Final Report (Group 15-22)

Ultrasound Imaging System

Project members

Advisor and Client: Timothy Bigelow
bigelow@iastate.edu

Aaron Tainter (Programming)
atainter@iastate.edu

Haoyu Wang (Hardware)
haoyu@iastate.edu

Jingyu Xie (Hardware)
bbsxjy@iastate.edu

Table of Contents:

Project Definition

MRI imaging is very expensive to run and maintain. Our research project will help develop a solution to provide a cheap alternative to fMRI imaging of the brain. Our client wants us to research and develop working components for a prototype that can be used for demonstration purposes.

In this project, our system consists of software and hardware components. The software side configures and deploys beams from a transmission device (transducer). The software can also acquire and analyze echo signals gathered by a receiver. Data is processed to generate greyscale "B-Mode" images that can be used to diagnose brain injuries (as an alternative to fMRI). We evaluated the accuracy and performance of the software to determine if the performance of the system is adequate to function in real time.

The hardware components function in conjunction with serial programs transmitted from the software. Our hardware produces a modulated sine pulse and amplifies it to work with the transducer. Protection circuitry also protects the proprietary hardware components from being damaged.

Professor Bigelow defined this project as a lab experiment rather than a working product. Creating the hardware needed for 256 channels is out of the budget of this project. Instead, our focus was on the design of a system that could be scaled. Our work designed components that could be used once the funding is available for all hardware needed to run the system.

System requirements

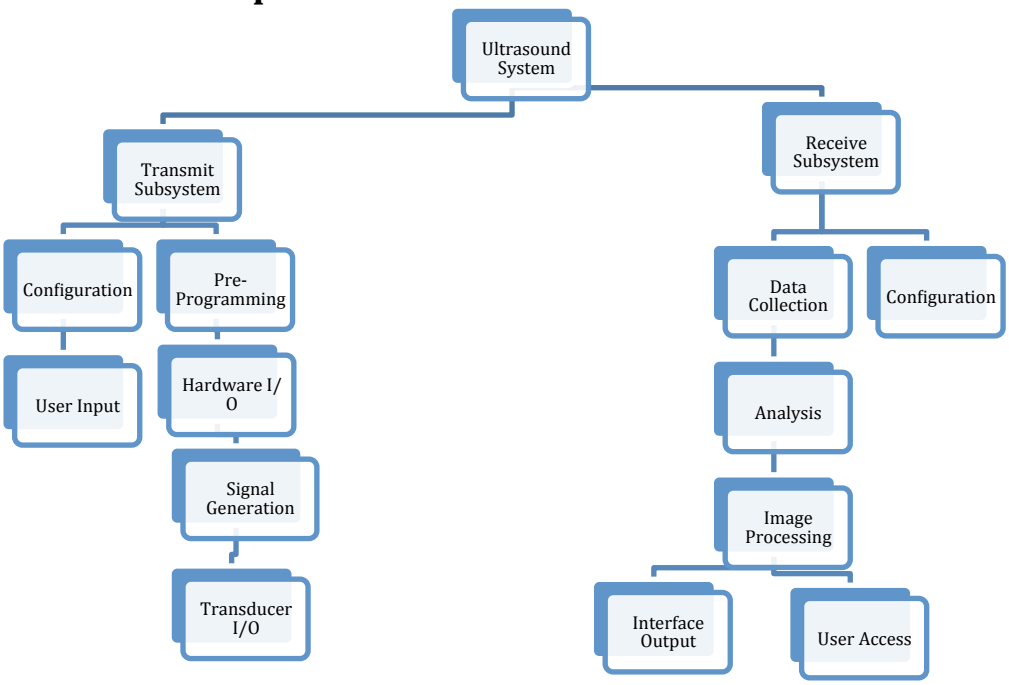
1. Transmission System
    - Serial Communication with the hardware
    - Signal generation hardware to transducer communication
    - Programming wave output via serial connection
    - Generates correct pulse waves
        - Over 1 channel (scalable to 512)
        - Ability to adjust pulse width modulation.

2. Receiver
    - Receive amplified and filtered signal from the receiver circuit.
    - NI 5752 card 8 bit input
        - Contains a variable input ADC

3. Analysis
    - Stream data continuously on input
    - Process 8 bit input

- Develop an algorithm that generates (B-Mode) images based on collected data
- Process real time (standard FPS)

Functional Decomposition



System Analysis

Overview

This system provides a quick and easy way to produce images of the brain. Due to the fact that this system will not be commercialized, more emphasis is placed on the performance and system accuracy rather than usability.

Transmitter

The configuration of each element can be controlled separately within the same UI. The signal generation hardware will be controlled by a subsystem within the labVIEW program. Our current design uses a separate module to test I/O. It contains default runtime settings that can be reconfigured for amplitude and pulse spacing by the user.

Receiver

The signal generation hardware can be programmed via serial data. Once programmed, the hardware will can emit pulses as a slave of the master program. Signals from the hardware will make the transmitter/transducer produce ultrasound pulses. The transducer can receive reflected signals and then amplify them to be sent back to the system. NI 5725 AI cards on our PXI controller can receive signals from the receiver within a range of -100mv to 100mv.

Analysis
Received data can be analyzed. An algorithm we created interpolates data to produce B-Mode images from the received signals. This subsystem executes when the NI 5752 DAQ receives data. We also enabled the module to run with sample input data.

The UI also includes elements that allow users to adjust the sensitivity of the receiver as well as the high and low bounds for data during input. This will allow images to have different levels of contrast. The system can save images to disk as well.

Block Diagram

Below is the block diagram of connected components in our system:

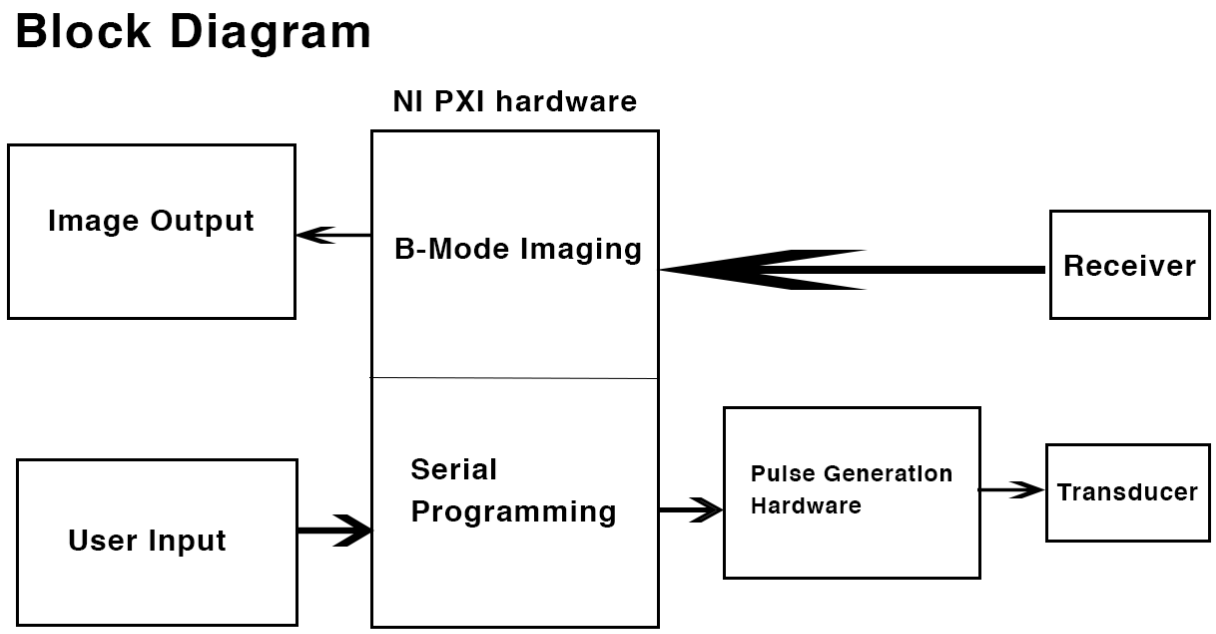


Figure 1- System Diagram

I/O Specification

Transmission system

Input

- Hardware Amplitude signal configuration (-/+ 50v)
- Channels (1 – 512 scalable)
- Variable gain for differing imaging depths (total 70 db)

- Frequency (1-15mHz)
- Switching delay time (32-167ns)
- On/Off

Output

- Signal generation hardware serial program.
- System initialization
- System termination

Receiver system

Input

- Data sensitivity (Max voltage input 200mVpp)
- Real time vs. Offline Imaging
- Image Contrast adjustments
- Data storage options

Output

- Raw signal data
- Interpolation algorithm
- Image file output
- Image real time output

Hardware/Software Specifications

Our signal generation hardware is interfaced with through NI 5752 cable I/O. We configured our input to work according the NI cable system I/O. These cables refer to different pins on each the analog and digital side of the 5752 cards. This is outlined later in Appendix 1.

The hardware part started in various ways at the beginning, after some meetings we specified the different hardware components. Those components are described below:

- 4 bits serial to parallel converter which is used for adjust our input to the next component 4 bits DAC. Since the DAC is 8 bits originally but our input only contain 4 bits, we came up to use this converter to arrive desired input

- The 4 bits DAC is designed to transform the incoming digital signal to analog signal. The reason we are using it is because the receiving board is based on analog signal which was originally designed for receive a beam former signal.

- After getting the analog signal, we designed a Band-pass filter which is used for smoothing signal and reduce the noise to ensure our input source clear enough before enter into the imaging system.

- While designing the Band-pass filter, we adjusted the resistor value with op-amps to make to output into a $\pm 50V$ and also with the protection circuit to make sure it won't burn the system out.

- After that, a power amplifier is also added to make sure the receiving part will work correctly.

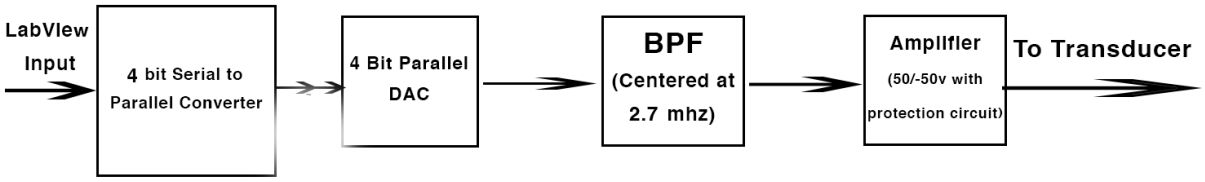


Figure 2 – Signal Generation Diagram

Additionally, transducer operates on specific voltage regulations. If the voltage input or impedance is out of specification, it might destroy the system. We must be careful to use the correct voltages during output. Also, because the parts are so sensitive, we also used proper safety precautions when handling parts to avoid short circuits or static discharge.

Software was created with modularity in mind to make it easy for other groups to use our products. Several subVIs were created in other to allow users to swap them if needed. The directory listings and description of each subVI can be seen below:

ultrasoundPanel.vi
- The main panel for configuration and setup.

Serial Programming (subVI).vi
- Used to program the signal generation hardware via NI 5752 digital output.

B-Mode (subVI).vi
- Analyzes input signal to produce a B-Mode image.
- Operates inefficiently (1500ms), but produces a nice image.

B-Mode_Optimized(subVI).vi
- Optimized version of the previous vi.
- Operates efficiently (150ms), but image is of lesser quality.

FiniteAcqMultipleChannels.lvproj
- Contains FPGA for AI acquisition.

SingleChannelExtClock.lvproj
- Contains FPGA for DO output testing.

ultrasoundData.csv
- Contains sample data for B-Mode imaging

B-Mode Imaging

The ultrasound imaging system works off of the acquired AI data. RF input data is demodulated using a Hilbert transform. This is done to remove negative frequencies in the RF data. Envelope detection is performed on the data to remove complex components in the signal. The data will contain values outside of the dynamic range for an 8bit greyscale image, so a log compression reduces the data point values to those within a black and white 0-255 value space. These are known as the "a-lines" because each line of transmitted data represents a pixel column in the image. In order to scale the image for proper viewing, we wrote a function that performs pixel compression on the image. Instead showing image to the user as a 512px by 4000px aspect ratio, the program compresses the image to a square aspect ratio, which is easier to view.

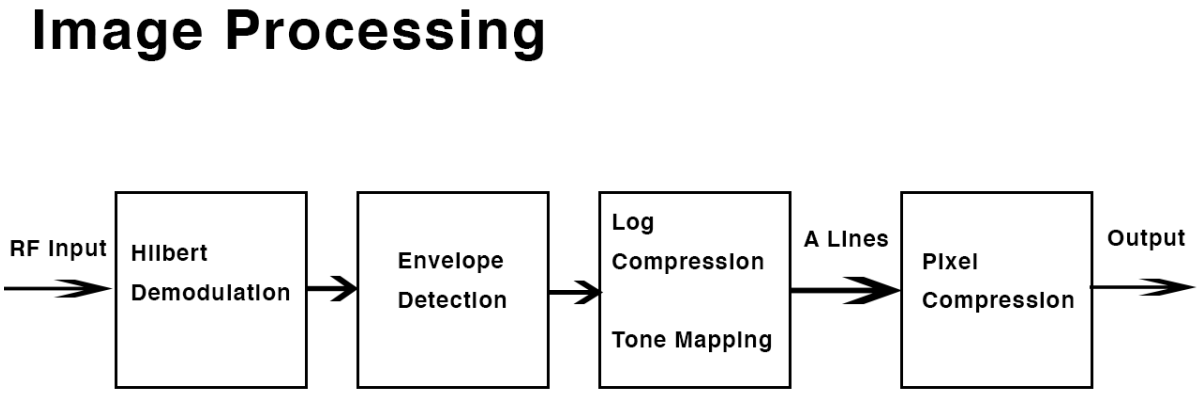


Figure 3 – B-Mode Imaging Diagram

Simulations and Modeling

The software side of testing was done using PCB test boards that allowed us to connect the PXI device to a function generator and oscilloscope. Simulations were created with the function generator. We tested uniform signals to prove the correctness of our interpolation algorithm. We also tested simple waves as input to the acquisition system.

B-Mode imagery was tested with open sample data acquired online. Sample data spanned 512 channels and 4000 data points. This allowed us to test efficiency at maximum input conditions.

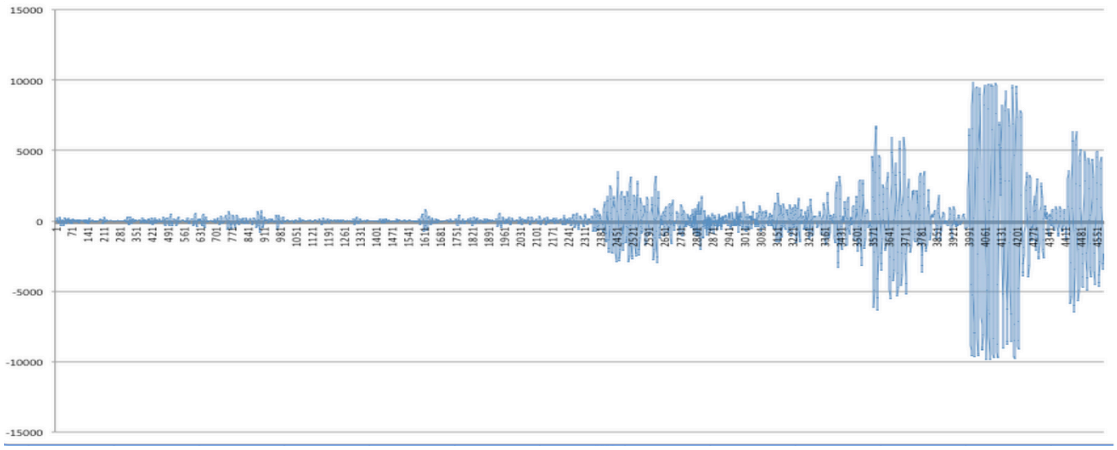
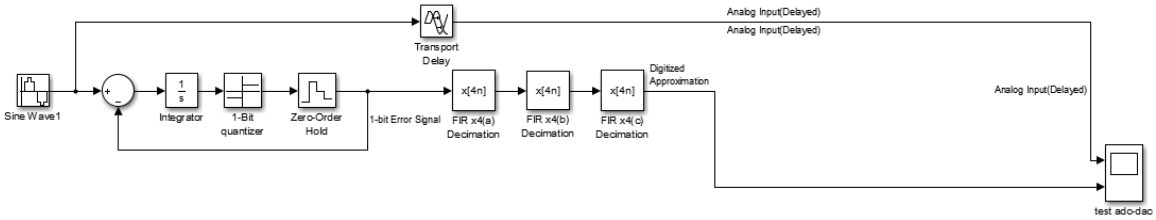
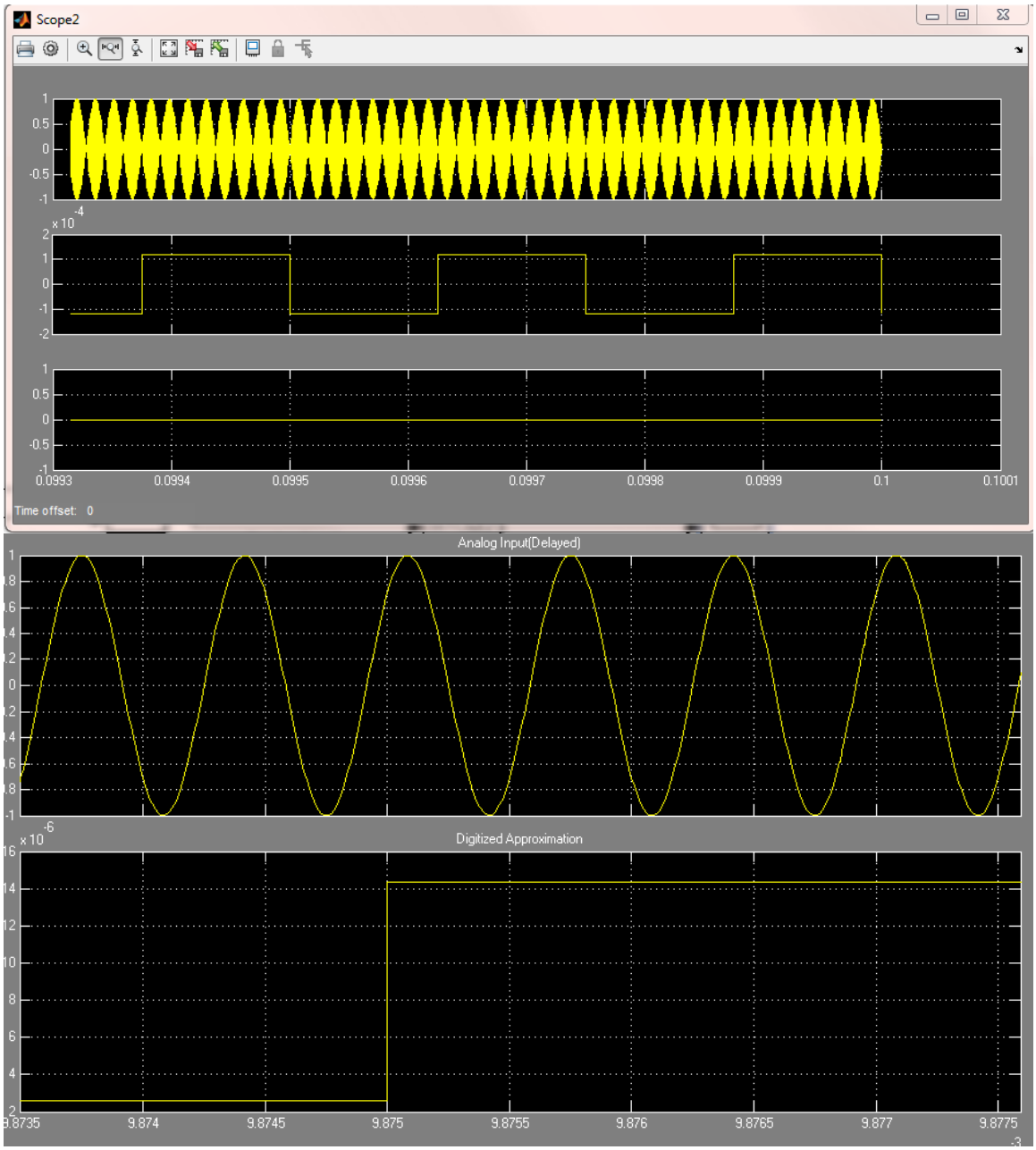


Figure 4 – One channel of sample RF data (data is acquired in reverse)

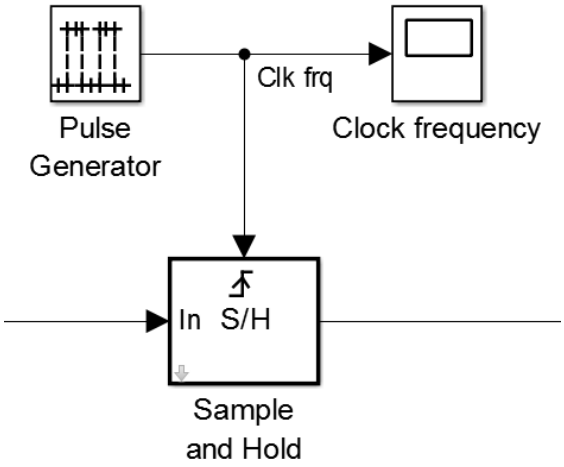
Simulations for Hardware

1. Entire system: Using Matlab Simulink
    a. ADC testing:

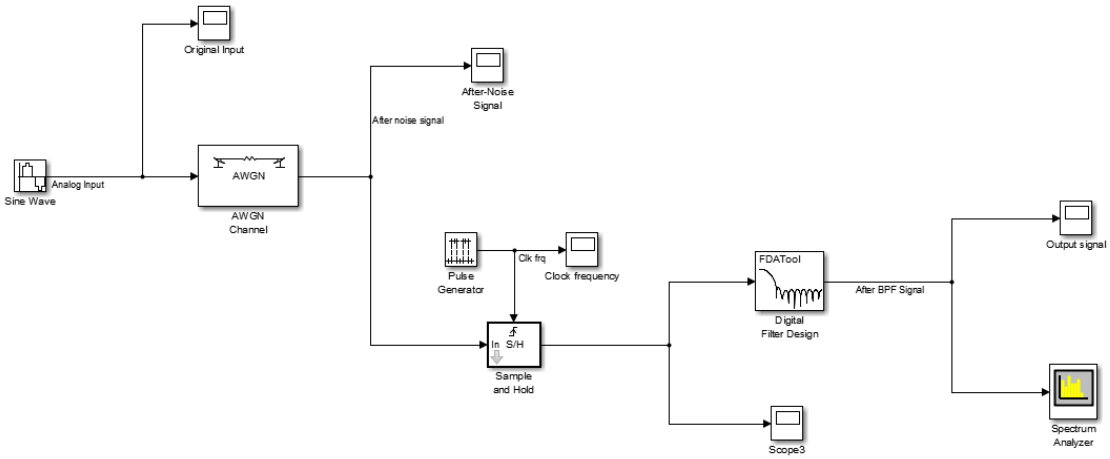




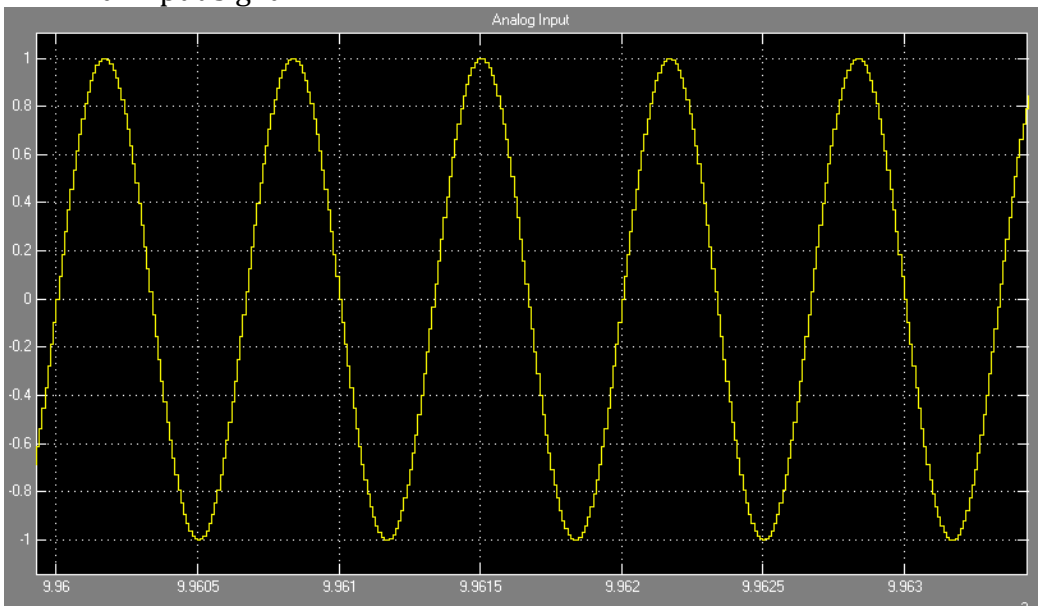
b. DAC testing:



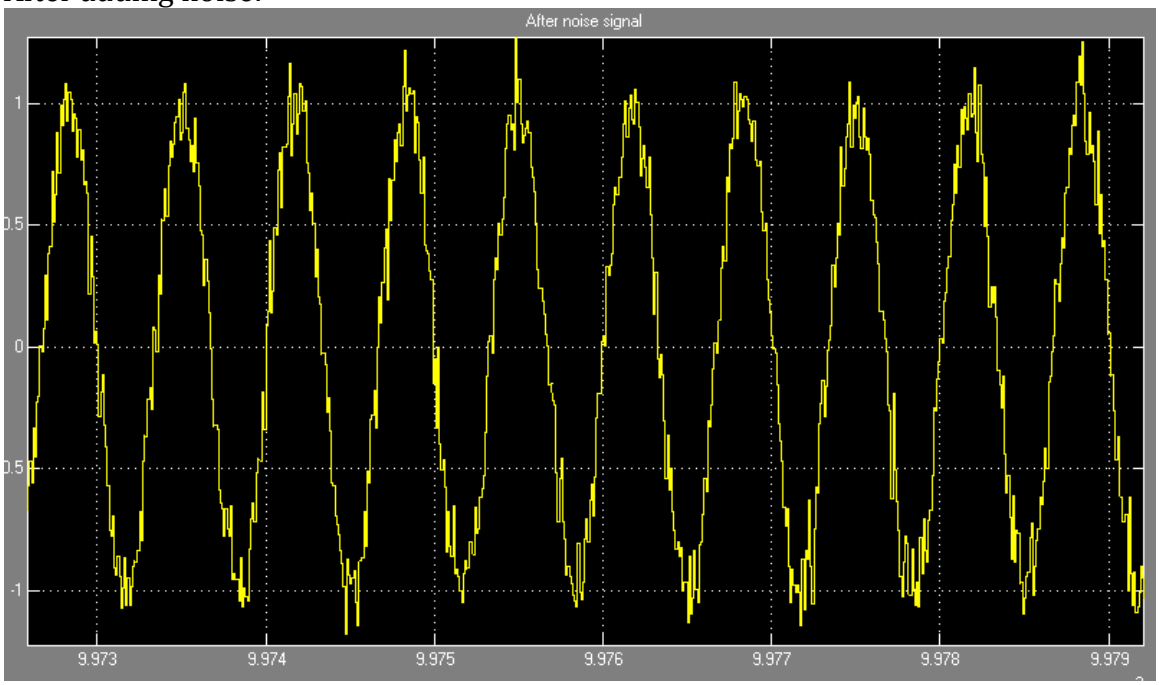
c. Entire System:



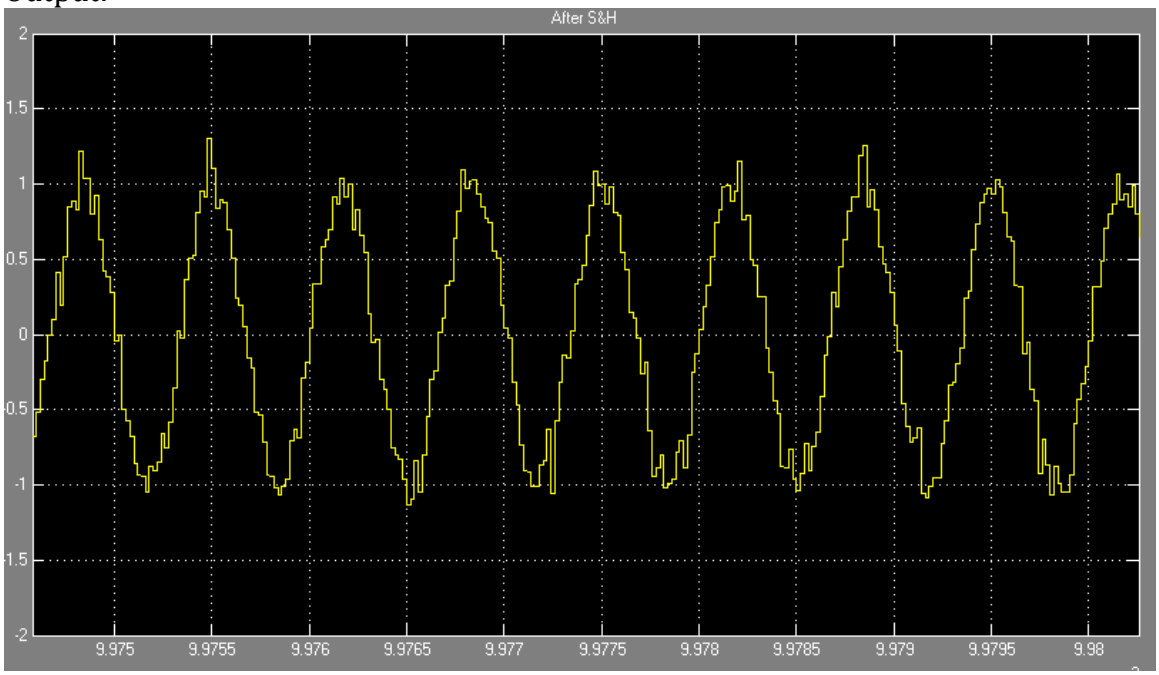
d. Input signal:



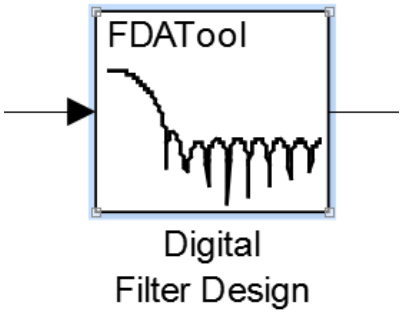
After adding noise:

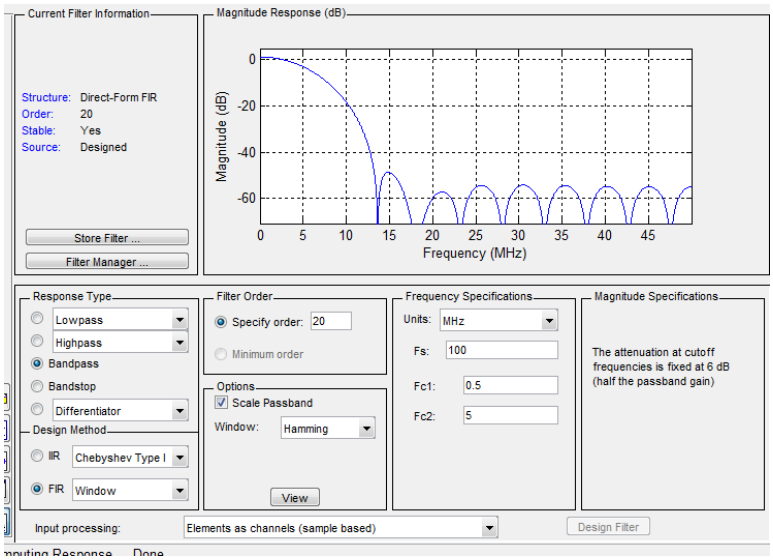


Output:

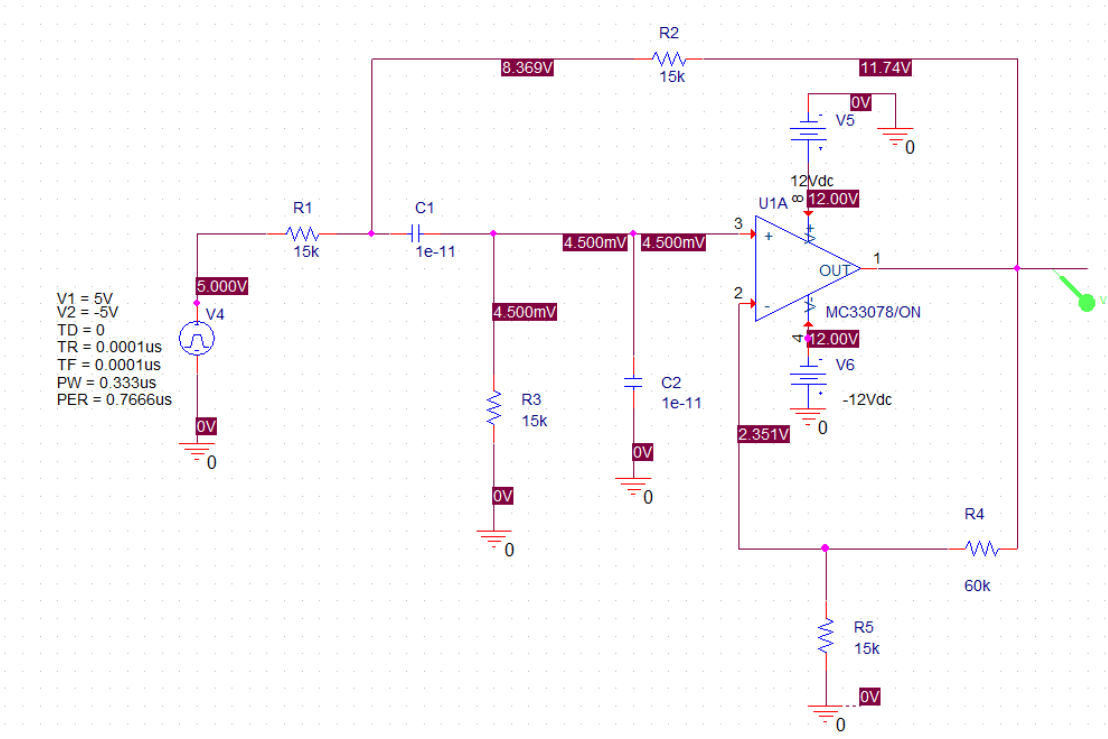


e. Band-pass filter simulation (simulink)

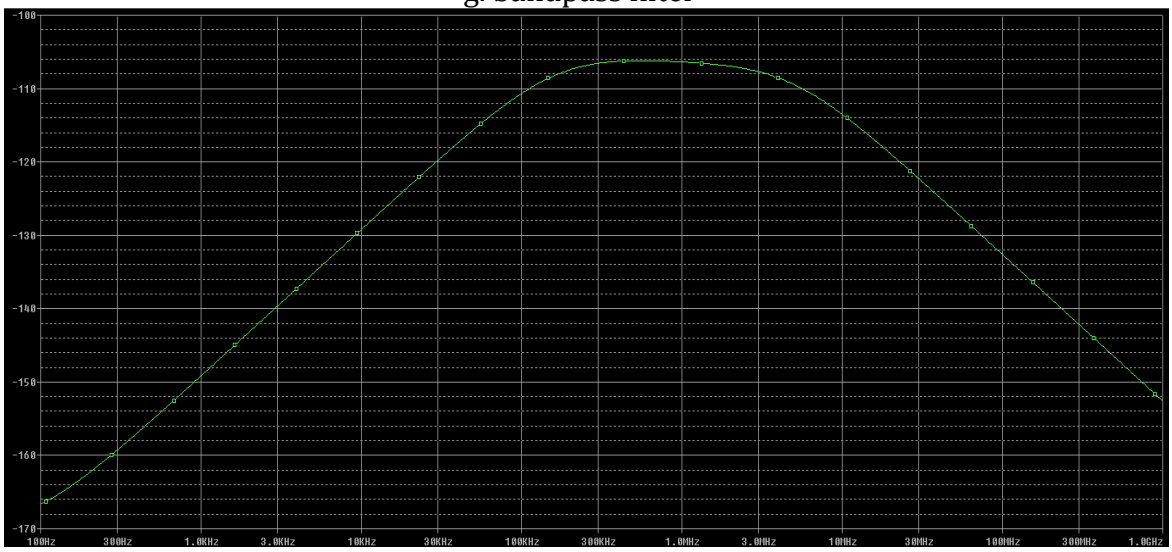




f. Using ECAD build schematic and simulation:



g. bandpass filter



f. bandwidth wave

Implementation Issues and Challenges

Issues

1. Proper settings and control programs
2. Hardware handling methods
3. Correct interpolation algorithm for raw data

Challenges

1. Performance

2. Image manipulation
3. Data translation

These issues are important because they are functional requirements for our system. Without these elements performing correctly, our whole system would fail. In order to get them working properly, we used the following testing procedure. Performance was also be measured through labVIEW controls and timing.

Testing, Procedures and Specifications

We used a circular development pattern in order to produce the products. This made us test, then make improvements, and redevelop. The procedure is shown as following:

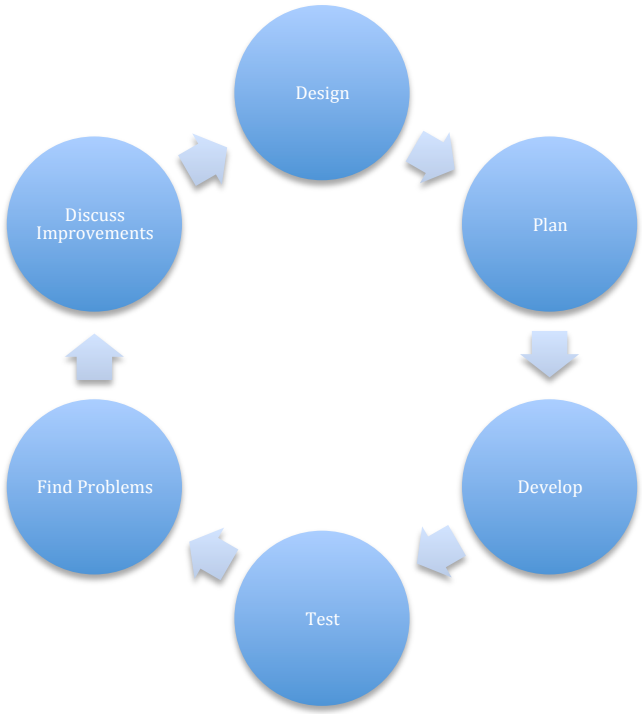


Figure 5 – Testing Process

Software Testing

Testing for this project is more difficult because it is not like typical software development with unit testing (labVIEW). We created a PCB test board that was used to analyze signals with the oscilloscope and send test input with the function generator.

In order to time algorithms to test efficiency, we used a frame timing function available in the labVIEW functions menu. Frames were drawn around each main module of code, and then timed. Adding all of these frames together gave us the total

time for all modules. Individual module times also helped us discover areas that needed to be improved to increase efficiency.

The labVIEW system was wired to a function generator with the analog test board and an oscilloscope with the digital test board. By sending signal through I/O within the specifications of the NI 5752 cards, we were able to test our programs for input and output.

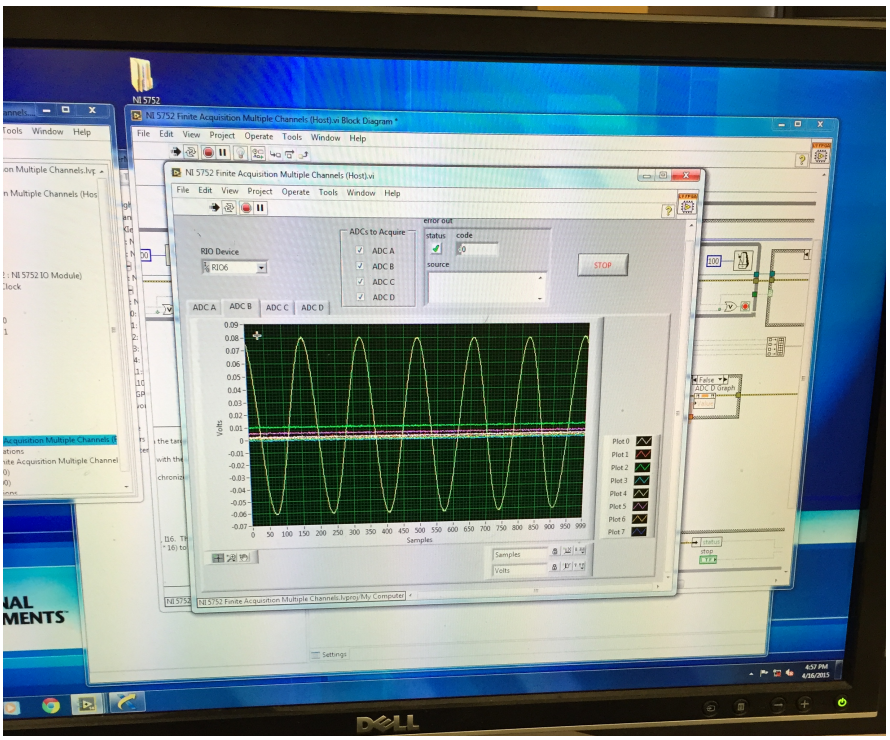


Figure 6 – AI input

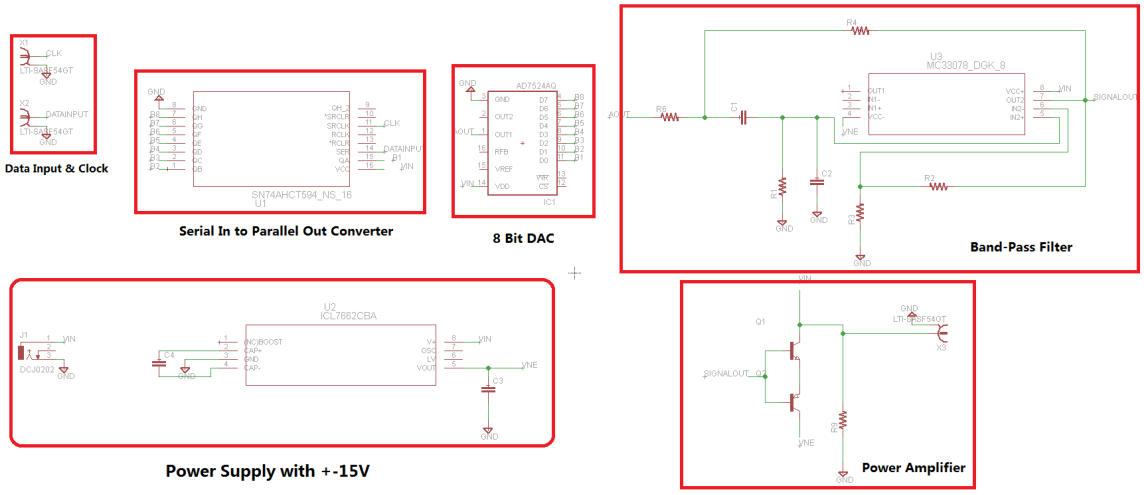
PCB Issues

The PCB used for testing is a very simple adapter component with no extra circuitry. When using the board, we careful that system input does not exceed 200mVpp input on the NI DAQ or else we could damage parts.

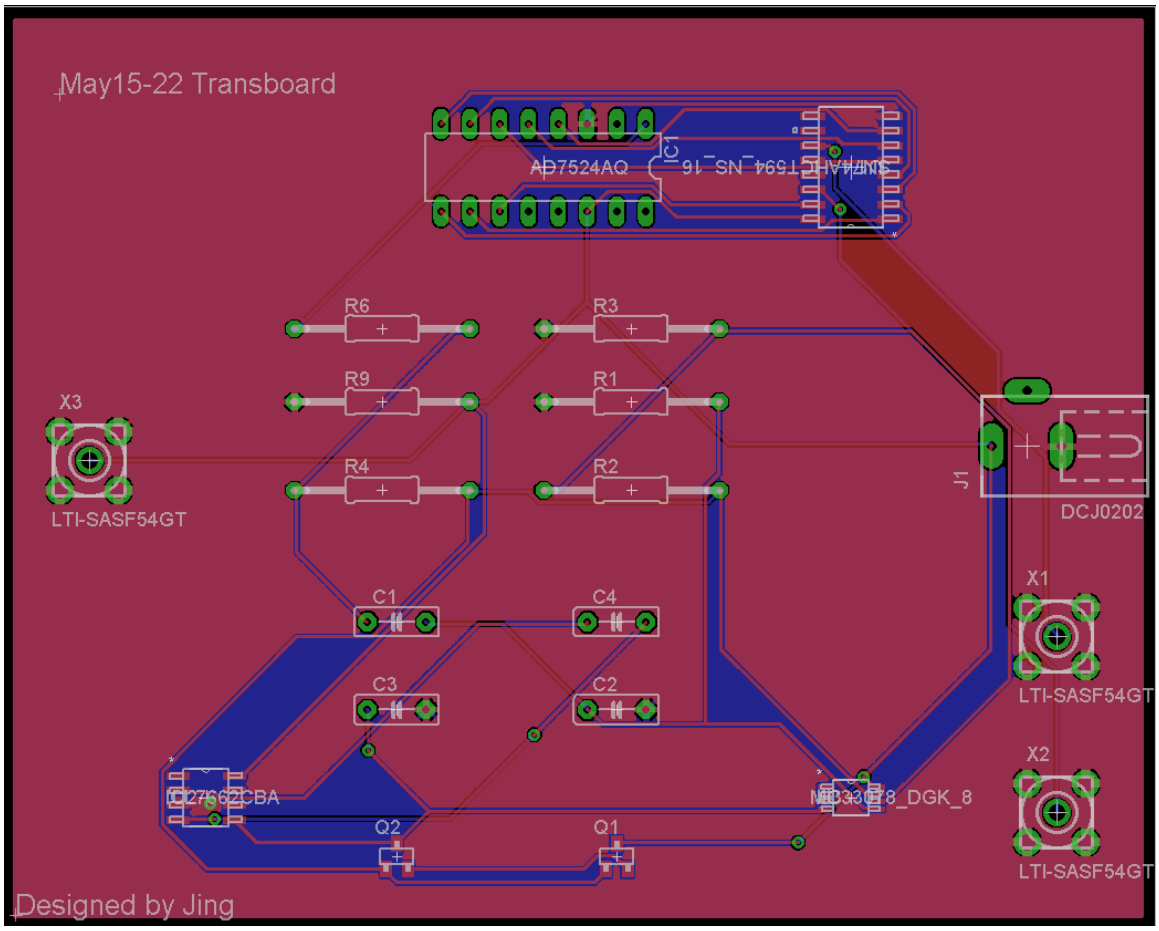
Hardware Testing

PCB Design:

1. Schematic



2. Board:



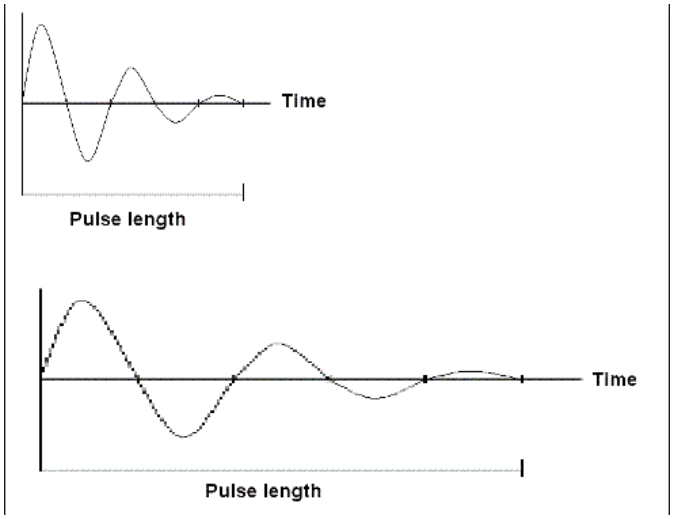
Testing Results

After testing the software side of the project, our results were successful for the majority of the components. The RF data translation and B-Mode imaging was able to produce an accurate image of the test input. The algorithm also function at a fast

150ms (8FPS) per image on the highest load conditions. If the system were to function at 256 channels (which is more realistic for this system), the frame rate would more than double and produce images at around 16FPS, which is close to standard for most medical ultrasound imaging devices. We discovered that the pixel compression algorithm was taking a long time to process data, because the labVIEW image functions were very slow. Manipulating the pixel data prior to converting them to image data vastly increased efficiency in or program.

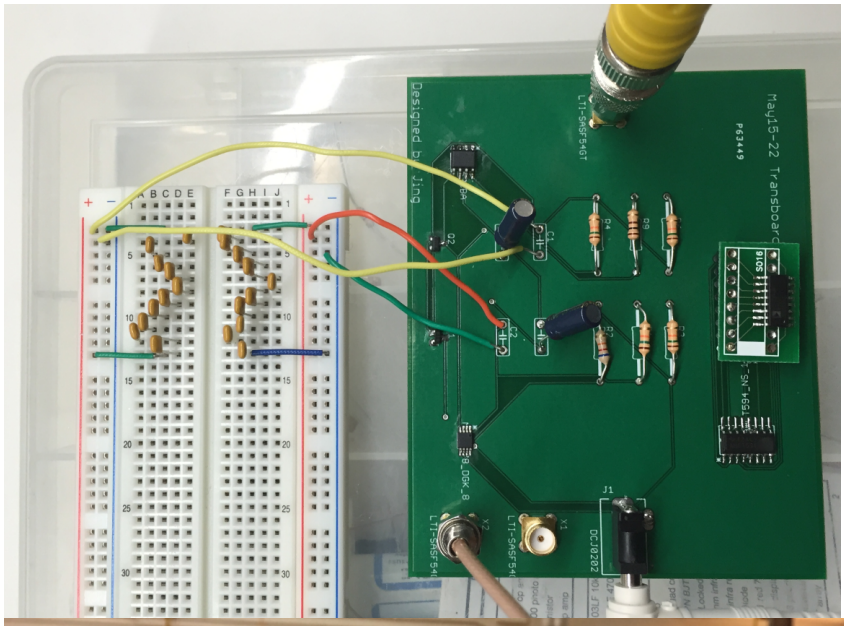
AI also tested to work correctly with the analog test PCB board and function generator input. We were able to test the signal across all channels and ADCs for each individual card. However, when testing the oscilloscope for the DO output, we were unable to read a signal. After testing the DO PCB board, we found that an error in the board design limited us from testing it. We tested the FPGA for DO according to the NI specifications and found that it was setup correctly.

Signal generation hardware testing was done using the function generator and oscilloscope connected to the signal pulse output.

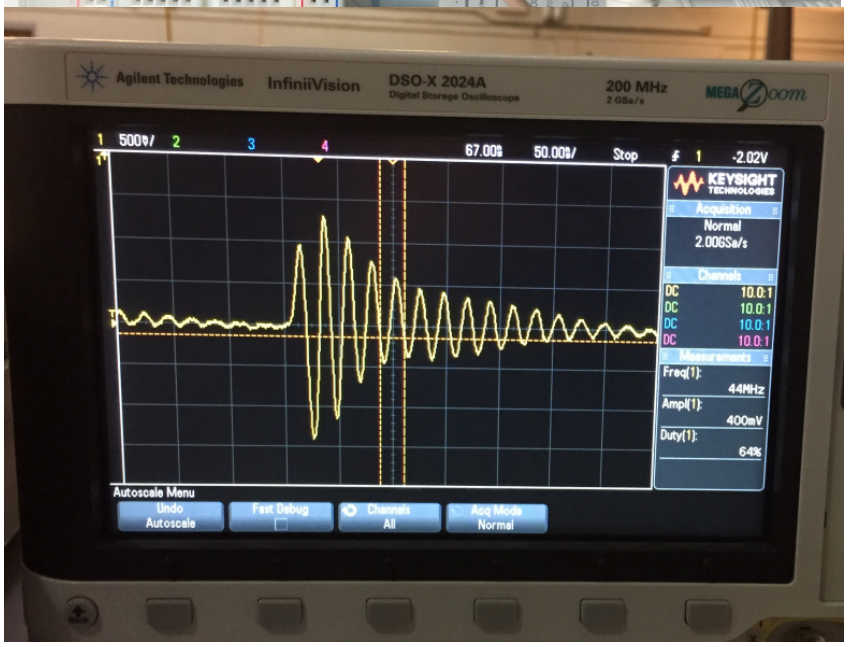


Theoretical result

The graph above is theoretical result of what we wanted to achieve



Actual board



Result

The result is the same as the theoretical result. This pulse generated from our board can be using as input for the Transducer. This result would be replicated across many more channels to produce the pulse needed at the transducer. Unfortunately, we do not have the resources to make multiple boards for all 256 channels.

Interface specifications

Due to the fact that this project is programmed entirely in labVIEW, there are no true Interface methods. The only interface used will be a GUI element created in labVIEW. The GUI conforms to the I/O requirements. Those controls are described below.

Inputs

UI Switch: on/off

Numeric Controls: Amplitude, Channels, Variable Gain, Frequency, Delay Time, Data sensitivity

Dropdown Controls: Realtime/Offline Imaging, Data Storage

Outputs

Image window: Image real time output

File Output: Image offline output, raw data

Serial Data: Programming and serial output

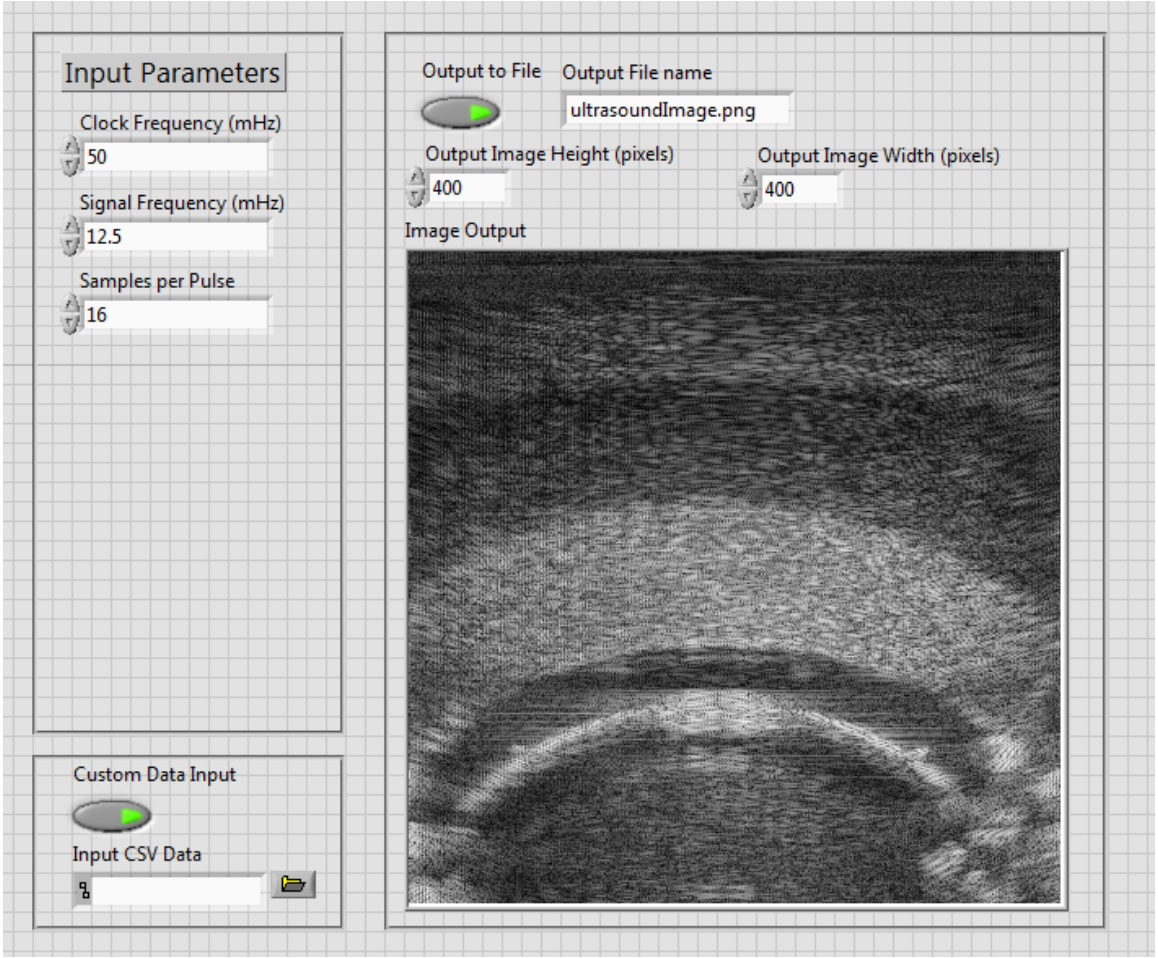


Figure 6 – Ultrasound Panel (Created from sample data)

Conclusion

The results of our project are a several different components necessary for the creation of a fully functioning system. While we did not have the resources available to produce the working prototype, our research should allow future teams to easily develop a functioning product, given they have the resources available to do so

Appendix I

Datasheet of each chip in PCB board:

8 bit serial to parallel signal converter:
http://www.ti.com/lit/ds/symlink/sn74ahct594.pdf

Digital to analog converter:
http://www.ti.com/lit/ds/symlink/sn74ahct594.pdf

Amplifier in the bandpass filter:
http://www.ti.com/lit/ds/symlink/mc33078.pdf

Negative voltage converter:
http://datasheets.maximintegrated.com/en/ds/ICL7662-Si7661.pdf

Ultrasound labVIEW Panel

1. Login to the NI PXIe-8135 system with credentials provided by Professor Bigelow
2. Locate the "Ultrasound" folder on the desktop. This folder contains all of the labVIEW programs written for the ultrasound project.
3. The ultrasoundPanel.vi is the main panel for the ultrasound protocol. Double click to open it and view the main panel.
4. Clicking on Window->Show Block Diagram shows the labVIEW code. The main panel code contains three subVIs that control the panel. The other subVIs are contained in the ultrasound folder. The subVIs are B-Mode_Optimized, Data Acquisition, Serial Programming.
5. Currently, because we do not have all 256 channels working, sample data was used to test the imaging algorithms. Make sure the "Use sample data" button is selected on the main panel and click the run arrow under the edit button.
6. The program will ask you to select the sample data. There should be a file in the ultrasound folder called "ultrasoundData.csv". The sample data contains results from a system with 512 active channels.

7. The image will output on the main panel and also save to file if specified. *Note* because the code is setup for realtime imaging, the main loop makes continues to call the file I/O for custom data. Pressing cancel or the stop button will end the program.
8. *Note* The B-Mode module is the only fully functioning module used in the main panel, the other subVIs are mainly placeholders for future groups. Other programs were used to test I/O for the system. And can be implemented once hardware for all 256 channels is available.

NI 5752 AI Module

1. Login to the system
2. Locate the folder
3. Open the 5752 folder included in the directory. This contains samples used for reading and writing from the 5752 cards.
4. Enter the "FiniteAcqMultipleChannels" folder and double click on the .lvproj file. There may be some dependency error, but ignoring them should be fine.
5. The project contains several FPGA files for different input interfaces. You can view the interface for the particular device in the NI Max program located on the desktop. Expanding the hardware component should show you the RIO devices and their specific interfaces, however, for this machine, the interface should be PXIe-7962R.
6. Double click on the host VI to open it. It should be the file with "(Host)" appended to the end of the name.
7. Click Window->Show Block Diagram to view the labVIEW code.
8. At the very beginning of the block diagram, there should be a LV FPGA module titled "PXIe-7962R". If not, go back the project explorer and expand the PXIe-7962R subdirectory. Drag the FPGA file from that directory over to the block diagram and replace the old one.
9. Try running the host VI by pressing the arrow under the edit button on the block diagram.
    a. If the error is broken, or the Host gives an error, you will need to recompile the FPGA.
    b. To do this, go back to the subdirectory under PXIe-7962R and open the Build specifications. Right click on the file in that directory and select build.
    c. This will take about 20 minutes to run.
    d. *Note* Every time you edit an FPGA or create a new one, you must build again.
10. Once the Host is running, you will see a series of colored data lines being acquired. These are seven channels on each ADC.
11. You can look at different ADCs by selecting the tabs on the graph. There are 4 ADCs on each card. *Note* In order to enable the ADC, you must select the box next to it.

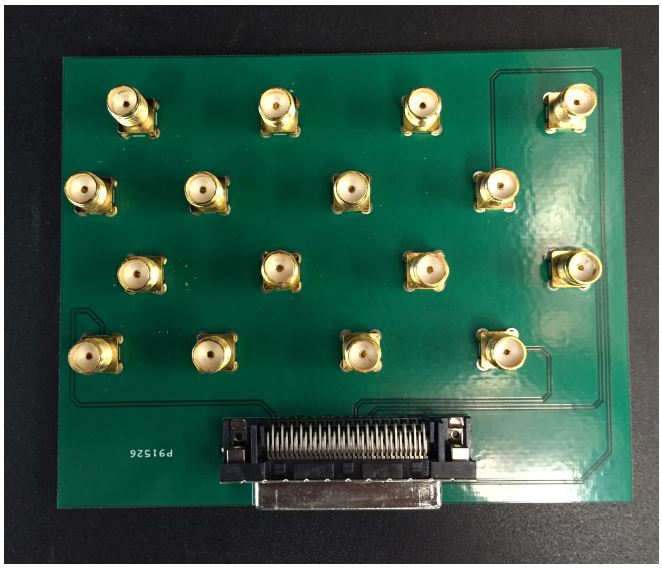
12. You can also change the card by selecting a different RIO device. You can find which RIO corresponds to which card in the NI Max program. It is under Devices and Interfaces-> NI PXIe-1065 “Chassis 1”. Click on each rio, then look at the slot number under the settings.

NI 5752 DO Module

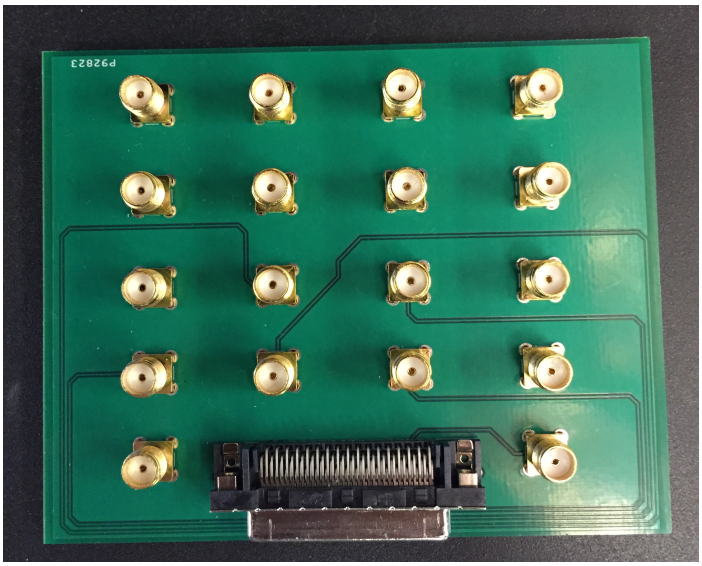
1. Repeat steps 1-8 on the AI section, except open the second project contained in the 5752 folder. The DO module is implemented in this program.
2. In order to view the code for this module, open the PXIe-7962R FPGA. As a test, all channels are output for each DO. A single toggle switch is assigned to all 16.
3. After running the program, pressing the DO toggle switch will toggle all channels on/off.

PCB Testing Boards

Analog



Digital



Appendix II

Version 1: Beamformer Hardware Components (TI Designed)

This version was originally planned by Professor Bigelow and expanded on previous years’ design. The system was to be controlled by the labVIEW PXI system, like our current design, but the hardware between the labVIEW code and the transducer was planned to be a beamformer-based chip designed by a 3rd party (Texas Instruments). Past senior design groups working on this project tried to produce the hardware components, but the results were unsuccessful. After a few

months, TI was unable to produce components that worked reliably and the department lost funding for the parts.

This design was scrapped because the beamformer produced unreliable results. In order to produce the correct waveform, we decided to design our own (more simple) hardware system.

Version 2: Serially Programmed Waveforms

After discovering that the TI created hardware would no longer be implemented in our system, we developed our own design based on existing technology. Because our PXI machine could output a series of bits, we decided to serially program DACs to produce the modulated sine waves needed for the transducer.

Appendix III

Project Pitfalls

When we began this project, there were several issues that were unknown by our group and Professor Bigelow that made the development very slow at the start. These issues included:

- Necessary software not included with the PXI system and it was not configured properly when sent.
- Project design was not solidified until halfway through the project.
- Hardware redesign caused software structure to change.
- Software testing was limited by the system I/O

The first portion of the project was mainly troubleshooting because of these issues. Although it limited the amount of work we were able to accomplish during the first semester, we learned how to communicate effectively with people in industry.

Communication with NI

In order to solve the problems with the misconfigured NI system, we got in contact with support engineers at NI. After multiple calls, we were able to reconfigure the system and install software necessary to start the project.