ekchian cekchian@bu.edu
Benjamin Havey benhavey@bu.edu
Andy Mo andy2m11@bu.edu
Thomas Nadovich tnadov@bu.edu
Christopher Woodall cwoodall@bu.edu

Submitted: December 9, 2013

Table of Contents

Executive Summary

dDOSI Spectrum Analysis Unit (previously Ultra Fast ADC)
Team 19 – Team dDOSI

The digital diffuse optical spectroscopic imaging (dDOSI) Spectrum Analysis Unit (SAU) is a device being developed for Professor Darren Roblyer and will aid in his research in the use of diffuse optical spectroscopic imaging for monitoring the effects of chemotherapy on cancer cells. Professor Roblyer wishes to use near-infrared lasers to measure the scattering and absorption properties of a media. This can be used as an imaging mode to find concentrations of lipids, hemoglobin and water [1]. We must produce a system that generates and measures six reference waveforms and stores the data in a user accessible manner.

Our technical approach includes two boards: a motherboard and a signal generator board. The motherboard will contain an ADC and a system on chip (SoC) to sample the returning waveform and send it to the user's computer. The signal generator board will contain 6 direct digital synthesizer (DDS) chips to produce the reference waveforms.

Due to the prohibitive costs of ADCs which are capable of sampling at Nyquist for the rates desired by our customer (1 GSPS), we are utilizing a method called undersampling. This allows us to use a much less expensive ADC with a minimal decrease in the accuracy of our data.

In summary, our final deliverable will be comprised of two hardware PCBs, firmware and software running on the processing system on the motherboard, an Ethernet protocol to the host PC, software and libraries to be run on the host PC, a power delivery system, and an enclosure to house the hardware. It will also include all documentation necessary for further use and development of the system.

1.0 Introduction

The customer, Professor Roblyer, needs a fully functional digital diffuse optical spectroscopy (dDOSI) imaging setup to advance his research into the applications of diffuse optics to breast cancer research. Professor Roblyer uses near-infrared lasers to measure the scattering and absorption properties of a media. This can be used as an imaging mode to find concentrations of lipids, hemoglobin and water [1]. An example of this technology can be seen in *Figure A3-1*. His current setup is very slow and the system itself is bulky. There is an older setup which utilized a network analyzer, but such equipment is large and expensive and provides a limit to its use in clinical trials. Our new setup should be able to take a sweep in less than 100ms with each step in the sweep having a sufficient number of samples to solve the diffusion equation and calculate the biochemical properties in the tissue.

Our device must be able to simultaneously create six frequency sweeps that will modulate external laser drivers through user specified frequency ranges between 50MHz and 500MHz. Secondly, the dDOSI Frequency Analyzer must also be able to measure two waveforms, one from the aforementioned output and the other received from an external amplified photodiode that will measure the light scattered off the tissue from the modulated laser at a point. Furthermore, the device must communicate this waveform information to a PC running custom software for recording and visualizing the data. This software must also be able to set frequency sweeps and otherwise control the analyzer.

In order to meet these goals we will design custom printed circuit board(s) (PCB) with six direct digital synthesizer (DDS) chips for producing the output waveforms, a two channel 14-bit 250MSPS ADC for measuring the inputs and a MicroZed SoC with custom firmware for controlling these devices and communicating with the PC. We will also be writing the control and data logging software that will run on the PC.

To sample the signals above the Nyquist rate we would require an ADC capable of sampling at 1GSPS. We found multiple ADCs capable of this, but such ADCs are prohibitively expensive. To overcome this price problem, and the constraint that we must implement a direct digital sampling methodology, we have decided to undersample [2]. At each step in the frequency sweep we will have a reference channel and know where the frequency is supposed to be. As a result, we will be able to resolve the amplitude degradation and phase shift between the reference waveform and the returned waveform from the sample, even if we are not sampling above Nyquist.

Once the data is on the computer we must package it in a way that can be easily analysed. The analysis algorithms are outside of the scope of this project, but our ability to generate data which can be easily brought into an analysis suite, such as MATLAB, is vital for the future use and utility of this project.

On the host computer a graphical user interface (GUI) will allow the user to control the board runs. The GUI must be user friendly and be able to display some sort of graphical information about the recent runs for debugging purposes.

Aside from the immediate results for our customer's research, there are also many real world applications of our project and his research. The dDOSI project can improve detection, monitoring and treatment of cancer, such as breast cancer. With the ability to take frequent, and safe measurements from human tissue comes the ability to monitor the response of the body to various cancer treatments as they are happening rather than the traditional before and after

measurements which tend to be taken with the more traditional imaging modes. Diffuse optical spectroscopic imaging is not meant to replace the traditional imaging modalities, but rather make new information available to aid in the treatment and diagnosis process.

2.0 Concept Development

There are a lot of considerations to be taken in a project of this size. Most of the hardware specifications were decided directly by our customer. The minimum requirements for our system are to produce and read a frequency sweep from 50MHz to 500MHz with a 1MHz step size (450 steps) with a maximum of 64kSamples/step.

A frequency sweep is constructed from a series of steps. A step is the amount that a frequency increases in a sweep. The sample size is the amount of data that is taken at every step. With a 1MHz step size, 4kSamples/step, and a frequency range of 50MHz to 500MHz, a run would go as follows: 4000 samples would be read from the ADC while a 50MHz signal was produced by the DDS, than 4000 samples would be read while there was a 51MHz signal, followed by a 52, 53, and 54MHz signals. The run would stop after 4000 samples were read at 500MHz. In this example Step 1 occurred at 50MHz, Step 2 occured at 51MHz, etc. A sweep includes all of the steps between the two frequencies. The speed at which samples are taken is determined by the sample rate of the ADC.

Our customer has requested be able to complete a run within 1 second. We strove to choose hardware that could meet these needs. After having a discussion with him we collectively agreed that gigabit ethernet was the best communication choice between the board and the host computer due to its ubiquity, as well as it's ease to port.

Having chosen Ethernet, we now had a limiting factor of 1Gbps. This gave us a concrete upper limit for our project. As we would see afterwards, this is the largest bottleneck in our project. This is not an issue, as using any type of accessible and cost effective communication method always leaves this as a bottleneck. *Figure A3-2* shows hypothetical run speeds for different types of run speeds. These results use a 50% gigabit ethernet throughput, but as of right now we've achieved closer to 70% throughput without optimization (see Section 4.4). This means that our final run speeds should be faster.

Once we had the ethernet chosen it was time to select an ADC. At first we were planning to use the ADC our customer had been using in a development board in his previous setup. This ADC met all of our requirements quite nicely. Running at 1.8GSPS with a bit depth of 12 bits, the ADC provided the speed and accuracy necessary for our needs. When looking to purchase this ADC we realized that the chip was more expensive than the original development board, which was already very expensive. This put the price above what our customer wished to pay.

After much research we were unable to find any ADCs that met Nyquist for our price point. This left us with a challenging solution: how would we sample the data accurately while staying in the price range of our customer. Discussing the issue with our customer we decided that we could sample under the Nyquist rate. This technique is often used to acquire information about high frequency signals where an oversampling ADC is either too high power, or expensive. The basic concept utilizes aliasing to see the higher frequency components mirrored down to a lower frequency. The drawback here is that noise in different Nyquist bands can be aliased and summed on top of the signal we care about. Since our expected signal can occur over a broad bandwidth from 50MHz to 500MHz, naive filtering will not help to reduce aliasing. Furthermore, ±3% accuracy and ±.1° phase accuracy compared to Darren's Network Analyzer version is acceptable. A student named, Justin Jung, who works with Darren Roblyer has

presented that undersampling is feasible and a viable option.

With a less tight sampling rate constraint we settled on an ADC, the ADS62P49, for our design. This ADC is spec'd at 250MSPS with a bit depth of 14 bits. This design change would later prove to be invaluable in lowering the complexity of our motherboard, specifically with the routing challenges of DDR RAM.

Selecting the frequency synthesis chips was simple. Our customer already has a working system and his direct digital synthesis chips, the AD9910, work fine. This is an area where cost cannot be lowered any more, so we are continuing to use the chips our customer is familiar with.

With the ADC and DDSs selected we needed to choose a way to interface with the chips as well as gigabit ethernet. The first options we considered were FPGAs, but designing gigabit ethernet on these board creates a very long critical path, allowing only the highest end of FPGAs capable of running at the speeds we wanted. These FPGAs added too much cost to our design so we needed another solution.

We considered using an ARM chip instead of an FPGA, as an ARM chip could do full speed gigabit communication. The issue with this method is there are no ARM chips available with enough LVDS pins to read from the ADC.

We finally chose to use a Zynq-7000 SoC, allowing us to meet the needs of high speed data collection from the ADC using the Virtex-7, as well as meet the communication needs utilizing the ARM A9. The two portions of the Zynq will communicate via high speed DMA.

Our first plan was to place a Zynq-7040 chip directly on the motherboard. This drastically increased the complexity of our motherboard, as the Zynq-7040 didn't have enough on board memory for our needs. The Zynq-7040 is designed to use DDR RAM for external memory, which is very difficult to route. The lines between the Zynq and DDR need to be impedance and length matched. The Zynq-7040 was necessary though, as our original ADC had many LVDS pairs and the 7040 was the first chip in the Zynq family which had enough pairs to read from it.

Once we changed our ADC, the Zynq-7040 was no longer necessary. A Zynq-7010 has sufficient LVDS pairs to read the new ADC. With the 7010 as an option, we were able to purchase a MicroZed for our board instead. This was useful as the MicroZed comes with 1GB of external memory and a built in gigabit ethernet port.

With all of the hardware chosen, we had to make decisions on how the user would interface with the board. Our customer had expressed a need to be able to control all of the sweep parameters from a host computer. Windows was our chosen operating system due to its ubiquity in our customer's lab and in medical research labs.

To program the GUI, Visual Studio was chosen as a framework/design suite. This decision was made because Visual Studio is the most developed GUI system for Windows. As stated, the GUI will allow the user to control sweep parameters in a user-friendly and approachable manner. A mockup of the design is shown on *Figure A3-3*. In addition to run control the GUI will also include data flow. This will allow the user to save the data in an easily parsable file such as a csv file. This is necessary as we are only collecting the data, not processing it in any meaningful manner.

Another aspect of the GUI is a graphical data field. To allow the user to see if the system is working. This field will display some useful recent data to the user, such as a frequency triggered plot. This data display is not meant to directly be used in research, but instead allows the user to debug the system and see if they are getting an expected result.

The customer has expressed an interest in easily portable board interfacing. The simplest solution to this issue was to make a library and wrap our GUI around. It is common for Windows applications to use dynamically linked libraries (DLL) for this type of thing, so we went with this method. To streamline the build process all of the menus, fields, and buttons used in our GUI will use the DLL we will ultimately give to the customer.

We have yet to choose a communication protocol, but have begun research on our options. Our initial tests have been on TCP and we have achieved 70% throughput with only simple TCP/IP. Although this is higher than we expected it, can be improved. A more complex protocol needs to be placed on to the TCP stack to achieve a higher throughput.

Our current top picks are a roll-your-own solution and ZeroMQ. The attractive part of doing roll-your-own is that you can optimize everything to your exact needs. The issue is that it is difficult to build a communication protocol, and there is a high chance that a pre-existing solution will have better performance. There is also a good chance that time would be better spent tweaking a pre-existing solution to make it more suited for our system.

ZeroMQ is one such pre-existing protocol that we are examining and highly considering for our communication. ZeroMQ provides a user friendly socket API that operates over TCP. It offers buffering and framing, features not available in simple TCP. Using such a system could increase our throughput with minimal effort, allowing us to concentrate on more pressing matters than developing our own TCP stack.

3.0 System Description

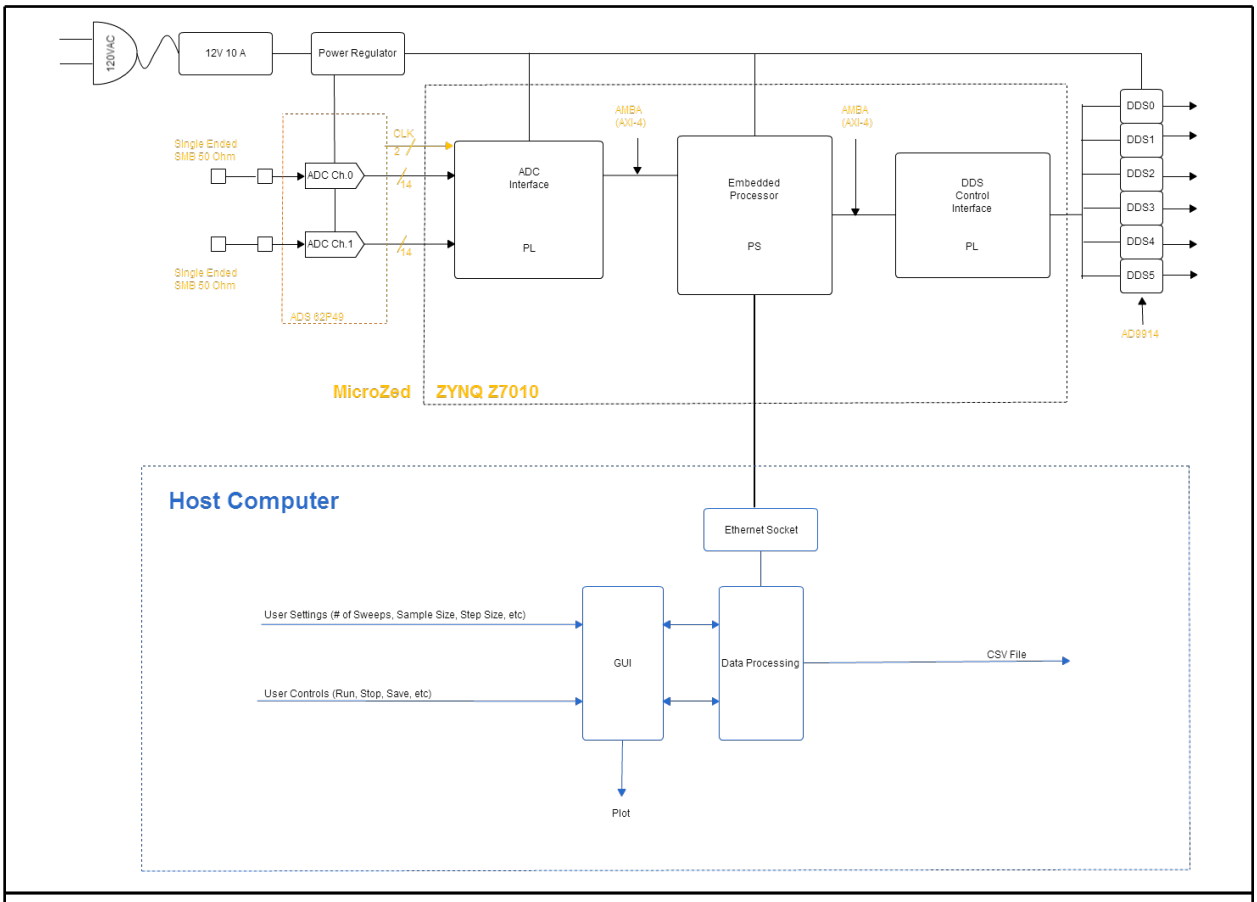


Figure 3-1: System Block Diagram.

In *Figure 3-1*, a system block diagram of our proposed design solution is presented. Illustrated is the basic power architecture, which includes a primary switching regulator which will give use 12V DC at a 10A maximum current draw to provide at most 120W. The 12V DC input voltage will be regulated on board to provide power to the microZed, ADC, DDS and all supporting devices. We will need to provide 25W of power in the worst case scenario, and 15W nominally with all devices turned on.

The hardware section is made up of the ADC, processing subsystem, and the 6 DDS controllers. The ADS62P49 dual-channel 250MSPS ADC which will digitize 2 channels—the reference and resultant waveforms—and send the samples to the processor. The ADS62P49 requires 15 LVDS pairs, and 7 CMOS control signals and communicates through the protocol specified in *Figure A3-10 [4]*. The AD9910 direct digital synthesis chips, provide a sampling rate of 1GSPS which will bring us right up to our upper range of 500MHz [5]. It is acceptable to only reach 450MHz. A block diagram of the AD9910 can be found in *Figure A3-5*. Each DDS

communicates using an SPI bus and we will be sharing their clocks. The data lines for the 6 chips will be fed in parallel and the each DDS will have its own chip select so that we don't need to write to them serialy, but they will still be run with a shared clock. This particular architecture is show in *Figure A3-4*, the SPI signals in this case are the chip select, data in and data out lines.

Both the ADC and the 6 DDS chips will be controlled by a Xilinx Zynq Z-7010 on a MicroZed development board. The MicroZed development board contains 1GB of on board RAM, and will run a Linux operating system. This MicroZed will communicate with a PC and our software client over a 1Gbps ethernet link as a server that a client program can attach to. Furthermore, the MicroZed will host firmware in the PL section to communicate with the ADC and the DDS chips. These peripherals will be done in either VHDL or verilog and will utilize the AXI4 standard bus to communicate with the Linux OS. The ADC firmware core will use the AXI4 DMA capabilities and the DDS chip interface will utilize a simpler AXI4 GPIO interface. A firmware block diagram is available in *Figure A3-6.*

The implementation of the hardware will be done in a two board architecture, the "motherboard" and the "frequency synthesis board". The motherboard will contain connectors to the MicroZed, the ADS62P49, extendable IOs for further expansion boards, and connectors to the frequency synthesis boards. A block diagram of this board can be seen in *Figure A3-4*. The frequency synthesis board is illustrated in *Figure A3-4* and contains 6 AD9910s, 6 50Ω SMB connectors, local power regulation, clocks for the AD9910s and the connector back to the motherboard. This architecture will allow us to do the two designs in parallel for two of our major components. Furthermore, Darren has a 3 AD9910 development boards which could be used to test the DDS control functionality in the case of a hardware fault in the frequency synthesis board, with proper connector design.

Figure A3-3 displays the design of our user interface. The purpose of the user interface is to allow the user of the system to set various parameters of the measurements that they will be taking. This design consists of three main tabs: Settings/Controls, Data, and Graphs/Post Processing. The Settings/Controls tab is where they will directly communicate with the Zynq. In addition to being able to start and stop the measurements, they can also set the maximum and minimum frequency in each sweep, the number of sweeps, the time delay between each sweep, the number of steps per sweep and the number of samples per each step in the sweep. They will then be able to save these settings to a profile name to allow them to reuse these settings again later on without having to re-enter them. There will also be a graph of the data that is being collected so that the user can see if there are any errors in the measurements, in which case the user will be able to stop the current sweep. The data tab will allow the user to view the raw data collected by the Zynq and the graphs and post processing tab enable the user to further analyze the data that was collected.

This GUI and its functionality serve as the client side of the software part of the project. The functionality of the GUI will all be contained in a Windows DLL. The purpose of creating the DLL is that it will allow new functionality to be easily added to the GUI in the future. The DLL will make it possible for the different functions of the GUI to be used in other code. The entire software portion of the system, *Figure A3-7*, consists of three main parts: a client, a server and the communications protocol between the them. The server will be the Zynq which will be running linux and the server will be a PC running windows. The communications protocol that will be used is still being decided.

4.0 First Semester Progress

4.1 Hardware

The primary focus this semester was on parts selection and high level design. We have chosen the AD9910 as the digital synthesis chip, the ADS62P49 as the ADC and the MicroZed development board for the Xilinx Z-7010 SoC as our processing system. A great deal of time was spent assessing the compatibility of the ADC with the MicroZed in order to be sure that the MicroZed possessed sufficient LVDS pins. The decision was also made to split the DDS functionally off from the main printed circuit board to allow for a more modular and revisable design. Finally, the hardware block diagrams for the motherboard (*Figure A3-8*) and the DDS board (*Figure A3-4*) have been produced with all non-power routings marked.

4.2 Embedded Processing System Selection

This semester Christopher Woodall and a Benjamin Havey worked on selecting the embedded processing system. One of the more important metrics for choosing our processing system were the interfaces with the high speed ADC converter and the DDS chips. The DDS chips chosen, the AD9910, contains a SPI bus with a few additional control signals. This interface is not very timing critical and can be implemented from a vanilla SPI peripheral, with some wrapper code. However, high-speed ADC's utilize a variety of parallel interfaces which often run at the sampling rate or double data rate at half the sampling rate. Our first choice was a 1.8GSPS dual channel ADC which communicates using 4 12-pin LVDS ports—two for each channel. The pin count and speed of such an interface made it obvious that a typical microcontroller would not have a sufficient peripheral interface.

We originally adopted a microcontroller and FPGA architecture, where the FPGA would implement the direct interface with the ADC and the microcontroller would interface with a computer over either UART or ethernet. However, we determined that transfer between the FPGA and microcontroller would be slow, creating a bottleneck for sweep time. From this architecture we naturally moved to the Xilinx Zynq System-On-Chip which integrates a dual core ARM-Cortex A9 and an Artix-7 FPGA in one package. Furthermore, those two subsystems share the AMBA bus, which has already been discussed, and allows for rapid communication between the two subsystems.

To test this part we started experimenting with a Zedboard, which is a development board for the Z-7020 series. First, we walked through the Avnet tutorials to understand the interactions between the FPGA and processor. Then we implemented a basic FPGA module that would turn LEDS on and off by receiving messages from a bare-metal processor over the AMBA bus. After this we tried to control a section of programmable logic using Linux, which requires us to compile a linux kernel from scratch and use a u-boot section. Currently, we are working on completing this build of Linux.

After moving to the undersampling ADC our number of datalines has dropped to 15LVDS pairs and data is being delivered at 250MHz. The lower number of data lines has allowed us to move to using the microZed development board with a Z-7010. This will plug into our motherboard via header pins available on the microZed. This design allows parallel firmware/hardware development and for reduced design time.

4.3 Software Progress

This semester we designed our GUI. We decided to split up all of the relevant information into three separate tabs. We chose to use three different tabs to split up the information in a clear and systematic way. After speaking with our customer, we determined that this would be the best way. The three tabs are Settings/Control, Data and Graphs/Post Processing. The Settings/Control tab will let the user set different parameters of their measurements and start and stop the measurements. The Data tab

will allow the user to view the raw data that was collected by the Zynq. The Graphs/Post Processing tab will let the user analyze the data that was collected. We also have basic setups of our TCP and ZeroMQ implementations for simple communication between computers.

4.4 First Deliverables Testing

We developed a testing interface between the PL and PS of the ZedBoard (Zynq development board) and started the communications protocol for the communication between the client and the server of our system.

The testing interface that was created used the PS to read in two hex characters using executable C code that the user entered over UART using PuTTY. The C Code parses the ASCII values and converts it to hex. It then sets the registers on the AXI bus and prints the hex value out on the LEDs of the Zedboard. This interface was a major step forward in understanding the Zynq SoC and the tools that are needed to program it. Through this test we were able to show that we can describe hardware for the Zynq PL and load code on to the PS to talk to it. Additionally, this provides us with our final boards validation test to ensure that the Zynq is working as expected and the UART connectivity will allow us to have an easy interface for hardware validation and debugging.

The communications protocol that was developed throughout the semester consisted of a client-server program that sent a file between two PCs. This program was written in Visual Studio using C++ and made use of TCP. This semester the client and server were both PCs. While this program provides us with base for client/server architecture, in the future the client will be a PC and the server will be an embedded linux device. We were able to test this protocol and determine its average throughput for 100 different trials. *Figure A3-9* shows a histogram of the throughput values that were collected during the trials. We found that there was an average throughput of 671.57Mbps. Because we need to transfer data at a minimum of 500Mbps to meet our customers speed requirements. This is a promising result because it provides us with a comfortable margin above this minimum. In addition to the protocol that was developed, two GUIs were made in order to run the program and display the throughput.

5.0 Technical Plan

Task 1. Motherboard

The motherboard will contain the ADC which will measure the incoming 50-500MHz signals and pass them to the PL on the MicroZed which will in turn send the data over an AMBA bus to the PS which will package the data and send it out over Ethernet to a PC. The MicroZed on the motherboard will also be connected to the DDS chips on the DDS board via SPI busses so it can set them when instructed over Ethernet.

Lead: Christopher Woodall, Assist: Thomas Nadovich

Task 2. DDS Board

The DDS circuit board must also be designed and ordered by January 30th. It will include 6 digital synthesis chips connected to the MicroZed via SPI that will produce signals that will be sent to SMB connectors out of the enclosure. It will also contain it's own power regulation.

Lead: Thomas Nadovich, Assist: Christopher Woodall

Task 3. Firmware Design

The Zynq bootloader must be finished. After this is complete the DMA module needs to be completed. First the PL portion will be designed and once that is complete the linux driver will be written. After testing and completing the DMA module, the ethernet packet parser and GPIO module interconnect will be completed. This will allow for commands to be taken from the user (over ethernet) and then sent to the FPGA. These commands will include things such as step size, frequency range, and number of samples. Once this is done the firmware will be ready for controls: the ADC, DDS, and GPIO controls will be programmed for the programmable logic, in that order. Once all of these are done there will be a rigorous testing and integration period. The system will be rigorously tested until the test date to ensure the system is as robust as possible.

Lead: Benjamin Havey, Assist: Christopher Woodall

Task 4. Ethernet Communications

There should be complete optimized TCP and ZeroMQ communications protocol implementations a week before testing date. At this time, we will be comparing the performance of the two. Whichever one shows more promise by the end of our test comparison phase will be the one that will receive full focus and further optimized in preparation for ECE day.

Lead: Andy Mo, Assist: Benjamin Havey, Caroline Ekchian

Task 5. Linux Software Development

The server for both protocol implementations will be integrated with the ZedBoard. Control commands received from the client will be transmitted to the microZed, which will parse the commands for ADC and DDS module controls. During a frequency sweep, the microZed will receive data from the DMA module, package it, and send it to the client. Whatever we choose as our final communication protocol will need to be implemented on Linux.

Lead: Andy Mo, Assist: Benjamin Havey, Caroline Ekchian

Task 6. Windows GUI

A GUI will be created using Visual C++ based on the design developed for the mock up. It will allow the user to set various parameters of the measurements that they are taking using the dDOSI system. As well as allowing the user to set parameters, the GUI will also enable to user to view the raw data that was collected by the Zynq and to analyze and graph the data that was collected. The GUI will then be integrated with the Zynq which will be used as the server.

Lead: Caroline Ekchian, Assist: Andy Mo

Task 7. Enclosure

The enclosure will be designed built and tested. It be large enough to mount all components of our design and provide openings to connectors mounted on the board. If heat is deemed to be an issue then the enclosure must provide adequate ventilation as well. The faceplates will be custom cut and must be designed and procured as well.

Lead: Thomas Nadovich

Task 8. Power Regulation Subsystem

Adequate power must be delivered to all systems in the device. A 12V 10A supply will be purchased and enclosed in the box to provide power to the motherboard. This 12V signal will be routed through the motherboard to the DDS through a connector. The voltage supply requirements for the DDS board will be met by a 1.8V and a 3.3V regulator and the board is projected in total to draw 5W in total nominally. The two components that will be consuming the majority of the power on the motherboard will be the ADC and the MicroZed. The MicroZed, running at full capacity, draws 4.8W at 5V from a dedicated regulator and the ADC will be drawing an additional watt, distributed over a 1.8V supply and a 3.3V supply. To avoid the electromagnetic noise from too many switching regulators, linear regulators will be used and these will require additional power as well. In total the power draw for the whole system is projected to draw approximately 15W.

Lead: Thomas Nadovich, Assist: Christopher Woodall

6.0 Budget Estimate

Item #	Quantity	Description	Indiv. Cost	Net Cost
1	6	AD9910 Direct Digital Synthesis Chips	$56.93	$341.98
2	1	ADS62P49 Dual Channel; 250MSPS ADC Chip	$207.97	$207.97
3	1	MicroZed Zynq SoC Development Board	$199	$199
4	1	Zedboard Zynq SoC Development Board	$395	$0
5	2	4 Layer PCB Board Manufacture	$66	$132
6	1	Enclosure	$50	$50
7	1	12V 10A Switching Regulator	$20	$20
8	8	SMB Connectors	$2.38	$19.04
9	N/A	Passive components estimate	$10	$10
10	N/A	Power subsystem estimate	$30	$30
11	N/A	I/O Breakout, USART, etc. connectors	$50	$50
		Total Cost		$1059.99

It should be noted first that all components and manufacturing runs will be paid for by our client. The three dominating items that make up more than half of our budget are the ADC chip, the DDS chips, and the MicroZed. These three items provide the core functionality of our project and the performance required to meet our clients needs can only be found at this price point. The next largest single expense is the printed circuit board manufacturing. Two runs have been been budgeted in case there are critical errors that must be fixed in the first run, however the ultimate cost of a revision will be variable on what exactly needs to be fixed. Beyond these costs, the individual items are relatively inexpensive on there own, but add up to a significant amount. They are estimates as opposed to direct quotes as many of the systems that inform their price have yet to be fully designed. Finally it should be noted that the Zedboard (not the MicroZed) is listed as a cost as we have been using it to familiarize ourselves with the Zynq chip. However it was donated temporarily to the project by a group member and does not need to be accounted for as a true expense.

7.0 Attachments

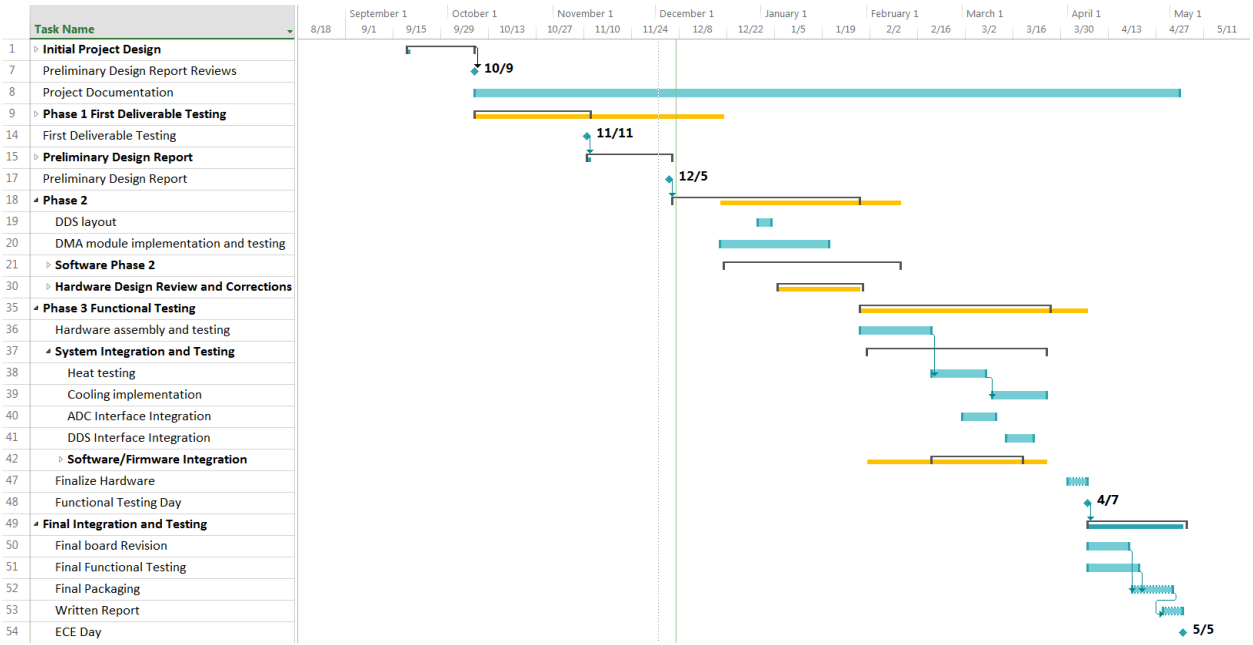
7.1 Appendix 1 – Engineering Requirements

Team # 19 Team Name: dDOSI

Project Name: Digital Diffuse Optical Spectroscopic Imaging Spectrum Analysis Unit

Requirement Name	**Parameters**
Frequency Range	50MHz-500MHz
Sample Size	14 Bits
Min Frequency Step Size	1MHz
Maximum Allowable Frequency Sweep Time	1s
Preferred Frequency Sweep Time	100ms at a sample size of 4kSamples/step and 450 steps
Max Sample Size	64kSamples/step
Max Steps	450 steps
Amplitude Error	±3% Amplitude Error
Phase Error	±0.1° Phase Error
Noise Floor	> -80dBm
ADC Input Impedance	50Ω
# of Simultaneous DDS Channels	6 Channels
Ethernet Speed	10/100/1000Gbps

7.2 Appendix 2 – Gantt Chart


Task Name
1 Initial Project Design
7 Preliminary Design Report Reviews
8 Project Documentation
9 Phase 1 First Deliverable Testing
14 First Deliverable Testing
15 Preliminary Design Report
17 Preliminary Design Report
18 Phase 2
19 DDS layout
20 DMA module implementation and testing
21 Software Phase 2
30 Hardware Design Review and Corrections
35 Phase 3 Functional Testing
36 Hardware assembly and testing
37 System Integration and Testing
38 Heat testing
39 Cooling implementation
40 ADC Interface Integration
41 DDS Interface Integration
42 Software/Firmware Integration
47 Finalize Hardware
48 Functional Testing Day
49 Final Integration and Testing
50 Final board Revision
51 Final Functional Testing
52 Final Packaging
53 Written Report
54 ECE Day
8/18
September 1
9/1
9/15
October 1
9/29
10/13
November 1
10/27
11/10
December 1
11/24
12/8
12/22
January 1
1/5
1/19
February 1
2/2
2/16
March 1
3/2
3/16
April 1
3/30
4/13
May 1
4/27
5/11
10/9
11/11
12/5
4/7
5/5

7.3 Appendix 3 - Figures

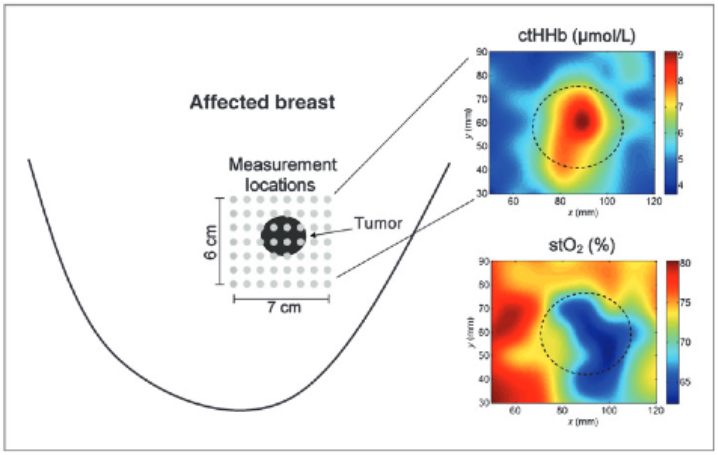


Figure 1. DOSI measurement procedure and optical property maps. Measurements are taken using a handheld probe that is moved in a grid or line pattern over tumor and normal breast tissue. Dots indicate measurement locations. In this example, an 6 cm by 7 cm region of tissue was measured containing a clinical stage II IDC measured to be 27 mm in the greatest dimension. Maps of optical properties are made by interpolating data values between measurement points. In this example, both deoxyhemoglobin (ctHHb) and oxygen saturation (stO_2) are shown. In both maps, which are from identical tissue locations, the dotted circle indicates the approximate tumor location determined by ultrasound and palpation. This subject was non-pCR. Note the relatively low oxygen saturation in the tumor region compared with surrounding normal tissue.

Figure A3-1: DOSI measurement procedure and optical properties map [3].

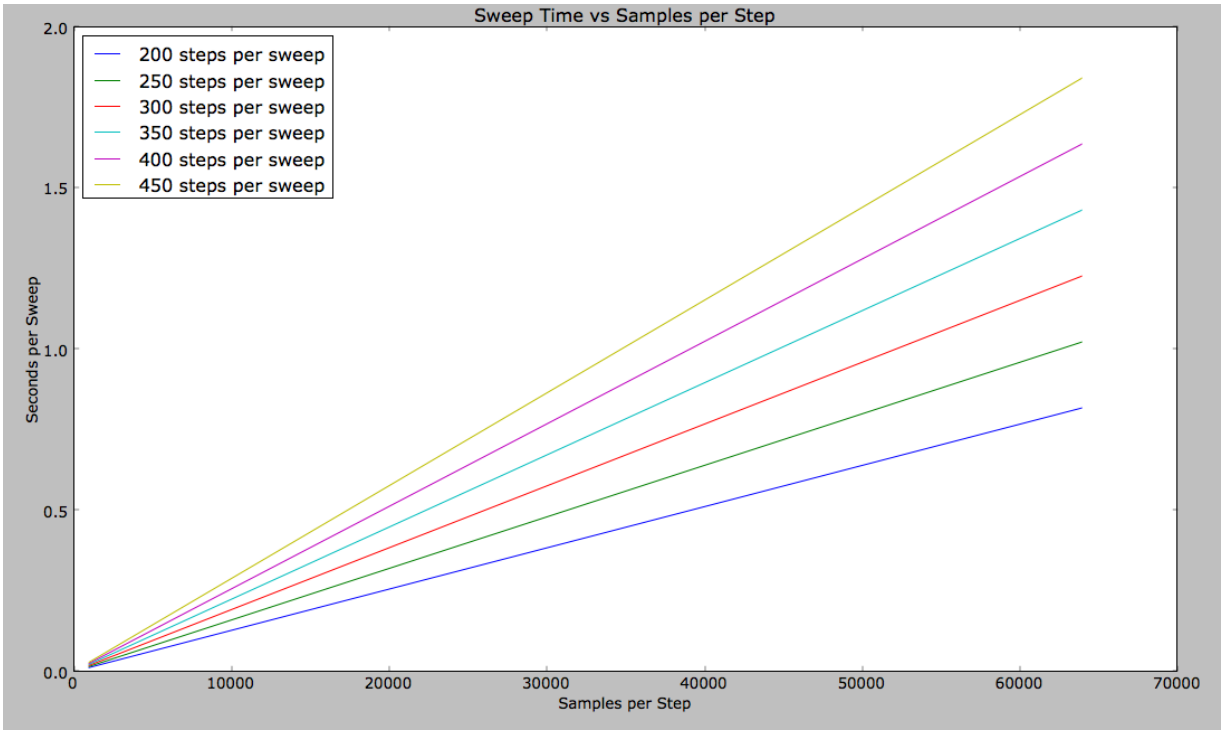


Figure A3-2: A plot of the sweep time vs samples per step. Computed using the code in Appendix A4-1 Assumes 50% throughput on Gigabit ethernet

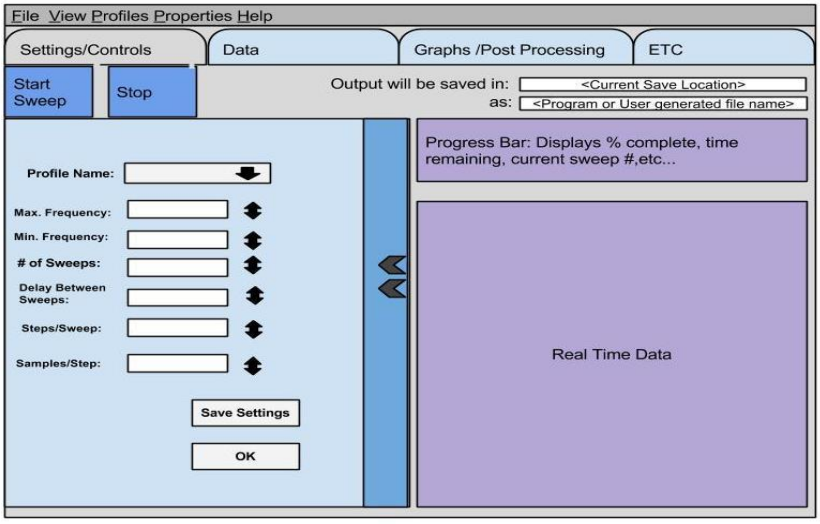


Figure A3-3: Software GUI mockup

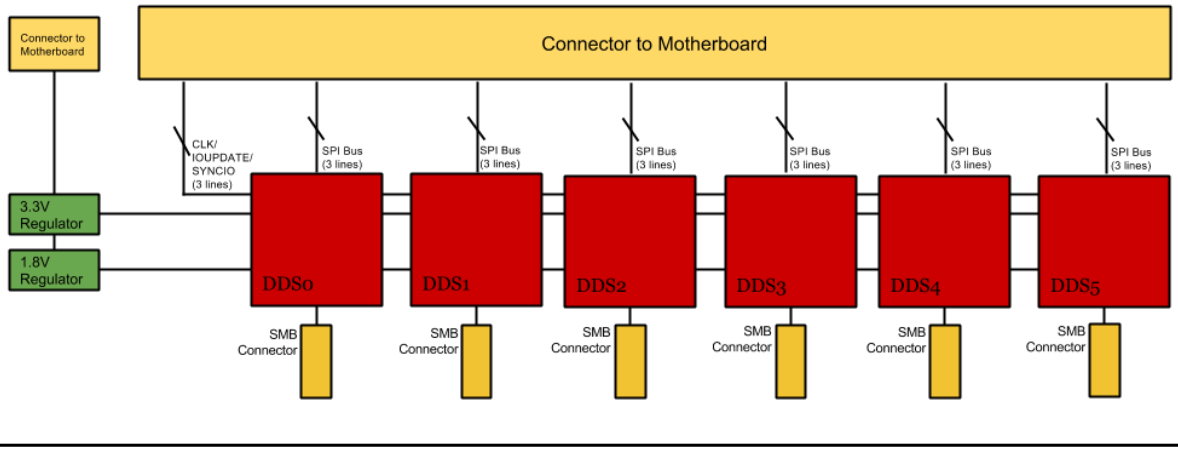


Figure A3-4: Hardware System Block Diagram

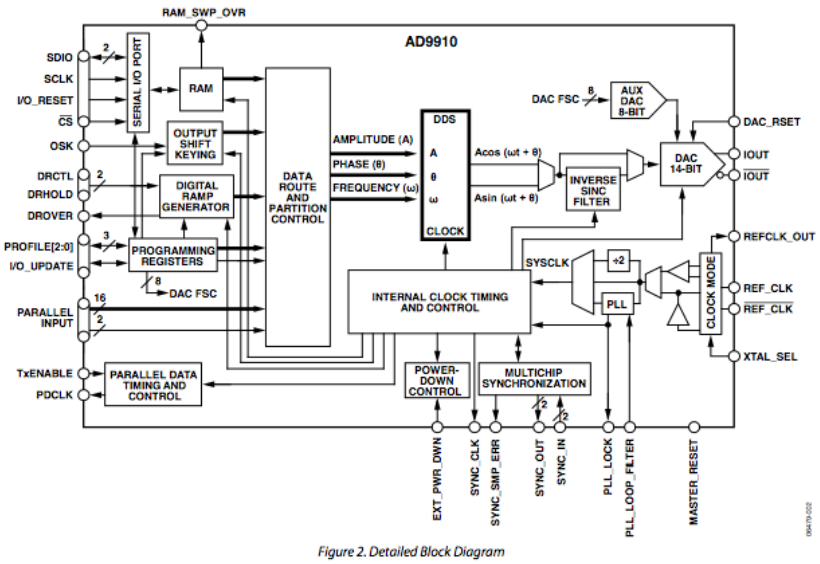


Figure 2. Detailed Block Diagram

Figure A3-5: AD9910 Block Diagram[5]

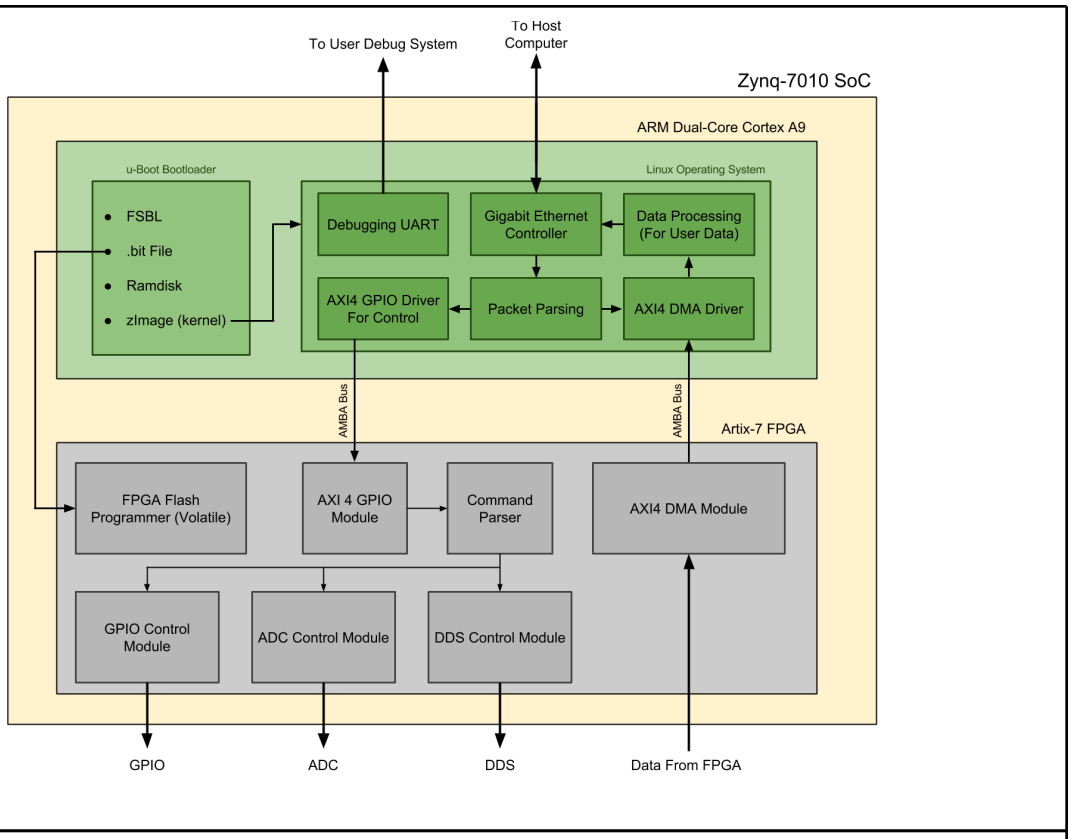


Figure A3-6: Firmware Architecture Block Diagram

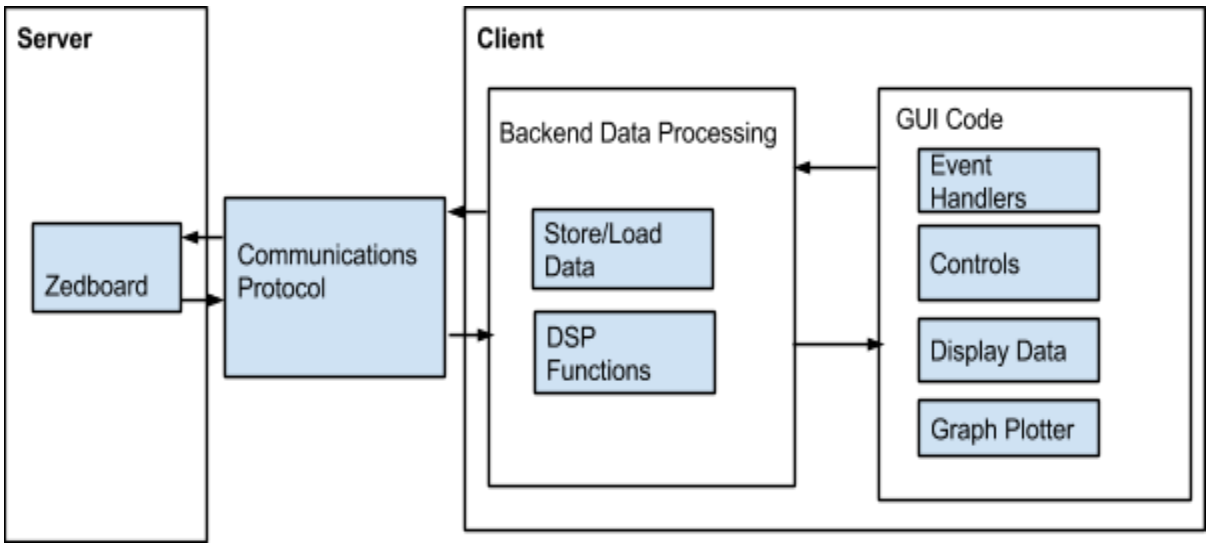


Figure A3-7: Software Architecture Block Diagram

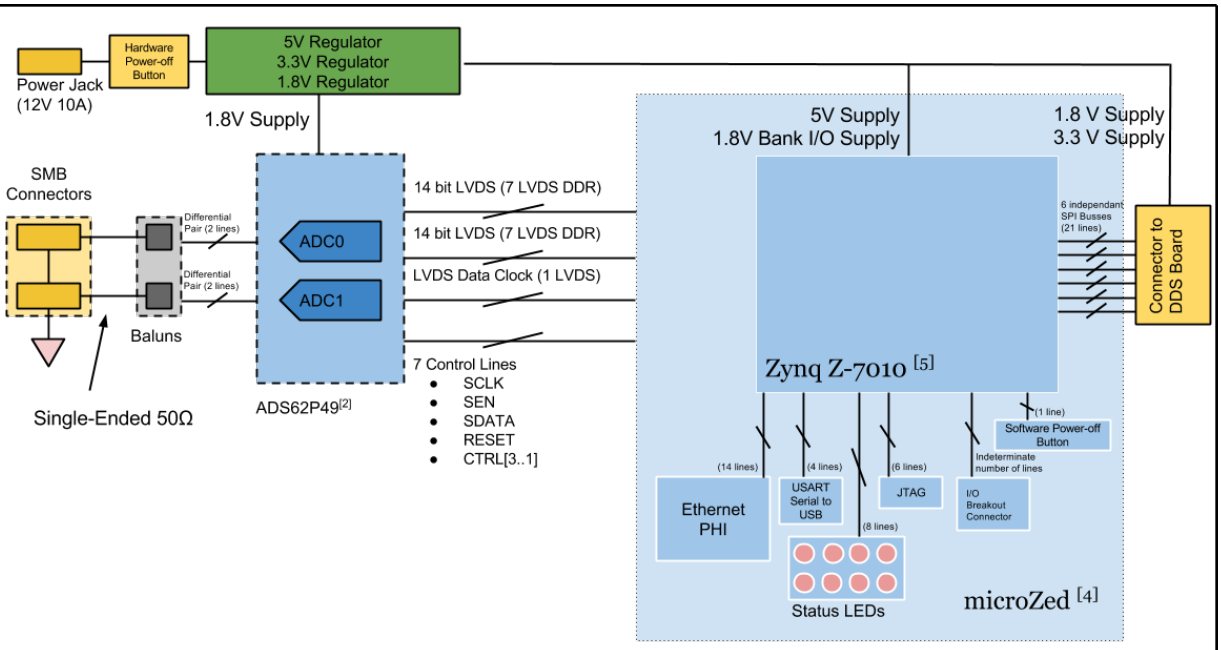


Figure A3-8: Hardware System Block Diagram

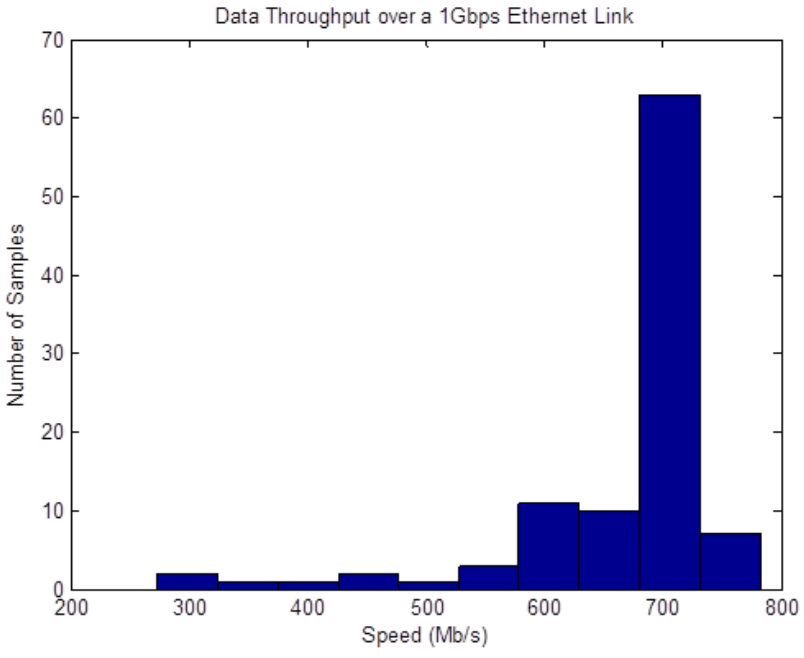


Figure A3-9: Histogram of throughput values, collected over 100 trials, in Mbps

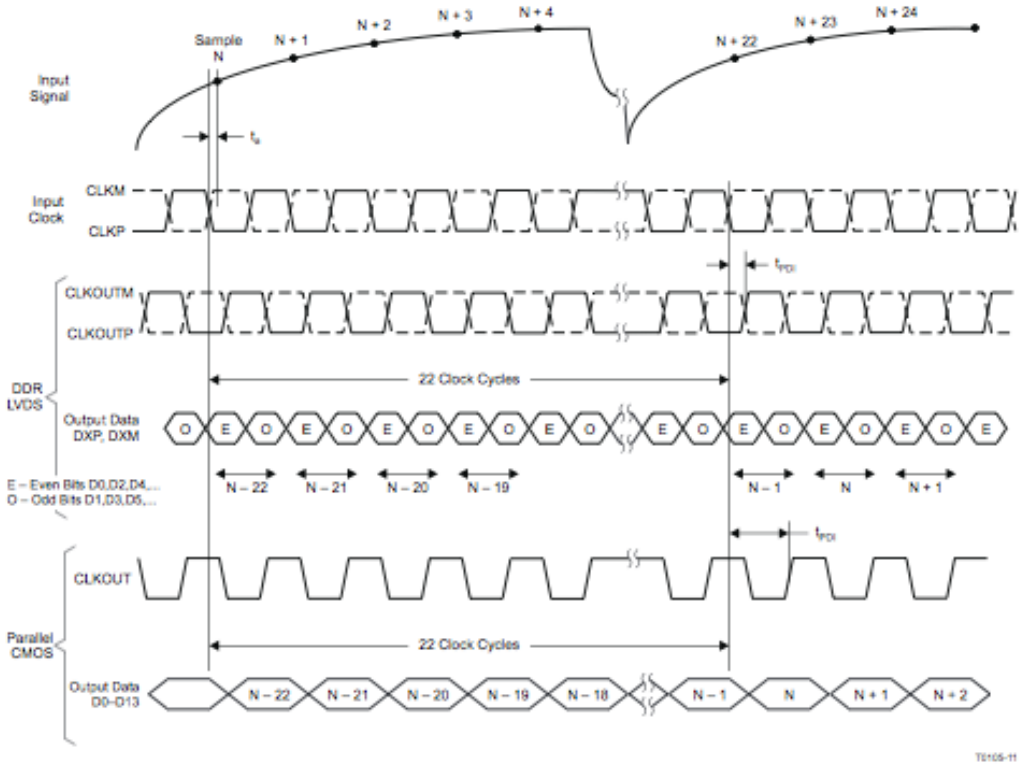
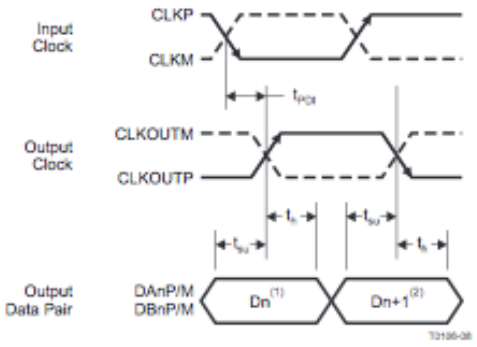


Figure 5. Latency Diagram



(1) Dn - Bits D0, D2, D4, ...
(2) Dn + 1 - Bits D1, D3, D5, ...

Figure 6. LVDS Interface Timing

Figure A3-10: ADS42D69 Block Diagram [4]

7.4 Appendix 4 – Code Snippets
Sweep Time vs Samples per Sweep Simulation

Requires: Python 2.7, and pylab.

```
import pylab
import sys

def time_per_sweep(n, bits_per_sample=16 ,num_steps=450,
        line_speed=1e9,efficiency=0.50, c = 2):

   return c*num_steps*n*bits_per_sample/(efficiency * line_speed)

if __name__ == "__main__":
   ns = [1e3, 2e3, 4e3, 8e3, 16e3, 32e3, 64e3]
   ss = [i for i in range(200,500,50)]

   lines = []
   for s in ss:
        t = []
        for n in ns:
                t.append(time_per_sweep(n,num_steps=s))

        lines.append((pylab.plot(ns,t), s))
        pylab.hold(True)


   pylab.legend([ln[0] for ln in lines],
                        ['{0} steps per sweep'.format(ln[1]) for ln in lines],
                        'upper left')
   pylab.xlabel('Samples per Step')
   pylab.ylabel('Seconds per Sweep')
   pylab.title('Sweep Time vs Samples per Step')
   pylab.show()
```

7.5 Appendix 5 – References

[1]: Pham, Tuan H., Olivier Coquoz, Joshua B. Fishkin, Eric Anderson, and Bruce J. Tromberg. "Broad Bandwidth Frequency Domain Instrument for Quantitative Tissue Optical Spectroscopy." *Review of Scientific Instruments* 71.6 (2000): 2500. Web.

[2]: Roblyer, Darren, et al. “Feasibility of direct digital sampling for diffuse optical frequency domain spectroscopy in tissue”. Meas. Sci. Technol. 24. 2013.

[3]: Uedoa, Shigeto, Darren Roblyer, et al. “Baseline Tumor Oxygen Saturation Correlates with a Pathologic Complete Response in Breast Cancer Patients Undergoing Neoadjuvant Chemotherapy”. Cancer Research. July 8, 2012.

[4]: Texas Instruments. “ADS62P49 Datasheet”. January 2011. http://www.ti.com/lit/ds/symlink/ads62p49.pdf

[5]: Analog Devices. “AD9910 Datasheet”. May 2012. http://www.analog.com/static/imported-files/data_sheets/AD9914.pdf

[6]: Avnet. “microZed Hardware User Guide”. November 2013. http://www.zedboard.org/sites/default/files/documentations/MicroZed_HW_UG_v1_2.pdf

[7]: Xilinx. “XC7Z010 Datasheet”. http://www.xilinx.com/support/documentation/data_sheets/ds187-XC7Z010-XC7Z020-Data-Sheet.pdf

7.6 Appendix 6 – Team Information

Name	Phone Number	Email Address	Role	Description
Caroline Ekchian	617-460-4397	cekchian@bu.edu	Software Lead	Electrical Eng. ‘14
Benjamin Havey	508-971-9784	benhavey@bu.edu	Firmware Lead	Computer Eng. ‘14
Thomas Nadovich	267-772-1898	tnadov@bu.edu	DDS Hardware, Enclosure, and Power Lead	Electrical Eng. ‘14
Andy Mo	617-272-6402	andy2m11@bu.edu	Communications Lead	Computer Eng. ‘14
Christopher Woodall	1-631-804-9822	cwoodall@bu.edu	Motherboard Hardware Lead	Electrical Eng. ‘14